ડિઝાઇન એન્ડ એનાલિસિસ ઓફ અલ્ગોરિધમસ

ગુજરાતી

Prof. Madhavan Mukund
Computer Science and Engineering
Chennai Mathematical Institute

cmi | CHENNAI MATHEMATICAL INSTITUTE

DESIGN AND ANALYSIS OF ALGORITHMS

Gujarati

Translated by

Name	Institute
Nikita Bhatt	Charotar University of Science and Technology

Verified by

Name	Institute
Ronakkumar Kantilal Panchal	Vidyabharti Trust College of Bca, Umrakh

Index

Sl. No	Topic	Page No
	Week 1 : ઈન્ટ્રોડક્શન, એનાલિસિસ ઓફ એલ્ગોરિધમસ	
1	કોર્સ ની આઉટલાઇન	4
2	એક્ઝામ્પલ : એર ટ્રાવેલ	7
3	એક્ઝામ્પલ: ઝેરોક્સ શોપ	10
4	એક્ઝામ્પલ: ડોક્યુમેન્ટ સિમિલારીટી	13
5	ઈન્ટ્રોડક્શન અને મોટીવેશન	17
6	ઈનપુટ સાઇઝ, વર્સ્ટ કેસ, એવરેજ કેસ	22
7	ક્વોંટઈફેઇન્ગ એફિશિયન્સી: o(),Omega(), Theta()	25
8	એક્ઝામ્પલ્સ: ઇટરેટિવ અને રિકરસીવ મેથડ નું એનાલિસિસ	30
	Week 2 : સર્ચઇંગ એન્ડ સૉર્ટિંગ	
9	એરે અને લિસ્ટ	35
10	એરે માં શોધવું	37
11	સિલેક્શન સૉર્ટ	40
12	ઇન્સરશન સૉર્ટ	44
13	મર્જ સૉર્ટ	48
14	મર્જ સૉર્ટ - એનાલિસિસ	52
15	ક્વીકસોર્ટ	56
16	ક્વીકસોર્ટ - એનાલિસિસ	61
17	સૉર્ટિંગ - કંકલુંડઈનગ રિમાર્ક	65
	Week 3 : ગ્રાફ્સ	
18	ઈન્ટ્રોડક્શન ટુ ગ્રાફ	68
19	રિપ્રેઝેન્ટિંગ ગ્રાફ	72
20	બ્રેડથ ફર્સ્ટ સર્ચ (BFS)	76

21	ડેપથ્ ફર્સ્ટ સર્ચ (DFS)	82
22	DFS અને BFS ની એપ્લિકેશન	86
23	ડાઇરેકટેડ એસાઇલિક ગ્રાફ્સ: ટોપોલોજિકલ સૉર્ટ	93
24	ડાઇરેકટેડ એસાઇલિક ગ્રાફ્સ: લોનગેસ્ટ પાથ	99
	Week 4 : વેઇટેડ ગ્રાફ્સ	
25	સિંગલ સોર્સ શોરટેસ્ટ પાથ : ડીજેકસ્ટ્રાસ એલ્ગોરિધમ	104
26	ડીજેકસ્ટ્રાસ એલ્ગોરિધમ : એનાલિસિસ	109
27	નેગેટિવ એજ વેઇટ્સ: બેલમેન ફોર્ડ એલ્ગોરિધમ	114
28	ઑલ પેર્સ શોરટેસ્ટ પાથ	119
29	મિનિમમ કોસ્ટ સ્પાનઇંગ ટ્રિસ	125
30	પ્રિમ્સ એલ્ગોરિધમ	130
31	ક્રૂસકલ્સ એલ્ગોરિધમ	139
	Week 5: ડેટા સ્ટ્રક્ચર્સ : યુનિયન ફાઇન્ડ અને હિપ્સ, ડિવાઈડ અને કોંકવર	
32	યુનિયન ફાઇંડ યુઝિંગ એરે	145
33	યુનિયન ફાઇંડ યુઝિંગ પોઇન્ટર્સ	153
34	પ્રાયોરિટી ક્યુઝ	159
35	હિપ્સ	164
36	હિપ્સ: અપડેટિંગ વેલયુઝ, સૉર્ટિંગ	175
37	કાઉન્ટિગ ઈન્વર્ઝનસ	181
38	કલોજહેસ્ટ પેર ઓફ પોઇન્ટ્સ	188
	Week 6: ડેટા સ્ટ્રક્ચર્સ : સર્ચ ટ્રિસ, ગ્રિડઈ	
39	બાઈનરી સર્ચ ટ્રિસ	194
40	બેલેન્સડ સર્ચ ટ્રિસ	209
41	ઇન્ટરવલ શિડ્યૂલિંગ	218
42	શિડ્યૂલિંગ વિથ ડેડલાઇન્સ : મિનિમાઈઝિંગ લેટનેસ	224
43	હફમેન કોડ્સ	230

Week 7: ડાયનેમિક પ્રોગ્રામિંગ

44	ઈન્ટ્રોડક્શન ટુ ડાયનેમિક પ્રોગ્રામિંગ	240
45	મેમોઈઝેશન	247
46	ગ્રીડ પાથસ	254
47	કોમન સબવર્ડ્સ અને સીક્વન્સીસ	267
48	એડિટ ડિસ્ટન્સ	276
49	મેટ્રીક્સ મલ્ટિપ્લીકેશન	281

Week 8 : લિનિયર પ્રોગ્રામિંગ અને નેટવર્ક ફ્લોવસ, ઇન્ટરેક્ટએબીલીટી

50	લિનિયર પ્રોગ્રામિંગ	287
51	એલપી મૉડેલિંગ : પ્રોડક્શન પ્લાનિંગ	294
52	એલપી મૉડેલિંગ: પ્રોડક્શન એલોકેશન	299
53	નેટવર્ક ફ્લોવસ	303
54	રીડક્શન્સ	310
55	ચેકિંગ એલ્ગોરિધમ્સ	314
56	પી અને એનપી	323

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ

વીક - 01
મૉડ્યૂલ - 01
લેક્ચર - 01

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:06)

તો, એનપીટીઈએલ એમઓઓસી(MOOC) માં ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms) માં આપનું સ્વાગત છે. તેથી, અહીં કેટલીક બાબતો છે જે આપણે આ કોર્સમાં જોઈશું. જ્યારે અમે એલ્ગોરિધમ્સ(algorithms) નો અભ્યાસ કરીએ છીએ, ત્યારે પ્રથમ વસ્તુ કે જે આપણે આપણી જાતને સમજાવવાની જરૂર છે તે એ છે કે એલ્ગોરિધમ(algorithm) સાચું છે અને તે એવી જ રીતે કરી રહ્યું છે જે અમે અપેક્ષિત છે. તેથી, અમે એલ્ગોરિધમ્સ(algorithms)ની સાચીતાને સાબિત કરવા માટેની વ્યૂહરચનાઓ પર ધ્યાન આપીએ છીએ. અલ્ગોરિધમનો બીજો મહત્વનો પાસું તેની કાર્યક્ષમતા છે. ઍલ્ગોરિધમ્સ ઈનપુટ્સ(inputs) પર કેટલો સમય લે છે? હવે, અલબત્ત, આપણે ઈનપુટ(input)ના કદમાં પરિબળ લગાવવું પડશે. અને અમને એવી રીતની જરૂર છે કે બે અલગ અલગ એલ્ગોરિધમ્સ(algorithms)ની તુલના કરવાની રીત છે, જે સમાન પ્રકારનાં ઈનપુટ્સ(inputs) પર કાર્ય કરે છે અને સમાન પ્રકારના આઉટપુટ બનાવે છે. તેથી, આ એસિમ્પટૉટિક જટિલતા(asymptotic complexity) તરીકે ઓળખાતી રીત દ્વારા પ્રાપ્ત થશે, જે એલ્ગોરિધમ(algorithm)નો ચાલી રહેલ સમય માપશે કેમ કે ઈનપુટ્સ(inputs)ના કાર્ય તરીકે ઈનપુટ્સ(inputs) મોટા અને મોટા થાય છે. અને અમે અમુક સંકેત વિકસાવીશું, ખાસ કરીને બિગ O(big O) સંકેત, જે કેટલાક

((સમયનોસંદર્ભ: 01:00))

એલ્ગોરિધમ્સ(algorithms)ને સરળ બનાવશે અને તેમને જે સમાન છે એવા મોટા હિસ્સામાં જૂથ કરશે. કોઈ પણ ડોમેન(domain) અને ખાસ કરીને એલ્ગોરિધમ્સ(algorithms)માં સમસ્યાનું નિરાકરણનું એક મહત્વપૂર્ણ ભાગ એ, યોગ્ય સ્તરની વિગતવાર સમસ્યાને મોડેલ કરવાની કળા છે. મોટા ભાગનાં ઍલ્ગોરિધમ્સ(algorithms)માં આપણે જોશું કે આપણને યોગ્ય ગાણિતિક મોડેલ શોધવાની જરૂર છે. આમાંથી એક આલેખ હશે. અમારા આલ્ગોરિધમ(algorithm)માં આ મૉડેલ્સની વિભાવનાઓને રજૂ કરવાની રીતની અમને જરૂર છે. આ માટે, અમને યોગ્ય ડેટા સ્ટ્રક્ચર(data structures)ની જરૂર છે. અને સામાન્ય રીતે કોઈ સમસ્યાને હલ કરવા માટે, તેને મેનેજ કરવા યોગ્ય સબ સમસ્યાઓમાં તોડી નાખવાની જરૂર છે. તેથી, આપણે નાની સમસ્યાઓમાં સમસ્યાઓનું વિઘટન કરવા માટેની વ્યૂહરચનાઓ પર ધ્યાન આપીશું અને તેની સમસ્યાને ઉકેલવા માટે તેમને કેવી રીતે મૂકવું તે જુઓ. સમય જતા, મોટી સંખ્યામાં સમસ્યાઓ ઉકેલવા માટે ઘણી સામાન્ય તકનીકો વિકસાવવામાં આવી છે. અને આપણે ઘણી તકનીકી સમસ્યાઓ પર આ તકનીકોને કેવી રીતે લાગુ કરી શકીએ તેના ઉદાહરણો જોઈએ છીએ જે આપણને વારંવાર મળે છે. આ તકનીકો વચ્ચે ડિવાઈડ એન્ડ કોનકોર(divide and conquer) છે જ્યાં, આપણે સમસ્યાને વ્યક્તિગત ઘટકોમાં ભાંગીએ છીએ જે એકબીજા સાથે ઓવરલેપ કરતું નથી અને પછી સમગ્ર સમસ્યાઓ માટે ઉકેલ મેળવવા માટે આ ઉકેલો ભેગા કરે છે. કેટલાક કિસ્સાઓમાં, અમે એવી સમસ્યાની ઓળખ કરી શકીએ જે સમસ્યાની સ્થાનિક સ્થિતિને જુએ છે અને શ્રેષ્ઠ માર્ગ પસંદ કરે છે અને તમામ શક્યતાઓને ધ્યાનમાં લીધા વિના અંતિમ ઉકેલ પર આવે છે. આ પ્રકારની ગ્રીડી(greedy) અલ્ગોરિધમ્સ(algorithms) ત્યાં છે. આવા એલ્ગોરિધમ્સ(algorithms) સાચા કેવી રીતે સાબિત કરવું તે જાણવું મહત્વપૂર્ણ છે. પરંતુ જો ગ્રીડી(greedy) એલ્ગોરિધમ્સ(algorithms) અસ્તિત્વ ધરાવે છે, તો તે સામાન્ય રીતે અન્ય પ્રકારના અલ્ગોરિધમ્સ કરતા વધુ કાર્યક્ષમ છે. જ્યારે ગ્રીડી(greedy) કામ કરતી નથી, ત્યારે આપણે બધી શક્યતાઓની શોધ કરવાનો અને શ્રેષ્ઠ વિકલ્પ પસંદ કરવાની વ્યવસ્થિત રીતની જરૂર છે. આ પ્રક્રિયામાં, કેટલીકવાર આપણે ઓવરલેપિંગ સમસ્યાઓને અવગણવાની હોય છે અને ખાતરી કરીએ છીએ કે અમે સંપૂર્ણપણે ફરીથી વિવાદિત વસ્તુઓને બગાડતા નથી. તેથી, આ ગતિશીલ પ્રોગ્રામિંગ(programming)ની કલ્પના દ્વારા આવરી લેવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:06)
હવે સિદ્ધાંત સાથે, આ કોર્સમાં આપણી પાસે કેટલીક પ્રોગ્રામિંગ(programming) અસાઈનમેન્ટ હશે. તેથી, અમે અપેક્ષા રાખીએ છીએ કે તમારી પાસે પ્રોગ્રામિંગ(programming)માં કેટલીક પૃષ્ઠભૂમિ છે. આ સી(c) અથવા સી પ્લસપ્લસ(C++) અથવા જાવા(Java) માં હોઈ શકે છે. અમે જે ભાષાનો ઉપયોગ કરીએ છીએ તેના વિશે અમે લવચીક છીએ, પરંતુ તમે અમારા અભ્યાસક્રમમાં ઉપયોગમાં લેવાતા પ્રમાણભૂત પ્રોગ્રામ્સ અને કેટલાક મૂળભૂત એલ્ગોરિધમ્સ(algorithms)ને અમલમાં મૂકવા જોઈએ. આ કરવા માટે, અમે આ કોર્સમાં કેટલાક નવા ડેટા સ્ટ્રક્ચર(data structures) આવરીશું. પરંતુ અમે અપેક્ષા રાખીએ છીએ કે તમે ઉપયોગમાં લો છો તે ભાષામાં, તમે એરે(Array) અને લિસ્ટ(lists) જેવી મૂળભૂત વિભાવનાઓથી પરિચિત છો અને સ્ટેક્(stack) અને ક્યુ(queue) જેવી વસ્તુઓ પણ છે જે આના પર નિર્માણ કરે છે. અલબત્ત, જ્યારે આપણે સ્ટેક્(stack) અને ક્યુ(queue)માં આવીએ છીએ ત્યારે અમે તેનો ઉપયોગ કેવી રીતે કરવામાં આવે તે વિશે કેટલીક વિગતો આપવાનો પ્રયાસ કરીશું. પરંતુ અમે અપેક્ષા રાખીએ છીએ કે તમે તેમને પહેલાં જોયા છે. તેથી, આ પહેલી વાર તમે આ ખ્યાલોને જોતા નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:54)
અહીં કોર્સમાં આવરી લેવાની અપેક્ષા રાખતી મુદ્દાઓની એક પ્રકારની અંદાજિત સૂચિ છે. તેથી, કેટલાક ઉદાહરણો જોવા પછી આપણે એસિમ્પટૉટિક જટિલતા(asymptotic complexity) સાથે પ્રારંભ કરીશું, જે એલ્ગોરિધમ્સ(algorithms)ની કાર્યક્ષમતાને માપવા અને આ માપને આ રીતે લખવાનું છે જે આપણે સરળતાથી અલ્ગોરિધમ્સ(algorithms)માં તુલના કરી શકીએ છીએ. પછી આપણે એરે(Array) માટે સૌથી મૂળભૂત સમસ્યા તરફ જઈશું, જે તત્વ માટે એરે(Array) શોધવાનું છે. અને પ્રક્રિયામાં આપણે સમજીશું કે એરે(Array)ને કાર્યક્ષમ રીતે ગોઠવવા માટે આપણે અસ્થાયી રીતે સૉર્ટ(sort) કરી શકીએ છીએ, જેથી આપણે એક અસરકારક રીતે શોધી શકીએ. તેથી, અમે શોધ(search) અને સૉર્ટિંગ(sorting) તરફ જોશું. તેથી, આપણે દ્વિસંગી શોધ((binary search)ને જોઈશું અને સૉર્ટિંગ(sorting)ના જુદા જુદા વિચારોને જોશું. તેમાંના કેટલાક વધુ પ્રાથમિક અને અંતર્જ્ઞાન જેવા છે ઈન્સેરશન અને પસંદગી સૉર્ટ(insertion sort and selection sort). પરંતુ હું કાર્યક્ષમતાના સંદર્ભમાં શ્રેષ્ઠ નથી. અને પછી આપણે વધુ કાર્યક્ષમ અલ્ગોરિધમ્સ જોશો જેમ કે મર્જ સૉર્ટ(merge sort), ઝડપી સૉર્ટ(quick sort); જે શરૂઆતથી સ્પષ્ટ નથી. શોધ(search) અને સૉર્ટિંગ(sorting)થી આગળ વધવું, અમે આલેખ અને ગ્રાફ એલ્ગોરિધમ્સ(algorithms) પર આવીએ છીએ. તેથી, અમે આલેખ રજૂ કરીશું. આપણે જોશું કે કેવી રીતે આપણે ચોક્કસ પ્રકારની સમસ્યાઓનું મોડલ કરવા માટે તેનો ઉપયોગ કરી શકીએ છીએ. ગ્રાફની રજૂઆત કેવી રીતે કરવી તે આપણે જાણવાની જરૂર છે. તમે કેવી રીતે કરવું? ગ્રાફ એ આવશ્યકપણે એક ચિત્ર છે, તો તમે આ ચિત્રને કઈ રીતે અનુવાદિત કરો છો કે જે એલ્ગોરિધમનો ઉપયોગ કરી શકે છે? તેથી, તે રજૂઆત છે. અમે ગ્રાફમાં માનક સમસ્યાઓ જોઈશું; પહોંચ અને જોડાણ (reachability and connectedness). અમે નિર્દેશિત એસાયકલિક ગ્રાફ્સ તરીકે ઓળખાતા ગ્રાફના વિશિષ્ટ વર્ગને જોઈશું, જે ચોક્કસ પ્રકારની સમસ્યાઓનું મોડેલિંગ કરવા માટે ખૂબ જ ઉપયોગી છે. અને પછી આપણે ટૂંકી પાથ અને સ્પાનિંગ ટ્રી(shortest paths and spanning trees) સહિતના ગ્રાફ પરની અન્ય કેનોનિકલ સમસ્યાઓ જોઈશું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:31)
જેમ આપણે પહેલા ઉલ્લેખ કર્યો છે, ત્યાં કેટલીક મૂળભૂત એલ્ગોરિધમિક(algorithmic) ડિઝાઈન તકનીક છે; જે સમસ્યાઓના વર્ગમાં લાગુ પડે છે. તેથી, આપણે ખાસ ડિવાઈડ એન્ડ કોનકોર(divide and conquer), ગ્રીડી(greedy) એલ્ગોરિધમ્સ(algorithms) અને ગતિશીલ પ્રોગ્રામિંગ(programming)માં જોશું. આ કોર્સમાં આપણે જે ડેટા સ્ટ્રક્ચર(data structures)નો સામનો કરીશું તે અગ્રતા ક્યુ(priority queues,) છે, જે ઘણી વાર ઢગલાઓ માટે લાગુ કરવામાં આવે છે. તમે દ્વિસંગી સર્ચ ટ્રી(binary search trees)ને જોશો, જે સૉર્ટ કરેલ ક્રમમાં માહિતીને જાળવી રાખવા માટે કાર્યક્ષમ રીતો છે જેમ માહિતી આવે છે અને જાય છે. અને સેટની ભાગલા
ને સબસેટ્સના અલગ સંગ્રહમાં જાળવવા માટે અમે ખૂબ જ ઉપયોગી ડેટા સ્ટ્રક્ચર જોશું; કહેવાતા યુનિયન-શોધી અલ્ગોરિધમ(union-find algorithm)નો. અને છેવટે, કેટલો સમય બાકી છે તેના પર આધાર રાખીને આપણે કેટલાક પરચુરણ મુદ્દાઓ જોઈશું. એ સમજવું અગત્યનું છે કે દરેક એલ્ગોરિધમ(algorithm) એક કાર્યક્ષમ ઉકેલ સ્વીકારે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:34)
તેથી, આપણે સમસ્યાને અવ્યવસ્થિતતા તરફ જોશું અને કેટલીક સખત મુશ્કેલીઓ અને અન્ય કેટલીક સમસ્યાઓ જે આ સ્થિતિ અજાણ્યા છે તે જોશે. આઠ અઠવાડિયાનો કોર્સ છે. નીચે મુજબની શેડ્યૂલ નીચે મુજબ છે: પ્રથમ સપ્તાહમાં, અમે કેટલાક પ્રેરણાત્મક ઉદાહરણો કરીશું અને અમે કેટલાક સૂચનો જોશું અને એસિમ્પટોટિક જટિલતા(asymptotic complexity)ને સમાવતી કેટલીક સમસ્યાઓનું કાર્ય કરીશું. બીજા અઠવાડિયામાં, અમે શોધ(search) અને સૉર્ટિંગ(sorting) પર આગળ વધીશું. ત્રીજા સપ્તાહમાં, અમે ગ્રાફ્સ રજૂ કરીશું અને મૂળભૂત ગ્રાફ એલ્ગોરિધમ્સ(algorithms) જોશું. અમે વધુ ગ્રાફ એલ્ગોરિધમ્સ(algorithms) સાથે ચાલુ રાખીશું. અને પછી પ્રક્રિયામાં, ડિજેક્ટો સેટ ડેટા સ્ટ્રક્ચર્સ રજૂ કરો. પછી ઔપચારિક રીતે ડિવાઇડ એન્ડ કોનકોર(divide and conquer), જે તમે પહેલાથી જ જોયા છે. પરંતુ આપણે ફરીથી તે જોઈશું. અને પછી આપણે ઢગલો રજૂ કરીશું. છઠ્ઠા અઠવાડિયામાં, અમે શોધ વૃક્ષો અને ગ્રીડી(greedy) એલ્ગોરિધમ્સ(algorithms)ના કેટલાક ઉદાહરણો અને તેમના પુરાવા કરીશું. સાતમી સપ્તાહમાં, અમે ગતિશીલ પ્રોગ્રામિંગ(programming) માટે સમર્પિત થઈશું. અને પછી છેલ્લા અઠવાડિયામાં આપણે વિવિધ વિષયોને આવરી લઈશું. હવે, તમારે યાદ રાખવું જોઈએ કે આ માત્ર એક અંદાજિત શેડ્યૂલ છે અને તેમાં કેટલાક ફેરફારો હશે. પરંતુ મોટેભાગે આ તે ક્રમ અને ગતિ છે જેમાં તમે કોર્સમાં સામગ્રીને આવરી લેવાની યોજના બનાવી છે. તમે અઠવાડિયાની સાથે પણ જોઈ શકો છો. કૅલેન્ડર અઠવાડિયા કે જે આ અનુરૂપ છે. તેથી, અભ્યાસક્રમના પહેલા અઠવાડિયામાં વર્તમાન સપ્તાહ છે જે જાન્યુઆરી પાંચમીથી નવમી છે. અને આ કોર્સ ફેબ્રુઆરીના છેલ્લા અઠવાડિયામાં જશે.
(સ્લાઈડસમયનો સંદર્ભ લો: 07:53)
હવે કોર્સ માટેના મૂલ્યાંકનના ભાગ રૂપે, સતત મૂલ્યાંકન કરવામાં આવશે. દર અઠવાડિયે, ક્વિઝ હશે. તમે પ્રોગ્રામિંગ(programming) અસાઈનમેન્ટ પણ કરશો; આઠ અઠવાડિયામાં લગભગ છ પ્રોગ્રામિંગ(programming) અસાઈનમેન્ટ. અભ્યાસક્રમ સમાપ્ત થયા પછી, પ્રમાણપત્ર પરીક્ષા હશે. તમે આ કોર્સ સફળતાપૂર્વક પૂર્ણ કર્યા છે તે પ્રમાણપત્ર મેળવવા માટે, તમારે ક્વિઝમાં અને પ્રમાણપત્ર પરીક્ષામાં 60 ટકા સ્કોર કરવાની જરૂર છે. તમારે ઓછામાં ઓછા પાંચ છ સબમિટ કરવાની જરૂર છે; છ સોંપણીઓમાંથી. અને ઓછામાં ઓછા, તેમાંના ચાર તમારે કંઈક કરવું આવશ્યક છે
((સમયનોસંદર્ભ લો: 08:27))
સમીક્ષા કરો.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:30)
આપણે બે મુખ્ય પાઠ્ય પુસ્તકોનું અનુસરણ કરીશું. તેમછતાં પણ, હું પાઠ્યપુસ્તકોનો કોર્સ સીધી રીતે ઉપયોગ કરતો નથી, પરંતુ જો તમે કેટલીક સામગ્રીને જોવા માંગતા હો, તો તમને આ બે પુસ્તકોમાં મળશે. જોન ક્લેનબર્ગ અને ઈવા ટેર્ડોસ (Jon Kleinberg and Eva Tardos) દ્વારા પ્રથમ પુસ્તકને "એલ્ગોરિધમ ડિઝાઈન"(algorithm design) કહેવામાં આવે છે. અને બીજી પુસ્તક સંજય દાસગુપ્તા, ક્રિસ્ટોસ પાપાડીમિત્રિઓ અને ઉમેશ વાઝિરા(Sanjay Dasgupta, Christos Papadimitriou and Umesh Vazirani)ની દ્વારા "એલ્ગોરિધમ્સ" (algorithms)કહેવામાં આવે છે. ક્લેનબર્ગ અને ટેર્ડોસ(Kleinberg and Tardos)નું પુસ્તક વધુ વિગતવાર છે અને તેમાં ઘણી બધી વિભાવનાઓ અને તદ્દન વિગતવાર પુરાવાઓને સમજાવવા માટે સંખ્યાબંધ સરસ ઉદાહરણો છે. દાસગુપ્તા પાપદિમિત્રિઉ અને વાઝિરાની(Dasgupta, Papadimitriou and Vazirani) દ્વારા લખાયેલી પુસ્તક એક નાજુક પુસ્તક છે. તે તમારી જાતે વાંચવા માટે વધુ સરળ અને ઍક્સેસિબલ છે. પરંતુ બીજી બાજુ તે વ્યાયામ તરીકે ઘણી બધી વિગતો છોડી દે છે. તેથી, સામગ્રીને ખરેખર સમજવા માટે, તમારે ખરેખર સ્વાધ્યાય કરવા માટે વધુ સમય પસાર કરવો પડશે અને અધ્યાયમાં હાજર સામગ્રી દ્વારા જ નહીં.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)

ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)

વીક - 01
મૉડ્યૂલ - 02
લેક્ચર - 02

તેથી, અમે કોર્સનો ઔપચારિક ભાગ પ્રારંભ કરીએ તે પહેલાં, હું કેટલાક દાખલાઓની ચર્ચા કરવા માંગું છું કે જે પ્રકારની સમસ્યાઓ છે તે માટે આપણે પ્રેરણા આપીશું..
(સ્લાઈડસમયનો સંદર્ભ લો: 00:10)
તેથી, અમે એર ટ્રાવેલની સમસ્યાથી પ્રારંભ કરીએ છીએ. તેથી, અમારી પાસે એરલાઈન છે; બાર્બેટ એરલાઈન્સ(barbet airlines), જે દેશમાં ઘણા શહેરોને સેવા આપે છે. અને અલબત્ત, જોકે તે અનેક શહેરોમાં સેવા આપે છે, તે ખરેખર આ બધા શહેરોને સીધી રીતે કનેક્ટ કરતું નથી. ફક્ત કેટલાક શહેરો સીધા ફ્લાઈટ્સ દ્વારા જોડાયેલા છે. અને શહેરોની અન્ય જોડી માટે તમારે હૉપ્પીંગ ફ્લાઈટ લેવાની રહેશે. તમારે મધ્યવર્તી શહેરમાંથપસાર થવું પડશે. તેથી, અમારું પ્રથમ ધ્યેય કદાચ શહેરોના દરેક જોડીની ગણતરી કરવા, જે આ નેટવર્ક દ્વારા ખરેખર નેટવર્ક દ્વારા જોડાયેલ છે. તેથી, આપણે એ, બી શહેરોના બધા જોડીઓ કેવી રીતે શોધી શકીએ? આ રીતે એ અને બી ફ્લાઈટ્સના અનુક્રમે જોડાયેલા છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 00:50)
તો, સૌ પ્રથમ, આપણે નેટવર્ક પર ધ્યાન આપવાની જરૂર છે. તેથી, આ લાક્ષણિક રીત છે કે આપણે નેટવર્ક શોધી શકીએ. ઉદાહરણ તરીકે, જો અમે કોઈ એરલાઈનના ઈન-ફ્લાઈટ મેગેઝિનને(in-flight magazin) ખોલીએ, તો તમને એક માર્ગ નકશો મળશે. અને તે આ રીતે લખ્યું છે. તમારી પાસે દેશનો ભૌતિક નકશો છે અને તમારી પાસે એવા શહેરો છે જે સેવા આપે છે; ચિલ્લિનત લગભગ દસ શહેરો છે; દિલ્હી, વારાણસી, અમદાવાદ, દક્ષિણમાં ત્રિવેન્દ્રમ સુધી. અને તમારી પાસે ફ્લાઈટ્સ સૂચવેલા કેટલાક તીર છે. હવે, આમાંની કેટલીક ફ્લાઈટ્સ એક દિશામાં છે. તમે દિલ્હીથી વારાણસી જઈ શકો છો, પરંતુ તમે સીધા દિલ્હી પાછા આવી શકતા નથી. તમારે અમદાવાદ જવું પડશે અને પછી પાછા આવવું પડશે. તેથી, આ ખૂબ સામાન્ય છે. જો તમે એરલાઈન્સમાં એરલાઈન્સના સમયપત્રક જુઓ છો, તો તમને મળશે કે આ પ્રકારના ત્રિકોણાકાર રસ્તાઓ છે. જ્યાં તમે એક ત્રિકોણની આસપાસ જાઓ છો અને તમે વચ્ચે હૉપ કર્યા વિના સીધા પાછા જઈ શકતા નથી. મહત્વપૂર્ણ શહેરોની કેટલીક જોડી; આ કિસ્સામાં મુંબઈ અને દિલ્હી બંને દિશામાં અથવા મુંબઈ અને કલકત્તામાં ફ્લાઈટ્સ દ્વારા જોડાઈ શકે છે. અને તેથી હવે આપણી પાસે આ દસ શહેરો છે. આપણે જાણીએ છીએ કે વારાણસીથી ત્રિવેન્દ્રમ કહેવાનું ખરેખર શક્ય છે અથવા તે શક્ય નથી, શું હૈદરાબાદથી દિલ્હી જવાનું શક્ય છે અથવા શક્ય નથી. તેથી, અમારું પ્રથમ પગલું આ સમસ્યાને આ રીતે મોડેલ કરવા માટે છે કે અમે આવશ્યક વિગતો જાળવી રાખીએ, જે સમસ્યાને હલ કરવા માટે સંબંધિત છે અને બિનજરૂરી વિગતો માટે કરવું.
(સ્લાઈડસમયનો સંદર્ભ લો: 02:03)

તેથી, આ કિસ્સામાં આપણે ખરેખર જે જાણવા માંગીએ છીએ તે આ નેટવર્કનું માળખું છે. તેથી, નકશા પોતે સંબંધિત નથી. આપણે જાણવાની જરૂર છે કે ત્યાં કેટલા શહેરો છે, જે નેટવર્ક પર છે અને તે ફ્લાઈટ્સ દ્વારા કેવી રીતે જોડાયેલ છે. તેથી, નીચેની ચિત્ર જે આ ગ્રે વર્તુળો ધરાવે છે અને તીરો આ નેટવર્કનું પ્રતિનિધિત્વ કરે છે. અને આ શહેરો ગ્રે વર્તુળો છે અને ફ્લાઈટ્સ તીર છે અને તીરના માથા દિશા સૂચવે છે. તેથી, જો એક દિશામાં એક તીર માથું હોય, તો તે એક દિશામાન ફ્લાઈટ છે; જો બંને બાજુઓમાં એક તીર માથું હોય, તો તેનો અર્થ એ છે કે તે એક બિડિરેક્શનલ ફ્લાઈટ છે. શહેરોના વાસ્તવિક નામ હું એટલા સુસંગત નથી. તેથી, આપણે તેમને 1, 2, 3, 4, 5 અથવા એ, બી, સી, ડી, ઈ અથવા ગમે તે કહી શકીએ અને સમસ્યાને હલ કરી શકીએ. તેથી, આ પ્રકારની એક ચિત્ર ગ્રાફ કહેવામાં આવે છે. જ્યારે આપણે આ મોડ્યુલમાં આપણા કોર્સમાં આવે ત્યારે આપણે આલેખનો વધુ ઔપચારિક રીતે અભ્યાસ કરીશું. પરંતુ આલેખ ફક્ત આ પ્રકારની એક ચિત્ર છે. તેથી તેમાં કેટલાક ગાંઠો, આ બિંદુઓ અને ધાર છે.(સ્લાઈડસમયનો સંદર્ભ લો: 03:00)
તેથી, હવે આ અમૂર્ત સ્તર પર જવા વિશે એક સરસ વસ્તુ એ છે કે વાસ્તવિક ચિત્ર તેના અર્થને બદલ્યાં વિના વિકૃત થઈ શકે છે. તેથી, અમે ખસેડી શકો છો. દાખલા તરીકે, જો આપણે આ શહેરને અહીં જોઈશું, તો જમણી બાજુએ આપણે તેને જમણી બાજુએ ખસેડી શકીએ અને સમસ્યાને હલ કરવા માટે કોઈ ફરક નથી. અથવા, ઉદાહરણ તરીકે, આ કિનારીને ખસેડીને આપણે ચિત્રને સરળ બનાવી શકીએ છીએ, જેથી અમને કોઈ ક્રોસિંગ ધાર મળે નહીં. અને આ ફરીથી એક જ ચિત્ર છે. જો કે આ ચિત્રથી તે જુદું જુદું લાગે છે, જેની સાથે આપણે પ્રારંભ કર્યું છે, તે ફરીથી સમાન નેટવર્ક છે. હવે, કેટલીક પરિસ્થિતિઓમાં એ સમજવું ઉપયોગી છે કે આપણે જે ગ્રાફ જોઈએ છીએ તે આ રીતે દેખાય છે; કે ત્યાં કોઈ ક્રોસિંગ ધાર છે. તકનીકી રીતે, આવા ગ્રાફને પ્લાનર ગ્રાફ(planner graph) કહેવાય છે. તે કોઈપણ ધારને પાર કર્યા વિના કાગળના ફ્લેટ ટુકડા પર ખેંચી શકાય છે. પ્લાનર ગ્રાફ માટે, અમારી પાસે વધુ સારી એલ્ગોરિધમ્સ હોઈ શકે છે અને મનસ્વી ગ્રાફ માટે. હવે તમે આવા ગ્રાફમાં શું કરવા માંગો છો?
(સ્લાઈડસમયનો સંદર્ભ લો: 04:00)
તેથી, આ સ્થિતિમાં આપણે જે પાથ કહીએ છીએ તેનું ગણતરી કરવા માંગીએ છીએ. તે એક શહેરથી બીજા શહેરમાં જવાના કિનારોનો ક્રમ છે; જ્યાં દિશા નિર્દેશ હોવો જ જોઈએ. તેથી, તમે પાછળથી, આજુબાજુ અને કિનારે જઈ શકતા નથી જે એ થી બી સુધી ઉડતી હોય. તમે ફ્લાઈટ બી થી એ લઈ શકતા નથી, સિવાય કે ત્યાં બીજી ફ્લાઈટ હોય. તેથી, અમારો પ્રથમ પ્રશ્ન એ છે કે આપણે આ ચિત્ર કેવી રીતે લઈશું અને તેને ફોર્મમાં મૂકીશું કે આપણે પ્રોગ્રામ અથવા એલ્ગોરિધમનો ઉપયોગ કરીને ચેપ લગાવી શકીએ છીએ. તેથી, આ ગ્રાફને રજૂ કરવા માટે અમને યોગ્ય ડેટા માળખુંની જરૂર છે. હવે આપણે ગ્રાફને રજૂ કરીએ છીએ તે રીતે, અમે તેને નિયંત્રિત કરીએ છીએ તે પ્રશ્નનો જવાબ આપવા માટે તેને બદલવાની જરૂર છે. આ કિસ્સામાં, કનેક્ટિવિટી. અમે A થી B કેવી રીતે જઈ શકીએ અથવા આપણે A થી B માં જઈ શકીએ અથવા બધા શહેરો B હું A થી પહોંચી શકીએ? તેથી, ગ્રાફમાં આ શહેરોનું પ્રતિનિધિત્વ કરતી રીત મુજબ, અમે આવા અલ્ગોરિધમનો કેવી રીતે ડિઝાઈન કરી શકીએ? શું તે પ્રતિનિધિત્વ અથવા ત્યાં અનેક રજૂઆતો પર આધાર રાખે છે, જેમાંથી કેટલાક આપણને ઓછા કે ઓછા કાર્યક્ષમ અલ્ગોરિધમ આપે છે? આ બધા સવાલો છે જેનો જવાબ આપતા પહેલાં આપણે જવાબ આપવો જરૂરી છે કે આપણે કાર્યવાહીમાં શ્રેષ્ઠ ઉકેલ મેળવ્યો છે કે કેમ. હવે, કાર્યક્ષમતાની દ્રષ્ટિએ આપણે કોઈ સમસ્યાની જટિલતાને નિર્ધારિત કરતી બાબતો પર ધ્યાન આપવું જોઈએ. આ ચોક્કસ કિસ્સામાં તે સ્પષ્ટપણે સ્પષ્ટ છે કે જો આપણી પાસે વધુ શહેરો હોય, તો સમસ્યા વધુ જટીલ છે. તેથી શહેરોની સંખ્યા, જેને આપણે એન કહી શકીએ, ચોક્કસપણે એક પેરામીટર છે જે એલ્ગોરિધમનો કેટલો જટિલ છે તે નક્કી કરે છે, અથવા એલ્ગોરિધમ કેવી રીતે જટીલ હશે તે નહીં, અથવા તેને ચલાવવા માટે કેટલો સમય લાગશે તે નક્કી કરશે. અન્ય પ્રશ્ન એ છે કે નેટવર્ક કેટલા જટિલ છે તે નક્કી કરે છે કે ત્યાં કેટલી સીધી ફ્લાઈટ્સ છે ... દેખીતી રીતે ઓછી ફ્લાઈટ્સ છે, ત્યાં ઓછા સ્થળો છે, જે જોડાઈ શકાય છે અને અમને ઓછી શક્યતાઓની શોધ કરવી પડશે. તો આનાથી, તે અનુસરે છે કે પાથનું ગણતરી એન અને એફ બંને પર આધારિત છે. તેથી, અમારી પાસે એલ્ગોરિધમ નથી જે હંમેશા 20 પગલાં લેશે. તેને એન અને એફ પર આધારીત કેટલાક પગલાઓ પર આધાર રાખવો પડશે. હવે આ નિર્ભરતા શું છે, તે કેવી રીતે વધે છે? જો એન ડબલ્સ, તો શું આપણું અલ્ગોરિધમ બે ગણા વધારે વખત લે છે? શું તે ચાર ગણા વધારે વખત લે છે? જો એન શબ્દ દસના પરિબળમાં પરિણમે છે, તો શું તે દસ ગણો વધુ અથવા સો ગણું વધુ સમય લે છે? આનાથી સંબંધિત અન્ય પ્રશ્ન એન અને

એફ પર આ નિર્ભરતાને આપવામાં આવે છે, જે વાસ્તવિક નેટવર્ક્સનાં કદને આપણે નિયંત્રિત કરી શકીએ છીએ? જો એરલાઈન્સ 20 જેટલી ફ્લાઈટ્સમાં વધારો કરશે, તો પણ અમે અમારા જવાબને વાજબી સમયમાં આપી શકીશું. યાદ રાખો કે જ્યારે કોઈ વ્યક્તિ ઑનલાઈન બુકિંગ અથવા કંઈક કરે છે ત્યારે આ પ્રકારનો જવાબ સામાન્ય રીતે આવશ્યક છે. અને તમને થોડા સેકંડમાં જવાબ આપવા માટે પરવાનગી નહીં મળે, બરાબર, એક કલાક પછી પાછા આવવું પૂરતું નથી અને "હા, ત્રિવેન્દ્રમથી કલકત્તા સુધીની ફ્લાઈટ છે". તો, આપણી કાર્યક્ષમતાની મર્યાદા શું છે? શું એરલાઈન્સ, બહુવિધ એરલાઈન્સને આવરી લેવા માટે અમે આ અલ્ગોરિધમનો સ્કેલ કરી શકીએ? તેથી, અમારી પાસે એક વેબસાઈટ છે જે વાસ્તવમાં વેચે છે. બધી એરલાઈન્સમાં હું તમને એ જગ્યાએથી બી સ્થાને લઈ જઈ શકું છું તે એન અને એફની મોટી કિંમતને નિયંત્રિત કરી શકે છે તેના પર આધાર રાખે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:55)

અને પછી અલબત્ત જે સમસ્યા આપણે જોયેલી તે એક ખૂબ જ સરળ સમસ્યા છે; હું એ થી બી સુધી મેળવી શકું છું પરંતુ ઘણીવાર તે એ થી બી મેળવવા માટે પૂરતો નથી. તમે અમુક વાજબી સમય ફ્રેમમાં A થી B મેળવવા માંગો છો. દાખલા તરીકે, વિમાન પર રાતોરાત મુસાફરી તોડવા સામાન્ય રીતે સ્વીકાર્ય નથી. તે જ સમયે, તમે ફ્લાઈટ્સ વચ્ચેની ચોક્કસ સમયની બેઠક કરતાં વધુ ખર્ચ કરવા માંગતા નથી. તેથી, ફક્ત કેટલાક જોડાણો છે. તેમ છતાં હોઈ શકે છે; ગેરકાયદેસર રીતે જોડાણ હોઈ શકે છે, તેમાંથી કેટલાક માત્ર વાસ્તવમાં સંભવિત હોઈ શકે છે. તેથી, હવે અમારી સમસ્યાઓ થોડી વધુ કમનસીબ બની જાય છે. તેથી, આપણે ફક્ત એ થી બી સુધીના કનેક્ટેડ પાથોને જોવા નથી માંગતા પરંતુ કનેક્ટ પાથ્સ A થી B, જે સમય અને અન્ય વસ્તુઓના સંદર્ભમાં કેટલીક વધારાની અવરોધોને પૂરી કરે છે. તેથી, આપણે આ સમસ્યાનો ઉકેલ મેળવી શકીએ છીએ કે અમે એક સરળ સમસ્યાને હલ કરીએ છીએ અથવા અમારે ધરમૂળથી અલગ અભિગમ લેવાની જરૂર છે અથવા સમસ્યાનો નિર્ણય લેવા અથવા ઉકેલવા માટે અમને વધુ માહિતીની જરૂર છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:53)

ધારો કે, તમે અપેક્ષા રાખશો કે આ વસ્તુ પર પ્રત્યેક ક્ષેત્રનો ખર્ચ થશે. પેસેન્જર તરીકે, કિંમત ટિકિટની કિંમત હશે. તેથી, જો તમે A થી B ની શ્રેષ્ઠ રીતની ગણતરી કરવાનો પ્રયાસ કરી રહ્યાં છો, તો તમારી પ્રેરણા ટિકિટ ખર્ચના સંદર્ભમાં સૌથી સસ્તી માર્ગ પસંદ કરવાનું હોઈ શકે છે. અલબત્ત ખર્ચ માત્ર પૈસા નથી, ખર્ચ પણ સમય હોઈ શકે છે. તમને એ થી બી સુધીનો સૌથી ઝડપી રસ્તો પણ જોઈએ છે, જેનો ઓછામાં ઓછો રાહ જોવાનો સમાવેશ થાય છે. તેથી, તે તમારી પ્રાધાન્યતા પર આધારિત છે. અથવા તમે તાત્કાલિક જ્યાંથી આવશ્યક છે, તે કિસ્સામાં તમને વધુ ચૂકવણી કરવાનું મન નથી. જ્યાં, હું મારા પરિવાર સાથે વેકેશન માટે જાઉં છું; આ સ્થિતિમાં, તમારી પાસે આરામ સમય શેડ્યૂલ છે, પરંતુ તમારે ખાતરી કરવી છે કે તમને પૈસામાંથી મૂલ્ય મળે છે. એરલાઈન્સના દ્રષ્ટિકોણથી અન્ય પ્રશ્નો હોઈ શકે છે. સમયાંતરે વિમાન જાળવણી માટે એક દિવસ માટે નીચે લાવવામાં આવે છે. હવે, તમે એટલા બધા એરક્રાફ્ટ્સ નથી ઈચ્છતા કે તમે બધા માર્ગો ઉડ્ડયન રાખો અને કચરાપૂર્વક વિમાનોનો ઉપયોગ ન કરો. તે જ સમયે જો તમે બે ઓછા વિમાનો રાખો છો, તો જ્યારે તમે જાળવણી માટે વિમાનને નીચે લાવો છો ત્યારે તમારે કેટલાક માર્ગોનું બલિદાન કરવું પડશે. હવે, આપણે કયા રસ્તો બલિદાન આપવું જોઈએ? તેથી, તમે ખાતરી કરો છો કે નેટવર્કની કનેક્ટિવિટી એક જ રહે છે. જો તમે અગાઉથી ત્રિવેન્દ્રમથી કલકત્તા જવાનું ચાલુ રાખી શકો છો, તો જાળવણી શટ ડાઉનમાં તમે હજી પણ ત્રિવેન્દ્રમથી કલકત્તા જઈ શકો છો; કદાચ અલગ દ્વારા. તેથી, એરલાઈન્સ દ્વારા સંબોધવામાં આ સમસ્યા છે. જ્યાં સસ્તું માર્ગ ગ્રાહક દ્વારા સંબોધવામાં સમસ્યા હોઈ શકે છે. તેથી, આ મૂળભૂત હવા નેટવર્ક વિશે તમે પ્રશ્નો પૂછી શકો છો જેનો અમે ગ્રાફ દ્વારા ઉપયોગ કરીને વર્ણન કર્યું છે. અને આ કોર્સમાં આપણે કેટલીક સમસ્યાઓનો જવાબ જોશું.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ **(Design and Analysis of Algorithms)**

પ્રોફેસર માધવન મુકુંદ **(Prof. Madhavan Mukund)**

ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ **(Chennai Mathematical Institute)**

વીક - **01**

મૉડ્યૂલ - **03**

લેક્ચર - **03**

તેથી, અમારું પહેલું ઉદાહરણ એરલાઈન્સ, યાર્દિંગ, નેટવર્ક અને ગ્રાફનો ઉપયોગ કરીને કરવાનું હતું. હવે, ચાલો તેના અલગ ભાગને જોઈએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:08)

તેથી, ધારો કે અમારી પાસે ફોટો કૉપિ શોપ છે, યુનિવર્સિટી કેમ્પસની અંદર ઝેરોક્સ(Xerox) કેમ્પસ છે. પ્રોજેક્ટ્સ માટેની સમય સીમા નજીક આવી રહી છે અને વિદ્યાર્થીઓની ટોળું તેમની યોજનાઓ તાકીદે નકલ કરવા માંગે છે. તેથી, તેઓ બધા જ જાય છે અને ફોટો રિપોર્ટની દુકાનમાં તેમની રિપોર્ટ સબમિટ કરે છે અને કહે છે કે અમે આવી સમયમાં ઘણી બધી નકલો બનાવવી જોઈએ. તેથી, હવે દુકાન માટેનો પ્રશ્ન એ છે કે આ કામને કેવી રીતે સુનિશ્ચિત કરવું તે શ્રેષ્ઠ છે. હવે, ત્યાં ઘણા વિવિધ માર્ગો છે જેમાં આ સમસ્યાને હલ કરી શકાય છે. તેથી, ચાલો આપણે તેમાંની એક તરફ જોઈએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:42)

તેથી, ધારો કે વિદ્યાર્થીઓ તેમની કામ સબમિટ કરે છે અને આ દુકાન કેમ્પસ ઝેરોક્સ કેટલાક પ્રતિસ્પર્ધી સામે સ્પર્ધા કરે છે. તેથી, તે એક ખાસ સોદો ઓફર કરે છે. તેથી, તે કહે છે કે તે દરેક ગ્રાહકને વચન વિતરણ સમય આપશે અને જો પીઝા દુકાન જેવી યાદીને પૂરી કરશે નહીં, તો તે ડિસ્કાઉન્ટ આપશે. તેથી, હું તમારી રિપોર્ટને 6 કલાકની અંદર વચન આપીશ અને જો તમે 6 કલાકની અંદર નહીં મળે, તો તમે ઓછું ચૂકવશો. હવે, આ સમયે ફ્રેમમાં કેટલીક મોટી કામ અને કેટલીક નાની કામ છે. તેથી, કેટલીક ફોટો કૉપીંગ કામ ઝડપથી સમાપ્ત થઈ જશે, કેટલાક લાંબો સમય લેશે, પરંતુ તે જ સમયે તેમને ઝેરોક્સની દુકાનની સમાન મશીનો પર ચાલવું પડશે. તેથી, હવે તમે વસ્તુઓને ફરીથી ગોઠવી શકો છો. તેથી, તમે કંઈક લઈ શકો છો જે પછીથી આવ્યું અને તેને મશીન પર પહેલા મૂકી દો અને આશા છે કે તે તેની અંતિમ મુદતમાં સમાપ્ત કરશે. તેથી, ડિસ્કાઉન્ટ આપવાની જરૂર નથી અને તે કંઈક લેવાનું છે જે લાંબા સમય સુધી લેશે અને કહીને તેને સ્થગિત

કરી દેશે, તેમ છતાં તમે મૂત રેખાને પહોંચી વળવા નથી માંગતા અને તે કામ પર ડિસ્કાઉન્ટ છોડશો નહીં. તેથી, ઝેરોક્સની દુકાનમાં સમસ્યા એ છે કે આ યાદી કેવી રીતે કરવું તે યોગ્ય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:50)

તેથી, બેકગ્રાઉન્ડમાં હંમેશાં રહે છે જેને જૂથ બળ અભિગમ કહેવામાં આવે છે. તમે હવે કહી શકો છો કે મને કેટલીક ફાઈલોમાં આ ફોટો કૉપિ કરવાની કામને મશીન પર ફાળવવાની રહેશે. તો, ચાલો હું દરેક સંભવિત ઓર્ડરનો પ્રયાસ કરું અને તે એક પસંદ કરું જે મને શ્રેષ્ઠ વળતર આપે. આનાથી સમસ્યા એ છે કે આ કરવા માટે ખૂબ મોટો સમય લેશે કારણ કે શક્યતાઓની સંખ્યા ઘટક છે. જો તમારી પાસે માત્ર 30 વિનંતી બાકી છે, તો ઑપ્ટિમાઈઝ(optimize) યાદી શોધવા માટે ઘણાં કલાક લાગી શકે છે અને તે ઘણાં કલાકો આગળ વધ્યા હોઈ શકે છે અને કેટલાક કાર્ય કરી શકે છે, જેથી તમને કામ મળી અને સંભવતઃ શ્રેષ્ઠ રૂપે થોડું પૈસા મળ્યું ન હોય. તે તેથી, અહીં છે જ્યાં આપણે વિઘટનની વિચાર પર આવીએ, જમણી બાજુ. તેથી, આપણે આ સમસ્યાને સરળ સમસ્યામાં ઘટાડીને હલ કરી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:30)

તેથી, ધારીએ છીએ કે આપણે પહેલી વાર એક જૉબને ફિક્સ કરીશું. જો આપણે આ કામને પ્રથમ ચલાવવા માટે ઠીક કરીએ, તો બાકીની કામની બાકીની કામમાં બાકી રહેવું એ સંખ્યામાં નાનું છે. તેથી, જો એન માઈનસ 1 જોબ માટે શ્રેષ્ઠ રીતે ઉકેલ લાવવાનો કોઈ રસ્તો હોય, તો પછી અમે પ્રથમ કામમાંથી દરેકને પસંદ કરી શકીએ છીએ, દરેક પ્રથમ રોજિંદા કામ પ્રથમ કામ હોઈ શકે છે અને તેમાંથી દરેકને તે નક્કી કરે છે કે તે કેટલો સમય અસરકારક રીતે લે છે બાકીના 1 ઓછા 1 કરી શકે છે અને શ્રેષ્ઠ પસંદ કરી શકો છોએક.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:13)

તો, આ આપણને રીકર્સિવ(recursive) સોલ્યુશનનો પ્રકાર આપશે, એક પસંદ કરશે અને બાકીનાને ઉકેલશે અને પછી આનો સમય ઉમેરશે. બીજો વિકલ્પ ફક્ત એક વ્યૂહરચના સાથે આવે છે. હજુ સુધી જે બધી કામ થઈ છે તે જોઈએ છીએ, આપણે કેટલાક માપદંડો શોધી કાઢીએ છીએ જેના દ્વારા આપણે આગળ આવવા માટે એક પસંદ કરીએ છીએ. હવે, અમે અલગ માપદંડ ધરાવી શકીએ છીએ જેના માટે અમે એક પસંદ કરી શકીએ છીએ જેમાં આગળની સંખ્યામાં પૃષ્ઠો છે જે તમે પ્રક્રિયા કરવા માટેનો સૌથી ટૂંકા સમય લે છે, અથવા અમે તેને એક પછીની બાજુએ કરવા માટે કરી શકીએ છીએ જે સમાપ્ત લાઈનની નજીક છે. તે એક છે જેના માટે અમે તેને સમયસર સમાપ્ત કરવાનું અને ડિસ્કાઉન્ટ આપવાનું ચૂકીએ છીએ. તેથી, આમાંના દરેક માટે, અમારી પાસે એક વ્યૂહરચના હોઈ શકે છે જે અમને બીજી કામ કરવાની બધી શક્યતાઓને ધ્યાનમાં લીધા વિના આગળ શું કરવું તે કહેશે, પરંતુ પછી આપણે જે વ્યૂહરચના પસંદ કરી છે તે વાસ્તવમાં શ્રેષ્ઠ છે. સૌથી ટૂંકી અથવા (())પ્રારંભિક

સમય સીમા પસંદ કરવાનું વધુ સારું છે અથવા હજુ સુધી બીજું માપદંડ છે અને આ માપદંડ પસંદગીને કેવી રીતે વાજબી ઠેરવે છે. આ વ્યૂહરચનાને પસંદ કરીને હું હંમેશાં મશીન પર શ્રેષ્ઠ શક્ય વળતર મેળવીશ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:17)

હવે, એરલાઈન નેટવર્ક સમસ્યા સાથે આપણે જોયું તેમ, મૂળ સમસ્યામાં ઘણી વિવિધતા છે જે શક્ય છે. દાખલા તરીકે, જો આપણે માનીએ છીએ કે દુકાનમાં ઘણા ફોટો કોપીરો છે, તો એ ધારવું વાજબી છે કે કેટલાક નવા છે અને કેટલાક જૂના છે. તેથી, જે લોકો નવું છે તે કરતા વધુ ઝડપથી કાર્ય કરી શકે છે તે જૂના છે. તેથી, કામ પૂરી કરવા માટેનો સમય એ છે કે આપણે કઈ મશીન પર કામ મૂકી છે. તેથી, જો તમારી પાસે આ અતિરિક્ત સ્પર્ધા હોય, તો આ પ્રકારની વ્યૂહરચનાઓ જે આપણે બધા પ્રકારની મશીનો માટે પસંદ કરી છે, જે સમાન છે, હજુ પણ જૂની છે, તો પછી અમને જુદી જુદી વ્યૂહરચના જોવાની છે. અન્ય પરિબળ એ છે કે કંઈક કરવાની કિંમત મશીનોમાં બદલાય છે. તેથી, જો આપણે મશીનનો ઉપયોગ કરીએ છીએ; અમે કેટલાક સંસાધનોનો ઉપયોગ કરીએ છીએ, અમે કેટલાક શાહીનો ઉપયોગ કરીએ છીએ, કાગળનો ઉપયોગ કરીએ છીએ, અમે વીજળીનો ઉપયોગ કરીએ છીએ અને આ કિંમત એક મશીનથી બીજામાં બદલાય છે. તેથી, હવે પ્રશ્ન એ પહેલા પ્રશ્ન સાથે સંબંધિત બને છે, અગાઉના પ્રશ્ન એ છે કે હવે જો હું મારી કામને મશીનોમાં વિભાજીત કરું છું, તો તે વધુ અથવા ઓછો સમય લેશે નહીં, તે દુકાનને વધુ અથવા ઓછો ખર્ચી શકે છે. તેથી, દુકાનની વાસ્તવિક આવક તે કઈ મશીન પસંદ કરે છે તેના આધારે કદાચ વધુ અથવા ઓછી અનુભૂતિ કરે છે. બીજી વસ્તુ કે જે આપણે ધ્યાનમાં રાખવાની ઈચ્છા રાખી શકીએ છીએ તે એ છે કે કોઈ વસ્તુ માટે કંઈક સમય માટે મશીનને અનિશ્ચિત સમય સુધી અટકાવી શકાશે નહીં, કાગળ લોડ કરવા માટે, કંઈક માટે. તેથી, આપણે વાસ્તવિકપણે એવું માનતા નથી કે દરેક મશીન સતત ઉપલબ્ધ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:29)

હવે, આ બધી પરિસ્થિતિઓમાં, તે હજી પણ એક માન્ય લોભી વ્યૂહરચના છે અથવા આપણે બીજું કંઈક કરવું પડશે. તેથી, તમે સામાન્ય વિચાર જુઓ છો. સામાન્ય વિચાર એ છે કે કેટલીક સમસ્યાઓ સાથે તમને મૂળભૂત સમસ્યા છે જે તમે હલ કરવા માંગો છો, પરંતુ તે નવી સમસ્યાઓને ઉમેરીને તે સમસ્યાને વિસ્તૃત કરી શકાય છે અથવા વધુ વાસ્તવિક બનાવી શકાય છે. જ્યારે તમે નવી સુવિધા ઉમેરો છો, ત્યારે તમારે જોવું જોઈએ કે તમારી પાસે સરળ સમસ્યાઓ માટેનો ઉકેલ હજી પણ કાર્ય કરે છે કે નવો અભિગમ નવી અભિગમ માટે ધરમૂળથી માંગે છે અને જો તમને તે કેવી રીતે મેળવવું જોઈએ.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)

ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)

વીક - 01

મોડ્યૂલ - 01

લેક્ચર – 04

અત્યાર સુધી અંતિમ ઉદાહરણમાં આપણે આ કોર્સમાં પ્રતિનિધિત્વ કરીએ તે પહેલા, ચાલો દસ્તાવેજોને સમાવી લેતી સમસ્યાની તપાસ કરીએ. તેથી, અમારી પાસે બે દસ્તાવેજો છે અને અમારો ધ્યેય એ શોધવા માટે છે કે તેઓ કેવી રીતે સમાન છે. તેથી, આ બે દસ્તાવેજો ખરેખર સમાન ક્ષેત્રની ભિન્નતા છે. હવે, ત્યાં ઘણી જુદી જુદી પરિસ્થિતિઓ હોઈ શકે છે જ્યાં આ સમસ્યા રસપ્રદ છે. તેથી, એક પ્રશ્ન ચોરી ચોરી શોધવા માટે હોઈ શકે છે. તેથી, તે હોઈ શકે છે કે કોઈએ અખબાર અથવા વેબસાઈટ પર લેખ માટે ફરજ પાડ્યો છે અને તમે માનો છો કે આ લેખકે ખરેખર લેખ જાતે લખ્યો નથી. તેઓએ આ લેખો ક્યાંકથી કૉપિ કર્યા છે અથવા જો તમે કોર્સમાં શિક્ષક છો, તો તમે ચિંતા કરશો કે વિદ્યાર્થી, બે વિદ્યાર્થીઓએ સમાન સોંપણીઓ સબમિટ કરી છે અથવા એક વિદ્યાર્થીએ કેટલાક સ્રોતમાંથી અસાઈનમેન્ટની નકલ કરી છે, કેટલીક વિગતો. તેથી, જો તમે બે સમાન માપદંડને માપવા અથવા સરખાવી શકો છો, તો તમે આ કલ્પનાને નિશ્ચિત કરવાનો પ્રયાસ કરી શકો છો કે કોઈકને કોઈ બીજા દ્વારા કોપી કરી છે. હવે, હંમેશાં આના જેવી નકારાત્મક અર્થ હોતી નથી. જ્યારે કેટલાક લોકો કોડ લખતા હોય ત્યારે કેટલીક પ્રકારની વસ્તુઓ જોવાનું પણ હોઈ શકે છે, ખાસ કરીને કેટલાક એપ્લિકેશન માટે પ્રોગ્રામ્સ લખવાનું, સમયના દસ્તાવેજોના સમયગાળા દરમિયાન પ્રોગ્રામ્સ વિકસિત થાય છે, જમણી બાજુએ. તેથી, લોકો લક્ષણો ઉમેરો. હવે, તમે કોડના બે જુદા જુદા ટુકડાઓ જોઈ શકો છો અને જે ફેરફારો થયા છે તે નક્કી કરવાનો પ્રયાસ કરો. તેઓ કેટલા સમાન છે, તે કેટલાં અલગ છે, વાસ્તવિક ફેરફારો કે જે થયું છે. ડોક્યુમેન્ટ સમાનતા તરફ હકારાત્મક વિચાર હોય તેવી બીજી જગ્યા એ વેબ શોધની શોધ કરવી. જો તમે કોઈ શોધ એંજિન પર કોઈ પ્રશ્ન પૂછો છો અને તે પરિણામોની જાણ કરે છે, તો સામાન્ય રીતે તે સમાન પરિણામ સાથે જૂથ બનાવવાની કોશિશ કરે છે કારણ કે તે ખરેખર અલગ જવાબો નથી. હવે, જો ત્યાં 10 અલગ-અલગ નકલો અથવા દસ્તાવેજની સમાન

કૉપિઝ હોય તો તે જ વસ્તુ વધુ અથવા ઓછી કહેવાતી હોય અને તે તમારા પ્રથમ 10 શોધ પરિણામો તરીકે બતાવે છે, તો પછીનો દસ્તાવેજ ખૂબ સુસંગત રહેશે અને આમાંથી તદ્દન અલગ હશે કારણ કે તે હવે ગુમાવશે શોધના પ્રથમ પૃષ્ઠ હશે. તેથી, સમાનતા દ્વારા શોધ ક્વેરીના પરિણામોને એકસાથે જૂથબદ્ધ કરવા માટે ઉપયોગી થઈ શકે છે, જેથી વપરાશકર્તા વાસ્તવમાં શોધ ક્વેરીના જુદા જુદા જવાબો વચ્ચેની અસરકારક પસંદગી દ્વારા પ્રસ્તુત થાય છે અને ફક્ત તે જ જવાબોની માત્ર મોટી સંખ્યામાં નહીં .

(સ્લાઈડસમયનો સંદર્ભ લો: 02:13)

તેથી, જો આ અમારું પ્રેરણા છે, તો દસ્તાવેજોની તુલના કરવાની રીતની અમને જરૂર છે દસ્તાવેજોની સમાનતા કેટલી સારી છે. હવે, ઘણા જુદા જુદા વિચારો છે જે લોકો સાથે આવ્યા છે. દેખીતી રીતે, તેને ક્રમમાં અને અક્ષરોની પસંદગી સાથે કંઈક કરવું પડશે, પરંતુ દસ્તાવેજને જોઈતી અંતરને માપવાના એક રીત એ છે કે સંપાદન અંતર જેને કહેવામાં આવે

છે તેનો ઉપયોગ કરવો, એટલે કે તમારે કેટલા ફેરફારો કર્યા છે એક ડોક્યુમેન્ટને બીજા ડોક્યુમેન્ટમાં પરિવર્તિત કરો, અને સંપાદનનો અર્થ એ છે કે તમે ખરેખર દસ્તાવેજ સંપાદક અથવા વર્ડ પ્રોસેસરમાં દસ્તાવેજમાં લોડ કર્યું છે, તો તમે કઈ રીતે કરી શકો છો તે તમે કયા સમયે કરી શકો છો, કારણ કે તમે અવરોધિત કરી શકો છો અને આખો દસ્તાવેજ કાઢી નાખો અને પછી બીજા ડોક્યુમેન્ટને કાપી અને પેસ્ટ કરો અને બે પગલામાં ફેરફાર કરો, પરંતુ આ પ્રકારની છેતરપિંડી થઈ શકે છે. તેથી, તમારે જે ઓપરેશન્સ કરવું છે તે મર્યાદિત કરવું પડશે, જેથી અમારી પાસે આ ગણવામાં એક સમાન રીત છે. તેથી, અમે કહી શકીએ કે સંપાદનમાં કેટલોક અક્ષરો બદલાઈ રહ્યો છે તે શામેલ છે. તેથી, સંપાદનના દરેક પગલામાં ક્યાંતો પત્ર ઉમેરશે અથવા દૂર કરશે અને કદાચ તમે એક અક્ષરને બીજા અક્ષર દ્વારા બદલી શકો છો અને તે એક કૉલને કૉલ કરી શકો છો. તેથી, હવે આપણે આને ગણતરીમાં લેવા અથવા પત્રને ઉમેરવા અથવા દૂર કરવા અથવા એક અક્ષરને બીજા સ્થાને બદલવાના મૂળભૂત પગલાં તરીકે ગણીએ છીએ અને એક ડોક્યુમેન્ટને સંપાદિત કરવા માટે કેટલા પગલાં લે છે તે તેને બીજા દસ્તાવેજમાં બનાવવા માટે બનાવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:42)

તેથી, ઓછામાં ઓછા સંપાદન ઑપરેશન પછી અંતર પરત કરશે. હવે, એલ્ગોરિધમ સમસ્યા તરીકે જે પ્રશ્ન છે તે એ છે કે, આ લઘુત્તમ અંતરની ગણતરી કેવી રીતે કરો. એક દસ્તાવેજને સંપાદિત કરવાનો અને તેને બીજો દસ્તાવેજ બનાવવાની શ્રેષ્ઠ રીત કઈ છે તે તમે કેવી રીતે નક્કી કરો છો. અલબત્ત, તે બ્લોક કટ(block cut) અને બ્લોક સ્પેસ(block space) જેવા ટૂંકા ઉકેલ છે. તમે બધા જ અક્ષરોને કાઢી શકો છો અને પછી નવા દસ્તાવેજો લખી શકો છો. તેથી, તે કરવા માટે એક બળવાન બળ માર્ગ છે, પરંતુ આ શ્રેષ્ઠ સંભાવના હોવાનું સંભવ નથી. તેથી, તમે બધા સંભવિત કાઢી નાખો અને અનુક્રમ શામેલ કરવાનો પ્રયાસ કરી શકો છો અને જુઓ કે તેમાંના કયા તમને શ્રેષ્ઠ ઉકેલો આપે છે, પરંતુ આ બધા ખૂબ અપૂર્ણ કાર્યવાહી છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:26)

તેથી, ફરીથી આપણે આ પ્રશ્ન પર જઈ શકીએ છીએ સમસ્યાને સમાપ્ત કરી રહ્યા છે. તેથી, પ્રથમ ધ્યેય સમજો તે ફક્ત એ છે કે બે ડોક્યુમેન્ટના પ્રથમ અક્ષર સમાન છે, જો તેઓ પહેલાથી જ સમાન છે, તો આપણે છોડી આગળ જઈએ છીએ. જો તે સમાન નથી, તો પછી આપણી પાસે બે વિકલ્પો છે. આપણે ક્યાં તો અક્ષરને બદલી શકીએ છીએ, પ્રથમ અક્ષર સમાન હોઈ શકે છે અથવા આપણે બે દસ્તાવેજોમાંથી એકમાં એક અક્ષર શામેલ કરી શકીએ છીએ. તેથી, ડોક્યુમેન્ટને ધ્યાનમાં રાખીને, પ્રથમ ડોક્યુમેન્ટ x થી શરૂ થાય છે અને બીજા ડોક્યુમેન્ટમાં z છે. કાં તો આપણે કહી શકીએ કે x ને z માં z અથવા x માં x બનાવવા માટે આપણે એક ઓપરેશન કરીએ છીએ અથવા x પહેલા z શામેલ કરીએ તે પહેલાં x દાખલ કરી શકીએ છીએ અથવા x પહેલા z શામેલ કરી શકીએ છીએ, પરંતુ પછી આપણને તે જ જવાબ મળતો નથી. પછી, એક વાર અમે આ કર્યું છે, એક વાર આપણે પહેલો અક્ષર બનાવ્યો છે, પછી આપણે બાકીના દસ્તાવેજોને ઠીક કરવાનો પ્રયાસ કરી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:27)

તેથી, હવે જ્યારે આપણે આ રીતે રિકર્ઝન(recusrion) કરીએ છીએ ત્યારે મુશ્કેલીઓમાંની એક એ છે કે સમાન ઉપ-સમસ્યા ઘણી વખત ઉકેલો માટે બને છે. તેથી, આનો એક વિશિષ્ટ ઉદાહરણ એન ફિબોનાકી(fibonaci) શોધવાનું પુનરાવર્તિત ઉકેલ છે. તેથી, ફિબોનાકી(fibonaci) સંખ્યાઓ વ્યાખ્યાયિત કરવામાં આવી છે, તે ખૂબ શાસ્ત્રીય અનુક્રમ છે. તેથી, પ્રથમ બે ફિબોનાકી(fibonaci) નંબર્સ 1 અને 1 છે. આ પછી તમને આગલા સાધનો ઉમેરીને આગલું

ફિબોનાકી(fibonaci) નંબર મળે છે. તેથી, 1 અને 1 પછી, આગલું એક 2 છે જે 1 વત્તા 1 છે. પછીનું એક 3 છે જે 1 વત્તા 2 છે અને તેથી આગળ. 5 એ 2 વત્તા 3 છે અને તેથી. તેથી, સામાન્ય રીતે પુનરાવર્તિત સંબંધ એ હકીકત

દ્વારા આપવામાં આવે છે કે fn એ પાછલા બે સંખ્યાનો સરવાળો છે, n માઈનસ 1 n માઈનસ 2, અને પછી તમારી પાસે બેઝ કેસ છે જે પ્રથમ બે નંબર્સ f1 અને f2 જેના માટે n માઈનસ 1 અને n માઈનસ 2 આ બે નંબરો માટે વ્યાખ્યાયિત થઈ શકશે નહીં, મૂલ્યો 1. હવે, સમસ્યા એ છે કે જ્યારે તમે રિકર્ઝન(recusrion)ને સીધું લાગુ કરો છો, તો ઉદાહરણ તરીકે સાતમો ફિબોનાકી(fibonaci) નંબરની ગણતરી કરવાનો પ્રયાસ કરો તો, તે આના માટે કહેશે મારે f6 પ્લસ f5 નું ગણતરી કરવાની જરૂર છે, પરંતુ ઉત્તેજક રીતે આ f6 અને f5 લાગુ કરીને, અમને એમ લાગે છે કે f4 જેવી વસ્તુઓ બે વાર આવી રહી છે. તેથી, આપણી પાસે f4 છે જે અહીં આવે છે અને પછી f4 અહીં આવે છે કારણ કે જ્યારે હું f5 કરું છું, ત્યારે મને આ લખવાની જરૂર છે, અને જ્યારે હું f6 કરું છું, મને વારંવાર અરજી કરવાની જરૂર છે. તેથી, જો હું તે કરું, તો હું f ની ઊંડાઈ બે વાર કરીશ અને હકીકતમાં, હું આ F5 નું ગણતરી કરીશ, હું ખરેખર બીજા F ને મેળવીશ. આ f4 હું ઘણી વાર ગણતરી કરું છું, f3 ઘણી વખત અને તેથી . તેથી, આ ગણતરી કરવા માટે ખરેખર અયોગ્ય રીત છે, જ્યારે હું અહીંથી આ માટે કરું છું, મને 1 1 2 3 3 8 8 13 21 મળે છે અને મને લાગે છે કે સાતમો ફિબોનાકી(fibonaci) નંબર વાસ્તવમાં તમે અનુક્રમણિકાનો ઉપયોગ નથી કરતા, જમણી બાજુ . તેથી, આ કરવા માટે ખૂબ જ ઝડપી માર્ગ છે. પુનરાવર્તનનો આદર કરવામાં આવે છે, પરંતુ જો હું તેને વારંવાર કરું છું, તો હું વારંવાર ઘણી બધી સમસ્યાઓ ઉકેલું છું. તેથી, આપણે તેની આસપાસ કેવી રીતે મેળવી શકીએ અને આ તે છે જેને આપણે ડાયનામિક(dynamic) પ્રોગ્રામ કહીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:17)

તેથી, ડાયનામિક(dynamic) પ્રોગ્રામિંગ કહે છે કે સમાન ઉપ-સમસ્યાઓનું બે વખત ગણતરી કરશો નહીં. જ્યારે પણ આપણે સમસ્યાઓને હલ કરીએ, જો 4 ની f મળી હોય, તો તેને જુઓ, તેને ક્યાંક સંગ્રહિત કરો, તેને જુઓ અને ખાતરી કરો કે તમે ફરી f4 કરશો નહીં. તેથી, આ તે તકનીકોમાંની એક છે જે આપણે શરૂઆતમાં જોયેલી હોવાથી આપણે આ કોર્સમાં જઈશું, અને તે મહત્વનું છે કે જ્યારે ઉપ-સમસ્યાઓમાં સમસ્યાઓ ઉભી થાય, ત્યારે તે હંમેશાં એવું નથી હોતું કે ઉપ સમસ્યાઓ જ્યાં સુધી આપણે પેટા-સમસ્યાના આ માળખામાં સહેજ વધુ ઊંડાણપૂર્વક ન જોયા ત્યાં સુધી કાર્યક્ષમ રીતે હલ કરીએ અને ખાતરી કરીએ કે અમે તેમને અસરકારક અનુક્રમમાં હલ કરીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:55)

હવે, હંમેશની જેમ આ સમસ્યા, બે દસ્તાવેજો વચ્ચેના તફાવત અથવા સમાનતા ઘણા જુદા સ્તરે હોઈ શકે છે. તેથી, આપણે શબ્દો, વાસ્તવિક લખાણ પર ધ્યાન કેન્દ્રિત કર્યું છે, પરંતુ જો આપણે ખરેખર શબ્દોના અનુક્રમને ન જોતા હોય, તો આપણે ફક્ત શબ્દોનો સમૂહ કરવા માંગીએ છીએ, તો પછી આપણે શબ્દોની અંતરને સંપાદિત કરી શકીએ છીએ કારણ કે આપણે તેને ફરીથી ગોઠવવાની જરૂર છે શબ્દો, પરંતુ સામગ્રી જો તમે ફક્ત કયા પ્રકારનાં શબ્દો છે તેના સંદર્ભમાં માપદંડ કરો છો, તો આ અમને દસ્તાવેજોના અર્થ વિશે સચોટ સમજ આપી શકે છે. તેથી, જો તમે વાસ્તવમાં કોઈ વિશિષ્ટ શોધ એંજિનમાં કોઈ દસ્તાવેજ શોધતા હોવ, તો તમે વારંવાર શોધી શકશો કે તમે જે શબ્દો માગતા હતા તે એકસાથે થઈ શકશે નહીં, તમે જે ક્રમમાં ઉલ્લેખ કરો છો તેમાં આવી શકશે નહીં. તે ફક્ત તે જ દસ્તાવેજમાં મળશે જેમાં શબ્દોનો સંગ્રહ હશે. તેથી, વેબ શોધ માટે આ ખૂબ જ ઉપયોગી છે અને તમે જે કરવા માંગો છો તે અન્ય વસ્તુ છે, શબ્દોની સમાનતા અને અર્થના શબ્દોને માપવા. તેથી, જો તમે કોઈ દસ્તાવેજ માટે શોધ કરો જેમાં શબ્દ કાર શામેલ હોય અને ત્યાં એક અન્ય દસ્તાવેજ છે જેમાં ઓટોમોબાઈલ શબ્દો શામેલ હોય, તો તે શોધ એન્જિન માટે ઓટોમોબાઈલ ધરાવતાં દસ્તાવેજો પર જવાનું એક સારું વિચાર હોઈ શકે છે, કારણ કે ઓટોમોબાઈલ અને કાર આવશ્યક જ છે વસ્તુ. તેથી, તમે પહેલા જોયું તે અન્ય ઉદાહરણની જેમ, સમસ્યાની વિવિધતા હોઈ શકે છે અને તમારી પાસે પ્રાદેશિક સમસ્યા માટેના સોલ્યુશન હોઈ શકે

છે, તે આ ફેરફારો માટે માન્ય હોઈ શકે છે અથવા નહીં પણ. તેથી, હલ કરવા માટે હંમેશાં નવી અને રસપ્રદ સમસ્યાની સંપૂર્ણ જગ્યા રહેલી છે.

((સમયનોસંદર્ભ લો: 09:25)).

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)

વીક - 01
મૉડ્યૂલ - 05
લેક્ચર - 05

તેથી, આ કોર્સને અલ્ગોરિધમ્સના ડિઝાઈન અને વિશ્લેષણ કહેવામાં આવે છે. તેથી, ડિઝાઈન એવી કંઈક છે જે સમજવામાં સરળ છે. અમને સમસ્યા છે, અમે એક ઍલ્ગોરિધમ શોધવા માંગીએ છીએ, તેથી અમે એક અલ્ગોરિધમનો ડિઝાઈન કરવાનો પ્રયાસ કરી રહ્યા છીએ. પરંતુ વિશ્લેષણનો અર્થ શું છે?
(સ્લાઈડસમયનો સંદર્ભ લો: 00:14)
તેથી, વિશ્લેષણનો અર્થ એ છે કે એલ્ગોરિધમ કેટલું કાર્યક્ષમ છે તેનો અંદાજ કાઢવાનો પ્રયાસ કરી રહ્યો છે. હવે, ત્યાં બે મૂળભૂત પરિમાણો છે જે આ સાથે સંકળાયેલા છે. એક સમય છે, હાર્ડવેરના આપેલા ભાગ પર ઍલ્ગોરિધમનો અમલ કેટલો સમય લાગે છે. યાદ રાખો કે ઍલ્ગોરિધમનો પ્રોગ્રામ તરીકે એક્ઝેક્યુટ કરવામાં આવશે, તેથી આપણે તેને પ્રોગ્રામિંગ ભાષામાં લખવું પડશે અને પછી તેને કોઈ ચોક્કસ મશીન પર ચલાવવું પડશે. અને જ્યારે તે ચાલે છે, ત્યારે અમારી પાસે તમામ પ્રકારનાં મધ્યવર્તી ચલ છે કે જેનો જવાબ આપવાની ગણતરી કરવા માટે અમારે ટ્રૅક રાખવાની જરૂર છે. તેથી, તે કેટલી જગ્યા લે છે? તેને કેટલી મેમરીની જરૂર છે?
(સ્લાઈડસમયનો સંદર્ભ લો: 00:47)
હવે, આપણે એવી દલીલ કરીશું કે આ કોર્સમાં, આપણે જગ્યા કરતાં સમય પર ધ્યાન કેન્દ્રિત કરીશું. આનો એક કારણ એ છે કે, હાર્ડવેરના સંદર્ભમાં સમય વધુ મર્યાદિત પરિમાણ છે. કમ્પ્યુટર લેવાનું અને તેની ગતિને બદલવાનું સરળ નથી. તેથી, જો આપણે કોઈ વિશિષ્ટ પ્લેટફોર્મ પર ઍલ્ગોરિધમનો ઉપયોગ કરીએ છીએ, તો પછી અમે પ્લેટફોર્મને ગતિની અવધિમાં આપી શકીએ તે પ્રભાવ સાથે અમે અટકેલા છીએ. બીજી તરફ મેમરી, કંઈક છે, જે પ્રમાણમાં વધારે લવચીક છે. અમે મેમરી કાર્ડ ઉમેરી અને મેમરીમાં વધારો કરી શકીએ છીએ. અને તેથી એક અર્થમાં, જગ્યા વધુ લવચીક આવશ્યકતા છે અને તેથી અમે તે રીતે વિચારી શકીએ છીએ. પરંતુ આવશ્યકપણે, આ કોર્સ માટે આપણે અવકાશ કરતાં વધુ સમય પર ધ્યાન કેન્દ્રિત કરીશું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:32)
તેથી, જો તમે સમય જોઈ રહ્યા છો, તો આપણે પૂછવું પડશે કે આપણે કેવી રીતે ચાલી રહેલ સમય માપીએ છીએ. તેથી, અલબત્ત, અમે આપેલા કમ્પ્યુટર પર પ્રોગ્રામ ચલાવી શકીએ છીએ અને સેકંડ અથવા મિલીસેકંડમાં જવાબની જાણ કરી શકીએ છીએ, પરંતુ આની સમસ્યા એ છે કે, અલબત્ત હાર્ડવેરના ચોક્કસ ભાગની દ્રષ્ટિએ માપવામાં આવેલા ઍલ્ગોરિધમનો સમય ચાલશે , એક મજબૂત માપ નથી. અમે તેને અલગ કમ્પ્યુટર પર ચલાવીએ છીએ અથવા અમે એક અલગ પ્રોગ્રામિંગ ભાષાનો ઉપયોગ કરીએ છીએ, અમને લાગે છે કે તે જ અલ્ગોરિધમ અલગ સમયનો સમય લે છે. વધુ અગત્યનું, જો આપણે બે અલગ અલગ એલ્ગોરિધમ્સની તુલના કરવાનો પ્રયાસ કરી રહ્યા છીએ, તો જો આપણે એક કમ્પ્યુટર પર એક ઍલ્ગોરિધમ ચલાવીએ, તો બીજા કમ્પ્યુટર પર બીજાને ચલાવો, અમને ગેરમાર્ગે દોરતી સરખામણી મળી શકે છે. તેથી, સમયના કેટલાક એકમોની દ્રષ્ટિએ કોંક્રિટ ચાલી રહેલ સમયને જોવાને બદલે, એલ્ગોરિધમનો કેટલો પગલા લે છે તેના કેટલાક અવતરણ એકમોના સંદર્ભમાં તે કરવું વધુ સારું છે. તેથી, આનો અર્થ એ છે કે, આપણે નક્કી કરવું પડશે કે એક પગલું શું છે. એક પગલું એ એક પ્રકારનું મૂળભૂત કાર્ય છે, એક સરળ પગલું પગલું છે, જે એક અલ્ગોરિધમ કરે છે અને પગલાની કલ્પના, અલબત્ત, અમે કયા પ્રકારની ભાષાનો ઉપયોગ કરી રહ્યા છીએ તેના પર આધાર રાખે છે. જો આપણે સંમિશ્રણ ભાષા જેવી વસ્તુ પર ખૂબ જ નીચા સ્તરે જોવું હોય, તો પછી પગલાંઓ મુખ્ય મેમરીમાંથી ડેટાને રજીસ્ટર કરવા, તેમાં પાછા જવા, સીપીયુમાં કેટલાક અંકગણિત કામગીરી કરવા અને તેમાં શામેલ હોવાનો સમાવેશ થાય છે. પરંતુ

સામાન્ય રીતે, અમે એલ્ગોરિધમ્સ પર ધ્યાન આપીએ છીએ અને અમે તેમને ડિઝાઈન કરીએ છીએ અને અમે તેમને ઉચ્ચ સ્તર પર લાગુ કરીએ છીએ. અમે સી, સી પ્લસ પ્લસ અથવા જાવા જેવી પ્રોગ્રામીંગ ભાષાઓનો ઉપયોગ કરીએ છીએ જ્યાં આપણી પાસે ચલો છે, આપણે વેરીએબલો(variables)ને વેલ્યુ અસાઈન કરીએ છીએ, આપણે અભિવ્યક્તિની ગણતરી કરીએ છીએ અને બીજું. તેથી, મોટાભાગના ભાગ માટે અમે મૂળભૂત કામગીરી તરફ ધ્યાન આપશું કેમ કે અમે એકલ કથન અથવા ઉચ્ચ-સ્તરની ભાષામાં પગલાને ધ્યાનમાં લઈશું. તેથી, આ એક ઑપરેશન હોઈ શકે છે જેમ કે X ની કિંમત 1 ની બરાબર કિંમત સોંપવું અથવા તુલના કરતા, જો B કરતા ઓછું હોય, તો પછી કરવું. તેથી, એ એ ચકાસવું કે બી એ કરતાં ઓછું છે કે કેમ તે આપણા માટે એક પગલું છે. હવે, આપણે પગલાંઓની થોડી વધુ વિગતવાર કલ્પનાને જોઈ શકીએ છીએ. ઉદાહરણ તરીકે, આપણે ધારવું જોઈએ કે, વાસ્તવમાં બે વેરીએબલ્સ(variables)માં મૂલ્યોનું વિનિમય કરવા માટે આપણે એક આદિમ ઓપરેશન ધરાવીએ છીએ, જો કે આપણે જાણીએ છીએ કે અમલીકરણ કરવું એ ખરેખર અસ્થાયી ચલ દ્વારા જવું પડશે. આપણે જોશું કે, આપણે કલ્પના સાથે આવીશું, જે મજબૂત છે તેથી મૂળભૂત કામગીરીની વાસ્તવિક કલ્પના એટલી સુસંગત નથી કારણ કે અમે ધ્યાન કેન્દ્રિત કરવાનું પસંદ કરીએ છીએ તે મુજબ આપણે તેને નીચે અને નીચે સ્કેલ કરી શકીશું. .

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:55)

અગત્યનું લાગે તેવું બીજું મહત્વનું છે, અલબત્ત એ છે કે એલ્ગોરિધમ તેની સાથે રજૂ કરવામાં આવેલી સમસ્યાના કદના આધારે જુદા જુદા સમય લેશે. તે એકદમ પ્રાકૃતિક અને સ્પષ્ટ છે, જો આપણે કોઈ એરે(array)ને સૉર્ટ કરવાનો પ્રયાસ કરી રહ્યા છીએ, તો ટૂંકા એરે(array)ને સૉર્ટ કરવા માટે તે લાંબી એરે(array)ને સૉર્ટ કરવા માટે વધુ સમય લેશે. તેથી, અમે તેના ઈનપુટ કદના ફંક્શન તરીકે ઍલ્ગોરિધમની કાર્યક્ષમતાને રજૂ કરવા માંગીએ છીએ. તેથી, જો ઈનપુટ કેટલાક કદ n ની છે, તો તે n નો સમય ટશે, જ્યાં ઈનપુટના આધારે ટી કાર્ય કરશે જ્યાં આ તાત્કાલિક એક સ્પષ્ટ વ્યાખ્યા નથી કારણ કે કદ ના બધા ઈનપુટ્સ એ જ લેશે નહીં. કેટલો સમય. કેટલાક ઈનપુટમાં ઓછો સમય લેશે, કેટલાક ઈનપુટ વધુ સમય લેશે. તેથી, અમે શું લેતા? તેથી, તે ચાલુ થશે, કે જે કલ્પના, આપણે સામાન્ય રીતે કદ n ના બધા ઈનપુટ્સને જોવું જોઈએ અને સૌથી લાંબી શક્ય તેટલું જોવું જોઈએ જે યોગ્ય લાંબો સમય લે છે. તેથી, આને સૌથી ખરાબ કેસના અંદાજ તરીકે ઓળખવામાં આવે છે, તે નિરાશાવાદી અંદાજ છે, પરંતુ આપણે સાથે જતા જ ન્યાયી ઠરાવીશું. પરંતુ આનો આપણે અર્થ કરીએ છીએ. હવે, જ્યારે આપણે એલ્ગોરિધમનો સમય કાર્યક્ષમતા જોઈ રહ્યા છીએ ત્યારે અમારું શું અર્થ છે, તે કદના ઈનપુટ્સ પર શું લેશે તે સૌથી ખરાબ શક્ય સમય છે અને n ના કાર્ય તરીકે અભિવ્યક્ત કરે છે? તેથી, અમે આને ઔપચારિક બનાવવા પહેલાં, ચાલો આપણે થોડા ઉદાહરણો જોઈએ અને આપણે જે ઉપલબ્ધ કમ્પ્યુટર્સ પર ઉપલબ્ધ છે તેના પર પ્રાયોગિક ચાલી રહેલ સમયના આધારે કાર્યક્ષમતા શું છે તેની અનુભૂતિ કરો.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:13)

તેથી, ચાલો સૉર્ટિંગ સાથે પ્રારંભ કરીએ, જે ઘણા અલ્ગોરિધમ્સમાં એક ખૂબ જ મૂળભૂત પગલું છે. તેથી, અમારી પાસે n એલિમેન્ટ્સ સાથે એક એરે(array) છે અને આપણે આગળ વધવા માટે આ ઘટકો ગોઠવવા, ચડતા અથવા ઉતરતા ક્રમમાં કહીએ છીએ. હવે, એક નમ્ર સૉર્ટિંગ એલ્ગોરિધમ તત્વોના પ્રત્યેક જોડીની તુલનામાં ઓછું કરશે અને તેને ફરીથી ગોઠવવાનું નક્કી કરવાનો પ્રયાસ કરશે. આ એન સ્ક્વેર માટે પ્રમાણસર સમય લેશે કારણ કે અમે તત્વોના બધા જોડીઓની તુલના કરીએ છીએ. બીજી તરફ, આપણે આ કોર્સમાં ટૂંક સમયમાં જ જોશું, કે ત્યાં ઘણા વધુ ચુસ્ત વર્ગીકરણ અલ્ગોરિધમ્સ છે, જે સમયે n લોગ n એ પ્રમાણમાં કાર્ય કરે છે. તેથી, અમે નિષ્ક્રીય અલ્ગોરિધમનો ઉપયોગ કરી શકીએ છીએ જે સમય n ચોરસને વધુ સારા અલ્ગોરિધમનો લે છે, જે સમય n લોગ n લે છે. તેથી, આપણે ખરેખર જે કર્યું છે તે છે, આપણે n નો પરિબળ લીધો છે અને લોગ n ના પરિબળ દ્વારા તેને બદલ્યો છે. તેથી, આ પ્રમાણમાં નાનો ફેરફાર લાગે છે. તેથી, આપણે જોઈ શકીએ કે આ પરિવર્તનની અસર શું છે? તેથી, આને જોવાનો એક રસ્તો વાસ્તવમાં કોંક્રિટ ચાલી રહેલા સમયમાં જોવાની છે. અમે કહ્યું હતું કે, અમે એલ્ગોરિધમનો કાર્યક્ષમતા માપવા માટે ચાલી રહેલ સમયને જોશું નહીં, પરંતુ અલબત્ત, ઍલ્ગોરિધમની કાર્યક્ષમતાને અમલમાં મૂકવા માટે કેટલો સમય લાગશે તેના પર અસર કરે છે, અને તેથી પ્રોગ્રામ કેવી રીતે ઉપયોગી છે તેના પર અમારા વ્યવહારિક દ્રષ્ટિકોણથી. તેથી, જો તમે લાક્ષણિક CPU ને લેતા હો કે જે ડેસ્કટોપ અથવા લેપટોપ પર મળે છે, જેનો આ દિવસો છે, તો તે લગભગ 10 થી 8 ઑપરેશન સુધી પ્રક્રિયા કરી શકે છે. આ

અસાઈનમેન્ટ જેવી ઉચ્ચ-સ્તરની ભાષાઓમાં મૂળભૂત કામગીરી છે અથવા તપાસવું કે એક મૂલ્ય અન્ય મૂલ્ય કરતાં ઓછું છે, આપણે કહી શકીએ કે, તે લેશે, તે લગભગ 10 થી 8 ઓપરેશન્સ કરી શકે છે. હવે, આ એક અંદાજીત નંબર છે, પરંતુ તે ખૂબ જ છે, આપણે નંબરો પર કેટલાક હેન્ડલ મેળવવાની જરૂર છે જેથી અમે આ અલ્ગોરિધમ્સની થોડી જથ્થાત્મક તુલના કરી શકીએ. તેથી, તે અંદાજિત ગણતરીઓ માટે ઉપયોગી સંખ્યા છે. હવે, યાદ રાખવાની એક વાત એ છે કે આ સંખ્યા બદલાતી નથી. દેખીતી વાત એ છે કે, આપણે એક એવી પરિસ્થિતિ ઊભી કરવા માટે ઉપયોગ કરીએ છીએ જ્યાં સીપીયુ(CPU) દર દોઢ વર્ષ સુધી ગતિ કરે છે, પરંતુ હવે આપણી પાસે હાલની તકનીકની ભૌતિક મર્યાદાઓ સુધી પહોંચેલું છે. તેથી, સીપીયુ(CPU) ઝડપી નથી. આ, આસપાસના, સમાંતર, મલ્ટિકોર(Multicore) અને આજુબાજુ મેળવવા માટે અમે વિવિધ પ્રકારની તકનીકોનો ઉપયોગ કરી રહ્યા છીએ. પરંતુ આવશ્યક રીતે, સીપીયુ(CPU)ની ક્રમશઃ ગતિ દર સેકન્ડમાં 8 થી 8 ઓપરેશન્સ પર અટકી જાય છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:46)

તેથી, હવે જ્યારે આપણે એક નાનો ઈનપુટ લઈએ, તો ધારો કે અમે અમારા મોબાઈલ ફોન પર અમારી સંપર્ક સૂચિને ફરીથી ગોઠવવાનો પ્રયાસ કરી રહ્યા છીએ. અમારી પાસે કેટલાક સો સંપર્કો, કદાચ હજાર સંપર્કો, કદાચ થોડા હજાર સંપર્કો હોઈ શકે છે. જો તમે કોઈ સંપર્ક સૂચિને સૉર્ટ કરવાનો પ્રયાસ કરો છો, તો નામ દ્વારા કહો, તે ખરેખર અમારામાં ઘણું ફરક કરશે નહીં કે પછી આપણે n ચોરસ અથવા n લોગ n એલ્ગોરિધમનો ઉપયોગ કરીએ છીએ. તે બન્નેના બીજા ભાગમાં કામ કરશે અને આપણે તે જાણતા પહેલા, અમારો જવાબ હશે. પરંતુ જો આપણે વધુ નજીવી કદના માપો પર જઈએ, તો આ તફાવત બદલે સ્થિર બનશે. તેથી, દેશભરના તમામ મોબાઈલ સબ્સ્ક્રાઈબર્સની સૉર્ટ કરેલ સૂચિ સંકલનની સમસ્યાને ધ્યાનમાં લો. તેથી, તે તારણ આપે છે કે ભારત પાસે લગભગ એક અબજ છે, એટલે કે 10 થી 9 મોબાઈલ સબ્સ્ક્રાઈબર્સ છે. આમાં ડેટા કાર્ડ્સ, ફોન, વિવિધ વસ્તુઓ શામેલ છે, પરંતુ આ તે બધા લોકો છે જે બધા કેટલાક ટેલિકોમ ઓપરેટર સાથે નોંધાયેલા છે અને એક સિમ કાર્ડ ધરાવે છે. તેથી, આ સંખ્યાબંધ સિમ કાર્ડ્સ છે, જે આજે ભારતમાં સક્રિય ઉપયોગમાં છે. તો, માની લો કે આપણે ભારતમાં સેમ કાર્ડ્સના તમામ માલિકોની સૂચિ સંકલન કરવા માંગીએ છીએ. કારણ કે અમારી પાસે 10 થી 9 સબ્સ્ક્રાઈબર્સ છે, જો અમે n ચોરસ અલ્ગોરિધમનો સ્કોર કરીએ, તો આમાં 10 થી લઈ જશેઈ 18 ઓપરેશન્સ, કારણ કે 10 થી 9 સ્ક્વેર 10 થી 18 છે. હવે, 10 થી 18 ઓપરેશન્સ સેકંડમાં, કારણ કે આપણે માત્ર 10 સેકન્ડમાં 8 ઓપરેશન્સ કરી શકીએ, 10 સેકન્ડમાં લઈશું. તેથી, 10 થી 10 સેકંડ કેટલા છે? ઠીક છે, લગભગ 2.8 મિલિયન કલાક છે; તે આશરે 115 હજાર દિવસ છે, જે લગભગ 300 વર્ષ છે. તેથી, તમે કલ્પના કરી શકો છો કે, જો આપણે એન સ્ક્વેર્ડ અલ્ગોરિધમનો ઉપયોગ કરીને ખરેખર આ કરવા માંગતા હોઈએ, તો તે ખરેખર વ્યવહારુ નહીં હોય કારણ કે તે વર્તમાન હાર્ડવેર પર તેનું ગણતરી કરવા માટે, વાસ્તવમાં ઘણી પેઢીઓ કરતાં વધુ, આપણા જીવનકાળ કરતાં વધુ લેશે. બીજી તરફ, જો આપણે સ્માર્ટ એન લોગ એન એલ્ગોરિધમ પર જવાનું હતું, જેનો અમે દાવો કરીશું તો અમે શોધીશું, પછી તે તારણ આપે છે કે આ મોટી સંખ્યામાં પોતાના વપરાશકર્તાઓને સૉર્ટ કરવું ફક્ત 3 ગુણ્યા 10 થી 10 થાય છે કારણ કે લોગ 10 થી 9 ની બેઝ 2 એ 30 છે. તે યાદ રાખવામાં ઉપયોગી છે કે 2 થી 10 એ 1000 છે. તો, બેઝ 2 ની 1000 માં 10 લોગ થાય છે. હવે, લોગ 1000 વખત ઉમેરે છે, 1000 થી 10 થાય છે. 6, ઠીક છે, તેથી 10 થી 6 નું લોગ 20 છે, 10 થી 9 ની સંખ્યા 30 છે. તેથી, 30 થી 10 માં 9 n લોગ n એ 3 ગુણ્યા 10 થી 10 થાય છે. તો આનો અર્થ એ થાય કે તે લગભગ 300 સેકન્ડ લેશે. તેથી, તે લગભગ 5 મિનિટ લેશે. હવે, 5 મિનિટ ટૂંકા સમય નથી, તમે જઈ શકો છો અને ચા મેળવી શકો છો, પરંતુ હજી પણ તે એકદમ ટૂંકા સમયમાં કરવામાં આવશે, જેથી અમે આ ઉપયોગી માહિતી સાથે કામ કરી શકીએ અને વિરોધ તરીકે આગળ વધીએ. 300 વર્ષ સુધી, જે તદ્દન અવ્યવહારુ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 10:35)

તેથી, ચાલો બીજો એક ઉદાહરણ જોઈએ. તેથી, ધારો કે અમે વિડિઓ ગેમ રમી રહ્યા છીએ, જમાણે. તેથી, આ એક્શન પ્રકારની રમતોમાંની એક હોઈ શકે છે જ્યાં સ્ક્રીનની ફરતે વસ્તુઓ ફરતી હોય છે અને આપણે જાણવાની જરૂર છે, ચોક્કસ ઑબ્જેક્ટ(object) ઓળખો, તેમને નીચે શૂટ કરો, તેમને કેદ કરો. તેથી, ચાલો ધારીએ કે સ્કોરની ગણતરી કરવા માટે રમતના ભાગરૂપે, તેને સમયાંતરે સ્ક્રીન પરની વસ્તુઓની નજીકની જોડી શોધી કાઢવી પડશે. તેથી, હવે, તમે પદાર્થોના સમૂહમાં સૌથી નજીકના પદાર્થો કેવી રીતે શોધી શકો છો? ઠીક છે, અલબત્ત, તમે તેમાંના દરેક જોડીને લઈ શકો છો, દરેક

જોડી વચ્ચેની અંતર શોધી શકો છો અને પછી સૌથી નાનું, જમણે લઈ શકો છો. તો, આ એન સ્ક્વેર્ડ અલ્ગોરિધમ હશે, જમણે. તેથી, તમે કોઈપણ બે વસ્તુઓ વચ્ચેની અંતરની ગણતરી કરો અને પછી દરેક જોડી માટે આ કર્યા પછી તમે સૌથી નાનું મૂલ્ય લો. હવે, તે તારણ આપે છે, કે ત્યાં એક હોંશિયાર એલ્ગોરિધમ છે, ફરીથી, જે સમય n લોગ n લે છે. તો, આ સંદર્ભમાં n ચોરસ અને n લોગ n વચ્ચે આ તફાવત શું છે?
(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:35)
હવે, આધુનિક પ્રદર્શન સાથેના આધુનિક પ્રકારનાં ગેમિંગ કમ્પ્યુટર પર, ધારી લેવા માટે ગેરવાજબી નથી, કે અમે કહી શકીએ કે 2500 પિક્સેલ(pixel) દ્વારા 2500 નો રિઝોલ્યુશન(resolution) હોઈ શકે છે. જો અમારી પાસે 20 ઈંચનાં મોનિટરનાં આ વાજબી કદમાંનો એક હોય, તો અમે સરળતાથી આવા પ્રકારના ઠરાવો મેળવી શકીએ છીએ. તો, સ્ક્રીન પર લગભગ 3.75 મિલિયન પોઈન્ટ હશે. હવે, આપણી પાસે આમાંના કેટલાક બિંદુઓ પર કેટલીક વસ્તુઓ મૂકવામાં આવી છે. તેથી, ચાલો ધારીએ કે અમારી પાસે પાંચ લાખ વસ્તુઓ છે, સ્ક્રીન પર પાંચસો હજાર પદાર્થો છે. તેથી, જો તમે હવે આ પાંચસો હજાર વચ્ચેની વસ્તુઓની જોડી ગણતરી કરો છો, જે એકબીજાના સૌથી નજીક છે અને અમે નમ્ર એન સ્ક્વેર્ડ અલ્ગોરિધમનો ઉપયોગ કરીએ છીએ, તો તમે 25 10 ને 10 પગલાંમાં લેવાની અપેક્ષા કરશો કારણ કે તે 5 માં છે 10 થી 5 ચોરસ સુધી. 10 થી 10 માં 10 એ 2500 સેકંડ છે, જે લગભગ 40 મિનિટ છે. હવે, જો તમે કોઈ રમત રમી રહ્યા છો, તો એક્શન રમત કે જેમાં તમારા સ્કોરને નિર્ધારિત કરે છે, દેખીતી રીતે, દરેક અપડેટમાં 40 મિનિટ લાગી શકે નહીં. તે એક અસરકારક રમત નથી. બીજી બાજુ, તમે સહેલાઈથી તપાસ કરી શકો છો, કે જો તમે 5 થી 10 ની 5 ઘાતની ગણતરી કરો છો, તો તમે 10 કે 6 થી 10 ને 7 સેકંડમાં લઈ જાઓ છો. તેથી, આ સેકંડનો ભાગ, 1-10 અથવા 1-100 મો ભાગનો હશે, જે તમારા માનવીય પ્રતિસાદ સમયથી નીચે છે. તેથી, આવશ્યકપણે, તે તરત જ હશે. તેથી, અમે એક રમતથી આગળ વધીએ છીએ, જે એક નિરાશાજનક રીતે ધીમું છે જેમાં તે ખરેખર તમારી પ્રતિક્રિયાને મનુષ્ય તરીકે ચકાસી શકે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:00)
તેથી, આપણે આ બે ઉદાહરણોમાં જોયું છે કે, એક બીજાની નજીકમાં કંઈક સમાન છે જેમ કે n લોગ n અને n સ્ક્વેર્ડ. સમસ્યાઓનું કદ, જેને આપણે એન સ્ક્વેર્ડ માટે હલ કરી શકીએ તે કદ કરતા ઘણી નાનું છે, કદાચ આપણે n લોગ n માટે હલ કરી શકીએ છીએ. તેથી, જ્યારે આપણે n ના આ ફંક્શનોને જુએ છે, ત્યારે સામાન્ય રીતે આપણે સ્થિરાંકોને અવગણશો. હવે, આ અંશે આ હકીકતને આધારે વાજબી ઠેરવવામાં આવશે, કે આપણે મૂળભૂત કામગીરીને ઠીક કરી રહ્યા નથી, પરંતુ આવશ્યકપણે સતત અવગણના કરીને આપણે કાર્યક્ષમતાના સંપૂર્ણ કાર્યને, n ના કાર્ય તરીકે, જેમ તે વધે તેમ, જમણે તેથી, તે એ છે જે આપણે એસિમ્પ્ટોટિક જટિલતા(asymptotic complexity) તરીકે ઓળખીએ છીએ, કારણ કે n એ મોટો કાર્ય કરે છે, કાર્ય કેવી રીતે વર્તે છે. તેથી, ઉદાહરણ તરીકે, ધારો કે અમારી પાસે કેટલાક ફંક્શન છે, જે મોટા ગુણાંક અને કંઈક સાથે n સ્ક્વેર્ડ જેવા થાય છે, જે 1 ની ગુણાંક સાથે n ક્યુબની જેમ જાય છે, શરૂઆતમાં તે n સ્ક્વેર્ડ જેવા દેખાશે, કહે છે કે 5000 n સ્ક્વેર્ડ n કરતાં વધુ છે ક્યુબ, પરંતુ ખૂબ ઝડપથી, 5000 ની બરાબર કહે છે. 5000 ની પાસે, બંને 5000 ક્યુબ અને 5000 કરતા વધારે હશે, ફંક્શન એન ક્યુબ 5000 એન સ્ક્વેર્ડ કરતા વધુ ઝડપથી વધશે, જમણી બાજુ. તેથી, એક બિંદુ કે જેનાથી n ક્યુબ n ચોરસ આગળ નીકળી જશે અને તે પછી તે ઝડપથી ખેંચાય છે. તેથી, આ તે છે જે આપણે સામાન્ય રીતે માપવા માંગીએ છીએ. અમે વ્યક્તિગત સ્થિરાંકોને ધ્યાનમાં લીધા વગર કાર્યોના કાર્ય તરીકે જોવાનું પસંદ કરીશું અને આપણે આ થોડું વધુ વિગતવાર જોઈશું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:30)
તેથી, આપણે માત્ર તીવ્રતાના ઓર્ડરમાં જ રસ ધરાવો છો, તેથી આપણે કયા પ્રકારનાં ફંક્શનો જેવો દેખાય છે તેના સંદર્ભમાં આપણે વિસ્તૃત રીતે કાર્યોને વર્ગીકૃત કરી શકીએ છીએ. તો, આપણે કાર્ય કરી શકીએ, જે લોગ એન માટે પ્રમાણસર છે; કાર્યો, જે એન, એન સ્ક્વેર્ડ, એન ક્યુબ, એન માટે કે પ્રમાણમાં છે; જે છે, તેથી નિશ્ચિત કે કોઈપણ n એ બહુમતિ છે. તો, આપણે આને બોલાવી શકીએ, આ ફંક્શન્સને બોલાવીશું, તમે, પોલિનોમિયલ, જમણી તરફ જોશો. તેથી, આ બહુમતિ છે. અને પછી આપણી પાસે કંઈક હોઈ શકે છે, જે 2 થી n ને 2 ના હોય તો આવશ્યક છે, આવો, બધા શક્ય સબસેટને જોવાનું બનેલું છે. તેથી, આ સામાન્ય રીતે બ્રુટ ફોર્સ(brute force) એલ્ગોરિધમ્સ છે જ્યાં આપણે દરેક

શક્યતાઓને જુએ છે અને પછી જવાબને નિશ્ચિત કરીએ છીએ, જમણે. તેથી, આપણે લોગરિધમિક પોલિનોમિયલ અને ઘાતાંકીય છે. તેથી, આ સંખ્યાઓની દ્રષ્ટિએ તમને શું લાગે છે
(તમેસમય: 15:23)).
(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:25)
તેથી, જો તમે આ ચાર્ટને જુઓ છો, તો ડાબી કૉલમ ઈનપુટ કદને 10 થી 1 થી 10 સુધી જુદી જુદી રાખે છે અને ત્યારબાદ દરેક કૉલમ વિવિધ પ્રકારનાં જટિલતા વર્તન તરીકે કાર્ય કરે છે, જે સ્થિરાંકોને અવગણે છે અને માત્ર તીવ્રતાને જુએ છે. તો, આપણી પાસે લોગ n છે અને પછી આપણી પાસે કંઈક છે, જે પોલિનોમિયલ એન, એન સ્ક્વેર, એન ક્યુબ છે. વચ્ચે આપણી પાસે n લોગ n છે જે આપણે પહેલા જોયું હતું અને પછી આપણી પાસે આ બે ઘાતાંકીય ફંક્શન છે, જે 2 ને n અને n ફેક્ટોરિઅલ, જમણી છે. તેથી, હવે, આમાં આપણે કાર્યવાહી કેવી રીતે કાર્યક્ષમતા નક્કી કરે છે તે નિર્ધારિત કરવાનો પ્રયાસ કરી રહ્યા છીએ. તેથી, હવે, જો તમે આ જુઓ, તો અમે કહ્યું કે, આપણે 1 સેકંડમાં 8 થી 8 ઓપરેશન કરી શકીએ છીએ. તેથી, કદાચ આપણે 10 સેકન્ડમાં કામ કરવા તૈયાર છીએ. તેથી, 10 થી 9 ની કામગીરી એ આપણે જે ધ્યાનમાં લઈશું તેની મર્યાદા વિશે છે. કાર્યક્ષમ 10 થી 10 ઓપરેશન 100 સેકંડનો અર્થ છે, જેનો અર્થ 2 મિનિટ અને 2 મિનિટ હોઈ શકે છે. તેથી, હવે સ્પષ્ટપણે, જો આપણે લોગરિધમિક સ્કેલમાં જોઈએ છીએ, તો કોઈ સમસ્યા નથી. 10 થી 10 સુધી, અધિકાર, આપણે લગભગ 33 પગલાંમાં બધું કરી શકીએ છીએ, જે ખૂબ જ લેશે
((સમયનોસંદર્ભ લો: 16:37)).
હવે, એક રેખીય સ્કેલ આપવામાં આવે છે, કે અમે 10 થી 9 સુધી અમારી મર્યાદા તરીકે અપેક્ષા રાખીએ છીએ, તે અમારી મર્યાદા બની જાય છે, જમણી બાજુ. તો, આપણે અહીં એક રેખા દોરી શકીએ અને કહી શકીએ કે, આ લાલ રેખા નીચે વસ્તુઓ અસ્થાયી ધોરણે ઓછી છે. n લોગ n એ n કરતા સહેજ ખરાબ છે. તેથી, આપણે કદ 10 થી 8 ની ઈનપુટ કરી શકીએ કારણ કે n લોગ n થી 10 ની 8 ઘાત 10 ની 9 ઘાતક. હવે, આ વિશાળ ખીલ છે જે આપણે જોયું છે. જો આપણે n લોગ n થી n squared તરફ જઈએ, તો સંભવનીયતા મર્યાદા ક્યાંક 10 થી 4 અને 10 ની વચ્ચે 5 અને એકસાથે, 10 થી 5 ની વચ્ચે છે, આપણે પહેલાથી 10 અને 10 ના સમય સુધી ચાલી શકીએ છીએ, જે છે આપણે જે જોઈએ છે તે ઉપરાંત અને હવે જો આપણે એન ક્યુબમાં જઈએ છીએ તો સંભવિતતા મર્યાદા વધુ ઘટશે. તેથી, આપણે 1000 ની જમણી બાજુના ઈનપુટ કદથી આગળ વધી શકતા નથી. તેથી, આપણે જોઈ શકીએ છીએ કે ત્યાં એકદમ ડ્રોપ છે અને જ્યારે આપણે કદ 10 અથવા 20 ની કેટલીક નાની બાબતોની તુલનામાં ઘાતાંકીય બાબતોમાં આવીએ છીએ, ત્યારે આપણે ખરેખર કંઈપણ કરી શકતા નથી, કંઈપણ, જે 100 પણ અશક્ય ધીમું હશે, બરાબર. તો, આપણી પાસે આ પ્રકારની, સંભવિતતાના અધિકારની તીવ્ર વલેંચણી રેખા છે. અને તેથી આપણે જોઈ શકીએ છીએ કે કાર્યક્ષમ એલ્ગોરિધમ મેળવવાનું મહત્વપૂર્ણ છે કારણ કે જો અમારી પાસે અયોગ્ય અલ્ગોરિધમનો હોય, તો પણ જો આપણે વિચારીએ કે તેઓ યોગ્ય રીતે કાર્ય કરે છે, તો અમે કોઈપણ વાજબી માપની સમસ્યાને હલ કરવામાં સમર્થ હશો નહીં. જ્યારે આપણે પીસી પર ચાલી રહેલ કમ્પ્યુટશનલ એલ્ગોરિધમ્સ જોઈ રહ્યા છીએ, ત્યારે તમે ખરેખર ઈનપુટ કદ 100, હજારો અને વધુ સાબિત થવાની અપેક્ષા રાખશો. તેથી, અમે વસ્તુઓ ટૂંકાવી રહ્યા છે. જો તમે વોટર્સ સૂચિ, વસ્તી માહિતી અથવા આનુવંશિક સામગ્રી વિશેના જીવવિજ્ઞાનના પ્રયોગમાંથી બહાર આવતા ડેટાને જોવા માંગતા હોવ તો અમે ખરેખર મોટી સંખ્યામાં ડેટાની અપેક્ષા રાખીએ છીએ, હવે આ વસ્તુઓ સામાન્ય રીતે ખૂબ મોટી માત્રામાં ડેટાની છે, તેથી તે ખરેખર મહત્વપૂર્ણ છે આ વસ્તુઓ કરવા માટે ઓછા શક્ય એલ્ગોરિધમ અન્યથા સમસ્યાને હાથમાં અસરકારક રીતે હલ કરવામાં આવશે નહીં.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)

વીક - 01
મૉડ્યૂલ - 06
લેક્ચર - 06

તેથી, અમે દલીલ કરી હતી કે અમે મૂળભૂત કામગીરીના સંદર્ભમાં ઍલ્ગોરિધમનો કાર્યક્ષમતા માપવા માંગીએ છીએ, અને અમે ચાલી રહેલ સમયની ગણતરી કરવા માંગીએ છીએ તેના ઈનપુટ કદ n ના કાર્યના સંદર્ભમાં ઍલ્ગોરિધમનો ઉપયોગ કર્યો છે અને અમે જોયું છે કે જો તમે n ચોરસથી n લોગ n થી જાઓ છો, તો તમે જે ઈનપુટ્સને અસરકારક રીતે હેન્ડલ કરી શકો છો તે કદ નાટકીય રીતે મોટો બને છે. હવે, આજે આપણે આમાંના કેટલાક વિચારોને વધુ સ્પષ્ટ રીતે બનાવવાની કોશિશ કરીશું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:27)
તો, પ્રથમ વસ્તુ ઈનપુટ કદ છે. તેથી, યાદ રાખો કે ઍલ્ગોરિધમનો ચાલી રહેલો સમય આવશ્યકપણે ઈનપુટના કદ પર આધારિત છે. તેથી, આપણે ચાલતા સમયને n ના કેટલાક ફંકશન ટી તરીકે લખવા માંગીએ છીએ. અને યાદ રાખવાની મુખ્ય વસ્તુ એ છે કે કદ ના બધા ઈનપુટ્સ એ જ ચાલી રહેલ સમય આપશે નહીં. તેથી, ખરાબ કેસના અંદાજની કલ્પના થઈ રહી છે જે તમને સમજાવવાની અને ન્યાયી કરવાની જરૂર પડશે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:54)
આ કરવા પહેલા આપણે ઈનપુટ કદની કલ્પના પર ધ્યાન આપીએ - આપેલ સમસ્યા માટે ઈનપુટ માપ કેવી રીતે નક્કી કરીશું? તેથી, ઈનપુટ કદ વધુ અથવા ઓછા સમસ્યાના વિતરણને લખવા માટે લેતી જગ્યાની સંખ્યાને રજૂ કરે છે અથવા સમસ્યાનો કુદરતી પરિમાણ બને છે. તેથી, ઉદાહરણ તરીકે, જ્યારે આપણે એરે(Array)ને સૉર્ટ કરી રહ્યાં છીએ ત્યારે ખરેખર તે મહત્વનું છે કે હલ કરવા માટે ત્યાં કેટલી વસ્તુઓ છે, તેથી અમને તેમને ખસેડવા અને તેમને ફરીથી ગોઠવવાની જરૂર છે. તેથી, કોઈ સૉર્ટિંગ સમસ્યા માટે એરે(Array)નું કદ એ ઈનપુટ કદની એક કુદરતી ધારણા છે. બીજી તરફ, જો તમે તત્વોને પુન: ગોઠવવા જેવી કંઈક કરવાની કોશિશ કરતા હતા અથવા કહેતા હતા કે અમારી પાસે કેટલીક વસ્તુઓ છે જે અમને કન્ટેનરમાં લોડ કરવાની જરૂર છે અને અમે વજન અથવા વોલ્યુમની દ્રષ્ટિએ મહત્તમ સબસેટ(subset) લોડ શોધી રહ્યા છીએ તો પછી વસ્તુઓની સંખ્યા કુદરતી ઈનપુટ પરિમાણ હશે. અમે પ્રારંભિક પ્રવચનોમાંના એકમાં હવાઈ મુસાફરીનો એક ઉદાહરણ જોયો હતો જ્યાં અમે એરલાઈન રૂટ નકશાનો ગ્રાફ બનાવ્યો હતો જ્યાં નોડ્સ જ્યાં શહેરો અને કિનારે જ્યાં ઉડાન હતી. અને અમે એવી દલીલ કરી હતી કે શહેરોની સંખ્યા અને ફ્લાઈટ્સની સંખ્યા બંનેને જે વિશ્લેષણની જરૂર છે તેના પર અસર પડશે. તેથી, આ તમામ ગ્રાફ્સની સામાન્ય મિલકત છે; જો આપણી પાસે ગ્રાફ હોય તો નોડ્સ અથવા શિરોબિંદુઓની સંખ્યા અને ધારની સંખ્યા બંને ઈનપુટ કદને નિર્ધારિત કરશે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:14)
હવે, ત્યાં સમસ્યાઓની એક અગત્યની વર્ગ છે જ્યાં આપણે ઈનપુટ કદ વિશે વાત કેવી રીતે કરીએ તે વિશે થોડું સાવચેત રહેવું જોઈએ, અને આ સંખ્યાઓને સમાવતી સમસ્યાઓ છે. ધારો કે આપણે પ્રાથમિકતા માટે એલ્ગોરિધમ લખવાનું હતું કે આપેલ નંબર મુખ્ય છે કે નહિ. હવે, આપણે ઈનપુટના કદ વિશે કેવી રીતે વિચારવું જોઈએ? દાખલા તરીકે, માની લો કે આપણે તેને 5003 કહેવા માટેના પ્રશ્નનો ઉકેલ લાવવા માટે અને પછી 50003 માટે, 50003 માટે આશરે 10 વખત 5003 નો સમય માનીએ છીએ. તેથી, આપણે સમયનો અંદાજ 10 સુધી અનુરૂપ થવાની અપેક્ષા કરીશું. તેથી, વાસ્તવમાં n ની તીવ્રતાને ધ્યાનમાં લેવી જોઈએ. ઈનપુટ કદ. હવે, તે સ્પષ્ટ છે કે જ્યારે આપણે હાથ દ્વારા અંકગણિત કરીએ છીએ, આપણે શાળામાં જે અંકગણિત કરીએ છીએ, તે આપણે તીવ્રતા દ્વારા નથી કરતા, આપણે અંકોની સંખ્યા દ્વારા જઈએ

છીએ. જ્યારે આપણે અંકગણિત કરીએ છીએ જેમ કે વહન સાથે ઉમેરવું, આપણે કૉલમ્સમાં નંબરોને જમણી બાજુ નીચે કરીએ છીએ, ત્યારે આપણે કોલમ દ્વારા સ્તંભ આગ્રુ કરીએ છીએ. તેથી, અંકોની સંખ્યા નિર્ધારિત કરે છે કે આપણે કેટલા કૉલમને ઉમેરવા જોઈએ. બાદબાકી અથવા ઉમેરણ અથવા ગુણાકાર અથવા લાંબી ડિવિઝન અને આ રીતે પણ તે જ છે. તેથી, સ્પષ્ટપણે, તે સંખ્યાબંધ અંકો છે જે મહત્વની છે. અને અંકોની સંખ્યા ખરેખર લોગ જેવું જ છે. જો તમારી પાસે આધાર 10 માં લખાયેલ સંખ્યા છે, તો જો તમારી પાસે 6 અંકનો આંકડો હોય તો તેનો લોગ 5 થશે. કંઈક. અને આપણે લોગ 6 પર જઈશું, આપણી પાસે 7 અંકોની સંખ્યા હશે અને બીજું. તેથી, અંકોની સંખ્યા લોગ પ્રત્યે સીધા પ્રમાણસર છે, તેથી આપણે ઈનપુટ કદ તરીકે સંખ્યાના લોગ વિશે વિચારી શકીએ છીએ. તેથી, આ એક ખાસ કેસ છે. તેથી, લોગ ઈન નંબર્સ સાથે અંકગણિત ફંક્શન માટે, તે સંખ્યા પોતે જ નથી જે ઈનપુટ કદ છે, પરંતુ સંખ્યાના કદને સંખ્યા લખવા માટે અધિક સાથે કેટલા અંકોમાં વ્યક્ત કરવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:00)

હવે, બીજી વસ્તુ, આપણે આનો ઉલ્લેખ કર્યો છે કે આપણે સ્થિરાંકોને અવગણવા જઈ રહ્યા છીએ. અમે આ કાર્યોને ઓર્ડર અથવા તીવ્રતાના સંદર્ભમાં જોવા જઈ રહ્યા છીએ, આ રીતે કાર્ય n, n ચોરસ, n ક્યુબ, અને બીજું ઘણું થાય છે. તેથી, આના માટે એક સમર્થન છે કારણ કે અમે મૂળભૂત કામગીરીની કલ્પનાને અવગણતા હોઈએ છીએ અથવા તેના બદલે આપણે તેના વિશે થોડું અસ્પષ્ટ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:20)

તેથી, ચાલો એક ઉદાહરણ જોઈએ. તેથી, ધારી રહ્યા છીએ કે, આપણે મૂળરૂપે અમારા મૂળભૂત કામગીરીને ચલો વચ્ચેની તુલના અથવા તુલનામાં અસાઈન કરવા માટે ધ્યાનમાં લઈશું. હવે, આપણે નક્કી કર્યું છે કે આપણે x અને y ની સમાવિષ્ટોને બેઝિક ઑપરેશન તરીકે બદલવાની 2 મૂલ્યોને સ્વેપિંગ શામેલ કરીશું. હવે, પ્રોગ્રામિંગમાં આપણે જે શીખ્યા તેમાંથી એક એ છે કે આ કરવા માટે અમારે અસ્થાયી ચલો દ્વારા જવાની જરૂર છે. તેથી, મોટાભાગની પ્રોગ્રામિંગ ભાષાઓમાં x અને y ને વિનિમય કરવા માટે તમારે સૌપ્રથમ અસ્થાયી ચલમાં x ને કહેવું પડશે પછી y થી x ની કૉપિ કરો અને પછી આ સ્ટોર y સ્થાયી ચલથી; આ 3 સોંપણીઓ લે છે. તેથી, જો ગણતરીની સરખામણીમાં અમે અમારી ભાષામાં મૂળભૂત કામગીરી તરીકે સ્વેપ લઈએ છીએ, જ્યાં આપણે ફક્ત સોંપણીઓ જોશું, ત્યારે 3 અસાઈનમેન્ટ્સને 1 મૂળભૂત કામગીરીમાં પતન કરીશું. તેથી, જો આપણે એક ઓપરેશન તરીકે સ્વેપ માટે એકાઉન્ટ કરીશું, તો આમાં 3 તફાવતોનો પરિબળ છે. તેથી, ઓર્ડરમાં 3 અને 2 ની આ પરિબળો વિશે ચિંતા કરવાથી દૂર રહેવા માટે અને તેથી, આ કરવાનું એક મહત્વપૂર્ણ રીત એ છે કે આપણે આ ગણતરી કરતી વખતે આ સ્થિરાંકોને અવગણવું જોઈએ. તેથી, તે માત્ર તીવ્રતાના ઓર્ડર્સને જોવા માટે અન્ય પ્રેરણામાં છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:30)

તેથી, ચાલો આપણે સૌથી ખરાબ કેસની આ કલ્પના પર પાછા આવીએ. તેથી, જેમ આપણે કહ્યું તેમ આપણે ખરેખર કદ n ના બધા ઈનપુટ્સને જોઈ રહ્યા છીએ; અને, આ ઈનપુટ્સમાં, જે મહત્તમ સમય લેતા અલ્ગોરિધમનો ડ્રાઈવ કરે છે. તો, ચાલો અહીં એક સરળ એલ્ગોરિધમ જોઈએ જે એક વર્ગીકૃત નઈ કરેલા એરે(Array) એમાં વેલ્યુ માટે જોઈ રહ્યું છે. તો, એક વર્ગીકૃત નઈ કરેલા એરે(Array)માં અમને કોઈ ખ્યાલ નથી કે મૂલ્ય કે જ્યાં હોઈ શકે છે. તો, એકમાત્ર વસ્તુ જે આપણે કરી શકીએ તે છે શરૂઆતમાં સ્વેપ. તેથી, અમે આ અનુક્રમણિકાને પ્રારંભ કરીને પ્રારંભ કરીએ છીએ જે 0 ની સ્થિતિમાં છે, અને પછી આપણી પાસે લૂપ છે જે કહે છે, તેથી આપણે એરે(Array) તત્વ શોધી નથી. તેથી, I, K ના થાય ત્યાં સુધી આપણે આગલા તત્વ તરફ આગળ વધીએ છીએ. તો, આ લૂપ છે, અને જ્યારે આપણે આ લૂપથી બહાર નીકળીએ છીએ ત્યારે 1 અથવા 2 શક્યતાઓ હોય છે, ક્યાં તો આપણે તે ઘટક શોધી કાઢીએ છીએ કે જે કિસ્સામાં હું 1 થી ઓછું છે, અથવા અમને આ કિસ્સામાં તત્વ મળ્યું નથી, હું n બની ગયો છું. તેથી, અમે તપાસો કે હું એન કરતા ઓછું છે કે નહીં; જો હું n કરતા ઓછું હોત તો આપણે તેને શોધી કાઢીએ, અને જો હું n કરતા મોટો હોય તો તેનો અર્થ એ નથી કે તે મળ્યું નથી. તેથી, આ એક સરળ અલ્ગોરિધમ છે. તેથી, હવે, આ અલ્ગોરિધમમાં, બોટલ ગરદન આ લૂપ છે, બરાબર. તેથી, આ પુનરાવર્તન સુધી લઈ શકે છે. હવે, જ્યારે એન પુનરાવર્તન લેશે? તે સૌથી ખરાબ કેસ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:47)

તેથી, આ ચોક્કસ અલ્ગોરિધમનો સૌથી ખરાબ કેસ એ છે કે તે છેલ્લો ઘટક છે કે નહીં તે જો અંતિમ ઘટક k છે, અથવા વધુ સામાન્ય રીતે જો એરે(Array)માં K ની કોઈ કૉપિ નથી. જો અરેમાં કોઈ K નથી, તો તે નક્કી કરવા માટે બધા તત્વોને સ્કેન કરવું પડશે કે K અસ્તિત્વમાં નથી. તેથી, આ અમારી સૌથી ખરાબ કેસ ઈનપુટ બની જાય છે. તેથી, એલ્ગોરિધમ જોવા માટે સક્ષમ થવું મહત્વપૂર્ણ છે અને વધુમાં વધુ સમય લેવા માટે તેમાં કયા ઈનપુટને ચલાવવા માટે તેને ફરીથી ગોઠવવાનો પ્રયાસ કરવો. તેથી, આ સરળ કિસ્સામાં, આ સરળ ઉદાહરણ આપણે જોઈ શકીએ છીએ કે જે કેસ આપણને સમગ્ર લૂપને ચલાવવા માટે દબાણ કરે છે તે એ K મૂલ્ય કે જે એરે(Array) એમાં નથી તે પસંદ કરીને જનરેટ કરી શકાય છે અને આ કિસ્સામાં, તેથી સૌથી ખરાબ કેસ એ એરે(Array) ના કદના પ્રમાણમાં છે. યાદ રાખવાની નિર્ણાયક બાબત એ છે કે સૌથી ખરાબ કેસ ઈનપુટ કયા છે તે નિર્ધારિત કરવા માટે, અમને એલ્ગોરિધમ સમજવાની અને તેને જોવાની જરૂર છે. સમસ્યારૂપ કાર્યને જાણ્યા વિના આપણે સૌથી ખરાબ કેસ શું છે તે ફક્ત આંખે નિર્ધારિત કરી શકતા નથી. વિવિધ એલ્ગોરિધમ્સ માટે આપણે એલ્ગોરિધમ શું કરવાનું માનવામાં આવે છે તેના આધારે, અને એલ્ગોરિધમ કેવી રીતે નિર્માણ કરે છે તેના આધારે અમને જુદા-જુદા કેસ સાથે આવવું પડશે. તેથી, સૌથી ખરાબ કેસ ઈનપુટ એલ્ગોરિધમનો પોતે જ કાર્ય છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:55)

હવે, આપણે એક અલગ માપ જોઈ શકીએ છીએ, જમણી બાજુ. તેથી, ધારો કે આપણે સૌથી ખરાબ કેસ તરફ ન જોતા, આપણે કહીએ છીએ, સરેરાશ કેસને જુઓ, જમણે. તેથી, તમામ સંભવિત ઈનપુટ્સ જુઓ અને પછી તે જોવાનો પ્રયાસ કરો કે જ્યારે દરેક ઈનપુટ્સ સમયસર સરેરાશ હોય ત્યારે કેટલો સમય લે છે. હવે, આ પ્રકારની સરેરાશ ગણતરી કરવા માટે, ગણિતમાં આપણે સમસ્યાની બધી સંભવિત ઈનપુટ્સ શું છે તેનું અનુમાન કરવાની સારી રીત હોવા જોઈએ. તેથી, જો કે આ ખૂબ આકર્ષક કલ્પના જેવું લાગે છે, ઘણી સમસ્યાઓમાં તે ખૂબ જ મુશ્કેલ છે. તેથી, અમને લાગે છે કે અમે એરલાઈન રૂટ સમસ્યા કરી રહ્યા છીએ, અમે આ બધા રસ્તાઓના નકશાઓના આ સ્થાનને કેવી રીતે ધ્યાનમાં લઈએ છીએ, અને એક સામાન્ય માર્ગ નકશો શું છે, અને બીજું. તેથી, આપણે આ બધા ઈનપુટ્સ જેટલી જ શક્ય છે તેના ઉપર સરેરાશ શું ચાલી રહ્યું છે, તેથી અમને સંભાવનાઓ જોવાની જરૂર છે. અને વિવિધ પ્રકારનાં ઈનપુટ્સની સંભવિતતાઓનો અંદાજ કાઢવો ઘણી વાર મુશ્કેલ છે. તેથી, વ્યવહારિક પરિસ્થિતિઓમાં એલ્ગોરિધમનો વર્તણૂકના દૃષ્ટિકોણથી તે વધુ સમજણ બનાવશે જો કે સરેરાશ કિસ્સામાં તે ઈનપુટ્સની જગ્યા પર કેવી રીતે વર્તે છે. વ્યવહારમાં, આ કરવું ખૂબ મુશ્કેલ છે કારણ કે આપણે સંભવિત ઈનપુટ્સની આ જગ્યાને ખરેખર પ્રમાણિત કરી શકતા નથી અને તેમને અર્થપૂર્ણ સંભાવનાઓ સાથે અસાઈન કરી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:09)

સારાંશ માટે, આપણે ખરાબ કેસ જોવું જોઈએ, તેમ છતાં તે અવાસ્તવિક હોઈ શકે છે કારણ કે સરેરાશ કેસ મુશ્કેલ છે, જો ગણતરી કરવી અશક્ય હોય. ત્યાં ખૂબ મર્યાદિત પરિસ્થિતિઓ છે જ્યાં સરેરાશ કેસ વિશ્લેષણ કરવું શક્ય છે, પરંતુ આ ખૂબ જ દુર્લભ છે. તેથી, ખરાબ કેસ વિશ્લેષણ વિશે સારી વસ્તુ એ છે કે જો આપણે સારી ઉચ્ચ બાઉન્ડ્રી કરી શકીએ, તો તેમાં, ખરાબ કિસ્સામાં પણ જો એલ્ગોરિધમ અસરકારક રીતે કાર્ય કરે તો પછી અમને એલ્ગોરિધમ વિશેની માહિતીનો એક ઉપયોગી ભાગ મળ્યો છે; આ હંમેશા સારી રીતે કાર્ય કરે છે. બીજી બાજુ, જો આપણે શોધી કાઢીએ કે આ અલ્ગોરિધમનો ખરાબ અપરાધ છે, તો અમારે થોડો આગળ જોવું પડશે, આ સૌથી ખરાબ કેસ કેટલો દુર્લભ છે, આ પ્રથામાં વારંવાર ઉદ્ભવે છે, કયા પ્રકારના ઈનપુટ સૌથી ખરાબ કેસ છે, શું તેઓ ઈનપુટ્સ છે જે આપણે સામાન્ય રીતે જોશું, ત્યાં કોઈ સરળ ધારણા છે કે જે આપણે કરી શકીએ છીએ જે આ ખરાબ કિસ્સાઓમાંથી બહાર નીકળશે. તેથી, જો કે ખરાબ કેસ વિશ્લેષણ તે કરવાનો સંપૂર્ણ રસ્તો નથી, તે કંઈક છે જે ગણિતશાસ્ત્રી વ્યવહારુ છે, તે એવું કંઈક છે કે જે આપણે ગણતરી કરવાની આશા રાખી શકીએ છીએ. તેથી, તે કરવા માટે તે એક સારું કારણ છે જેથી આપણે ખરેખર જથ્થાત્મક અંદાજ સાથે આવી શકીએ. અને બીજું, તે આપણને કેટલીક ઉપયોગી માહિતી આપે છે; ભલે તે કેટલીક પરિસ્થિતિઓમાં ન હોય, તે વાસ્તવમાં સૌથી વાસ્તવિક પરિસ્થિતિઓ હોઈ શકે નહીં જેનો આપણે વ્યવહારમાં આવવાની શક્યતા છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)

વીક - 01
મૉડ્યૂલ - 07
લેક્ચર - 07

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:02)
તેથી, આપણે કહ્યું છે કે અમે માત્ર તીવ્રતાના ક્રમમાં જ અલ્ગોરિધમ્સની કાર્યક્ષમતાને માપશે . તેથી, આપણે ઈનપુટ કદ n ની ફંક્શન ટી તરીકે ચાલી રહેલ સમય વ્યક્ત કરીશું, પરંતુ અમે સતત અવગણના કરીશું. તેથી, આપણે માત્ર જ્યાં કહ્યું છે કે n ની t એ n ચોરસ અથવા n લોગ n અથવા 2 ની n ને પ્રાયોગિક છે. તેથી, હવે, આગળનું પગલું એ એક સરખા અસરકારક રસ્તો છે જે એલ્ગોરિધમ્સમાં ચાલી રહેલ સમય છે. જો હું 1 અલ્ગોરિધમનો તીવ્રતાના ઓર્ડરને જાણું છું, અને અન્ય અલ્ગોરિધમનો તીવ્રતાના ક્રમમાં હું તેની તુલના કેવી રીતે કરી શકું?
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:34)
તેથી, આપણને જે સંકેત જોઈએ છે અથવા આપણે જે વિચારની જરૂર છે તે એ ઉપલા બાઉન્ડની છે જે મોટા O નો સંકેત આપે છે. તેથી, આપણે કહીએ છીએ કે n નો ફંક્શન g એ ઉપલા બાઉન્ડ છે. n નો બીજો ફંક્શન ટી n ના કેટલાક બિંદુ g થી વધુ છે. હવે, યાદ રાખો કે n નો g હવે એક ફંક્શન બનશે જે ક્રમાનુસાર ઓર્ડર છે. તેથી, આપણે બધા સતત પરિબળો ફેંકી દીધા છે જે આપણે n ના ભાગમાં ભૂમિકા ભજવીએ છીએ. તેથી, અમે આ સતત આપણી જાતને મંજૂરી આપીએ છીએ. તેથી, આપણે કહીએ છીએ કે તે n એ બધા નું g નથી જે n ના t પર પ્રભુત્વ ધરાવે છે, પરંતુ n ના g ના કેટલાક સ્થિરાંકો. તેથી, ત્યાં અમુક નિશ્ચિત સ્થિરાંકો છે અને કેટલાક મર્યાદાઓથી આગળ છે. તેથી, ત્યાં પ્રારંભિક ભાગ છે જ્યાં આપણે કાળજી આપતા નથી, પરંતુ આ સીમાની બહાર આપણી પાસે n એ n ની સી વખત g ની નીચે રહે છે. આ કિસ્સામાં n ની c વખત g અને n ની t માટે ઉપલા બાઉન્ડછે અને આપણે કહીએ છીએ કે n ની t n ના g ના મોટા O છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:34)
તેથી, ચાલો એક ઉદાહરણ જોઈએ. તેથી, ધારો કે આપણી પાસે n ના આ ફંક્શન ટી 100 અને પ્લસ 5 છે, તો પછી આપણે દાવો કરીએ છીએ કે તે n ચોરસનો મોટો O હવે યાદ છે કે n એ ઈનપુટ કદ હોવાનું માનવામાં આવે છે. તેથી, સમસ્યાનું ઈનપુટ કદ હંમેશાં ઓછામાં ઓછું 1 થશે, જો કોઈ શૂન્યમાં ઈનપુટ હોય તો નિરાકરણની જરૂર નથી અને ચોક્કસપણે આપણે નકારાત્મક હોઈ શકતા નથી. તેથી, આપણે હંમેશાં પરિસ્થિતિને ધ્યાનમાં રાખીએ છીએ કે, n એ 1 થી મોટા અથવા બરાબર છે. તેથી, જો આપણે હવે આપણા કાર્ય 100 એન પ્લસ 5 થી શરૂ કરીએ તો, જો તમે n કરતાં 5 કરતા વધારે પસંદ કરો તો n મોટો થશે આ મૂલ્ય કરતાં. તેથી, આપણે કહી શકીએ કે 100 એન પ્લસ 5 એ 100 એન પ્લસ n કરતાં સમાન છે. અને હવે આપણે આ 101 બરાબર પતન કરી શકીએ છીએ. તેથી, 100 એન પ્લસ 5 એ 101 થી નાના છે, જો n એ 5 કરતા મોટો છે, હવે, કારણ કે n એ ઓછામાં ઓછું 1 n ચોરસ n કરતાં મોટો છે. તો, 101 વખત n એ 100 અને 1 ચોરસ કરતા નાના હશે. તેથી, 0 ને 5 અને c થી 101 ની પસંદગી કરીને આપણે n ચોરસ સ્થાપિત કર્યું છે તે 100 n વત્તા 5 ની ઉપરની બાઉન્ડ છે. તેથી, 100 n વત્તા 5 n ચોરસ મોટો છે. હવે, આપણે થોડું ભિન્ન ગણતરીઓનો ઉપયોગ કરીને આ કરી શકીએ છીએ, આપણે કહી શકીએ કે 100 એન વત્તા 5 100 ના વત્તા 5 ના કરતા નાનું નાનું છે, n એ 1 કરતા મોટું છે કારણ કે n એ ઓછામાં ઓછું 1 છે. તેથી, 5 ગુણ્યા n એ હશે ઓછામાં ઓછું 5. તેથી, હવે, જો તમે પતન કરો છો તો આપણને 105 એન મળે છે, પરંતુ 105 નો વર્ગ 105 એન સ્ક્વેર કરતાં પણ ઓછું છે, જ્યારે પણ n એ 1 કરતા વધારે હોય છે. તો, એ જ હકીકત સ્થાપિત કરવાની નવી રીત, જ્યાં આપણે પસંદ કરેલ છે n 0 ની બરાબર 1 અને

સી બરાબર 105 છે. તેથી, 0 અને સી કે અનન્ય ગણતરી નથી તેના આધારે આપણે ગણતરી કેવી રીતે કરી શકીએ તેના આધારે આપણે અલગ 0 અને જુદા જુદા સી શોધી શકીએ છીએ. પરંતુ તે કોઈ વાંધો નથી કે આપણે તેમને કેવી રીતે પસંદ કરીએ, જ્યાં સુધી આપણે એ હકીકતને સ્થાપિત કરી શકીએ કે ચોક્કસ એન 0 થી આગળ એક સમાન સતત સી હોય છે, જેમ કે c ના g g n ના t પર પ્રભુત્વ ધરાવે છે. નોંધ લો કે આ જ ગણતરી આપણને વધારે તીવ્ર બાઉન્ડ આપી શકે છે, આ પ્રકારનું છૂટક અપર બાઉન્ડ છે જે તમે 100 ચોરસ એન કરતાં નાના છે તેવી અપેક્ષા રાખશો. પણ આપણે એમ પણ કહી શકીએ કે આ મોટી એન છે, તે શા માટે છે? કારણ કે જો તમે ફક્ત આ બિંદુએ ગણતરી કરો છો, તો આપણે આ તબક્કે આવીશું નહીં કે તમે સ્થાપિત કર્યું છે કે 100 એન વત્તા 5 એ 101 જેટલું ઓછું છે. પણ, સમાન મૂલ્યો n બરાબર 5 અને c બરાબર 101 છે. અમને કહે છે કે 100 n વત્તા 5 મોટું O. આ જ સમયે જો તમે આ પગલાને અવગણો તો આપણે કહીશું કે 100 એન વત્તા 5 એ 105 ના કરતા નાનું છે. તેથી, 0 ની બરાબર 1 અને સી બરાબર 105 માટે તમે આ 1 ને સ્થાપિત કર્યું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:15)

ચાલો આપણે બીજો એક ઉદાહરણ જોઈએ કે આપણે 100 એન સ્ક્વેર પ્લસ 20 એન પ્લસ 5 જુઓ. હવે, ફરીથી એમ ધારી રહ્યા છીએ કે n એ 1 કરતા મોટો છે, આપણે જાણીએ છીએ કે આપણે n દ્વારા ગુણાકાર કરી શકીએ છીએ અને n ને નાનું મળે છે. તેથી, 20 n એ તેને 20 એન સ્ક્વેર અધિકાર પર પ્રભુત્વ અપાશે અને 5 5 વખત n વખત n 5 n ચોરસમાં પ્રભુત્વ પામશે. તેથી, હવે મારી પાસે 100 એન ચોરસ પ્લસ 20 એન સ્ક્વેર પ્લસ 5 એન સ્ક્વેર છે, તે મારા મૂળ કાર્ય કરતાં 100 મીટર વધારે છે, તેથી હું તેને જોડું છું, મને 125 એન સ્ક્વેર મળે છે અને હવે હું ધારું છું કે તે, n બરાબર 1 કરતા મોટી છે. તેથી, n બરાબર 1 અને c બરાબર 125 ની જેમ, તે n ચોરસ 100 એન સ્ક્વેર વત્તા 20 એન વત્તા 1 પર પ્રભુત્વ ધરાવે છે. તેથી, તમે સરળતાથી તે જોઈ શકો છો, સામાન્ય રીતે જો હું એક ચોરસ વત્તા બી.એન. વત્તા સી ધરાવે છે, આ એક વત્તા, બી વત્તા, સી વખત n ચોરસ જમણે પ્રભુત્વ ધરાવે છે. તેથી, આ બધા કરતાં ઓછા એન માટે 1 કરતા ઓછા હશે. તેથી, આપણે સામાન્ય રીતે કહીએ છીએ, આની જેમ કાર્ય કરી શકીએ છીએ અને નીચલા શરતોને અવગણવી જોઈએ કારણ કે તેઓ મૂલ્ય પર ધ્યાન કેન્દ્રિત કરવામાં ઉચ્ચતમ શબ્દ પ્રભુત્વ ધરાવે છે. સૌથી વધુ exp1nt સાથે. તેથી, આ કિસ્સામાં આ વર્ગમાં n ચોરસ સૌથી મોટો શબ્દ છે, આ સમગ્ર વસ્તુ એ n ચોરસ કરતા વધારે મોટી હશે. તો, આ એક ખૂબ જ સામાન્ય શૉર્ટકટ છે જે આપણે લઈ શકીએ છીએ, તમે માત્ર સમીક્ષકોને અવગણી શકો છો, સૌથી વધુ exp1nt પસંદ કરો અને તે મોટા O જમણી પસંદ કરો

(સ્લાઈડસમયનો સંદર્ભ લો: 05:37)

હવે, આપણે પણ બતાવી શકીએ છીએ તે વસ્તુઓ સારી હોતી નથી. તો, દાખલા તરીકે, તે સ્પષ્ટ રીતે સ્પષ્ટ કરે છે કે, એન ક્યુબ હવે એન સ્ક્વેર કરતા મોટો છે, આપણે ઔપચારિક રીતે કેવી રીતે બતાવીએ છીએ કે n ક્યુબ n ચોરસનું મોટું નથી. ઠીક છે, એવું માનવામાં આવે છે કે, કેટલાક એન 0 અસ્તિત્વમાં છે, જેમ કે બધા n એ બરાબર n ની બરાબર છે, n ઘન ઘાત બરાબર c વખત n ચોરસ બરાબર કરતા નાના હોવું જોઈએ. જો આ કામ કરે છે તો આપણે n ચોરસ ઉપર જઈએ છીએ આ તે છે જે આપણી પાસે હોવી જોઈએ. હવે આપણે વિચારીએ છીએ કે, આપણે n બરાબર c પસંદ કરીએ તો આપણી પાસે ડાબા હાથની સી ક્યુબ છે, જમણી બાજુએ આપણી પાસે c ક્યુબ છે અને ચોક્કસપણે આપણી પાસે c ક્યુબ સમાન ઘન સમઘનથી ઓછા છે. જો હું સી વત્તા 1 પર જાઉં તો મારી પાસે સી વત્તા 1 નું પૂર્ણ સમઘન હશે અને આ બાજુ મારી પાસે સી ગણો સી વત્તા 1 સંપૂર્ણ સ્ક્વેર હશે અને હવે સમસ્યા છે, આ મોટી છે કારણ કે c વત્તા 1 નું સંપૂર્ણ ઘન c ગણો c વત્તા કરતાં મોટું છે 1 સંપૂર્ણ ચોરસ. તેથી, આપણે ગમે તે સી પસંદ કરીએ, જો આપણે n બરાબર c પર જઈએ તો આપણને મળશે કે અસમાનતા જે આપણે ઈચ્છીએ છીએ તે ફરતે ફરે છે. તેથી, ત્યાં કોઈ સી નથી કે જે આપણે નિશ્ચિત બિંદુથી સી એન ચોરસ કરતાં નાના સમઘનનું પસંદ કરી શકીએ અને તેથી આ મોટી નથી. તેથી, આપણું સાહજિક વિચાર કે એન ક્યુબ એન સ્ક્વેર કરતા ઝડપથી વધે છે, આ પગલું કાર્યનો ઉપયોગ કરીને ઔપચારિક રૂપે સાબિત થઈ શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:07)

હવે, અહીં મોટા O વિશે ઉપયોગી હકીકત છે, જો મારી પાસે ફંક્શન એફ 1 છે જે જી 1 નું મોટું છે અને બીજુ ફંક્શન એફ 2 છે જે જી 2 નું મોટું છે, તો પછી એફ 1 પ્લસ એફ 2 ખરેખર જી 1 અને જી 2 ની મહત્તમ દ્વારા વર્ચસ્વ ધરાવે છે. તમને લાગે છે કે તે જી 1 પ્લસ જી 2 છે, આ દેખીતી વાત છે જે ધ્યાનમાં રાખીને આવે છે, એફ 1 પ્લસ એફ 2 જી કરતાં નાના છે. વત્તા જી 2, પરંતુ વાસ્તવમાં તે મહત્તમ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:31)

આપણે આ કેવી રીતે સાબિત કરીએ? સારું, ખૂબ મુશ્કેલ નથી. વ્યાખ્યા દ્વારા જો એફ 1 જીનું મોટું છે, તો કેટલાક એન 1 અસ્તિત્વમાં છે જેમ કે એન 1 એફ 1 ની સી 1 થી 1 જી 1 સી 1 વખત જી પ્રભુત્વ ધરાવે છે. એ જ રીતે, જો એફ 2 એ 2 ની મોટા O છે, તો ત્યાં 2 છે એન 2 જેમ કે n 2 f 2 ની બહાર સી 2 વખત g 2 જમણે પ્રભુત્વ ધરાવે છે. તેથી, હવે, આપણે શું કરી શકીએ તે છે કે આપણે n 3 ને મહત્તમ n 1 અને n 2 તરીકે પસંદ કરી શકીએ અને આપણે c 3 ને મહત્તમ c 1 અને c 2 તરીકે પસંદ કરી શકીએ. તો, ચાલો હવે જોઈએ એન 3 ની બહાર થાય છે, n 3 ની બહાર આ બંને અસમાનતાઓ અસરકારક છે. તેથી, આપણી પાસે એફ 1 વત્તા એફ 2 સી 1 અને જી 1 વત્તા સી 2 ગણી જી 2 થી ઓછું હશે. કારણ કે, આ એન 1 અને એન 2 ની બહાર છે તેથી, બંને એફ 1 સી 1 અને જી 1 થી ઓછું છે અને સી -2 જી 2 કરતા ઓછું એફ 2 છે. તેથી, હું 2 ઉમેરી શકું છું અને, આ પહેલી સ્પષ્ટ વસ્તુ છે જે આપણે કહ્યું છે કે તે જી 1 વત્તા જી -2 છે, પરંતુ હવે આપણે થોડું ચપળ હોઈ શકીએ છીએ, આપણે કહી શકીએ કે આપણી પાસે સી 3 છે. તેથી, સી 1 કરતા નાના છે. સી 3 કારણ કે આ મહત્તમ સી 2 સી કરતાં નાના છે. તેથી, હું આને ભેગા કરી શકું છું અને કહી શકું છું કે આ સી 3 જી 1 વત્તા સી 3 જી કરતાં ઓછું છે. હવે, આને જોડીને હું અલબત્ત તેને દબાણ કરી શકું છું સાથે મળીને કહેવું કે આ સી 3 ગણી જી 1 પ્લસ જી 2 ઓછું છે, પરંતુ જી 1 વત્તા જી 2 જો હું મહત્તમ તેટલું વધારે લઉં તો તેના કરતાં 2 ગણી મહત્તમ. તેથી, મને સી મહત્તમ કરતા 3 ગણો 2 ગણી મળશેજી 1 અને જી 2 અધિકાર. હું આ 2 લઈ શકું છું અને તેથી કહું છું કે, તે 2 C 3 ની બરાબર અને મહત્તમ G 1 અને G 2 જમાણા છે. તેથી, હવે, જો હું આને મારા n તરીકે અને આને મારા c તરીકે લેતો હોત તો મેં નક્કી કર્યું છે કે n અને મહત્તમ n 1 અને n 2 કરતા મોટા દરેક n માટે એક સતત છે જે c1 c ની મહત્તમ 2 ગણી છે. 2 જેમ કે એફ 1 પ્લસ એફ 2 એ જી 1 જી 2 ની સી વખત મહત્તમ દ્વારા પ્રભુત્વ ધરાવે છે 2. કેમ આ ગાણિતિક તથ્ય આપણા માટે ઉપયોગી છે?

(સ્લાઈડસમયનો સંદર્ભ લો: 09:51)

તેથી, ઘણીવાર જ્યારે આપણે ઍલ્ગોરિધમનો વિશ્લેષણ કરી રહ્યા છીએ, ત્યારે તેમાં વિવિધ તબક્કા હશે. તે એક ભાગમાં કંઈક કરશે પછી તે બીજી વસ્તુ ચાલુ રાખશે અને ચાલુ રહેશે. તેથી, આપણી પાસે 2 તબક્કાઓ હોઈ શકે છે, તબક્કો A જે સમયાંતરે O(Ga) અને તબક્કો b લે છે જે O(Gb) લે છે. તેથી, હવે, એલ્ગોરિધમનો એકંદરે ચાલતા સમય માટે સારી ઉપલા બાઉન્ડ શું છે. તેથી, સહજ વસ્તુ Ga પ્લસ Gb છે. પરંતુ આ પરિણામ આપણને શું કહે છે તે એ છે કે તે જી પ્લસ જીબી નથી જે ઉપલા બાઉન્ડ માટે ઉપયોગી છે, પરંતુ મહત્તમ ગા અને જીબી અધિકાર. બીજા શબ્દોમાં, જ્યારે આપણે ઍલ્ગોરિધમનો વિશ્લેષણ કરીએ છીએ ત્યારે બોટલ ગરદન જોવા માટે પૂરતું છે. તમે ઘણા પગલાઓ પર જાઓ છો જે પગલાને મહત્તમ સમય લે છે, તેના પર ધ્યાન કેન્દ્રિત કરો અને તે અન્યના એકંદરે ચાલતા સમયને નિર્ધારિત કરશે. તેથી, જ્યારે આપણે ઍલ્ગોરિધમ પર ફંક્શનને જોઈએ છીએ જેમાં લૂપ હોય છે, આપણે સામાન્ય રીતે લૂપને જોઈએ છીએ કે લૂપ કેવી રીતે જાય છે. અમે અવગણીએ છીએ, લુપ અથવા કેટલીક પ્રિંટ સ્ટેટમેન્ટ પહેલાં લૂપ પહેલા થાય છે તે પ્રારંભિક હોઈ શકે છે કારણ કે લૂપ પોતે જ ઠીક છે તેટલી જટીલતા જેટલી ફાળો આપે છે. તેથી, જ્યારે આપણી પાસે બહુવિધ તબક્કાઓ હોય, ત્યારે તે સૌથી બિનકાર્યક્ષમ તબક્કો છે જે સમગ્ર વર્તન પર પ્રભુત્વ ધરાવે છે અને તે આપણે જે પરિણામ જોયું તેના આધારે ઔપચારિક બને છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:01)

હવે, નીચલા બાઉન્ડ એટલે કે નીચલા બાઉન્ડ પર એક સમપ્રમાણિક કલ્પના છે. તેથી, જેમ આપણે કહ્યું કે n ની t હંમેશા n ની cfc વખત g ની નીચે રહે છે. આપણે કહી શકીએ કે n ના t હંમેશા n ના c વખત g ઉપર રહે છે અને આ નોંધ ઓમેગા(omega) નો ઉપયોગ કરીને વર્ણવવામાં આવે છે. તેથી, આ માત્ર એક સમપ્રમાણ વ્યાખ્યા છે જે ફક્ત કહે છે કે n ના t, g ના n નો ઓમેગા(omega) છે, જો n ની n ની 0 n ની બહાર દરેક n એ અમુક ચોક્કસ મતવિસ્તાર માટે c ના g

ના n ઉપર છે. તેથી, અહીં આપણી પાસે તે જ વસ્તુ છે જેની પાસે અમારી પ્રારંભિક વસ્તુ છે કે જેમાં અમને રુચિ નથી, કારણ કે આ બિંદુએ કશું પણ કહી શકાતું નથી. પરંતુ આ n 0 ની બહાર આપણી પાસે n ની ઉપર n આવેલ છે તેથી, n ની t હંમેશા n ના c વખત g ઉપર છે

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:50)

તેથી, આપણે પહેલા જોયું કે n ઘન બરાબર n ચોરસનું નથી. , પરંતુ એન ક્યુબ એન ચોરસ ઉપર આવેલા હોવા જોઈએ અને આ ચોક્કસપણે આ કેસ છે કારણ કે, એન ઘન બરાબર 1 બરાબર કરતાં મોટા દરેક n માટે સમાન ચોરસ કરતાં વધુ છે. તેથી, n બરાબર 1 ની બંને બરાબર 1 છે, પણ n બરાબર 2 આ 8 હશે, આ 4 હશે અને તેથી, ચાલુ છે. તેથી, જો 0 ને સમાન 0 અથવા n બરાબર 1 અને c બરાબર 1 આપવામાં આવે તો આપણે તેને સ્થાપિત કરી શકીએ છીએ. હવે અલબત્ત, જ્યારે આપણે ઉપરની બાઉન્ડ બનાવી રહ્યા છીએ ત્યારે અમે અમારી પાસે ઍલ્ગોરિધમ વિશે વાત કરી રહ્યા છીએ. તમે એમ કહો છો કે આ અલ્ગોરિધમનો આટલો બધો બાઉન્ડ છે, તેથી ખૂબ જ અને તેથી, હું આ સમયમાં ઘણીવાર સમસ્યાને હલ કરી શકું છું. હવે, જ્યારે આપણે નીચલા સીમા વિશે વાત કરીએ છીએ ત્યારે તે ચોક્કસ અલ્ગોરિધમનો વિશે વાત કરવી ઉપયોગી નથી. તે એટલું જ નહીં, કહેવું ઉપયોગી છે કે આ અલ્ગોરિધમનો ઓછામાં ઓછો સમય લાગે છે. અમે આ સમસ્યા જેવી કંઈક કહેવા માગીએ છીએ, ઓછામાં ઓછું, ખૂબ સમય, ભલે તમે એલ્ગોરિધમ કેવી રીતે લખો તે ભલે તે ઓછામાં ઓછું, વધુ સમય લેશે. તેથી, સામાન્ય રીતે અમે ઉપયોગી નિમ્ન બંધન નિવેદન બનાવવા માટે શું કરવું ગમશે તે કહેવું છે કે કોઈ સમસ્યા ચોક્કસ સમય લે છે, પછી ભલે તમે તેને કેવી રીતે હલ કરવાનો પ્રયાસ કરો છો. તેથી, એલ્ગોરિધમનો ઓછો બાઉન્ડ હોવાને બદલે સમસ્યા ઓછી બાઉન્ડ છે. હવે, તમે કલ્પના કરી શકો છો કે આ એકદમ જટિલ છે કારણ કે તમે બતાવવા માટે સક્ષમ છો કે તમે કેટલી હોંશિયાર છો, ભલે તમે કોઈ એલ્ગોરિધમ કેવી રીતે ડિઝાઈન કરો, તમે કોઈ કરતા વધુ સારું ન કરી શકો

(સમય 13:13 નો સંદર્ભ લો)

.મારી પાસે આ કરવાનું એક વિશિષ્ટ રીત છે તે કરતાં આ ઘણું મુશ્કેલ છે અને હું તે કેવી રીતે કરવું તેનું વિશ્લેષણ કરું છું. તેથી, નીચલા બાઉન્ડ્સની સ્થાપના ઘણી વખત મુશ્કેલ છે. તે વિસ્તારોમાંનું એક જ્યાં નીચલા બાઉન્ડની સ્થાપના કરવામાં આવી છે તે સૉર્ટ છે. તેથી, તે બતાવી શકાય છે કે, જો તમે મૂલ્યોને મૂલ્ય આપવા માટે મૂલ્યોની તુલના કરી રહ્યાં છો, તો તમારે ઓછામાં ઓછા એન લોગ n તુલનાઓ કરવી જોઈએ, ભલે તમે ખરેખર સૉર્ટિંગ કેવી રીતે કરો છો. તમે કોઈ પણ પ્રકારની હોશિયાર એલ્ગોરિધમને સૉર્ટ કરી રહ્યાં છો, તે ઘટકોની સરખામણીમાં તે n લોગ n કરતા વધુ ઝડપી હોઈ શકતું નથી, પરંતુ યાદ રાખવું મુશ્કેલ છે, કારણ કે તમે ખરેખર એલ્ગોરિધમનો આ સ્વતંત્ર દર્શાવો છો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:51)

હવે, આપણે એક સરસ પરિસ્થિતિ મેળવી શકીએ છીએ જ્યાં આપણે ઉચ્ચ અને નીચલા સીમા સાથે મેળ ખાતા હોઈએ છીએ. તેથી, આપણે કહીએ છીએ કે t એ g ના n નો g of ta છે, તો આપણે g ના n અને ઓમેગા(omega) g ના g જઈએ. બીજા શબ્દોમાં કહીએ તો, n ની યોગ્ય સ્થિરાંકો ટી n ના g દ્વારા પ્રભુત્વ પ્રાપ્ત કરી શકાય છે, અને તે કોર્સના બે જુદા જુદા સ્થિરાંકો માટે n ની ઉપર પણ છે. તેથી, આનો ખરેખર અર્થ શું છે તે છે કે, n અને n ના g ની મૂળભૂત રીતે તીવ્રતાના સમાન ક્રમ છે, તેથી તે આવશ્યક રૂપે સમાન કાર્ય છે, તમે એક શ્રેષ્ઠતમ મૂલ્ય સુધી પહોંચી ગયા છો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:26)

તેથી, ઉદાહરણ તરીકે આપણે કહી શકીએ કે તે n થી ઓછા 1 ભાગ્યા 2 એ s વર્ગ ની થીટા છે. આના જેવું કંઈક સાબિત કરવા માટે, આપણે બતાવવું પડશે કે ત્યાં ઉચ્ચ બાઉન્ડ છે, તે આપણે એક સ્થિરાંકો શોધી શકીએ છીએ

(સમયનોસંદર્ભ લો: 14:37)

આ અને નીચલા બાઉન્ડ પર પ્રભુત્વ ધરાવે છે. ત્યાં એક અન્ય સ્થિર છે

(સમયનોસંદર્ભ લો: 14:41)

આ નીચે છે. તેથી, ઉપલા બાઉન્ડ માટે આપણે ફક્ત n ને n ઓછા 1 થી 2 વડે વિસ્તૃત કરીએ. તેથી, આપણને n એ 2 વડે n નો વર્ગ મળશે અને n માર્ક્સ n દ્વારા 2 મળશે. હવે, કારણ કે તે ઉપરની બાહ્ય n એ 2 મા ઓછા n વર્ગ છે. 2 દ્વારા, જો હું 2 વડે n અવગણું છું, તો તે 2 ની વડે n વર્ગ થી ઓછું થશે. તેથી, હવે મારી પાસે એક ઉચ્ચ બાઉન્ડ છે જે, સતત

અડધો છે, n એ 0 થી મોટા n માટે n વર્ગ છે. બીજી તરફ, જો હું નિમ્ન બાઉન્ડ કરવા માંગું છું, તો હું તે જ કહું છું કે હું n થી n માં 2 સુધી વિસ્તૃત થઈશ, મને સમાન વિસ્તરણ મળશે. અને હવે હું બંધ કરવા માંગુ છું, તેથી હવે હું શું કરીશ, હું આને પણ નાનું કરીશ. હું કહું છું કે હું 2 દ્વારા n ને બાદ કરું છું પરંતુ 2 વખત n દ્વારા 2 ગુણ્યા 2 ને બાદ કરું છું. તેથી, આ આનાથી મોટું હશે, કારણ કે હું વધુ બાદ કરી રહ્યો છું. N ની 2 ઘાત વત્તા 2 વડે 2 ની વત્તા 2 ની વત્તા 2 ની વડે મોટું થશે, પણ આ ફરીથી n વડે ચોરસ થાય છે. તેથી, મારી પાસે n વત્તા 2 વડે n વર્ગ ભાગ્યા વર્ગ, તેથી n વર્ગ બીજા શબ્દોમાં કહીએ તો, મેં બતાવ્યું છે કે n માં n 1 ની 1 ઘાત બરાબર n ચોરસ 4 કરતા મોટું છે. પરંતુ હવે, આને યોગ્ય ઠરાવવા માટે, n ને 2 વડે વધારો થાય તે ન્યાયી બનાવવા માટે, n એ ઓછામાં ઓછું હોવું આવશ્યક છે 2. કારણ કે જો 2 કરતા નાના n એ આ અપૂર્ણાંક છે તેથી હું ખરેખર ઘટાડી રહ્યો છું. અહીં, મારી પાસે 2 ની બરાબર 2 કરતા વધારે મોટી છે. મેં નીચલા બાઉન્ડ પરિણામને સ્થાપિત કર્યું છે કે n ની બરાબર 2 n માં n ઓછા 1 ભાગ્યા 2 એ n ચોરસની એક ચોથા ઉપર છે. તેથી, તેથી હવે જો આપણે 2 કરતા મોટા મૂલ્યો માટે સતત 2 હોવાનું પસંદ કર્યું હોય, તો મારી પાસે n એ n ઓછા 1 ભાગ્યા 2 n વર્ગ ની અડધી કરતા ઓછી છે અને n માં ઓછા 2 એ n ચોરસ કરતા ચોથા કરતા મોટો છે. . તેથી, મને આ બંધબેસતી ઉપલા અને નીચલી બાઉન્ડ મળી છે જે દર્શાવે છે કે n એ એન સ્ક્વેરના 1 ની બાદબાકી 1 ની અવતરણ છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 16:44)
તેથી, જ્યારે આપણે મોટા O નો ઉપયોગ કરીએ છીએ ત્યારે સારાંશ આપવા માટે, આપણે ઉચ્ચ બાઉન્ડ શોધ્યું છે. જો આપણે n ના f નો n ના g ના મોટા ઓ છે, તો એનો અર્થ એ છે કે n n નું પ્રભુત્વ n ના f નો n ના g કરતાં મોટો નથી. અને આ સૌથી ખરાબ કેસ ચાલી રહેલ સમયની મર્યાદાને વર્ણવવા માટે ઉપયોગી છે. તો, આપણે કહી શકીએ કે ખરાબ સમયનો સમય એ n ના g દ્વારા વધારે છે. બીજી તરફ, જો આપણે ઓમેગા(omega)નો ઉપયોગ કરીએ છીએ તો આપણે કહીએ છીએ કે n નો f n ના ઓછામાં ઓછા g છે, n ના g ની નીચલી બાઉન્ડ છે
(સમયનોસંદર્ભ લો: 17:14).
જેમ આપણે વર્ણન કર્યું છે કે આ સંપૂર્ણ સમસ્યાઓ માટે વધુ ઉપયોગી છે, વ્યક્તિગત અલ્ગોરિધમનો બદલે સામાન્ય સમસ્યા તરીકે સૉર્ટ કરવું. કારણ કે તે અમને જણાવે છે કે તમે કઈ રીતે કંઈક કરો છો, તમારે ઓછામાં ઓછું તેટલો સમય પસાર કરવો પડશે, પરંતુ આ અધિષ્ઠાપિત કરવું મુશ્કેલ છે. અને જો તમારી પાસે કોઈ સમસ્યા હોય કે જ્યાં કોઈ સમસ્યા માટે નિમ્ન બાઉન્ડની સ્થાપના કરવામાં આવી હોય અને તમને ઍલ્ગોરિધમ મળે છે જે ઉચ્ચ બાઉન્ડ્રી જેટલું જ બંધાયેલું હોય, તો તમે અમુક અર્થમાં શ્રેષ્ઠ સંભવિત અલ્ગોરિધમનો અનુભવ કર્યો છે. કારણ કે, તમે n ની તુલનામાં વધુ સારું કરી શકતા નથી કારણ કે અમારી પાસે n ના g ની નીચી બાઉન્ડ છે અને તમે n ના g પ્રાપ્ત કરી છે કારણ કે તમે n એ g ના n ના મોટા O તરીકે તમારી એલ્ગોરિધમ બતાવી છે. તેથી, થીટા એ દર્શાવવાનો એક રસ્તો છે કે તમને ઍલ્ગોરિધમ મળ્યું છે જે શક્ય તેટલી અસરકારક છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)

વીક - 01
મૉડ્યૂલ - 08
લેક્ચર - 08

આ એકમના છેલ્લા ભાષણ, આપણે શું કરવા જઈ રહ્યા છીએ તે વાસ્તવમાં એલ્ગોરિધમ્સના કેટલાક ઉદાહરણો જુઓ અને જુઓ કે કેવી રીતે તેમના ઉપલા ક્રમાંકની ગણતરી કરવી

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:12)

તેથી, આ લેક્ચરમાં, આપણે આ એકમમાં બે મૂળભૂત વર્ગના એલ્ગોરિધમ્સ જોશું. તેથી, આપણે કેટલાક પુનરાવર્તિત ઉદાહરણો અને કેટલાક પુનરાવર્તિત ઉદાહરણો જોશું. તેથી, પુનરાવર્તિત ઉદાહરણ મૂળરૂપે કંઈક હશે જ્યાં લૂપ અને અલબત્ત પુનરાવર્તિત પ્રોગ્રામ હશે, તે મોટી સમસ્યાને ઉકેલવા પહેલાં તમારે નાની સમસ્યાને ઉકેલવી પડશે. તેથી, તમારે નાના ઈનપુટ સાથે સમાન એલ્ગોરિધમનો ફરીથી ઉપયોગ કરવો પડશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:34)

તેથી, અમારું પહેલું ઉદાહરણ એક ખૂબ માનક સમસ્યા છે જે તમારે તમારા મૂળભૂત પ્રોગ્રામિંગ કોર્સમાં કર્યું હોવું જોઈએ. ધારો કે, આપણે એરે(Array) માં મહત્તમ તત્વ શોધવા માંગીએ છીએ. તેથી, આપણે શું કરીએ છીએ તે છે કે આપણે પ્રારંભમાં એવું માનીએ છીએ કે મહત્તમ મૂલ્ય એ પ્રથમ મૂલ્ય છે અને પછી અમે બાકીના એરે(Array)ને સ્કેન કરીએ છીએ અને જ્યાં પણ આપણે મૂલ્યને જોઈએ છીએ તે વર્તમાન મહત્તમ મૂલ્ય કરતાં મોટું છે, મહત્તમ મૂલ્ય તેને બદલશે. અને આ સ્કેનના અંતે આપણે મહત્તમ મૂલ્યનું મૂલ્ય પાછું મેળવીશું જે આપણે શોધી કાઢ્યું છે જે સૌથી મોટું મૂલ્ય હોવું જોઈએ જે અમે સંપૂર્ણ મૂલ્ય બતાવ્યું છે. હવે, યાદ રાખો કે જો આપણે આ કિસ્સામાં બે તબક્કાઓ ધરાવીએ છીએ, તો આપણી પાસે એક તબક્કો છે જ્યાં આપણે પ્રારંભિકરણ કરીએ છીએ, આપણી પાસે ત્રણ તબક્કાઓ છે, વાસ્તવમાં આપણે આંતરિક લૂપ કરીએ છીએ અને પછી આપણે વળતર કરીએ છીએ, તે જોવા માટે પૂરતું છે. સાંકડા તબક્કામાં, અમે કહ્યું કે જો આપણી પાસે બે ભાગ એફ્ 1 અને એફ્ 2 હોય તો એફ્ 1 પ્લસ એફ્ 2 ની તીવ્રતાના ક્રમમાં એફ્ 1 અને એફ્ 2 ની તીવ્રતાના ક્રમમાં મહત્તમ છે, તેથી આ સ્થિતિમાં તે છે સ્પષ્ટ કરો કે આ લૂપ એ છે કે આપણે સૌથી વધારે સમય લેવો જોઈએ. તેથી, આ લૂપની જટિલતાનું વિશ્લેષણ કરવા માટે તે પૂરતું છે. તેથી, હવે, આ લૂપ બરાબર n ઓછા 1 પગલાં લે છે. તેથી, સૌથી ખરાબ કેસ, કોઈપણ ઈનપુટ સૌથી ખરાબ કેસ છે, કારણ કે આપણે મહત્તમ મૂલ્ય શોધવા માટે ક્રમમાં શરૂઆતથી અંત સુધી જવું જોઈએ, મહત્તમ મૂલ્ય ક્યાં છે તેના વિશે અમે કોઈ પણ અનુમાન કરી શકતા નથી. હવે, જ્યારે આપણે દરેક પુનરાવૃત્તિમાં લૂપ સ્કેન કરી રહ્યા છીએ ત્યારે આપણે ઓછામાં ઓછા એક પગલું કરીશું. તેથી, આ સરખામણી, એક મૂળભૂત કામગીરી છે અને આ થઈ શકે છે અથવા થઈ શકે છે. તેથી, અસાઈનમેન્ટ થાય છે, જો આપણને નવી વેલ્યુ A મળે છે જે મેક્સ વેલ(maxval) કરતા મોટો છે. પરંતુ, આપણે સ્થિરાંકોને અવગણતા હોઈએ છીએ તેથી આપણે તેને કેટલાક સી ઓપરેશન્સ તરીકે સારવાર આપી શકીએ છીએ, પુનરાવર્તન દીઠ કેટલાક સતત સંખ્યામાં ઓપરેશન્સ. તેથી, આપણી પાસે કેટલાક સી વખત n ઓછા 1 મૂળભૂત કામગીરી છે અને જો આપણે સી અવગણીએ છીએ અને આપણે આ અવકાશી અવગણના કરીએ છીએ, આ એકંદરે આ અલ્ગોરિધમ રેખીય છે, તે ઓર્ડર n સમય લે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:36)

તેથી, ચાલો હવે એક ઉદાહરણ તરફ આગળ વધીએ કે જેમાં આપણી પાસે બે નેસ્ટ થયેલ લૂપ્સ છે. તેથી, ધારી રહ્યા છીએ કે આપણે શોધી કાઢવાનો પ્રયાસ કરી રહ્યા છીએ કે કોઈ એરે(Array)માં બધા વિશિષ્ટ મૂલ્યો છે કે જે એરે(Array)માં કોઈ બે મૂલ્યો નથી એ એ જ છે. તો, આપણે શું કરીશું, તમે આ એરે(Array) A ને લેશો અને પછી આપણે દરેક A અને A એ j ની તુલના કરીશું અને જો મને ક્યારેય A જે સમાન મળશે તો હું ખોટું પાછી આપીશ. જો મને આવા કોઈ I અને A j

મળ્યા નથી, તો પછી હું ખોટું નહીં પાડીશ, હું સાચું વળું છું. હવે, જો હું પોઝિશન પર હોઉં તો આને ઑપ્ટિમાઈઝ કરવા માટેનો મુદ્દો છે, પછી હું ફક્ત તેના ઘટકોને જ જોઈ શકું છું. તેથી, હું મારી વત્તા 1 થી શરૂ કરીશ અને n ઓછા 1 પર જઈશ અને આ j માટે મારી રેન્જ હશે. તેથી, A, A j અને તે પછી j એ સરખામણી કરવા માટે, આપણે ફક્ત આ ડુપ્લિકેટ વસ્તુને ટાળવા માટે જે લખ્યું છે તે ટાળવા માટે હું 0 થી n ઓછા 1 બરાબર છે. તેથી, હું જે માટે દરેક તત્વને જોઉં છું. હું પ્લસ 1 થી n ઓછા 1 ની બરાબર છે, તે બરાબર છે, એ તપાસો કે હું એ સમાન એ જ છે. તેથી, હવે જો હું ખરેખર આ સંખ્યા એક્ઝેક્યુટ કરું છું, તો જો હું 0 ની બરાબર હોઉં તો, j એ 1 થી n માઈનસ 1 થી બદલાય છે. તેથી, n ઓછા 1 પગથિયાં છે, જ્યારે હું 1 ની બરાબર છે. ઓછા 2 પગલાંઓ અને તેથી, ચાલુ. તો, જ્યારે હું નીચે જાઉં છું ત્યારે હું 1 ની બરાબર 1 ની બરાબર છે, ત્યાં એક પગલું હશે, જ્યારે હું 2 ની બરાબર હોત ત્યાં એક પગલું હશે. જ્યારે હું n બરાબર 1 ની બાદબાકી કરીશ ત્યારે બાહ્ય લૂપ સમાપ્ત થશે, પણ આંતરિક ભાગ્યે જ ચાલશે, કારણ કે તમે n થી n માઈનસ 1 માં જઈ શકો છો. તેથી, આપણે જે કરીએ છીએ તે એકંદરે આપણે 0 વત્તા 1 વત્તા 2 કરી રહ્યા છીએ. , પ્લસ એન માઈનસ 2, પ્લસ એન બાદ 1 પગલાં. તો, આ એક પરિચિત સારાંશ છે, હું આ સારાંશ 1 થી 1 ની બાદબાકી 1 ની બરાબર છે અને તમારે જાણવું જોઈએ કે n ઓછા 1 ને n માં છે, આ એક ખૂબ પરિચિત પુનરાવર્તન છે અને આ આપણે વાસ્તવમાં જોયું છે, આ ઓ એન ચોરસ છે. તેથી, આપણે માત્ર 2 ઓછા એન સ્ક્વેરને 2 ઓછા એન 2 વર્ગ દ્વારા અવગણીએ છીએ, વાસ્તવમાં આપણે આ થીટા એન સ્ક્વેર પર દર્શાવ્યા છે, પરંતુ આ ક્ષણે આપણે ફક્ત ઉચ્ચ સીમા જોઈ રહ્યા છીએ. તો, ચાલો આપણે કહીએ કે આ દલીલો O n ચોરસ છે. તેથી, તે નજીવી નથી, અર્થમાં ઓ n ચોરસ એ સમાન કદના બે નેસ્ટેડ લૂપ્સ નથી, તે 0 થી n બરાબર નથી, જે બરાબર 0 થી n બરાબર છે, હું 0 થી n ઓછા 1 બરાબર છે અને j બરાબર હું વત્તા 1 થી n ઓછા 1. પરંતુ, હજી પણ આ સારાંશ 1, 2, 3, 4 ની ઉપર n એ ચોરસ પર છે અને આ તે છે જે આપણે વારંવાર જોશું. તેથી, આ યાદ રાખવામાં ઉપયોગી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:10)

તો, આ નેસ્ટેડ આંટીઓનું બીજું ઉદાહરણ છે અને આ તે છે જેને આપણે 3 નેસ્ટેડ લૂપ્સ પસંદ કરીએ છીએ. હવે, અહીં આપણે જે કરવાનો પ્રયાસ કરી રહ્યા છે તે છે, આપણે 2 ચોરસ મેટ્રિક્સ એ અને બીને ગુણાકાર કરવાનો પ્રયાસ કરી રહ્યા છીએ તેથી, આપણી પાસે બે મેટ્રિક્સ A અને B છે અને આપણે product C. નું ગણતરી કરવાની કોશિશ કરી રહ્યા છીએ, આ પ્રોડક્ટ C માં, જો હું ઈ, જે એન્ટ્રી જોઈએ છે, તો હું જે કરું છું તે હું પ્રથમ મેટ્રિક્સમાં પંક્તિને જોઉં છું, કૉલમ જે બીજા મેટ્રિક્સમાં છે અને પછી મારે જોડવું પડશે, મને અહીં પ્રથમ એન્ટ્રી કરવી પડશે, આ માર્ગ હું કરીશ પ્રથમ એન્ટ્રી કે જે કોલમ હું તે બંનેને ગુણાકાર કરીશ, પછી મને બીજી એન્ટ્રી ગુણાકાર કરવી પડશે. તેથી, આગળ અને પછી મારે છેલ્લી એન્ટ્રી ગુણાકાર કરવી પડશે અને પછી તે ઉમેરવું પડશે. તેથી, આ પ્રોગ્રામ શું છે તે છે. તેથી, આ માટે દરેક પંક્તિ માટે હું સમાન 0 થી n ઓછા 1, પછી દરેક સ્તંભ માટે જે બરાબર 0 થી ... તેથી, આ બધી શક્ય એન્ટ્રીઝ સી i j દ્વારા થઈ રહ્યું છે. હવે, હું કહું છું કે આ નવી એન્ટ્રી માટે હું ધારું છું કે સી આઈજે 0 છે અને ત્યારબાદ હું આ પંક્તિમાંથી કે બરાબર 0 થી n ઓછા 1 સુધી દોડું છું, હું એ આઈક તરફ જોઉં છું જે પંક્તિમાં kth ઘટક છે, બી કેજે આ સ્તંભમાં kth ઘટક, તેમને ગુણાકાર કરો અને C i j માં ઉમેરાય છે. તેથી, આ કદ n નો લૂપ બાહ્ય લૂપ છે, આ બીજો બાહ્ય લૂપ, કદ n ની આંતરિક લૂપ અને કદ n નો આંતરિક મોટા લૂપ છે અને આ ક્રમમાં n ક્યુબ છે. તેથી, આ એન ક્યુબ મૂલ્યનું એક કુદરતી ઉદાહરણ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:33)

તેથી, આપણું અંતિમ પુનરાવર્તિત ઉદાહરણ n ની દ્વિસંગી પ્રતિનિધિત્વમાં બિટ્સ(bits) ની સંખ્યા શોધવાનું છે. તેથી, આ 1 ને 2 સુધી વિભાજીત કરવા બરાબર છે, જ્યાં સુધી આપણે 1 અથવા 0 સુધી પહોંચીએ નહીં. તો, ચાલો ધારીએ કે n એ બિન-નકારાત્મક સંખ્યા છે. તેથી, n 0 અથવા 1 છે. તો, આપણે સ્વીકારો કે બિટ્સ(bits) ની સંખ્યા ઓછામાં ઓછી 1 અને તે પછી છે. તેથી, જ્યાં સુધી આપણી પાસે એક સંખ્યા છે જે 1 કરતા મોટી છે, આપણે આંકડાઓની ગણતરીની સંખ્યામાં એક વધુ ઉમેરીશું અને પછી આ પૂર્ણાંક વિભાગ માટે ટૂંકા સ્વરૂપ છે. તો, આપણે n ને n દ્વારા 2 ને બદલીશું. તો, આપણે તરત જ વિચારીએ કે આપણે 9 જેવી સંખ્યાથી શરૂઆત કરીએ, તો આપણે ગણતરી સાથે પ્રારંભ કરીશું 1 બરાબર 1, કારણ કે મેં તે જ કહ્યું છે. જ્યારે જ્યારે n એ 2 કરતા મોટો હોય, ત્યારે હું 2 વડે ભાગીશ અને 1 ઉમેરો. તેથી, હું

ગાણિતને બદલીશ, તેને 2 બનાવીશ, હવે આ 4 બનાવો, પછી હું કહીશ કે 1 થી વધારે. તેથી, હું આ 3 બનાવું અને હું આ 4 થી 2 બનાવીશ, તો હું કહું છું કે આ હજી પણ મોટું છે. 1. તેથી, હું આ 1 બનાવું, પછી હું આ 4 બનાવું. તો, હવે, મારી પાસે 4 ની બરાબર અને n બરાબર 1 છે. તો, આ લૂપ નીકળી જાય છે અને હું ગણતરી પાછું કરું છું. તેથી, તે કહે છે કે તે 4 અંકને તે પ્રિંટ નંબર 9 ની જરૂર છે જે સાચું છે, કારણ કે નંબર 9 અને બાયનરી તે 1001 છે. તેથી, હવે આ લૂપની જટિલતા શું છે? સારુ, આ કેટલી વાર ચલાવશે? સારુ, તેમાંથી નીકળવા માટે તે ઘણાં વખત અમલમાં આવશે જે મૂલ્ય 2 છે. તેથી, હું n, n 2 by 4, 4 વગેરે થી નીચે આવવા માંગું છું. તો, હું કેટલી વખત ભાગલા કરું? 2 1 સુધી પહોંચવા માટે અને આ પાછળ પાછળ જવા જેવું જ છે, હું 1 થી 2 સુધી કેટલી વખત n સુધી પહોંચવું જોઈએ. તેથી, 1 સુધી પહોંચવા માટે 2 દ્વારા વારંવાર ભાગ્યા એ સમાન છે જે 1 થી 2 વાર વારંવાર ગુણાકાર કરવા સમાન છેach n અને આ કંઈપણ નથી, લોગની વ્યાખ્યા, 2 ની કઈ શક્તિ n સુધી પહોંચે છે. તેથી, આ પુનરાવર્તિત લૂપ વાસ્તવમાં 1 દ્વારા ઘટાડો થયો નથી, દરેક સમયે છિદ્ર કરીને ઘટાડો, અમે હજી પણ તેને સ્પષ્ટ રીતે ગણતરી કરી શકીએ છીએ કારણ કે બેઝ 2 પગલાઓ પર લોગની આવશ્યકતા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:00)

તેથી, આપણે લીનિયર ટાઈમ(linear time), ક્વાડ્રાઈટીક ટાઈમ(quadratic time), ક્યુબિક ટાઈમ (cubic time), આ એન, એન સ્ક્વેર્ડ, એન ક્યુબ અને પુન: એન ટાઈમ સાથે રેખીય ઉદાહરણના પુનરાવર્તિત ઉદાહરણો જોયા છે. તેથી, હવે ચાલો આપણે એક પુનરાવર્તિત ઉદાહરણ જોઈએ, તે જોવા માટે કે જ્યારે આપણે પુનરાવર્તિત ઉકેલ ધરાવો ત્યારે આ કેવી રીતે કરવાનો પ્રયાસ કરીશું. તેથી, અમે ઔપચારિક પઝલ કરતાં વધુ ઔપચારિક અલ્ગોરિધમનો નજર રાખીએ છીએ. તેથી, આ જાણીતા ટાવર્સ ઓફ હનોઈ (Towers of Hanoi) છે. તેથી, ટાવર્સ ઓફ હનોઈ (Towers of Hanoi)માં અમારી પાસે આ ચિત્ર જોવા મળે છે, તેથી આપણી પાસે 3 લાકડાનાં ડબ્બાઓ છે જે આપણે એ, બી અને સી ક્ષણે બોલાવીશું તેથી, આપણે એ, બી અને સી લઈશું અને અમારો ધ્યેય છે. આ n ડિસ્કને A થી B. માં ખસેડો તેથી, જે વસ્તુ આપણને કરવાની મંજૂરી નથી, તે નાની ડિસ્ક પર મોટી ડિસ્ક મૂકવી છે. તેથી, જો આપણે નાની ડિસ્ક લઈએ અને તેને અહીં ખસેડો. તેથી, આપણે અહીં પહેલી ડિસ્કને ખસેડીએ, પછી આપણે બીજી ડિસ્ક લઈ જવી જોઈએ અને તેને ત્યાં ખસેડીશું, કારણ કે આપણે બીજી ડિસ્કને પ્રથમ ડિસ્કની ટોચ પર મૂકી શકતા નથી. તેથી, લક્ષ્ય અસરકારક રીતે આ કરવાનું છે. તેથી, પ્રત્યક્ષ ધ્યેય એ બધું A થી B માં ખસેડવાનું છે અને આ મધ્યવર્તી વસ્તુ છે, કારણ કે આપણે જોયું તેમ, આપણે પહેલી ડિસ્ક A થી B માં ખસેડીએ છીએ, આપણે ત્રાટક્યું છે, અમે કંઈપણ ખસેડી શકતા નથી

((સંદર્ભઆપો: 10 : 09)).

તેથી, તમારે આ કામ કરવા માટે અસ્થાયી ઑક્સિટરી પેગ લેવા માટે એક પ્રકારની ટ્રાન્ઝિટ તરીકે સીનો ઉપયોગ કરવો આવશ્યક છે. તેથી, જો તમે પહેલાં આ સમસ્યા ન જોવી હોય, તો તમે તેના વિશે થોડો સમય વિચારી શકો છો, પરંતુ આ એક શાસ્ત્રીય વ્યક્તિ છે કે જેનું પ્રમાણભૂત રીકર્સિવ(recursive) સોલ્યુશન છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:27)

અને સ્ટાન્ડર્ડ રિકર્સિવ(recursive) સોલ્યુશન એ નીચે મુજબ છે કે તમે પહેલા ધારે છે કે તમે જાણો છો કે n ઓછા 1 ડિસ્ક માટે સમસ્યા કેવી રીતે ઉકેલવી. તેથી, આ ક્ષણે તમે એન ડિસ્કમાંથી n ડિસ્ક ખસેડવા માંગો છો તેથી, આપણે શું કરીએ છીએ તે તમે પ્રથમ n ઓછા 1 ડિસ્ક ખસેડો. તેથી, તમારી પાસે ફક્ત એક જ નીચેની ડિસ્ક બાકી છે અને હવે તમારી પાસે ખાલી ખાલી છે અને તમે બીજા બધા n માઈનસ 1 ડિસ્ક પર સી ખસેડો છો. તેથી, અહીં n માઈનસ 1 disk છે. તેથી, તમે માની રહ્યા છો કે તમે મારા ટ્રાંઝિટ પીપ્સ તરીકે B નો ઉપયોગ કરીને આ કરી શકો છો. તેથી, હવે હું વસ્તુઓ A થી C માં ખસેડીશ, હવે હું જે કરું છું, પછી આ ડિસ્કને અહીં ખસેડો. તેથી, મારી પાસે હવે ડિસ્ક છે અને હવે મારી પાસે અહીં કંઈપણ નથી. તેથી, હવે, મારી પાસે બી પર એક મોટી ડિસ્ક છે તેથી, હું તેના પર કંઈપણ મૂકી શકું છું અને મારી પાસે સી પર n ઓછા 1 ડિસ્ક છે. તેથી, હું જે કરું છું તે હું અહીંથી વસ્તુઓ ખસેડવા માટે n માઈનસ 1 માટે સમાન એલ્ગોરિધમનો ઉપયોગ કરું છું. અહીં મારા ટ્રાંઝિટ ડબ્બાઓ તરીકે મારા એનો ઉપયોગ કરીને. તેથી, આ સમસ્યાને ઉકેલવા માટેનું પુનરાવર્તિત રીત, તમે એન બાદ 1 ડિસ્ક A થી C ને ખસેડો, A થી B ની સૌથી મોટી ડિસ્ક ખસેડો અને

પછી n માઈનસ 1 ડિસ્ક ખસેડો, બેક અપ સી થી બી. તેથી, આપણે જે પ્રશ્ન કરવા માંગીએ છીએ અમને પૂછો કે હું સૂચું છું, આ પ્રક્રિયામાં કેટલી વાર ડિસ્ક ખસેડવું?
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:37)
તેથી, એમ લાગે છે કે તમે એન ડિસ્કને એક પીગથી બીજા પીગ પર ટ્રાન્સફર કરવા માટે ચાલતી સંખ્યાઓની સંખ્યા સૂચવવા માટે એમ ના લખો. તેથી, આપણે જોયું છે કે n ડિસ્કને સ્થાનાંતરિત કરવા માટે, આપણે પ્રથમ એ -1 થી સી નાં n ઓછા 1 ડિસ્કને સ્થાનાંતરિત કરીએ છીએ, પછી આપણે એક ડિસ્કમાંથી એક ડિસ્કને સ્થાનાંતરિત કરીએ છીએ અને પછી n થી માઈનસ 1 disk back સી થી બી સુધી પરિવહન કરીએ છીએ. તે એન માઈનસ 1 નું એમ છે, તે એન ડિસ્ક્સ વત્તા 1 ને એમ 1 ના મી ઓછામાં 1 ડિસ્ક માટે સ્થાનાંતરિત કરે છે. તેથી, હું n ઓછા 1 વત્તા 1 ની 2 વખત એમ તરીકે સરળ બનાવી શકું છું. તેથી, એમનો n સામાન્ય રીતે 2 ગુણ્યા એમ ની 1 વત્તા 1 અને જો મારી પાસે માત્ર એક જ ડિસ્ક સ્થાનાંતરિત હોય, તો ત્યાં કોઈ સમસ્યા નથી કે અમે તેને એક પગથિયું સીધી કરી શકીએ છીએ. તેથી, 1 ની એમ જ્યાં n એ બરાબર 1 છે. તેથી, મૂડી મૂલ્યના નાના મૂલ્યોના સંદર્ભમાં એમ n ને વારંવાર વર્ણવવાની આ પ્રકારની અભિવ્યક્તિ, આને પુનરાવર્તન કહેવાય છે. તેથી, અમારી પાસે એમ એન માટે પુનરાવર્તિત અભિવ્યક્તિ છે, હવે આપણે આને ઉકેલવી પડશે. તેથી, જ્યાં આપણે હલ કરવા જઈ રહ્યા છીએ તે મોટેભાગે પુનરાવર્તિત સ્થાનાંતરણનો ઉપયોગ કરવા માટે છે, અમે આ અભિવ્યક્તિને સરળ બનાવવા માટે સમાન નિયમનો વારંવાર ઉપયોગ કરીશું, જ્યાં સુધી આપણે એમ 1 માં દરેક વસ્તુ સુધી પહોંચીએ નહીં અને પછી અમે મૂલ્યને પ્લગ કરી શકીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:00)
તેથી, અમે મૂળભૂત અભિવ્યક્તિથી પ્રારંભ કરીએ છીએ. તેથી, m ની n એ n માઈનસ 1 વત્તા 1 માં 2 વખત છે, હવે આપણે જે કરીએ છીએ તે એમ છે કે આપણે એમ n માઈનસ 1 માટે સમાન શબ્દપ્રયોગમાં એ જ અભિવ્યક્તિને બદલીએ છીએ કે n ઓછા 2. તેથી, એમ n માઈનસ 1 એ જ અભિવ્યક્તિ દ્વારા, તે છે 2 ગણી એમ એન માઈનસ 2 વત્તા 1, કારણ કે સામાન્ય રીતે કોઈપણ એમ માટે, આપણી પાસે n માઈનસ 1 વત્તા 1 ની વત્તા 2 ગુણ્યા એમ છે. તેથી, આ એક સામાન્ય અભિવ્યક્તિ છે. તો, આપણે લઈ રહ્યા છીએ ... તો, આપણે આ કરીએ અને આપણે સરળીકૃત કરીએ, આપણને બે શબ્દો મળે છે. તેથી, આપણને આમાંથી 2 ચોરસ મળે છે અને n ઓછા 2 મળે છે અને પછી આપણે 2 ગુણ્યા 1 લે છે જે આપણને આ 2 વત્તા 1 આપે છે. તેથી, આપણે તેને ફક્ત 2 ચોરસ એન બાદ 2 તરીકે ફરીથી લખી લીધું છે. હવે, જો તમે આ અભિવ્યક્તિ 2 વખત એમ n ઓછા 2 કરો છો જે 2 ગણી થાય છે એમ n ઓછા 3 વત્તા 1 અને પછી આ 2 ચોરસ વત્તા 1 છે, આ 2 સ્ક્વેર યાદ 4 છે. તેથી, મને 4 અંદર અને 2 ચોરસ વખત 2, 2 સમઘન મળે છે. તેથી, મને 2 ક્યુબ એમ ના માઈનસ 3 પ્લસ 2 સ્ક્વેર પ્લસ 2 વત્તા 1. હવે, તમે જોઈ શકો છો કે જો હું આ K વખત કરું તો મારી પાસે 2 ની K ની M ની ઓછા હશે. યાદ રાખો, દરેક જગ્યાએ મારી પાસે આ છે અને મારી પાસે આ છે, તે સમાન સંખ્યા છે અને તે 1 વત્તા 2 વત્તા 4 છે, પછીના સમયમાં 1 વત્તા 2 વત્તા 4 વત્તા 8 હશે. તો, આ ખરેખર 2 ની બાદબાકી 1 થી 2 છે. આ બીજું કશું નથી, 2 ક્યુબ ઓછા 1 તે બીજું નથી, 2 ચોરસ ઓછા 1 છે. તેથી, k પગલાઓ પછી સામાન્ય રીતે મેં આ કર્યું હતું, હવે જ્યારે હું આ n ઓછા 1 વખત કરીશ ત્યારે n માઈનસ n માઈનસ 1 એ કશું જ નથી, 1 n માઈનસ n plus 1. તેથી, જો હું આ 2 n માઈનસ 1 time kn માઈનસ 1 કરું તો n માઈનસ k 1 થાય અને તે element n માઈનસ 1 થાય. તેથી, n માઈનસ 1 1 is 1 થી હું આને બાદ કરી શકું છું. તેથી, મારી પાસે 2 થી n ઓછા 1 વત્તા 2 ની બાદબાકી 1 ની 1 ઘાત છે. પણ આ બીજું કંઈ નથી, પણ 2 ગુણ્યા 2 ની ઘાત 1 છે જે 2 ની છે. તેથી, હું આને 2 થી એન સાથે જોડી શકું છું. તેથી, તેથી, આ પુનરાવર્તિત વિસ્તરણ દ્વારા, તમે જે પણ તેને કૉલ કરવા માંગો છો તેને સ્થાનાંતરિત કરો. આપણી પાસે એમ છે કે એમ ના માઈનસ એમ ના બીજા 2 માં n માઈનસ 1 છે, તે આ પઝલને ઉકેલવા માટે ઘાતાંકીય સંખ્યાઓ લે છે. તેથી, લેખન ઘડિયાળ દ્વારા એક ખૂબ પ્રસિદ્ધ વાર્તા છે, જે આ મંદિરના કેટલાક શબ્દોની વાત કરે છે અને તેમાં 64 ડિસ્ક છે અને તે કહે છે કે જ્યારે 64 ડિસ્ક સ્થાનાંતરિત થાય ત્યારે વિશ્વનો અંત આવશે. તેથી, તમે 64 ડિસ્ક સાથે પઝલ હલ કરવા માટે, 64 ડિસ્કમાં 2 ને સ્થાનાંતરિત કરવા કેટલો સમય લેશે તે વિશે વાત કરી શકો છો. યાદ રાખો, આપણે કહ્યું કે 2 થી 30 એ 1 અબજ છે. તેથી, આ અતિશય સમય લાગે છે કે મને લાગે છે કે તમારે ખરેખર આ વિશે ચિંતા કરવાની જરૂર છે કેમ કે જો તે ગંભીર સમસ્યા છે, તો તે કેસ વાંચવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 16:00)
તેથી, સારાંશ માટે આપણે કેટલાક દાખલાઓ જોયા છે કે જે આપણે વાસ્તવિક એલ્ગોરિધમમાં પહેલા જે રીતે અભ્યાસ કર્યો છે તેના આધારે આપણે કેવી રીતે વિભાવનાઓને લાગુ કરીએ છીએ તેના સ્વાદને સમજાવવા માટે, તમે વાસ્તવમાં એલ્ગોરિધમ કેવી રીતે જુઓ છો કાઢવું તે જટિલતા છે. તેથી, પુનરાવર્તિત પ્રોગ્રામ માટે મૂળભૂત રીતે આંટીઓ પર ધ્યાન કેન્દ્રિત કરવું, કારણ કે આંટીઓ હું જે સમય લે છે અને તમને સમજવાનો પ્રયત્ન કરવા માટે કેટલીકવાર અપડેટ કરવા માટે કેટલોક સમય લાગે છે, કેટલી વાર લૂપ્સ એક્ઝેક્યુટ થાય છે, જ્યાં આપણે એક ઉદાહરણ પર ફરીથી યાદ કરાયેલા પ્રોગ્રામો બતાવીએ છીએ. , તમે વધુ સાથે જોશો અમે સાથે જઈશું. પરંતુ, મુખ્ય વિચાર એ છે કે તમે પ્રોગ્રામ માટે સમયની જટિલતાને પુનરાવર્તન તરીકે વ્યક્ત કરો છો, તમે એમ લખી શકો છો કે તમે m પગલાં લેવા માટે સમય લેવો છો, જે રિકર્સિવ(Recursive) કૉલથી પ્રાપ્ત થાય છે તે નાના મૂલ્યના સંદર્ભમાં. તેથી, હનોઈ કેસ માટે, આપણે n અને n ઓછા 1 હતા. તેથી, n ડિસ્ક માટે સમસ્યાને ઉકેલવા માટે, તેને n ઓછા 1 ડિસ્ક્સ માટે સમસ્યાને ઉકેલવા માટે આપણે બે વાર ઉકેલવાની જરૂર છે. અમે ચોક્કસપણે શોધીશું કે આમાંથી કોઈ પણ ઉદાહરણ યોગ્ય નથી, તે સરળ લૂપ નથી જે આપણે ગણતરી કરી શકીએ છીએ અને પછી આપણે ખરેખર ઓપરેશન્સને કેવી રીતે ગણવું તે વિશે થોડી વધારે કાળજી રાખવી પડશે. તેથી, જ્યારે હું આ કરું છું ત્યારે વાસ્તવમાં કાર્યક્ષમતાનો અંદાજ કાઢે છે, તે ખરેખર એકાઉન્ટિંગ જેવું છે. તેથી, તમારી પાસે તમામ મૂળભૂત કાર્યવાહીનો ટ્રૅક રાખવાનો પ્રકાર છે અને તમારે તેની ખાતરી કરવાની સારી જોગવાઈ કરવી છે કે તમે તેને શ્રેષ્ઠ સંભવિત રૂપે ટ્રૅક રાખવા માટે કરો છો. તેથી, તમને આર્ટ્સની વાસ્તવિક ચિત્ર મળે છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)

પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મૉડ્યૂલ - 01
લેક્ચર - 09
એરે(Array) અને સૂચિ

તેથી, આ એકમમાં આપણે મૂલ્યોની સૂચિમાં મૂલ્ય માટે શોધ અને સૉર્ટિંગ(sorting) તરફ્ જોવું જોઈએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:06)
તેથી, ચાલો આપણે સમીક્ષા કરીએ કે મૂલ્યોની સૂચિ ખરેખર કમ્પ્યુટરમાં કેવી રીતે સંગ્રહિત થાય છે. તેથી, તમે અનુક્રમ સંગ્રહિત કરવાના બે મૂળભૂત માર્ગો છે જેમ કે તમે એરે(Array) અને સૂચિ જાણો છો. હવે, વિધેયાત્મક રીતે તેઓ સમાન દેખાય છે, પરંતુ એક જટિલતા સિદ્ધાંત દ્રષ્ટિકોણથી, કાર્યક્ષમ દૃષ્ટિકોણથી, ડેટા કેવી રીતે ગોઠવાય છે તે તફાવત બનાવે છે. તેથી, ચાલો આપણે એરે(Array) અને સૂચિ વિશે કેટલાક બેઝિક્સ વિચારોની સમીક્ષા કરીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:30)
તેથી, પ્રારંભ કરવા માટેની એરે(Array) મેમરીનું એકલ બ્લોક છે. તેથી, જો તમારી પાસે કોઈ એરે(Array) હોય, તો તેને કંઈક એવું માનવું જોઈએ કે જેમાં એરે(Array)માં મૂલ્યો સંગ્રહિત સતત ઘટકો હોય. તેથી, જો તમારી પાસે કદ n ની એરે(Array) એ છે, તો એ 0 એ તરત જ એ 1 થી અનુસરશે અને તેથી એ, એ 1 ની બાદબાકી 1 સુધી. તેથી, નિર્ણાયક હકીકત એ છે કે જો મને ખબર હોય કે અરે ક્યાં છે, તો જો હું. કેટલાક મૂલ્ય A ને મેળવવા માંગે છે, પછી મારે એ છે કે, હું ક્યાં છું તે શોધવા માટે એરે(Array)ના દરેક એકમનું કદ, i દ્વારા ગુણાકાર કરવો પડશે. તેથી, આ કહેવાનું પ્રમાણ છે કે અમે કોઈ પણ મૂલ્ય A, કોઈપણ સમયે હું એરે(Array)માં કોઈપણ સ્થાનને સતત સમયમાં વાપરી શકીએ છીએ, પછી ભલેને હું શરૂઆત અથવા અંતે છે, પછી ભલેને આપણે માત્ર ઓફ્સેટની ગણતરી કરી શકીએ. તેથી, અમારી પાસે પ્રારંભિક સ્થિતિ છે અને પછી હું આપેલ છે, અમે એક શૉટમાં સીધી ગણતરી કરી શકીએ છીએ, શરૂઆતના સરનામાંના સંદર્ભમાં માત્ર અંકગણિત કરી રહ્યા છીએ, એ I ની સ્થિતિ. હવે, બીજી તરફ્, જો હું A અને A i plus 1 ની વચ્ચે કોઈ ઘટક શામેલ કરવા માંગું છું, તો તે એક મોટી જટિલ બની જાય છે કારણ કે, હવે વિચારે છે કે અહીં મૂલ્યને દબાણ કરવું છે; તેનો અર્થ છે, મારે આ મૂલ્યો લેવા અને તેમને નીચે ખસેડવા પડશે; તેનો મતલબ એ છે કે, આ દરેક મૂલ્યને 1 દ્વારા ખસેડવું પડશે. તેથી, તત્વ શામેલ કરવા માટે લેવાયેલા સમયને સ્થિતિ પર આધારીત છે, પરંતુ ખરાબ કિસ્સામાં મારે બધા ઘટકોને 1 દ્વારા ખસેડવા પડશે. તેથી, આમાં સમય લાગી શકે છે એરે(Array)ના કદના પ્રમાણમાં. તેથી, આ ઓર્ડર એન કામગીરી હોઈ શકે છે. આ જ રીતે સંકોચનથી મને એરે(Array) લેવાની જરૂર પડી શકે છે અને પછી જો હું આ તત્વ દૂર કરવા માંગું છું, તો મને આ બધા ઘટકોને 1 થી ઉપર ખસેડવા પડશે. તેથી, જો હું વિસ્તૃત કરવા અથવા અરેને કરાર કરવા માંગું છું, પરંતુ હું પોઝિશનથી સ્વતંત્ર હોય તેવા નિશ્ચિત કિંમત સમયમાં એરે(Array)માં આપેલા કોઈપણ ઘટકને ઍક્સેસ કરી શકું છું. અને હું તેને કોઈપણ અન્ય એક્સેસની જેમ સારવાર કરી શકું છું, જેમ કે વેરિયેબલ(variable) એક્સ અથવા વાય અથવા મારા પ્રોગ્રામમાં સરળ કંઈપણ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:24)
બીજી તરફ્ની સૂચિ સામાન્ય રીતે લવચીક માળખું છે અને તત્વો ઉમેરવામાં આવે છે, તેથી તે મેમરીની આસપાસ ફેલાયેલા હોય છે અને એકબીજા સાથે કોઈ ચોક્કસ સંબંધ નથી. તેથી, સામાન્ય રીતે સૂચિમાં મેમરીના જુદા જુદા ભાગોમાં મૂલ્યો હશે અને આ વિચાર એ છે કે દરેક મૂલ્ય આગલા તરફ્ નિર્દેશ કરશે. તેથી, આ સૂચિની શરૂઆત છે તેવું અનુમાન કરો, પછી તે આગલા તત્વ તરફ્ નિર્દેશ કરશે, તે ત્રીજા તત્વ તરફ્ નિર્દેશ કરશે, તે ચોથા તત્વ તરફ્ નિર્દેશ કરશે અને તેથી, આગળ. તેથી, આ લિંકિંગ માળખું સામાન્ય રીતે મોટાભાગના પ્રારંભિક માહિતી માળખાં અભ્યાસક્રમોમાં હોવાથી, સૂચિને હંમેશાં

લિંક સૂચિ તરીકે ઓળખવામાં આવે છે. તેથી, લિંક્સ સૂચિ આવી ફ્લેક્સિબલ પગલું માળખું અમલમાં મૂકવાની એક નક્કર રીત છે. તેથી, હવે આનો અર્થ શું છે કે, જો હું l 2 ક્યાં કહું તે શોધવા માંગુ, તો ધારો કે આ l 0, l 1 અને l 2 છે. પછી, મને સામાન્ય રીતે કોઈ ખ્યાલ નથી, જ્યાં l 2 સ્થિત છે, પરંતુ મને ખબર છે કે l 0 ક્યાં છે. સ્થિત છે, કારણ કે મારી સૂચિનું નામ અને મારું પ્રોગ્રામ l 0 તરફ નિર્દેશ કરશે. તેથી, હું 2 તીર સુધી પહોંચવા માટે આ તીરને અનુસરવું પડશે. તેથી, સામાન્ય રીતે જો હું એલ એ કરવા માંગું છું, તો મને A પર પ્રારંભ કરવું પડશે. 0, પછી એ 1 અને તેથી, મારા પરની મારી લિંકને અનુસરો. તેથી, આ મને પગલાં લેશે. તેથી, હું સૂચિમાં વધુ નીચે જવાની જરૂર છે, તે લાંબો સમય લે છે. તેથી, એનો ઉપયોગ કરવો એ i ના પ્રમાણમાં છે. તેથી, સામાન્ય રીતે જો મારે i નો ઉપયોગ કરવો હોય તો, l ની સીમા તે રેખાકીય પરિમાણ છે. બીજી બાજુ, દાખલ કરવું અને કાઢી નાખવું એ પ્રમાણમાં સરળ છે, જો હું જાણું છું કે હું ક્યાં છું, મને લાગે છે કે હું l2 અને l 3 વચ્ચે કંઈક શામેલ કરવા માંગું છું, તો હું શું કરું છું તે હું પહેલા નવો નોડ બનાવું છું અને પછી હું શું હું પ્લમ્બિંગ કહીશ. તેથી, હું આ લિંકને રદ્ કરીશ અને હું તેના બદલે, l 2 થી નવા નોડ અને નવા નોડમાંથી l 3 સુધી એક લિંક દાખલ કરું છું અને મને ખબર છે કે કઈ લિંક્સ ઉમેરવાનું છે, કારણ કે આ એક નોડ મેં હમણાં જ બનાવ્યો છે. તેથી, હું તેનું સ્થાન જાણી શકું છું અને એલ 2 પોઈન્ટ 3 સુધી. તેથી, હું તે માહિતીને નવા નોડ પર સ્થાનાંતરિત કરી શકું જેથી કરીને આ લિંક્સને સ્થાપિત કરી શકાય. અને જ્યારે હું નોડને કાઢી નાખવા માંગું છું ત્યારે સમાન વસ્તુ થાય છે, જો હું આ નોડને ઉદાહરણ તરીકે કાઢી નાખવા માંગુ છું, તો મારે શું કરવું છે, હું આ લિંક લઈશ અને હું તેને બાયપાસ કરી સીધું જ જાવ. તેથી, આ લિંક્સને આસપાસના સ્થાને, સ્થાયી સમયસર સ્થાનિક અર્થમાં, અમે સૂચિમાં કોઈપણ સમયે શામેલ અથવા કાઢી શકીએ છીએ. પરંતુ, સ્થિતિ શોધવા માટે મને શરૂઆતથી શરૂ થવું અને અંતમાં જવાની જરૂર છે, આ રેખીય સમય લે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:47)

તેથી, આ ભેદભાવનો આપણે શું કરી શકીએ તેના પર અસર છે. ધારો કે, આપણે મૂલ્યોનું અનુક્રમણિકા લેવા માંગીએ છીએ અને અમે પોઝિશન I અને પોઝિશન j પર મૂલ્ય લેવા માંગીએ છીએ અને હું તેને બદલવું છે. તેથી, જો હું એ અને જે સ્થિતિમાં પોઝિશનને જાણું છું, તો હું એરે(Array) અને એજેને સતત સમય પર મેળવી શકું છું અને તેમને વિનિમય કરી શકું છું. તેથી, આ બરાબર x અને y ને અદલાબદલ કરવું અને x અને y ને અદલાબદલ કરવામાં કોઈ તફાવત નથી, હું એ અને એ જે અદલાબદલ કરી રહ્યો છું. પરંતુ, જો તે સૂચિ છે, તો મને આ શોધવા માટે નીચે જવું પડશે, પછી મને તે શોધવા માટે નીચે જવું પડશે. તેથી, મને આ બે વસ્તુઓ મળી છે અને પછી મને વિનિમય કરવો પડશે. તેથી, સ્થાનોનું વિનિમય કરવા માટે મને માત્ર એક રેખીય સમય લાગશે. બીજી બાજુ, જેમ આપણે પહેલેથી જોયું છે, જો તમે કોઈ સૂચિમાંથી કોઈ તત્વ કાઢી નાખવા માંગતા હો અથવા અમે સૂચિમાં એક તત્વ શામેલ કરવા માગીએ, તો આ સતત સમય લે છે, જો કે અમે પહેલાથી જ A પર છે. તેથી, જો આપણે A સુધી પહોંચીએ અને આપણને આ સ્થિતિમાં મળે, તો તમે કંઈક કરવા માંગો છો અમે સતત કાર્યમાં તે ઑપરેશન, નિવેશ અથવા કાઢી નાખવું કરી શકીએ છીએ. પરંતુ, તે સ્થાને, જો તમે કોઈ મૂલ્ય અથવા એરે(Array) કોન્ટ્રાક્ટ શામેલ કરવા માંગો છો, તો પછી તમારે સંપૂર્ણ સંખ્યામાં આગળનાં અથવા પાછળના ભાગોને પાળી છે અને આ રેખીય સમય લેશે. તેથી, કેટલીકવાર આ આપણે જે કરી શકીએ તેના પર અસર કરે છે, એક ડેટાગ્રામ પર એક એલ્ગોરિધમ્સ સારી રીતે કાર્ય કરે છે, તે એક જ વસ્તુમાં કામ કરશે નહીં, જો કે તે અમૂર્ત રીતે એક જ વસ્તુનું પ્રતિનિધિત્વ કરે છે, એકથી n અથવા 0 થી n minus 1 ની વેલ્યુનો ક્રમ. આનાં ઉદાહરણ રૂપે, આપણે ટૂંક સમયમાં બાઈનરી શોધ જોશું. બાઈનરી શોધ એરે(Array) પર કામ કરે છે, પરંતુ તે સૂચિ પર કાર્ય કરતું નથી, કારણ કે તે અમને કેટલાક સૂચકાંકમાં કાર્યક્ષમ રીતે વારંવાર અરેની તપાસ કરવાની જરૂર છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મૉડ્યૂલ - 02
લેક્ચર - 10
સરચિંગ ઈન એરે(Searching in an Array)

ચાલો એરેમાં મૂલ્ય શોધવા માટેની સમસ્યાને જોઈએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:05)
તેથી, સામાન્ય રીતે શોધ(Search) સમસ્યા એ છે કે મૂલ્ય કે હાજર છે કે નહીં તે મૂલ્યો A ના સંગ્રહમાં હાજર છે અને આપણા કિસ્સામાં આપણે એ વિચારીશું કે એ મૂલ્યોના અનુક્રમે છે. અને વધુમાં, આપણે એમ પણ માનીએ છીએ કે અનુક્રમણિકા પૂર્ણાંક જેવી કંઈક છે, જ્યાં આપણે એક મૂલ્યની વાત બીજા મૂલ્ય કરતાં ઓછી હોઈ શકે છે. તેથી, મૂલ્યો એકબીજા સાથે ઓર્ડર કરી શકાય છે. તેથી, આપણે પહેલાથી જોયું છે કે આપણે આવા અનુક્રમોને એરે અને સૂચિ તરીકે, બે અલગ અલગ રીતે રાખી શકીએ છીએ. અને આપણે તેને એરે અથવા સૂચિ તરીકે રાખીએ છીએ તેના આધારે, અમે તત્વોને કેવી રીતે ઍક્સેસ કરી શકીએ તે અલગ છે. તેથી, આપણે પૂછી શકીએ છીએ કે પહેલો પ્રશ્ન એ છે કે શોધ એરેની સૂચિની સૂચિમાં તફાવત બનાવે છે અને બીજા પ્રશ્નને આપણે કહી શકીએ છીએ, ક્રમમાં મૂલ્યો કેવી રીતે ગોઠવવામાં આવે છે તે મહત્વનું છે, જો તે હોય તો તે મદદ કરે છે ચઢતા અથવા ઉતરતા ક્રમમાં અથવા તે કોઈ વાંધો નથી. તે મૂલ્યના રેંડમલી ઓર્ડર કરેલા સંગ્રહમાં કંઈક અથવા જ્યારે તે કોઈ ચોક્કસ રીતે રચાયેલ હોય ત્યારે કંઈક શોધવા માટે સમાન છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:03)
તો, અનસૉર્ટ કરેલ કેસમાં આપણી પાસે કોઈ વિકલ્પ નથી, આપણી પાસે અનુક્રમ એ છે જે 0 થી n ઓછા 1 થી ચાલે છે. તેથી, આપણે બધા મૂલ્યોને જોવું જોઈએ, કારણ કે અમને કોઈ ખ્યાલ નથી. કે જ્યાં કદાચ. તેથી, તે કરવા માટેની વ્યવસ્થિત રીત એ સ્થિતિ 0 થી શરૂ કરવી અને ફક્ત n ઓછા 1 સુધી બધી રીતે સ્કેન કરવી. તેથી, અમારી પાસે અહીં આ લૂપ છે, જે સ્કેન કરે છે અને જ્યારે તમે તેને શોધ્યા વિના અંત સુધી પહોંચો ત્યારે આ સ્કેન સમાપ્ત થાય છે. કેટલાક સ્થાને હું શોધી કાઢું છું કે હું એ કે સમાન છે અને પછી તેના પર આધાર રાખીને આપણે કહીએ છીએ કે તે મળી આવ્યું છે, તે સ્થિતિમાં આપણે પોઝિશન પાછી મેળવીએ છીએ અથવા આપણે પહોંચી ગયા છીએ, હું સમાન છે, જેનો અર્થ એ છે કે આપણે એ n માઈનસ 1 અને તેથી, આપણે 1 નહી 1, બાદબાકી 1 જે અમાન્ય સ્થિતિ છે તે સૂચવે છે કે તે મળી નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:53)
તેથી, આપણે જોયું કે તે પહેલા ખરાબ કેસ ખરેખર બને છે જ્યારે કે એ એ ના ઘટક નથી, કે કે એ એમાં નથી આવે, તો આપણે A 0 થી A માઈનસ 1 ને સ્કેન કરવું પડશે કેસ કે જે નક્કી નથી. કારણ કે, અમારી પાસે અગાઉથી કોઈ પુરાવા નથી જે સ્થિતિ હોવાની શક્યતા છે. તેથી, આનો અર્થ એ છે કે તત્વ અને બિન સૉર્ટ કરેલી એરે માટે શોધતા સૌથી ખરાબ કેસમાં લીનિયર પ્રકાર લે છે. અને અલબત્ત, તે કોઈ અરે અથવા સૂચિ છે કે કેમ તે હવે કોઈ વાંધો નથી, કારણ કે સૂચિમાં આપણે રેખાવારનો સમય પણ પ્રથમ તત્વથી પ્રારંભ કરી શકીએ છીએ અને અંત સુધી બધી રીતે લિંક્સને અનુસરી શકીએ છીએ, એરેમાં આપણે એ સાથે પ્રારંભ કરીએ છીએ. 0 અને એ માઈનસ 1 થી બધી રીતે જાઓ, તે બંને રેખીય સમય લે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:29)
બીજી તરફ જો ક્રમ ક્રમિત છે અને ખાસ કરીને જો તે એરે છે, તો આપણે થોડું વધુ બુદ્ધિશાળી હોઈ શકીએ છીએ. તેથી, આપણે જાણીએ છીએ કે મૂલ્યો ચડતા ક્રમમાં સોંપવામાં આવે છે. તેથી, જો તમે મધ્યમાં મૂલ્યની ચકાસણી કરો અને તપાસ કરો કે તે સમાન છે, તો આપણી પાસે જે મૂલ્ય છે તે કેલ્કની બરાબર છે, આપણે તે શોધી કાઢ્યું છે. જો તે અહીં મૂલ્ય કરતાં નાનું હોય, તો આપણને ફક્ત આ અર્ધ શોધવાની જરૂર છે, અને જો તે આ અર્ધમાં શોધવાની જરૂર કરતાં મોટો હોય તો. હવે, આ કંઈક છે જે આપણે હંમેશાં કરવું જોઈએ, આ રીતે આપણે શબ્દકોષમાં શબ્દો બોલવા અથવા જ્યારે

આપણે 20 પ્રશ્નો વગાડતા હોઈએ છીએ, ત્યારે આપણે કોઈ વ્યક્તિની ઉંમર વિશે પ્રશ્નો પૂછવાનો પ્રયાસ કરીએ છીએ, જ્યારે તમે આ વ્યક્તિ કરતાં ઓછી કહો છો 40, આ વ્યક્તિ 65 કરતા વધારે નથી અને તેથી, ચાલુ. તેથી, આ કંઈક છે જે આપણે આત્મવિશ્વાસથી જાણીએ છીએ, પરંતુ અમે ઔપચારિક બનાવી શકીએ છીએ. તેથી, અમે જે શ્રેણી શોધી રહ્યાં છે તે મિડપૉઈન્ટ લઈએ છીએ. જો મધ્યબિંદુ એ એક મૂલ્ય છે જે આપણે જોઈતું હોય તો આપણે તેને શોધી શકીએ; અન્યથા, અમે જે મૂલ્યને શોધી રહ્યા છીએ તેના આધારે અને મધ્યબિંદુ શું મૂલ્ય છે તેના પર આધાર રાખીએ છીએ, અમે ક્યાં તો તળિયે અડધા અથવા ટોચની અડધી શોધ કરીએ છીએ. તેથી, આ એક એવું નામ છે જે તમારામાંના ઘણા પહેલાથી જ જાણી શકે છે, આને બાઈનરી(Binary) શોધ કહેવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:38)

તેથી, અહીં દ્વિસંગી શોધ માટે એક સરળ રિકર્સિવ(recursive) એલ્ગોરિધમ છે. તેથી, સામાન્ય રીતે તે એરેને શોધે છે, યાદ રાખો કે જ્યારે અમે શોધ કરીએ છીએ ત્યારે અમે શોધને કેવી રીતે પ્રગતિ કરીએ છીએ તેના આધારે અમે વિવિધ સેગમેન્ટ્સ શોધતા હોઈએ છીએ. તેથી, સામાન્ય દ્વિસંગી શોધમાં એરે અને બે અંત પોઈન્ટ, ડાબે અને જમણે શોધવા માટે મૂલ્ય કે લે છે અને ખાતરી કરો કે આપણે બધું બરાબર મેળવીશું, તો તમે તે સંમેલન કરો છો જે તે ઈન્ડેક્સ l થી ઈન્ડેક્સમાંથી શોધે છે. બાદબાકી 1, મને l થી r માં શોધવા દો, પરંતુ r ને પોતે શામેલ કરશો નહીં. તેથી, હવે જો l અને r ખરેખર સમાન છે, તો આપણી પાસે ખાલી એરે છે, કારણ કે l થી માઈનસ 1 એ a છેખરેખર કંઈક કે જે ક્ષેત્રોમાં કોઈ તત્વો નથી. તેથી, અમે કહીએ છીએ કે અમને તે મળ્યું નથી. તેથી, જ્યારે આપણે જે અંતરાલ શોધી રહ્યા છીએ તે ખાલી બને છે, એરેમાં ચોક્કસપણે મૂલ્ય શામેલ હોતું નથી

((સમયનોસંદર્ભ લો: 04:27)).

નહિંતર, અમે સરવાળા અને આર વચ્ચે મધ્યબિંદુની ગણતરી કરીએ છીએ અને 2 દ્વારા ભાગ લે છે અને આ એક વિચિત્ર સંખ્યા હોઈ શકે છે, તેથી આપણે પૂર્ણાંક વિભાગનો ઉપયોગ કરીએ છીએ. હવે, આપણે આ બિંદુએ તપાસ કરીશું, આપણે મધ્યબિંદુ શોધીશું. તેથી, હવે, આપણે તપાસ કરીએ છીએ કે જો આપણે જે મૂલ્ય ઇચ્છીએ છીએ તે મધ્યબિંદુ પર છે, જો. તેથી, અમે સાચા પાછા ફરો. નહિંતર, જો મુલ્ય કે જે આપણે મધ્યબિંદુ કરતાં નાના કરવા માંગીએ છીએ, તો આપણે ડાબી બાજુએ જઈએ છીએ અને જે મૂલ્ય આપણે ઇચ્છીએ છીએ તે મધ્યબિંદુ કરતા મોટું છે, પછી આપણે જમણી બાજુએ જઈએ છીએ. તેથી, આ કાં તો ડાબેથી મધ્યમ 1 અથવા મધ્યમ વત્તા 1 થી જમણે જાય છે. બીજા શબ્દોમાં કહીએ તો અમે અમારી શોધમાંથી મધ્યમને બાકાત રાખીએ છીએ. તેથી, આ પહેલો કેસ ડાબેથી મધ્યમ 1 ની શોધ કરે છે કારણ કે તે અમારી ધારણા છે, આપણે તેને મધ્ય સુધી કહીએ છીએ, તે મધ્યમ 1 થી લઈ જાય છે. આ એક મધ્યમ વત્તા 1 થી શરૂ થાય છે અને જમણી બાજુ 1 થી જાય છે. તેથી , મૂળ વસ્તુ ડાબીથી જમણી બાજુ 1 થી હતી અને હવે આપણે આમાંથી મધ્ય ભાગને બાકાત રાખ્યો છે અને અમારી પાસે સંપૂર્ણ શોધ પણ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:25)

તેથી, બાયનરી શોધનો નિર્ણાયક ફાયદો એ છે કે દરેક પગલામાં શોધવા માટેનો અંતરાલ છે અને કોઈક સમયે આપણે 1 સુધી પહોંચશું અને પછી જ્યારે 1 હશે, ત્યારે આપણને 0 કદનો અંતરાલ મળશે અને તેથી, અમે એક તાત્કાલિક જવાબ મળશે. તેથી, જેમ આપણે આવા પુનરાવર્તન કાર્યો માટે જોયું તેમ આપણે લખી શકીએ છીએ, આપણે લખી શકીએ છીએ પુનરાવર્તન કહેવાય છે. સમાન અભિવ્યક્તિવાળા નાના મૂલ્યોના સંદર્ભમાં પુનરાવર્તન એ સમય માટે માત્ર એક અભિવ્યક્તિ છે. તેથી, મૂળ કેસ એ છે કે જ્યારે આપણી પાસે T ની n હોય ત્યારે તેનો અર્થ એ છે કે સૂચિમાં શોધવા માટેનો સમય અથવા કદ n એ વાસ્તવમાં કદ. તેથી, 0 નું T એ 1 છે. તેથી, જો આપણી પાસે ખાલી એરે હોય તો અમારે કંઈ કરવાનું નથી અને સામાન્ય રીતે T ની n એ શોધવા અને તુલના કરવા એક પગલું છે. તેથી, આ ખરેખર મધ્યબિંદુની સરખામણી કરવા માટે સતત સંખ્યામાં પગલાં છે અને તે ઉપર, નીચે અને તે બધાને નક્કી કરવાનું નક્કી કરે છે. તો, તે ઓપરેશનો વત્તા તે શોધવાનો સમય જે આપણે અડધા પર ધ્યાન કેન્દ્રિત કર્યું છે, ડાબા અર્ધ અને જમણા અર્ધ. યાદ રાખો કે, આપણે ડાબી બાજુએ જોઈએ છીએ, તમે ક્યારેય જમણી અડધી તરફ જોશો નહીં.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:25)

તેથી, આવા પુનરાવર્તનને ઉકેલવાની એક રીત એ છે કે તેને અનિશ્ચિત કરવું. તો, આપણી પાસે ટી n એ 1 વત્તા T n 2 છે. તેથી, આપણે 2 વડે n લઈએ છીએ અને આપણે 2 વડે 2 વડે ભાગીએ અને લખીને બદલે n એ 4 વડે લખીએ છીએ, આપણે લખીએ છીએ કે n એ 2 ચોરસ છે. અને આ છે કારણ કે હવે, જો હું એક વધુ સમય કરું તો તે 1 વત્તા 1 વત્તા 1 ક્યુબ દ્વારા વિભાજિત થશે અને તેથી, ચાલુ રહેશે. તેથી, સામાન્ય રીતે તમે જોઈ શકો છો કે જો હું તે સમયે કરું તો મારી પાસે K1 ની પાસે હશે, જો હું 3 વખત કરું તો મારી પાસે 1 વત્તા 1 વત્તા 1 વત્તા ટી n થી 2 થી 3, 4 અથવા 4 વત્તા T n હશે. 2 અને તેથી, દ્વારા. તેથી, k પગલાઓ પછી મારી પાસે k ના પ્લસ ટી n થી 2 ની છે, હવે જ્યારે n થી 2 ની સાથે k એ આગલા પગલા પર આવે છે તો મને 0 ની T મળશે. તેથી, આ ક્યારે બને છે 1, જ્યારે n એ બીજા શબ્દોમાં k થી 2 છે જ્યારે k એ n નો log 2 છે. તેથી, જ્યારે હું n નો લોગ 2 મેળવું, તો તે 1 ની T અને બીજો પગથિયું બની જાય છે, તે 0 ની 1 વત્તા ટી બનશે. તેથી, આ લોગ n 1s બનશે અને તેથી, બધી જટિલતા ઉપર બાઈનરી(Binary) શોધ ફક્ત લોગ n નો ક્રમ છે. તેથી, સૉર્ટ કરેલા એરેના કેસમાં લૉગરિધમિક શોધમાં અનૉર્ટ કરેલા એરેના કેસમાં અમે રેખીય શોધ સાથે સમાધાન કરીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:50)

તેથી, આપણે અગાઉના એકમમાં એરે અને સૂચિ વિશે ઉલ્લેખ કર્યો છે કે, સૂચિ પર કામ કરતી વસ્તુઓ કદાચ એરે અને તેનાથી વિપરીત કામ કરી શકશે નહીં. તેથી, અહીં કંઈક ઉદાહરણ છે જે તે ફક્ત એરે માટે કામ કરે છે. મધ્યબિંદુને જોવામાં અને પછી ડાબી તરફ જવું એ વિચાર, જો તમે મધ્યબિંદુ સતત સમય શોધી શકો છો, જો તમારે મધ્યબિંદુ માટે સતત જોવાની સમય કાઢવો પડે, તો તમને આ પુનરાવર્તન હવે મળી શકશે નહીં જે 1 બનશે નહીં પ્લસ ટી ના n થી 2, પણ તે n ની વત્તા ટી 2 વત્તા n વત્તા 2 વડે થશે અને પછી આપણે વાસ્તવમાં રેખીય બનીશું. તેથી, સૂચિ માટે દ્વિસંગી શોધ વાસ્તવમાં રેખીય બનશે, કારણ કે તે અમને મધ્યબિંદુ પર જવા માટે લે છે. તેથી, આ ફક્ત એરે માટે જ કાર્ય કરે છે, પરંતુ બાઈનરી(Binary) શોધ વિશે ખરેખર નોંધપાત્ર વસ્તુ એ છે કે માત્ર ક્રમના ખૂબ નાના ભાગને જોઈને, આપણે નિષ્કર્ષ કરી શકીએ છીએ કે તત્વ અસ્તિત્વમાં નથી. તેથી, આપણે ઉદાહરણ તરીકે જાણીએ છીએ કે 2 થી 2, 2 થી 10 ની 10 છે. તેથી, જો હું તમને 1000 મૂલ્યો આપીશ, તો આપણે 10 અથવા 11 જોઈ શકીએ છીએ અને કહી શકીએ કે કંઈક નથી. તેથી, અમે મૂલ્યવાન સંખ્યા અથવા મૂલ્યોને ધ્યાનમાં લઈએ છીએ કે મૂલ્ય ત્યાં છે કે નહીં તે નિર્ધારિત કરવા માટે આપણે ન જોવું જોઈએ અને બાયનરી(Binary) શોધ અસામાન્ય પ્રક્રિયા હોઈ શકે છે, જો તમે અહીં તેના વિશે વિચારો છો.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મોડ્યુલ - 03
લેક્ચર - 11
સિલેકશન સોર્ટ (selection sort)

તેથી, એરે(Array)માં તત્વ શોધવા માટે કેવી રીતે જોવું જોઈએ, હવે ચાલો સૉર્ટિંગ(sorting) કરીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:07)

તેથી, સૉર્ટિંગ(sorting) માટેનો સૌથી મૂળભૂત પ્રેરણા શોધમાંથી આવે છે. જેમ આપણે જોયું છે, જો તમારી પાસે કોઈ વાણગોઠવેલ એરે(Array) હોય, તો તમારે સંપૂર્ણ એરે(Array)ને શરૂઆતથી અંત સુધી સ્કેન કરીને તેને શોધવું પડશે. તેથી, તમે તત્વ શોધવા માટે રેખીય સમય વિતાવો છો, જ્યારે તમે એરે(Array)ને સૉર્ટ(sort) કર્યું છે, તો તમે તેને મધ્યબિંદુ પર શોધી શકો છો અને બાઈનરી શોધનો ઉપયોગ કરી શકો છો અને સમાન પરિણામોને લઘુગણક સમયમાં પ્રાપ્ત કરી શકો છો અને લોગેરિધમિક સમય રેખીય સમય કરતાં નોંધપાત્ર રીતે ઝડપી છે. હવે, ઘટકો સૉર્ટ(sort) કરવાના અન્ય ફાયદા છે. દાખલા તરીકે, જો તમે સરેરાશ શોધી શકો છો, તે મૂલ્ય કે જેના માટે અડધા તત્વો મોટા હોય, તો અડધા તત્વો નાના હોય છે. સારુ, સૉર્ટ(sort) કરેલા એરે(Array)નું સરેરાશ એરે(Array)ની મધ્યબિંદુ સ્પષ્ટપણે છે. જો તમે કોઈ અન્ય પ્રકારની આંકડાકીય બાબતો કરવા માંગો છો, જેમ કે મૂલ્યોની ફ્રીક્વન્સી ટેબલ બનાવવી, જ્યારે તમે આ મૂલ્યોને સૉર્ટ(sort) કરો છો ત્યારે તે બધા એકસાથે આવે છે, સાચું. તેથી, બધા સમાન મૂલ્યો એક સુસંગત બ્લોકમાં હશે. તેથી, ત્યાં દરેક મૂલ્યની કેટલી નકલો છે તે શોધવાનું ખૂબ સરળ છે. અને ખાસ કરીને, જો તમારે દરેક મૂલ્યની ફક્ત એક જ નકલ જોઈએ છે, જો તમે એરે(Array)ના બધા ડુપ્લિકેટ્સને દૂર કરવા માંગો છો, તો તમે સૉર્ટ(sort) કરેલ એરે(Array)ને પ્રારંભથી અંત સુધી સ્કેન કરો છો અને મૂલ્યોના દરેક બ્લોક માટે ફક્ત એક કૉપિ રાખો. તેથી, એરે(Array) ઘણા વધુ સરળ કારણો છે કે જેથી કોઈ એરે(Array) પર વધુ ગણતરી કરવા માટે એરેને સૉર્ટ(sort) કરવા માંગે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:16)

તેથી, ચાલો કલ્પના કરીએ કે તમે એરે(Array) કેવી રીતે સૉર્ટ(sort) કરવા માંગો છો. તેથી, એરે(Array) ભૂલી જાઓ અને કંઈક સૉર્ટ(sort) કરવા વિશે વિચારો, જે તમને પદાર્થના ભૌતિક સંગ્રહ તરીકે આપવામાં આવે છે. તેથી, કહો કે, તમે અભ્યાસક્રમ અને પ્રશિક્ષક માટે સહાયક શિખર રહ્યા છો, જે શિક્ષકએ અભ્યાસક્રમનો અભ્યાસ કર્યો છે તે પરીક્ષામાં સુધારો કર્યો છે, હવે તમે માર્કસના પરીક્ષાના કાગળોને સૉર્ટ(sort) કરવા માગો છો. તેથી, કહો, તમારું કાર્ય તેમને ઉતરતા ક્રમમાં ગોઠવવાનું છે, તમે આ કાર્ય કેવી રીતે કરશો?

(સ્લાઈડસમયનો સંદર્ભ લો: 01:47)

તેથી, એક સરસ વ્યૂહરચના નીચે મુજબ છે. તમે સમગ્ર સ્ટેકથી પસાર થશો અને તમે જોયેલી સૌથી નાનો માર્કનો ટ્રેક રાખો. તમારા પાસના અંતમાં તમારા હાથમાં પેપર છે, જે તમામ વિદ્યાર્થીઓને ફાળવવામાં આવેલા માર્કસમાં સૌથી નાનું ચિલ્ન છે. તેથી, તમે તેને નવા ખૂંટો પર ખસેડો, પછી તમે પ્રક્રિયાને પુનરાવર્તિત કરો. નવા પાઈલમાં સૌથી નાનું છોડી દેવાના પછી અને પછીના સૌથી નાનાને શોધવા પછી બાકીના પેપરોમાંથી પસાર થાઓ, નવા ખૂંટોની ઉપર અને આગળ વધો. તેથી, બીજા પાસ પછી તમારી પાસે ખૂણા પર બીજો સૌથી નાનો ચિલ્ન છે. ત્રીજા પાસ સુધી તમારી પાસે ત્રીજી સૌથી નાની છાપ છે અને તમે કલ્પના કરી શકો છો, એન પાસ કર્યા પછી તમે બધા નવા કાગળો જૂના ઢગલામાંથી નવા ખૂંટા પર ઉતરતા ક્રમમાં ખસેડ્યા છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:30)

તેથી, આ વિશિષ્ટ વ્યૂહરચના નીચે મુજબ દર્શાવી શકાય છે. તેથી, ધારો કે અમારી પાસે આ સૂચિ અથવા છ ઘટકોની સૂચિ છે. તેથી, પ્રથમ પાસમાં આપણે સૌથી નાનું મૂલ્ય શોધીએ છીએ. તેથી, આ કિસ્સામાં સૌથી નાનું મૂલ્ય 21 છે. તેથી, આપણે 21 ને નવી સૂચિમાં ખસેડીએ છીએ અને તેને મૂળ સૂચિમાંથી દૂર કરીએ છીએ. હવે, અમે બાકી મૂલ્યો વચ્ચે સ્કેન

પુનરાવર્તન કરો. મૂલ્ય 32 સૌથી નાનું છે, તેથી અમે તેને દૂર કરીએ છીએ અને તેને નવી સૂચિમાં ખસેડીએ છીએ. અમે આ કરવાનું ચાલુ રાખીએ છીએ. તેથી, આગળના પગલામાં, આપણે 55 ખસેડીશું અને પછી આપણે 64 ને ખસેડીશું અને પછી આપણે 74 ને ખસેડીશું અને પછી આપણે 89 ને ખસેડીશું. તેથી, આ પ્રક્રિયામાં જો આ એક ઊભી સ્ટેક હોત, તો 21 તળિયે હશે, 89 ટોચની હશે, અને તેથી અમે નીચે ઉતરતા ક્રમમાં ટોચથી નીચેની ક્રમમાં સૂચિબદ્ધ કરીશું. આ કિસ્સામાં, ઉતરતા ક્રમમાં જમણેથી ડાબેથી અથવા ડાબેથી જમણે ચડતા ક્રમમાં.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:22)

તેથી, આ વ્યૂહરચનાને પસંદગી સૉર્ટ(sort) કહેવામાં આવે છે. તેથી, અમે આગામી ઘટકની દરેક રાઉન્ડમાં પસંદ કરીએ છીએ, જે સૌથી નાનું છે, અને તેથી આગળ આપણે ક્રમબદ્ધ ક્રમમાં મૂકીએ છીએ અને તેને તેના સાચા સ્થાને ખસેડીએ છીએ, જે અંતિમ ક્રમાંકિત સૂચિના અંત સુધી છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:40)

હવે, પસંદગીની આ આવૃત્તિ જે આપણે હમણાં વર્ણવેલ છે, બીજી સૂચિ બનાવે છે. બીજા શબ્દોમાં કહીએ તો, એક ખૂંટોને સૉર્ટ(sort) કરવા માટે આપણે પેપર્સનું બીજું ખૂંટો બનાવવું પડશે. હવે, આપણે નીચે આપેલા કાગળોના બીજા ખૂણાને સરળતાથી દૂર કરી શકીએ છીએ. જ્યારે આપણે ન્યૂનતમ ઘટક શોધીએ છીએ, ત્યારે આપણે જાણીએ છીએ કે તે સૂચિની શરૂઆતમાં જ જવું જોઈએ. તેથી, નવી સૂચિ બનાવવાને બદલે આપણે તેને વર્તમાન સૂચિની શરૂઆતમાં ખસેડીએ છીએ. અલબત્ત, વર્તમાન સૂચિની શરૂઆતમાં બીજી કિંમત છે. તેથી, આપણે તે મૂલ્ય સાથે કંઈક કરવું પડશે. તમે માત્ર પોઝિશનનું વિનિમય કરો. તેથી, અમે ન્યૂનતમ પોઝિશન શોધીએ છીએ અને મૂલ્યને પ્રથમ સ્થાને મૂલ્ય સાથે ન્યૂનતમ સ્થાન પર અદલાબદલ કરીએ છીએ. હવે, સમગ્ર એરે(Array)માં સૌથી નાનું મૂલ્ય શરૂઆતમાં ખસેડ્યું છે. તેથી, હવે, અમે બીજા તત્વથી શરૂ કરીએ છીએ અને ન્યૂનતમ માટે જુઓ. ફરીથી, આ બીજો સૌથી નાનો હશે. તો, હવે, આપણે શોધી કાઢ્યું છે કે આપણે તે એરે(Array) મૂલ્યમાં બીજા સ્થાને ખસેડીશું. તેથી, અમે આ કરવાનું ચાલુ રાખીએ છીએ અને બીજી સૂચિનો ઉપયોગ કર્યા વગર અમે તે જ પ્રકારની સૉર્ટિંગ(sorting) કરી શકીએ છીએ જે અમે પહેલા કર્યું હતું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:33)

તો, આ કેવી રીતે કાર્ય કરે છે તે જોવા માટે આપણે તે જ એરે(Array)ને લઈએ જે પહેલા આપણી પાસે હતી. જેમ આપણે કહ્યું હતું કે, પ્રથમ રાઉન્ડમાં આપણે 21 ની લઘુતમ તત્વ તરીકે ઓળખી કાઢેલ છે, જે લાલ રંગીન છે અને પહેલી સ્થાને 74 છે. તેથી, અમે આ સૂચિની શરૂઆતમાં 21 ને પસંદ કરવા માંગીએ છીએ. તેથી, 74 થી ત્યાં છે, અમે 21 અને 74 નું વિનિમય કરીએ છીએ અને અમને આ નવી સૂચિ મળી છે. આ નવી સૂચિમાં, હવે 21 એ લીલા રંગમાં ચિલ્નિત છે જે સૂચવે છે કે તે તેના અંતિમ સ્થાને છે. તે સૌથી નાનું મૂલ્ય છે, તે સૉર્ટ(sort) કરેલ સૂચિમાં હોવું જોઈએ જ્યાં તે ખસેડવામાં આવે છે. તેથી, હવે આપણે બાકીના તત્વોને સ્કેન કરીએ છીએ અને અમે ઓળખીએ છીએ કે 32 સૌથી નાનો તત્વ છે. હવે, 32 પહેલાથી જ બીજા સ્થાને છે. તેથી, અમે તેને તમે કેવી રીતે અર્થઘટન કરવા માંગો છો તેના આધારે તેને બદલવું અથવા કંઈપણ કરવાનું વિચારી શકતા નથી. તેથી, 32 હવે લીલામાં ચિલ્નિત થાય છે, જેથી તે યોગ્ય સ્થિતિમાં રહે છે. હવે, બાકીની સૂચિમાં 55 એ લાલમાં સૌથી નાનો તત્વ ચિલ્ન છે, પરંતુ તેને પોઝિશન 3 પર હોવું જોઈએ, જે 89 દ્વારા પીળા રંગમાં છે. તેથી, આપણે આ બે વિનિમય કરીએ છીએ અને હવે આપણી પાસે ત્રીજી સ્થાને 55 છે. તેથી, અમે આ કરવાનું ચાલુ રાખીએ છીએ.તો, આગળના પગલા પર, આપણે 64 ને ઓળખીશું અને ચોથા સ્થાને તેને અદલાબદલ કરીશું અને પછી આપણે શોધી શકીએ કે 74 ની બરાબર સાચી સ્થિતિમાં છે. તેથી, આપણે તેને છોડી દઈએ છીએ અને છેવટે, અલબત્ત, 89 એકમાત્ર તત્વ છે. તેથી, જ્યારે તમારી પાસે એક તત્વની સૂચિ હોય ત્યારે ત્યાં કોઈ સૉર્ટિંગ(sorting) કરવું નહીં.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:57)

તેથી, આ પ્રક્રિયાને ખૂબ જ સરળ પુનરાવર્તિત અલ્ગોરિધમ દ્વારા વર્ણવી શકાય છે. તો, આપણે જે કરીએ છીએ, તે છે, જો તમને યાદ છે, તો આપણે આખા એરે(Array)ને સ્કેન કરીને પ્રારંભ કરીએ છીએ, એટલે કે 0 થી n ઓછા 1, આપણે તે સેગમેન્ટમાં ન્યૂનતમ ઘટક શોધીએ છીએ. તેથી, આપણી પાસે પુનરાવર્તન છે, જે મૂળરૂપે શરૂઆતની સ્થિતિમાં શરૂ થાય છે અને ત્યારબાદ પ્રારંભિક સ્થિતિથી તે ધારણા કરે છે કે પ્રારંભ એ છે, તે ન્યૂનતમ મૂલ્ય છે અને જ્યારે પણ તમને કોઈ

નાની કિંમત મળે છે, ત્યારે તમે લઘુત્તમ તરીકે ચિલ્નિત કરો છો પોઝિશન. હવે, આ બધા અધિકાર કર્યા પછી આપણે હવે 0 થી એ 1 ઓછા 1 થી ન્યૂનતમ મળ્યા છે, પછી તમે આ મૂલ્યને શરૂઆતમાં ખસેડો. તેથી, તમે પ્રારંભિક સ્થિતિ અને ન્યૂનતમ સ્થિતિને અદલાબદલ કરો છો. હવે, આપણી પાસે એ 0. ની સૌથી નાનું મૂલ્ય છે. તેથી, હવે તમે આ લૂપ પર પાછા જાઓ અને હવે તમે 0 થી 1 ની શરૂઆતની સ્થાને ખસેડો. તેથી, તમે એ 1 થી 1 ની બાદબાકી માટે સમાન સ્કેન કરો, હમણાં અને હમણાં પ્રારંભિક પોઝિશન 1 છે. તેથી, અંતે તમે બીજા નાના ઘટકને એ 1 માં અદલાબદલ કરો, તો તમે તે 2 થી એ 1 ઓછા 1 માટે કરો અને તેથી જ, બરાબર. તેથી, આ પસંદગી સૉર્ટ(sort)નું એક સરળ પુનરાવર્તિત સંસ્કરણ છે જ્યાં આપણે ફક્ત સંપૂર્ણ એરે(Array)થી પ્રારંભ કરીએ છીએ, સૌથી નાનો તત્વ પ્રથમ તત્વ, પ્રથમ સ્થાને ખસેડો, પછી આપણે એરે(Array)ના બાકીના ભાગને A 1 થી આગળ લઈ જઈએ પછી નાના ઘટકને A પર ખસેડો. 1, પછી 2 થી પ્રારંભ કરો, સૌથી નાનો તત્વ શોધો, A 2 પર જાઓ અને બીજું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:23)

તો, આ અલ્ગોરિધમ કેટલો સમય લે છે? તેથી સ્પષ્ટપણે, લંબાઈના ક્રમાંકિત સેગમેન્ટમાં ન્યૂનતમ ઘટક શોધવા માટે આપણે સંપૂર્ણ સેગમેન્ટને સ્કેન કરવું પડશે અને આમાં K પગલાં લેવાય છે અને દરેક પુનરાવર્તનમાં સેગમેન્ટને સ્કેન કરવા માટે 1 દ્વારા ઘટાડે છે. તેથી, અમે બધા n સ્કેન કરીને પ્રારંભ કરીએ છીએ. તત્વો, પછી આપણે n ઓછા 1 તત્વોને સ્કેન કરીએ છીએ, પછી અમે n ઓછા 2 ઘટકો સ્કેન કરીએ છીએ અને બીજું. તેથી, અમારી પાસે બધા પગલાંઓ છે n વત્તા n ઓછા 1 વત્તા n માઈનસ 2 ડાઉન 1, જે ફક્ત સામાન્ય સારાંશ છે જે હું 1 ના n થી 1 ની બરાબર છે. અને આ આપણે જાણીએ છીએ n એ n વત્તા 1 વડે 2 જે ઓર્ડર n ચોરસ છે, જમણે. તેથી, આ પુનરાવર્તિત અમલીકરણમાં આ નિષ્ક્રીય અલ્ગોરિધમ પસંદગી સૉર્ટ(sort) એ ચોરસ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 08:10)

હવે, પસંદગી સૉર્ટ(sort) જોવાની બીજી રીત છે. તેથી, યાદ રાખીએ કે આપણે કહ્યું છે કે, અમે ન્યૂનતમ તત્વને આગળના ભાગમાં ખસેડીને અને પછી તે જ વસ્તુ કરીને શોધીએ છીએ. જ્યારે પણ તમે એમ જ કહો છો ત્યારે આને યાદ રાખવું ઉપયોગી છે કે આને પુનરાવર્તિત અલ્ગોરિધમનો સ્વરૂપે સમજાવવા માટે, તમે જે કરો છો તે સાચું છે. તેથી, સામાન્ય સ્થિતિ A થી i ની n ની 1 ઘાતમાં ક્રમમાં ગોઠવવા માટે, આપણે જે કરીએ છીએ તે એ છે કે આપણે સેગમેન્ટમાં ન્યૂનતમ મૂલ્ય શોધીએ છીએ, એટલે કે A થી i થી A i minus 1 સુધી અને તેને શરૂઆતમાં ખસેડો, જે હું, અધિકાર છે. તો, આપણી પાસે એરે(Array) એ છે અને પછી આપણે પોઝિશન I અને છેલ્લા સ્થાને n માઈનસ 1 પર છીએ. અમે આને સૉર્ટ(sort) કરવાનો પ્રયાસ કરી રહ્યા છીએ. તેથી, આપણે જે કહી રહ્યા છે તે છે, આપણે લઘુત્તમ ક્યાંક શોધીશું અને પછી અહીં મૂલ્યનું વિનિમય કરીશું અને હવે આ દ્વારા પ્રાપ્ત કરવામાં આવે છે તે, આ સ્થિતિ હવે ઠીક છે, બરાબર. તેથી, A પરનું મૂલ્ય હવે યોગ્ય સ્થિતિમાં છે અને તેથી, હવે આપણે એક, હું વત્તા 1 થી n ઓછા 1 થી ક્રમમાં ગોઠવવાની જરૂર છે અને આપણે આ કેવી રીતે ફરીથી કરીશું? અમે ફક્ત પસંદગી સૉર્ટ(sort) લાગુ કરીએ છીએ. તેથી, અમે વત્તા 1 થી n ઓછા 1 ની પસંદગી સૉર્ટ(sort)ને લાગુ કરીએ છીએ અને પછી જ્યારે ફક્ત એક જ ઘટક હોય ત્યારે આ પ્રક્રિયા બંધ થાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:30)

તેથી, જો તમે A થી પ્રારંભ કરવા જઈ રહ્યા છો, તો હું ના બરાબર 1 થી 1 ની બાદબાકી 1, તો પછી આપણે કાંઈ જ નથી, અધિકાર. તો, આનું પુનરાવર્તિત રીતે વિચાર કરવાનો આ જ કારણ છે. એક કારણ એ છે કે, કોડ વાંચવા માટે સહેલું બને છે. તેથી, તમે કહો છો, કે તમે સૉર્ટ(sort) એને પ્રારંભથી n શરૂ કરવા માંગો છો જ્યાં શરૂ થાય છે તે વણગોઠવેલાં સેગમેન્ટની અનુક્રમણિકા જ્યાં તે પ્રારંભ થાય છે અને n એ કુલ કદ છે. તેથી, છેલ્લી સ્થિતિ સંભવતઃ ઓછા 1 ની છે. તેથી, જો પ્રારંભ એ n ઓછા 1 અથવા વધારે છે, તો તમે કંઈપણ કરશો નહીં, તમે હમણાં જ પાછા ફરો, અન્યથા તમને આ સેગમેન્ટમાં શરૂઆતથી નાનું 1 ઘાત, નાનું સૌથી નાનું મૂલ્ય મળે છે. તો, આ તે જ લૂપ છે જે આપણે પહેલા કર્યું હતું. અમે સેગમેન્ટની શરૂઆતની ન્યૂનતમ સ્થિતિથી પ્રારંભ કરીએ છીએ અને ત્યારબાદ અમે આગળ વધતા જતા રહેવું જોઈએ અને જ્યારે પણ આપણે મૂલ્યને ન્યૂનતમ પોઝિશન કરતા નાના હોય ત્યારે શોધીશું, ત્યારે અમે તેને અપડેટ કરીશું અને આ લૂપના અંતમાં આપણે જે સ્થાન મેળવ્યું છે તે બદલીશું પ્રારંભિક સ્થિતિ સાથે તે સ્થિતિ પર મૂલ્ય. તેથી, હવે,

સાચું મૂલ્ય, તેથી અહીંથી કંઈક મૂલ્ય સાચું સ્થાન પર ખસેડ્યું છે અને હવે અમે પુનરાવર્તિત રીતે પોઝિશન શરુઆતથી શરૂ કરીને 1 સુધી n ને સૉર્ટ(sort) કરીએ છીએ. તેથી, શું આ કોઈ જટિલતા છે?
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:33)
તેથી, ચાલો, ચાલો, જમણી, યાદ રાખીએ કે જ્યારે આપણે પુનરાવર્તિત ગણતરીઓ કરી ત્યારે અમે કહ્યું કે, સામાન્ય રીતે, તમે પુનરાવર્તન લખશો, જમણે. તમે એલ્ગોરિધમ્સની જટિલતાને પુનરાવર્તિત રીતે વ્યક્ત કરશો. તો, માની લો કે tn એ લંબાઈ n માં પસંદગી સૉર્ટ(sort) ચલાવવા માટે જરૂરી પગલાંઓની સંખ્યા છે, બરાબર. તેથી, પ્રથમ વસ્તુ એ છે કે, ન્યૂનતમ શોધવા માટે અને તેને શરૂઆતમાં ખસેડવા માટે તેને n પગલાની જરૂર છે અને પછી તેને શરૂઆતમાં ખસેડવામાં આવે છે બાકીનું તે એરે(Array) છે જે n માઈનસ 1 ના કદને સૉર્ટ(sort) કરવા માટે બાકી છે. તેથી, તમારે સમયને વારંવાર ટીન ઓછા 1 ની જરૂર છે, બરાબર. તેથી, તેથી, તમારી પાસે ટી. તેથી, આ લઘુત્તમ શોધવાનો સમય છે. tn એ, tn n વત્તા tn minus 1 છે જે પુનરાવર્તિત પગલું છે. અને અમે કહ્યું કે, જો આપણે કદ 1 ના એરે(Array)ને ગોઠવી રહ્યા છીએ, તો અમારે કંઈપણ કરવાની જરૂર નથી, અમે ફક્ત પાછા ફરો. તેથી, 1 ની ટી 1 છે. તેથી, n નો n એ n plus વત્તા tn minus 1 છે. પરંતુ હવે, જો હું tn minus 1 લે અને તે જ એક્સપ્રેશનનો ઉપયોગ કરીને તેને વિસ્તૃત કરું, તો મને n ઓછા 1 વત્તા tn minus 2 મળશે. હવે, જો હું ટી.ન. ઓછા 2 ને વિસ્તૃત કરું છું, મને એન n minus 2 tn minus 3 મળશે અને તેથી આગળ. અને તેથી આ પુનરાવર્તન માટે જે મળ્યું તે બરાબર વિસ્તૃત થશે, પુનરાવર્તિત અલ્ગોરિધમ માટે, એટલે કે એન પ્લસ એન ઓછા 1 વત્તા એન બાદ 2 ની નીચે, બરાબર. તેથી, આપણે આ એકમમાં જે જોયું છે તે એ છે કે આપણે જોયું છે કે, એક ખૂબ જ પ્રાકૃતિક એલ્ગોરિધમ, જે આપણે લાગુ કરીશું, જો આપણે હાથ દ્વારા વસ્તુઓને પસંદગી સૉર્ટ(sort) તરીકે ઓળખાવી હોય તો તેને પુનરાવર્તન અને વારંવાર બંને ઔપચારિક બનાવવામાં આવે છે અને તે આપણને જટિલતા આપે છે, જે છે ઓર્ડર n નો વર્ગ.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મોડ્યુલ - 04
લેક્ચર - 12
ઈન્સેરશન સૉર્ટ(Insertion Sort)

તેથી, ચાલો સૉર્ટિંગ(sorting) ની ચર્ચા સાથે ચાલુ રાખો, અને અન્ય સરળ સોર્ટિંગ એલ્ગોરિધમ જુઓ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:06)
તેથી, આપણે તે પહેલાં કહ્યું હતું કે સૉર્ટિંગ(sorting) માટે વધુ પ્રેરણા છે; શોધ કરવાથી, ડુપ્લિકેટ્સને દૂર કરવા, કેટલાક આંકડાકીય ગુણધર્મોની ગણતરી કરવા, જેમ કે આવર્તન કોષ્ટકો, વગેરે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:18)
અને આપણા હાથમાં જે ઉદાહરણ છે, તે એક રીત છે કે આપણે ગુણના ઉતરતા ક્રમના પરીક્ષા કાગળોનો સમૂહ ગોઠવવા માટે કહીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:28)
તેથી, પરીક્ષાના કાગળોના આ સમૂહને સૉર્ટ(sort) કરવાની બીજી વ્યૂહરચના નીચે મુજબ હશે. તેથી, તમે આ સ્ટેકમાં ટોચનું સૌથી વધુ પેપર લઈ લો છો અને તમારી પાસે એક નવું સ્ટેક બનાવવું છે. હવે તમે બીજા કાગળનો ઉપયોગ કરો અને તેને પ્રથમ કાગળ સાથે સરખાવો. જો માર્ક મોટું હોય, તો તમે તેને વિશે મુકો, કારણ કે તમે તેને ઉતરતા ક્રમમાં ઈચ્છતા હતા. જો માર્ક નાનું હોય, તો તમે તેને નીચે મુકો. તેથી, આ પગલા પછી, તમારી પાસે નીચે આવતા ક્રમમાં કાગળના બે સ્ટેક્સ છે. હવે બે પેપરોની સ્ટેક. અને હવે તમે ત્રીજો પેપર લો, અને હવે તમે જુઓ કે તે પહેલા બે સંબંધમાં ક્યાં બંધબેસે છે; કાં તો તે તળિયે જાય છે અથવા તે બે અથવા બધી રીતે ચાલે છે. આ રીતે દરેક બિંદુએ તમે ક્રમમાં ગોઠવેલ સ્ટેકમાં ટોચનું સૌથી વધુ કાગળ પસંદ કરો છો અને તમે તેને બિલ્ડ અપ કરેલ સૉર્ટ(sort) કરેલ સ્ટેકમાં સાચું સ્થાન શામેલ કર્યું છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:18)
તો ચાલો જોઈએ કે કેવી રીતે કામ કરવું જોઈએ. તેથી, માનીએ છીએ કે અમારી પાસે ક્રમમાં ગોઠવાયેલ ઘટકોનો રોલ્ડ એરે(Array) છે. તેથી, આપણે જે કરીએ છીએ તે છે, આપણે 74 ની પહેલી તત્વ સાથે પ્રારંભ કરીએ છીએ. તો, આપણે 74 લઈશું, અને આપણે એક નવું સ્ટેક શરૂ કરીશું. હવે આપણે લેવાની જરૂર છે. હવે આ બિંદુએ સૌથી વધુ ઘટકો હવે 32 છે, આ કિસ્સામાં ડાબી બાજુ સૌથી વધુ છે. તેથી, હવે આપણે 32 લેવું પડશે, અને તે નાના હોવાથી 74 ની ડાબી બાજુએ જવું જોઈએ. તો, આપણે આ કરીએ, તેથી હવે આપણે આ એરે(Array) મેળવીશું. હવે આપણે આગળના તત્વ તરફ જોશું; એટલે કે 89. અને ફરી એક વખત તે આ બંનેની આદર સાથે યોગ્ય સ્થિતિ સાથે જવું જોઈએ, તેથી તે 74 ની જમણી બાજુએ જવું જોઈએ, તેથી આપણને આ એરે(Array) મળે છે. હવે આપણે 55 લઈએ, અને તે શોધવાનું છે કે તે ક્યાં જાય છે. તેથી આપણે અંતથી શરૂ કરી શકીએ અને કહી શકીએ કે તે અહીં નથી, તે ડાબી બાજુએ જવું જોઈએ. પછી આપણે છેલ્લે, શોધી શકીએ કે તે જવા માટે આ એક સાચી જગ્યા છે. તેથી, આ દાખલ પગલું છે. અમે દરેક તત્વ લઈએ છીએ, અને આપણે તે બિંદુએ નીચે જતાએ છીએ જ્યાં આપણે શામેલ કરવા માંગીએ છીએ. તેથી, 55 32 અને 74 ની વચ્ચે આવે છે. હવે તે 21 છે જ્યાં. ઠીક છે જો આપણે તેને શામેલ કરવાનો પ્રયાસ કરીએ તો, તે ચાલુ થાય છે, તે પ્રારંભ કરવા માટે તમામ રીતે જવું પડશે, કારણ કે તે પછી તે બધું નાનું છે, જે અત્યાર સુધી બધું છે. તેથી, આ આગલું પગલું છે. અને છેલ્લે, જ્યારે આપણે 64 કરીએ, તે 55 અને 74 ની વચ્ચે આવશે. તેથી, દરેક પગલા પર આપણે આગલા તત્વને પસંદ કરીએ છીએ, અને આપણે તેને પહેલાથી સૉર્ટ(sort) કરેલા સેગમેન્ટ્સમાં શામેલ કરીએ છીએ જે આપણે પહેલાના બધા તત્વો સાથે બનાવી છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:42)

તેથી, આપણે એક તત્વ સાથે સૉર્ટ(sort) કરેલ ક્રમ બનાવવાની શરૂઆત કરીએ છીએ. તેથી, આને ઈન્સેરશન સૉર્ટ(Insertion Sort) કહેવામાં આવે છે. અને અમે દરેક વણગોઠવેલ ઘટકને પહેલાથી વણગોઠવેલ અનુક્રમમાં યોગ્ય સ્થાને શામેલ કરીએ છીએ. તેથી, તે દાખલ કરવાના પગલાને કારણે છે, કે દરેક તત્વ તેના સાચા સ્થાને શામેલ છે. આ ઈન્સેરશન સૉર્ટ(Insertion Sort) કહેવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:05)

તેથી, આ ફરીથી એકદમ સહેલું આગળ વધવું અમલીકરણ છે. તો, શું થશે, આપણી પાસે પ્રારંભિક એરે(Array) છે, અને 0 થી n ની 1 ઘાત છે. તેથી, શું કરશે; અલબત્ત, શરૂઆતમાં આપણે કશું જ નથી. તેથી, આપણે ધારી શકતા નથી કે આપણે ખરેખર સ્થિતિ સાથે પ્રારંભ કરીએ છીએ, તેથી, આપણે પોઝિશન 1 થી પ્રારંભ કરીએ છીએ, અને પછી આપણે પાછળ તરફ જોવું જોઈએ. તેથી, આપણે વર્તમાન સ્થિતિથી પ્રારંભ કરીએ છીએ, અને અમે પાછળથી કામ કરીએ છીએ, અને જ્યાં સુધી આપણે જે મૂલ્ય જોઈ રહ્યા છીએ તે મૂલ્ય તેના ડાબેથી મૂલ્ય કરતાં નાના છે. અમે આગળ વધી રહ્યા છીએ. તો, આપણે ધારી લીધું છે કે આપણી પાસે આ અદલાબદલ ઑપરેશન છે, જે મૂળભૂત રીતે બે પોઝિશન લે છે, એક એરે(Array) અને તેમને વિનિમય કરે છે. તો, આગળનાં વિકલ્પોને આગળની બાજુની બાદબાકી 1, અદલાબદલ કરવાનો અર્થ એ છે કે મૂલ્યને આગળના મૂલ્યમાં લઈ જાઓ, આગળના શુલ્કનું મૂલ્ય 1 લો અને તેમને અદલાબદલ કરો. જેમ આપણે શરૂઆતમાં વિસ્તૃત કરીએ છીએ, આપણે ધારી શકીએ છીએ કે આવી વસ્તુઓ એ આપણા માટે ઍલ્ગોરિધમને વ્યક્ત કરવા માટે અનુકૂળ કામગીરી છે, કારણ કે આ ફક્ત અમારે એકંદર કાર્યવાહીની સંખ્યા માટે સતત પરિબળ ઉમેરે છે, જેને આપણે અવગણી શકીએ છીએ એક

((સમયનોસંદર્ભ લો: 04:06))

જટિલતા. તેથી, આપણે દરેક ઘટકને ડાબી બાજુ તત્વ સાથે અદલાબદલ કરીએ છીએ, તેટલું લાંબો છે, તે ડાબી બાજુનો તત્વ છે, અને જ્યારે આપણે તે સ્થાને આવીએ છીએ જ્યાં ડાબી બાજુનું મૂલ્ય, વર્તમાન મૂલ્યની બરાબર કરતા વધારે છે. , આ બિંદુએ આપણે બંધ. તેથી, આ તે મૂલ્ય લાવે છે જેની સાથે આપણે પ્રારંભ કર્યું છે, તે સાચું સ્થાન છે. તેથી, સામાન્ય રીતે, જો આપણે હોય તો; હું સરખી રીતે કહું છું અને પછી હું પાછળથી ચાલવાનું શરૂ કરું છું. ત્યારબાદ જ્યાં સુધી હું 1 ઓછા કરતાં નાના હોય, હું બદલામાં આવીશ. જ્યાં સુધી હું પોઝિશન સુધી પહોંચું નહીં ત્યાં સુધી હું વિનિમય રાખું છું, જ્યાં ડાબી બાજુની કિંમતો, આ મૂલ્ય કરતાં નાની હોય છે અને હું રોકાઈશ. તો, આ મૂળભૂત લૂપ છે, અને હું આ બધા ઘટકો માટે કરું છું. તેથી, દરેક તત્વ માટે તેને દાખલ કરવું પડશે. તેથી, પ્રથમ વખત મને તેને નાના સેગમેન્ટમાં દાખલ કરવું પડશે. જેમ હું સાથે જાઉં છું, તે સેગમેન્ટ્સને તેમાં શામેલ કરવું પડશે, તે લાંબી અને લાંબી થઈ જશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:54)

તો, આપણે જોઈ શકીએ છીએ કે આ કેવી રીતે કાર્ય કરે છે, આપેલ એરે(Array) પર. તો, આ સેગમેન્ટને જોઈને, સૌપ્રથમ આપણે શરૂઆત કરીએ છીએ. તેથી, આ મારી પ્રારંભિક ગોઠવણી છે. તેથી, આ ક્રમમાં ગોઠવવામાં આવે છે, અને આ ક્રમાંકિત નથી. તેથી, હવે, હું 32 ને તરત જ જોઉં, તે 74 કરતા મોટો છે, તેથી હું વિનિમય કરીશ, અને પછી તે શરૂઆતમાં પહોંચશે. તેથી, તે લૂપની શરતમાંની એક, એ છે કે જો હું લૂપની શરૂઆતમાં પહોંચું, તો આગળનો ભાગ 0 ની બરાબર છે, હું પણ બંધ કરીશ, જો મને ડાબી બાજુએ સૌથી વધુ સ્થાન મળે તો loops બંધ થાય છે. તેથી, તે કર્યા પછી, મારી પાસે આ છે. તેથી, હવે હું આમાં 89 શામેલ કરવાનો પ્રયત્ન કરું છું. તેથી, મને લાગે છે કે 89 પહેલાથી 74 કરતા વધારે છે, કશું કરવાનું નથી. તેથી, પ્રથમ નોનપ્રિવિયલ પગલું જે થાય છે તે 55 વડે છે. તેથી, જ્યારે હું 55 કરું છું, હું તેની સરખામણી 89 થી કરું છું. મને લાગે છે કે 89 એ 55 કરતા મોટો છે, હું તેમને બદલીશ. હવે હું 74 થી એક વખત તત્વ સાથે 55 ની સરખામણી કરીશ. અને પછી ફરીથી તે ખોટો રસ્તો છે, તેથી હું બદલામાં આવીશ. છેવટે, જોયું કે 55 હવે 32 કરતા વધારે છે, હું રોકાઈશ. તેથી, હવે આ જગ્યાનો અંત છે. હવે હું આગળના ઘટકને જોઈશ, જે 21 છે. તેથી, હું 21 તરફ જોઉં છું. તો, 21 પછી વિનિમય થશે 21 પછી વિનિમય કરશે, તેથી તે ડાબી બાજુથી બધી રીતે વિનિમય કરશે, તેથી હું 21 જે આદાનપ્રદાન કરે છે 74. પછી હું 21 સાથે 55 નું વિનિમય કરીશ. પછી હું 21 નું વિનિમય કરીશ અને હવે ફરી એકવાર હું રોકાઈશ, કારણ કે દરેક (()) મોટા ભાગની સ્થિતિ છોડી દે છે, ત્યાં ડાબે કંઈ નથી. અને છેલ્લા રાઉન્ડમાં હું 64 લઈશ. તેથી, હવે, આ ભાગ સૉર્ટ(sort) થાય છે. તેથી, હું 64 લઈશ અને તેને અહીં દાખલ કરવાનો પ્રયાસ કરીશ, તેથી તે 89

સાથે અદલાબદલ કરશે. તે 74 સાથે અદલાબદલ થશે અને પછી બંધ કરશે. આ કેવી રીતે ઈન્સેરશન સૉર્ટ(Insertion Sort) કામ કરે છે. આ એક ખૂબ જ સાહજિક સૉર્ટ(sort) છે. જો તમે કાર્ડનો પેક લો અને તેને સૉર્ટ(sort) કરવાનો પ્રયાસ કરો, તો સામાન્ય રીતે આ તમે કેવી રીતે સૉર્ટ(sort) કરશો.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:59)

તેથી, જો તમે વિશ્લેષણને જુઓ છો, તો તે સીલ જેવું જ છેકેપ્શન સૉર્ટ(sort) કે જે આપણે પહેલા સૉર્ટ(sort) કરીએ છીએ. તેથી, મૂલ્ય શામેલ કરવાનો અર્થ એ છે કે આપણે ખૂબ જ અંત સુધી તે સેગમેન્ટમાં જવું પડશે. હવે અલબત્ત, કોઈએ દલીલ કરી હતી કે દાખલ કરવાની સ્થિતિ શોધવા માટે, તમે દ્વિસંગી શોધનો ઉપયોગ કરી શકો છો. આપણને એક સમયે એક તત્વની જરૂર નથી. જો તમારે ખરેખર તે જગ્યાની શોધ કરવી હોય, જ્યાં તેને જવું જોઈએ, તો આપણે બાઈનરી શોધનો ઉપયોગ કરી શકીએ છીએ, પરંતુ જો તમને લોગરિધમિક સમયમાં તે જગ્યાની જરૂર હોય તો પણ તે સ્થાન મળશે. તમારે બધા તત્વોને બદલવાની જરૂર છે, અને તે ખરેખર રેખીય સમય લે છે. તમારે વાસ્તવમાં આ જગ્યા બનાવવાની જરૂર છે. તેથી ખરાબ કિસ્સામાં, તમારે તેને ડાબી બાજુની સૌથી વધુ સ્થિતિમાં મુકવાની જરૂર છે. તેથી, બધા કે તત્વો જે પહેલેથી જ છે ત્યાં જ 1 થી જમણે ખસેડવું જોઈએ, અને તે કેસ પગલાં લેશે. તેથી, દ્વિસંગી શોધ, જો કે તે સ્થાનને વધુ ઝડપથી શોધવામાં મદદ કરી શકે છે, વાસ્તવમાં કોઈપણ ઝડપથી શામેલ કરવા માટે અમને ખરેખર સહાય કરતું નથી. તેથી, સેગમેન્ટ માટે રેખાંકનનો સમય દાખલ કરે છે, અને આ સેગમેન્ટ વધતો જ રહે છે. શરૂઆતમાં હું 1 ને કદ 1 ના સેગમેન્ટમાં દાખલ કરું છું, પછી 2 ને કદ 2 ના સેગમેન્ટમાં અને બીજું. તેથી, મારી પાસે 1 વત્તા 2 વત્તા 3 અપ ટુ માઈનસ 1 છે; છેવટે, આ સેગમેન્ટમાં 1 ઓછા 2 નો સમાવેશ કરવો આવશ્યક છે, 0 ની એક અવતરણ 2, જે લંબાઈ n માઈનસ 1 છે. તેથી, ફરીથી મારી પાસે n એ 1 વત્તા 2 થી n ઓછા 1 છે, અને આ માત્ર બદલાવ છે આ સારાંશ આપણે ઘણી વખત પહેલા જોયું છે, n એ n minus 1 થી 2 છે. તો, આ ફરીથી ઓર્ડર n ચોરસ શોધ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:16)

તો, ફરી એક વાર આપણે પસંદગી સૉર્ટ(sort) માટે જોયું. પુનરાવર્તિત વસ્તુ તરીકે ઈન્સેરશન સૉર્ટ(Insertion Sort)નો વિચાર કરવાની એક કુદરતી રીત છે. અમે એરે(Array)ના ભાગને સૉર્ટ(sort) કરીએ છીએ, અને પછી તેને વધવા માટે તેમાં એક તત્વ શામેલ કરીએ છીએ. તેથી, આ કિસ્સામાં આપણે એરે(Array)ને ઘટકોમાં હોવાનું વિચારીએ છીએ. તેથી, અમારી પાસે એક સૉર્ટ(sort) કરેલો ભાગ છે, અને કોઈ રદ કરેલ ભાગ છે. તેથી, આપણે શું કરીએ છીએ, આપણે અહીં પ્રથમ તત્વ લઈએ અને તેમાં શામેલ કરીએ છીએ અને પછી અમે બાકીના ભાગમાં બાકીના ભાગમાં ઍલ્ગોરિધમનો ઉપયોગ કરીએ છીએ. તેથી, જો 0 થી હું ઓછા 1 થાય. તો, આ સ્થિતિ હું છે અને આ સ્થિતિ હું ઓછા 1 છે. તેથી, હું ઓછા 1 એ છેલ્લી ક્રમાંકિત સ્થિતિ છે, હું પહેલી ક્રમાંકિત સ્થિતિ છે. પછી અમે ગોઠવેલ ભાગમાં Ai શામેલ કરીએ છીએ. અને પછી આપણે વારંવાર એઆઈ પ્લસ 1 ને સૉર્ટ(sort) કર્યું. અને ફરી એકવાર જ્યારે હું ખરેખર અહીં, minus 1 પર, તો આપણે કંઈપણ કરવાની જરૂર નથી, તેથી આપણે ફક્ત ત્રણ વાર પાછા આવી શકીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 09:17)

તો, હવે આપણી પાસે બે ભાગોમાં પુનરાવર્તિત ફોર્મ્યુલેશન છે. તેથી, આપણી પાસે ઈન્સેરશન સૉર્ટ(Insertion Sort) છે, જે વણગોઠવેલ સેગમેન્ટને શરૂઆતથી n ઓછા 1 સુધી સૉર્ટ(sort) કરે છે. તેથી, જો પ્રારંભ પહેલેથી જ n ઓછા 1 વળતર પર છે; અન્યથા સ્થિતિના મૂલ્યને બાકીના ભાગમાં શામેલ કરો, જે નીચે જોશે તો તમે જોશો. અને પછી બાકીના એરે(Array)ને સૉર્ટ(sort) પ્લસ 1 થી પાછળથી સૉર્ટ(sort) કરો. તેથી, શામેલ છે શું? વેલ તે શરૂ થાય છે જો પોઝિશન શરૂ થાય છે અને તેને 1 ની શરૂઆત કરવા માટે સેગમેન્ટ 0 માં શામેલ કરવાનો પ્રયાસ કરે છે. તો, તે પાછલા ભાગમાં બરાબર કાર્ય કરે છે જેમ આપણે તેને પુનરાવર્તિત વસ્તુમાં કર્યું છે. તે પ્રથમ સ્થાને શોધે છે કે ડાબી બાજુનું મૂલ્ય ઓછામાં ઓછું નાનું છે, જે હાલમાં મૂલ્ય શોધી રહ્યું છે અને ત્યાં અટકે છે. તેથી, આ શામેલ મૂળભૂત રીતે ઈટેરેટ લૂપનું શરીર શું છે, પરંતુ બાહ્ય લૂપ શામેલ કરો, અમે તેને પુનરાવર્તિત કરીએ છીએ. તેથી, આ કેટલો સમય લે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:10)

ફરી એકવાર તે તમને વારંવાર એલ્ગોરિધમ્સ તરફ જોવામાં અને વિશ્લેષણને લખવામાં પ્રેક્ટિસ આપે છે. તેથી, જ્યારે પણ આપણી પાસે વારંવાર અલ્ગોરિધમનો હોય ત્યારે આપણે પુનરાવર્તન લખીએ છીએ; એટલે કે, આપણે t ની નાનાં મૂલ્યોના

સંદર્ભમાં n ના ફોર્મ્યુલેશન લખીએ છીએ. તેથી, ચાલો આ આક્રમક અલ્ગોરિધમનો વિશ્લેષણ કરવાનો પ્રયાસ કરીએ. તેથી, આપણે એલ્ગોરિધમ બનાવવાની રીતથી થોડું અલગ રીતે જોઈશું, તેનું વિશ્લેષણ કરીશું. તેથી, જો તમે 0 ને 1 ઓછા કરવા માંગો છો, તો આપણે જે કરીએ છીએ તે છે, આપણે આ સેગમેન્ટને વારંવાર સૉર્ટ(sort) કરી રહ્યા છીએ અને પછી આ મૂલ્ય શામેલ કરી રહ્યા છીએ. તેથી, સમગ્ર વસ્તુને સૉર્ટ(sort) કરવા માટે સમય લાગે છે. પ્રથમ ભાગને n ઓછા 2 સુધી સૉર્ટ(sort) કરવા માટે, અને પછી n ઓછા 1 પગલું શામેલ કરવા માટેનો સમય આ ટ્રાક ટન ટન 1 લે છે. તેથી, આપણે સમાન પુનરાવર્તન કરીએ છીએ જેમ આપણે પસંદગી સૉર્ટ(sort) ટી માટે n કર્યું છે n ના t એ n એ અવકાશ જગ્યા છે, અને પ્લસ t ની n માઇનસ 1 જે recursive phase છે. અને જો આપણે આ પહેલા વિસ્તૃત કર્યું છે જેમ આપણે પહેલા કર્યું હતું, તો આપણને n ઓછા 1 વત્તા n ઓછા 2 ની નીચે n મળશે, જે n માં minus 1 by 2 થાય છે, જે ઓર્ડર n ચોરસ છે. તેથી ફરીથી, પુનરાવર્તિત અને પુનરાવર્તિત સંસ્કરણ માટે સમય વચ્ચે કોઈ જુદો જ નથી; સિવાય કે સામાન્ય રીતે ધ્યાનમાં રાખવું જોઈએ કે, પુનરાવર્તિત કૉલ્સ વધુ વિસ્તૃત પછી પુનરાવર્તિત આંટીઓ છે. અમે થોડા સમય પછી આ મુદ્દા પર પાછા આવીશું, પરંતુ અન્યથા જો આપણે ફંક્શનને બોલાવતા પુનરાવર્તિત કોલ્સને એક મૂળભૂત ઓપરેશન ગણતા હોઈએ, તો પુનરાવર્તિત અને પુનરાવર્તિત સંસ્કરણ વચ્ચે કોઈ તફાવત નથી. પુનરાવર્તિત સંસ્કરણ કેટલીક વખત વિભાવનાપૂર્ણ રીતે સમજવા અને કોડ કરવા માટે સરળ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:46)
તેથી, આપણે જે જોયું છે એ છે કે, બે પ્રાકૃતિક એલ્ગોરિધમ્સ કે જે આપણે જાતે જ કંઈક કરીએ ત્યારે સામાન્ય રીતે લાગુ કરીશું; પસંદગી સૉર્ટ(sort) અને ઈન્સેરશન સૉર્ટ(Insertion Sort) બંને ઑર્ડર n ચોરસ છે. ત્યાં એક બીજું ઍલ્ગોરિધમ છે જે તમે પાર કરી શકો છો, જેમાં આપણે બબલ સૉર્ટ(sort) તરીકે ઓળખાતા આ કોર્સમાં ચર્ચા કરીશું નહીં, જે કુદરતી રીતે અમે કરીશું તેવું કંઈક નહીં, પરંતુ તે કમ્પ્યુટર પરની વસ્તુઓ કેવી રીતે કરવું તેનું વર્ણન કરવાની એક પ્રિય રીત છે. તેથી, બબલ સૉર્ટ(sort) મૂળભૂત રીતે આ નિવેશ પ્રકારની સ્વૅપિંગ કરે છે, સિવાય કે તે તમે જે કરવા માંગો છો તેના આધારે મહત્તમ અથવા ન્યૂનતમ મૂલ્ય લે છે. કહો કે તે મહત્તમ મૂલ્ય લે છે અને તેને એરે(Array)ના એક અંતમાં ખસેડે છે. તેથી, પછી તમને પસંદગી સૉર્ટ(sort) ગમે છે, એક અંતમાં સૌથી મોટું મૂલ્ય. પછી તમે શરૂઆતમાં પાછા જાઓ છો, અને ફરીથી તમે એડજસ્ટિંગ મૂલ્યો તરફ ધ્યાન રાખો છો, અને બીજું સૌથી મોટું મૂલ્ય બીજા ક્રમની સૌથી મોટી સ્થાને લે છે. તેથી, બબલ સૉર્ટ(sort) પણ એન સ્ક્વેર છે, પરંતુ આપણે એન ચોરસ એલ્ગોરિધમ્સ જોયું છે તે મોટા મૂલ્યો માટે ખરેખર શક્ય નથી. તેથી, જો આપણી પાસે લગભગ 10000 ની ઉપર હોય, તો ક્યાંક 10 થી 4 ની વચ્ચે 10 થી 5 ની વચ્ચે, ત્યારબાદ n ચોરસ અમલમાં મૂકવા માટે કેટલાક સો સેકંડનો સમય લેશે, અને તેથી, આ વિશાળ માહિતીના ડેટા માટે ઉપયોગી બનશે નહીં . પરંતુ આ કહેવાનું નથી કે આ અલ્ગોરિધમ બધા સમાન ખરાબ છે. ખાસ કરીને તે તારણ આપે છે કે જો વાસ્તવમાં પ્રાયોગિક પુરાવા જોવા મળે છે, તો એન ઈન્સેરશન સૉર્ટ(Insertion Sort) સામાન્ય રીતે પસંદગી સૉર્ટ(sort) કરતા વધુ સારી રીતે વર્તે છે, અને તે બબલ સૉર્ટ(sort) કરતા બન્ને સારા છે. અને શામેલ શામેલ શામેલ શા માટે વર્તે છે તે વિશે વિચારો, જ્યારે કલ્પના કરો કે અમે પહેલેથી સૉર્ટ(sort) કરેલા સૂચિમાં શામેલ સૉર્ટ(sort) લાગુ કરીએ ત્યારે શું થાય છે. જ્યારે પણ આપણે પહેલાથી ક્રમમાં ગોઠવેલ સૂચિમાં કંઈક શામેલ કરવા માંગીએ છીએ, ત્યારે અમે તે શોધીશું કે તે પહેલાથી જ તે સાચું સ્થાન છે. તેથી, સાંકડુ, જે શામેલ તબક્કો છે, તે સાંકડુ છે. શામેલ ચલેરો મૂળભૂત રીતે એક તુલના પછી સમાપ્ત થશે. તેથી, દાખલ કરેલ સૉર્ટ(sort) આવશ્યક રૂપે રેખીય સમય બનશે, જો તમે પહેલાથી સૉર્ટ(sort) કરેલ અરજી પર લાગુ કર્યું છે. તેથી, આ ઘણી પરિસ્થિતિઓ છે; ઈન્સેરશન સૉર્ટ(Insertion Sort) વાસ્તવમાં અન્ય ઑર્ડર n સ્ક્વેર પ્રકારની તુલનામાં થોડું સારું વર્તન કરે છે, પરંતુ સામાન્ય ક્રમમાં એન સ્ક્વેર સૉર્ટિંગ(sorting) એલ્ગોરિધમ્સ સ્વીકાર્ય નથી અને અમે ડેટાને સૉર્ટ(sort) કરવાના દૂરસ્થ હોંશિયાર રસ્તાઓ પર લૂપ કરવા માંગીએ છીએ, કારણ કે મોટા પ્રમાણમાં ડેટાને સૉર્ટ(sort) કરવા માટે આ અવ્યવહારુ છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મૉડ્યૂલ - 05
લેક્ચર - 13
મર્જ સૉર્ટ (Merge Sort)

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:06)

તેથી, અમે બે અંતર્જ્ઞાન સૉર્ટિંગ અલ્ગોરિધમ્સ જોયા છે; પસંદગી સૉર્ટ(selection sort) અને નિવેશ સૉર્ટ(Insertion sort), પરંતુ કમનસીબે અમારા માટે, પછી બંને ઓર્ડર n સ્ક્વેર હોઈ ચાલુ થાય છે. અને આપણે જાણીએ છીએ કે ઓર્ડર n ચોરસ એ મોટી એરે(Array)ને સૉર્ટ કરવા માટે ખરેખર સારી નથી. તેથી, આપણે તેના બદલે શું કરી શકીએ?

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:18)

તેથી, અરે(Array)ને વધુ અસરકારક રીતે સૉર્ટ કરવા માટે અહીં એક રીત છે. તો, માની લો કે આપણે એરે(Array)ને બે સમાન ભાગોમાં વિભાજીત કરીએ છીએ. તો, આપણે મધ્યમાં ફક્ત કૌંસ જ જોઈએ, આપણે ડાબેથી જુદા જુદા અને જમણે જુદા જુદા જોઈએ છીએ. હવે ધારો કે આપણે સ્વતંત્ર રીતે સૉર્ટ થયેલ છિદ્રમાં ડાબે અને જમણે સૉર્ટ કરી શકીએ છીએ. તો, આપણી પાસે ડાબી અડધી સૉર્ટ કરેલ છે, જમણી અડધી સૉર્ટ કરેલ છે. હવે જો સંપૂર્ણ રીતે સૉર્ટ કરેલ એરે(Array) મેળવવા માટે બે ભાગોને એકીકૃત રીતે જોડવાનો રસ્તો છે, તો આપણે કહી શકીએ છીએ કે અમે તેને સૉર્ટિંગને બે નાની પેટા સમસ્યાઓમાં તોડીને પરિણામને સંયોજન કર્યું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:54)

તેથી, ચાલો પહેલા આ સંયોજન પગલું જોઈએ. તેથી, અમને બે સૉર્ટ કરેલી સૂચિ આપવામાં આવી છે a અને b અથવા બે સૉર્ટ કરેલ એરે(Array) એ અને બી છે, અને અમે તેને એક સૉર્ટ કરેલી સૂચિમાં ભેગા કરવા માંગીએ છીએ. તેથી, આ કંઈક છે જે આપણે સરળતાથી કરી શકીએ છીએ. ધારો કે અમારી પાસે બે સ્ટેક(stack) કાર્ડ છે. તેમાંથી દરેક એક જ રીતે ચઢતા ક્રમમાં છે. પછી આપણે શું કરીશું તે છે કે આપણે સૌથી વધુ કાર્ડ્સને સ્ટેક(stack)ની જરૂર છે, અને બંનેને નાનાં લો અને તેને નવા સ્ટેક(stack)માં ખસેડો. અને આપણે આ કરવાનું ચાલુ રાખી શકીએ, ત્યાં સુધી આપણે બધા તત્વોને નવા ક્રમમાં ગોઠવેલ સ્ટેક(stack)માં ખસેડીશું. તેથી, આ એક અંતર્ગત વિલીનીકરણ છે. તેથી, મારી પાસે બે સ્ટેક(stack) છે. તેથી, મારી પાસે બે સ્ટેક(stack) કાર્ડ છે, જમણી બાજુ. તેથી, હું દરેકમાં સૌથી વધુ ટોચના કાર્ડને જોશો, અને પછી તેમાંથી એક નવું સ્ટેક(stack) જશે. અને પછી આ ગયો અને તેથી હવે હું આગળની વેલ્યુ પર ધ્યાન આપીશ, અને તેનાથી તેની તુલના કરીશ, અને પછી આમાંથી એક અહીં પણ આવશે. તેથી, અંતે હું સૉર્ટ કરેલ સૂચિની નવી સ્ટેક(stack) બનાવી શકું છું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:53)

તેથી, ચાલો આપણે આને એક ખૂબ સરળ ઉદાહરણમાં જોઈએ. તેથી, ધારો કે તમે બે સૉર્ટ કરેલ સૂચિને મર્જ કરવા માંગો છો. તેથી, આ ચઢતા ક્રમમાં સૉર્ટ કરવામાં આવે છે અને તેથી આ ચડતા ક્રમમાં ગોઠવવામાં આવે છે. તેથી, પ્રથમ પગલું, ટોચની તરફ્ જોવું છે. તેથી, ચાલો ધારે છે કે ટોચની દ્રષ્ટિએ આ જેવા લખેલા છે. તેથી, મારી પાસે આ સ્ટેક(stack) છે, અને મારી પાસે આ સ્ટેક(stack) છે. તેથી, હું ટોચની સૌથી વધુ ઘટકોને જોઉં છું, અને પછી હું કહું છું કે બંનેના નાનાં નાના મારા નવા સ્ટેક(stack)નો સૌથી મોટો ભાગ હોવો આવશ્યક છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:23)

તેથી, આ કિસ્સામાં હું 21 આઉટ કરીશ, અને હું તેને અહીં ખસેડીશ. હવે મને સરખાવવું પડશે, બે સ્ટેક(stack) માં સૌથી વધુ ઘટકો શું છે; 32 અને 55 હોઈ શકે છે. તેથી, હવે, હું આ નાના 2 ને 32 વડે સરખાવીશ અને તેને ખસેડીશ. હવે હું 55 અને 74 ની સરખામણી કરું છું, અને તેથી 55 ચાલે છે, અને હું 74 અને 64 ની સરખામણી કરું છું. તેથી, હવે 64 ની બહાર નીકળો. હવે મારી પાસે બીજા સ્ટેક(stack)માં કંઈ બાકી નથી, તેથી પ્રથમ સ્ટેક(stack) સૉર્ટ કરેલ ક્રમમાં છે, તેથી હું તેને ક્રમમાં કૉપિ કરીશ. તેથી, પ્રથમ હું 74 ને ખસેડે છે અને પછી હું ખસેડીશ, આ ખૂબ જ સરળતાથી સમાવિષ્ટ વસ્તુ

છે કે જ્યારે આપણે ભૌતિક રૂપે બે સૉર્ટ કરેલી સૂચિ સાથે વ્યવહાર કરીએ છીએ ત્યારે અમે ફરીથી કુદરતી રીતે કામ કરીએ છીએ, અને અમે તેને સામાન્ય એરે(Array) સાથે પણ કરી શકીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:05)
તેથી, હવે, આપણે તેનો ઉપયોગ કેવી રીતે કરવા માટે કરીએ છીએ. જેમ આપણે કહ્યું તેમ અમારું લક્ષ્ય સમસ્યાને બે સમાન પેટા સમસ્યાઓમાં તોડવું છે. પેટા સમસ્યાઓનું નિરાકરણ કરો અને અંતિમ ઉકેલમાં બે સોલ્યુશનને મર્જ કરો. તો, આપણે તેને 0 થી n ને 2 થી 2 ને 2 n ઓછા 1 ને અલગ કરીશું. તો, આપણી પાસે 0 થી n ઓછા 1 ની સૂચકાંક છે. તેથી, આપણે 2 ઓછા 1 અને n એ 2 ની મધ્યબિંદુ તરીકે લઈએ છીએ. તેથી, આપણે આ અલગ રીતે સોર્ટ કરીએ છીએ, આપણે આ અલગ રીતે સોર્ટ કરીએ છીએ, અને પછી આપણે તેને મર્જ કરીએ છીએ. તેથી, આ અમારી પાસે જે વ્યૂહરચના છે, અને આ તે છે કારણ કે અંતિમ પગલું બે ઉકેલોને મર્જ કરે છે. આ એકદમ કુદરતી રીતે મર્જ સૉર્ટ છે. હવે મેં કહ્યું છે કે આપણે આ વસ્તુને બે પેટા સમસ્યાઓમાં ભાંગીશું. તો, હું આ કેવી રીતે ઉકેલું? સારુ હું વારંવાર એક જ વસ્તુ કરીશ. હું આ બેને બે પેટા સમસ્યાઓમાં ભરીશ, અને હું આને મર્જ કરીશ. હું આને બે પેટા સમસ્યાઓ સાથે ભંગ કરીશ અને આને મર્જ કરીશ. તેથી, હું સબ સમસ્યામાં વસ્તુને ભંગ કરું છું, જ્યાં સુધી તમે સબ સમસ્યા પર ન પહોંચો, જે આપણે જોયું છે, જે 1 નું કદ છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 04:10)
તો ચાલો આગળ વધતા પહેલાં આપણે એક ઉદાહરણ જોઈએ છીએ. તેથી, અહીં એરે(Array)નું ઉદાહરણ છે જે આપણે સૉર્ટ કરવા માંગીએ છીએ. તેથી, પ્રથમ પગલું તેને બે ભાગોમાં તોડવો છે. તેથી, આપણે ડાબે ભાગ લઈએ છીએ. તેથી, આ એક મધ્યબિંદુ છે. તેથી, આપણે ડાબે ભાગ લઈએ છીએ, જે પ્રથમ ચાર તત્વો છે, અને જમણી બાજુ. અને હવે આપણે આ ભાગમાં મર્જ સૉર્ટને અલગથી લાગુ કરીશું; છેવટે, આપણે જવાબને મર્જ કરીશું. તેથી, ડાબે ભાગમાં મર્જ સૉર્ટ લાગુ કર્યા પછી, તમારે તેને ફરીથી બે ભાગમાં ભંગ કરવું આવશ્યક છે. હું આ બિંદુને બે ભાગોમાં વિભાજીત કરવા માટે લઈશ. અને જમણી બાજુએ, મારે આ બિંદુ લેવી પડશે, અને બે ભાગમાં વહેંચવું પડશે. હવે આપણે કહી શકીએ કે આપણે સરળતાથી કદ બે એરે(Array) કરી શકીએ છીએ, પરંતુ ચાલો આપણે બેઝ કેસ સુધી પહોંચ્યા ત્યાં જ ચાલીએ. બેઝ કેસો જ્યારે આપણી પાસે માત્ર એક જ તત્વ છે અને કોઈ સૉર્ટિંગ આવશ્યક નથી. અમે આમાંના દરેકને બે ભાગમાં વિભાજીત કરીશું. તેથી, નોંધ લો કે સગવડ માટે આપણે કંઈક લીધું છે જ્યાં હું 2 વડે ભાગાકાર કરી શકું છું, પરંતુ સૉર્ટ કરેલ મર્જમાં કોઈ મર્યાદા નથી, તે કોઈપણ કદ માટે કામ કરશે. અમુક સમયે જ્યારે તમે અસમાન વિભાજન કરો છો, ત્યારે બે ભાગ એક સરખા કદમાં નહીં હોય, પરંતુ તે ખરેખર ફરજ પાડે છે. હવે આપણે છેલ્લા ભાગને બે ભાગમાં ભાંગીશું. તેથી 43 હવે છે. તેથી, આ સૂચવવા માટે લીલું છે કે આ હવે એક તત્વ છે અને તેથી સૉર્ટ કરેલું, આ પણ છે. અને તે જ રીતે આપણે આ છેલ્લા પગલાંઓમાંથી દરેકને લઈ શકીએ છીએ અને હવે 8 સિંગલ તત્વ મેળવી શકીએ છીએ, જે સૉર્ટ થાય છે. હવે અમે મર્જિંગ શરૂ કરો. તો, આપણે આ બંનેને મર્જ કરવા ઈચ્છીએ છીએ,
(()),અને તેવી જ રીતે આપણે આ બંનેને મર્જ કરવા માંગીએ છીએ. તો, આપણે પહેલા બે મર્જ કરીએ, અને પછી આપણે આને મર્જ કરીએ, નાના ને, પછી તે પહેલા જાય. તેથી, આ હવે સૉર્ટ થયેલ છે. એ જ રીતે, આપણે બીજા જોડીને મર્જ કરીએ છીએ, તે ક્રમમાં ફેરફાર કરે છે, કારણ કે 78 કરતા ઓછું 22. આપણે ત્રીજી જોડીને મર્જ કરીએ છીએ અને તેઓ વિનિમય થાય છે, અને આપણે ચોથી જોડીને ફરીથી મર્જ કરીએ છીએ તે પછી તેઓ વિનિમય થાય છે. ત્યાં એક મહત્વપૂર્ણ નોંધ છે કે વિનિમય કદ સીધી સીધી જોતાં, વિનિમય થયો ન હતો, પરંતુ આ બે મૂલ્યને જોઈને, નાનાને એક પછી નીચે નીચે લેતો. હવે હું આ બંને એરે(Array)ને લંબાઈ 4 ના સૉર્ટ કરેલ સેગમેન્ટમાં મર્જ કરવા માંગું છું, અને આ બે એરે(Array) તે લંબાઈ 4 ના સેગમેન્ટને સૉર્ટ કરે છે. તેથી, ફરીથી મર્જિંગ લાગુ કરો. તેથી, નોંધ લો કે 22 પહેલા આવશે, 32 અહીં આવશે, પછી 43 અહીં આવશે, અને પછી 78 અહીં આવશે. જો હું આ લાગુ કરું તો મને 22 32 43 અને 78 મળશે. તેવી જ રીતે અહીં 30 અહીં આવે, પછી 57, પછી 63, અને પછી 91. તેથી, આ છેમર્જિંગ અસર. અને અંતે, મારે આ બંનેને મર્જ કરવું પડશે. તો, આ સૌથી નાનું 1 અને 13 અહીં જશે, પછી હું 22 કરીશ, પછી હું 32 કરીશ, પછી મને 43 મળશે, પછી મને 57 મળશે, પછી મને 63 મળશે, અને મને 78 મળશે, પછી મને 91 મળશે. તેથી, કદ ચારના આ બે

સબ એરે(Array)ને મર્જ કરવાથી મને અંતિમ જવાબ મળશે. તો, આ કેવી રીતે કામ કરે છે સૉર્ટ કામ કરે છે. તમે તેને બે ભાગોમાં તોડો, એક જ વ્યૂહરચનાનો ઉપયોગ કરીને બે ભાગોને ફરીથી ઉકેલવા અને તેમને મર્જ કરો.
(સ્લાઈડસમયનો સંદર્ભ લો: 07:02)
તેથી, આ સામાન્ય રીતે એક સિદ્ધાંત છે જે ઘણી સમસ્યાઓ પર લાગુ થઈ શકે છે. તેથી, જો તમે કોઈ સમસ્યા લઈ શકો છો, અને તેને બે અથવા વધુ ભાગોમાં વિભાજિત કરો; જેમ કે આ ભાગને તે ભાગથી સ્વતંત્ર કરી શકાય છે, અને કોઈ ઓવરલેપ નથી. તમે આને અલગથી હલ કરો છો, તમે આને અલગ રીતે હલ કરો છો, અને હવે તમે કોઈક રીતે જોડવા માંગો છો. આ ચોક્કસ અલ્ગોરિધમ્સ સંયોજનમાં મર્જ થઈ રહ્યું છે. અમે પછીથી તેને અન્ય ભાગમાં જોઈશું અને એલ્ગોરિધમ્સની તુલના કરીશું, જો મિશ્રાણને કોઈ અલગ વ્યૂહરચનાની આવશ્યકતા હોય, પરંતુ સંપૂર્ણ વિચારો એ છે કે તમે સમસ્યાને નાની સમસ્યાઓમાં ભંગ કરી શકો છો અને પછી તેને એકીકૃત કરી શકો છો. પછી તમે કાર્યક્ષમતાના સંદર્ભમાં ઘણીવાર લાભ મેળવી શકો છો. અગત્યની વાત એ છે કે સમસ્યાને કેવી રીતે સમાવી શકાય છે અને ઉપ સમસ્યાઓમાં સમાધાન કેવી રીતે કરવું તે અને સમસ્યાને ઉકેલવા માટે આ પેટા સમસ્યાને ઉકેલવા માટે કેવી રીતે અસરકારક રીતે જોડવું તે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:50)
તો, ચાલો આપણે આપણા મર્જ સૉર્ટ પર પાછા આવીએ અને વાસ્તવિક કોડની દ્રષ્ટિએ એલ્ગોરિધમ્સને ઔપચારિક બનાવવાનો પ્રયાસ કરીએ. તેથી, હું કેવી રીતે બે શૉર્ટવાળી સૂચિ અથવા બે સૉર્ટ કરેલી એરે(Array) A અને B ને ત્રીજી ક્રમમાં ગોઠવેલ સૂચિમાં જોડું છું. સારુ જો આપણે જોયું કે બંનેમાંથી એક ખાલી છે, તો મને ખબર છે કે જે કંઈક હું ફક્ત બીજાની નકલ કરવા માંગું છું. તેથી જો A માં કોઈ તત્વ બાકી નથી, તો હું ફક્ત કૉપિ કરી શકું છું બી માટે C માં. જો B ખાલી હોય તો હું A માં C ને કૉપિ કરી શકું છું; અન્યથા આપણે એ અને બી ના પ્રથમ ઘટકની સરખામણી કરીએ અને નાનાને પછી ખસેડીએ, અને બધું જ ખસેડ્યા ત્યાં સુધી આપણે આને પુનરાવર્તન કરીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:25)
તેથી, અહીં એક સરળ અનુરૂપ મર્જ ફંક્શન છે. તેથી, ઈનપુટ તરીકે બે એરે(Array)ઝ લે છે. એરે(Array) એક કદ એમ લઈ લો. તેથી, 0 થી n ઓછા 1. ચાલો એરે(Array) બી લઈએ, જે વિવિધ કદ 0 થી n ઓછા 1 હોઈ શકે છે, અને તેનું લક્ષ્ય બનાવવું અને નવું એરે(Array) સી છે. હવે આપણે સીનું કદ જાણીએ છીએ, કારણ કે ત્યાં બધું જ આવવું જોઈએ અહીં તેથી, તે 0 થી m વત્તા n ઓછા 1 બરાબર થશે. તેથી, આપણે શું કરીએ છીએ, આપણે અમુક સ્થાન જાળવી રાખીએ છીએ. તેથી, આપણે કહીએ છીએ કે ઠીક છે, હું વર્તમાન સ્થિતિ જે હું જોઈ રહ્યો છું, જે બી, વર્તમાન સ્થિતિ હું બી તરફ જોઈ રહ્યો છું, અને કે બી વર્તમાન સ્થિતિ હું સી ભરવાનો પ્રયાસ કરી રહ્યો છું તેથી હવે, આપણે જાણીએ છીએ કે ત્યાં એન પ્લસ 1 પગલાં છે. તેથી આપણે લૂપ મુકીએ છીએ જે કહે છે કે આ વિચાર એન પ્લસ 1 મી પ્લસ n વખત ચાલે છે. તેથી, જો આપણે પહેલાથી જ અંત સુધી પહોંચી ગયા છે, જો j એ પહેલાથી જ અંત છે, અથવા જો હું બી j કરતાં નાની છે. તો, આ વાસ્તવિક મર્જ પગલું છે. જો હું A ના નાના અથવા બરાબર બ. જે આપણે કરીશું, તે છે કે આપણે અહીં મૂલ્ય કરીશું, અને પછી આપણે કરીશું અને j વધારો કરીશું. તેથી, આપણે સી કે હું ખસેડીશું અને હું વધારીશ અને કે વધારો થશે. પણ જો j એ સમાન છે, તો તે પણ થાય છે. જો j એ સમાન છે, તો તેનો અર્થ એ છે કે, આ પોઈન્ટર ખરેખર ત્યાં પહોંચે છે. તેથી, સ્કેન કરવા માટે કંઈપણ બાકી નથી. તેથી, j એ બરાબર n નો અર્થ છે કે ખરેખર જમણી બાજુની બહાર છે. તેથી, તે બિંદુથી આગળ છે. તેથી, સ્કેન કરવા માટે કંઈ નથી. તેથી, આપણે ફક્ત એ થી સી સુધી દરેક વસ્તુની નકલ કરીશું, તેથી, જો આપણે એલિમેન્ટની નકલ કરીએ છીએ, તો વર્તમાન ઘટક A થી C જો જો હું બીજે કરતા નાના હોય અથવા બીમાં કંઈ ન હોય તો, આપણે વર્તમાન ઘટકની નકલ કરીએ છીએ, કારણ કે આપણે ફક્ત બધું એક થી સીની નકલ કરો. તેથી, સમપ્રમાણતા કેસ છે, જ્યારે આપણી પાસે A મોટી હોય ત્યારે બી. j. તેથી, જો હું બી j કરતાં મોટો હોય, તો પછી આપણે શું કરવા માંગીએ છીએ, આ મૂલ્ય અહીં કૉપિ કરો. તેથી, આપણે j અપ અને k ખસેડવા માંગીએ છીએ. તેથી, આપણે સી કે જે ઈક્વલ ટુ બી જે નકલ કરીએ છીએ, બી જે થી સી કે પર વેલ્યુની નકલ કરીએ છીએ, અને જે અને કે બંનેને વધારીએ છીએ. અને જેમ આપણે પહેલા કર્યું હતું તેમ, બીજું કારણ આપણે જે કરવાનું ઈચ્છીએ છીએ તે છે, જેમ કે આપણે પહેલાથી લંબાઈ કરી દીધી છે. તેથી, જો આપણે અત્યારે અહીં છીએ. તેથી, એ થાકી ગઈ છે, પછી આપણે

બીજો સી માંથી સીથી આગળનો ઘટક પણ ખસેડીશું, તો આ સરળ છે જ્યારે લૂપ જમણે છે. તેમાં જવા માટેના ઘટકો બરાબર ઘણા પગલાં લે છે, અને દરેક પગલામાં હું એક વધુ ઘટક સી પર ખસેડો; વર્તમાન ઘટકના માપદંડને આધારે એક અથવા બીમાંથી ક્યાં તો હું જોઈ રહ્યો છું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:56)
તેથી, માર્ગથી મર્જ થઈ ગયો છે. હવે, આપણે સૉર્ટ કરેલી મર્જને જોઈ શકીએ છીએ. તેથી, જો આપણે કદ n નિર્દેશાંક C 0 થી n minus 1 ને સૉર્ટ કરવા માંગીએ છીએ, તો આપણે એક નવી એરે(Array) બનાવવી પડશે જે આપણે કહ્યું છે, કારણ કે મર્જિંગને તે બે વસ્તુઓને નવી વસ્તુમાં કૉપિ કરવી પડશે. તેથી, આખરે તે જ કદના નવા એરે(Array) બીને લઈ અને સૉર્ટ કરવામાં આવશે. તેથી, જો n એ 1 છે. જો આપણી પાસે કદ એક તત્વ હોય, તો અમારી પાસે મૂળ કેસ છે, કશું કરવું જોઈએ નહીં. નહિંતર આપણે ડાબા ભાગને સબ એરે(Array)માં ગોઠવીશું l. ડાબી અને જમણી બાજુ માટે આપણે જમણી અડધી અર્ધ એરે(Array) આર માં સૉર્ટ કરીશું, અને પછી તે ફંક્શનનો ઉપયોગ કરીને તે બેને મર્જ કરીશું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:38)
તેથી, આ ખૂબ જ સ્પષ્ટ રિકર્સિવ એલ્ગોરિધમ્સ છે. તેથી, આપણે સૉર્ટને ડાબેથી જમણે મર્જ કરવા માંગીએ છીએ. તેથી, આપણે તે ગ્રહણ કરીશું, અને આપણે ડાબેથી જમણે કહીએ છીએ, અમારું મતલબ છે કે ડાબેથી મોટાભાગના સ્થાનને ડાબે કહેવામાં આવે છે, અને જમણી બાજુની મોટાભાગની અનુક્રમણિકા વાસ્તવમાં બરાબર 1 છે. તેથી, આ ખરેખર એક પછી વધુ છે કે અમે સૉર્ટ કરવા માંગો છો. તો, આપણે આ સેગમેન્ટને ડાબેથી શરૂ કરવા અને એક તરફ જવાનું છે, જેમાં 1 ની બાદબાકી છે. તેથી, જો આ સેગમેન્ટ લંબાઈનો હોય તો સૌ પ્રથમ, તો આપણે ફક્ત મૂલ્યને બીમાં કૉપિ કરીએ. અમે મૂલ્યની કૉપિ કરીએ છીએ ડાબી બાજુ નહિંતર આપણે શું કરીએ છીએ, અમે કરીએ છીએ, અમે મિડપોઈન્ટ શોધીએ છીએ, અને પછી અમે કૉપિ કરીએ છીએ, પરંતુ મધ્યનો સમાવેશ નહીં કરીએ. તેથી, આનો અર્થ એ છે કે આપણે ઉપનામથી મધ્યમ 1 ઓછા છે. તેથી, આ ડાબેથી મધ્યમ 1 ડાબી બાજુએ રહેશે, અને તે મધ્યથી જમણી બાજુ 1 થી સૉર્ટ કરશે. તેથી, તમે શું કરો છો ખાતરી કરવા માગીએ છીએ કે આપણે આકસ્મિક રીતે લપિંગ(Looping) કરતા નથી. તેથી, અમે ખાતરી કરવા માંગીએ છીએ કે મધ્યમ મૂલ્ય જે બંને વસ્તુઓમાં દેખાય છે. મધ્યમાં મૂલ્ય ફક્ત તેમાંથી કોઈ નથી; એટલે કે જમણા બાજુ
((સમયનોસંદર્ભ: 13.02)),
તેથી આ કર્યું. પછી હું મારા મર્જ ફંક્શનનો ઉપયોગ કરીશ l, મર્જ કરવા માટે, જે માપ મધ્યમ ડાબા પણ છે, અને આર બરાબર ઓછા માધ્યમ બી છે. તેથી, આ તે ફંકશન છે જે આપણી પાસે છે. તેથી તે એક ખૂબ જ સરળ રીકર્સિવ (recursive) વસ્તુ છે. મિડપોઈન્ટ શોધો, ડાબા અડધાને સૉર્ટ કરો, જમણી અડધી સૉર્ટ કરો અને તેને મર્જ કરો. જેમ આપણે કહ્યું તેમ, તે વાસ્તવમાં મહત્વ ધરાવે છે કે A એ લંબાઈનો પણ છે અથવા તે બે ગુણાંક છે. તેથી, તે કદાચ અડધું અને જમણા અડધા સમાન લંબાઈનું નહીં હોય; એક તે લાંબા સમય સુધી રહેશે, એક ટૂંકા હશે. તે ખરેખર કોઈ વાંધો નથી. આ બધા કિસ્સાઓમાં કામ કરે છે. આનું વિશ્લેષણ કરવા માટે, તે ખૂબ સીધું આગળ નથી, તેથી અમે તેને આગામી મોડ્યુલ પર સ્થગિત કરીશું.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મૉડ્યૂલ - 06
લેક્ચર - 14
મર્જ સૉર્ટ: એનાલિસિસ (Merge sort: Analysis)

તેથી, આપણે વિભાજન અને વિજયની વ્યૂહરચનાનો ઉપયોગ કેવી રીતે કરવો તે જોયું છે, અમે સોર્ટિંગ એલ્ગોરિધમને મર્જ સૉર્ટ તરીકે ઓળખાતા અમલમાં મૂક્યા છે. તેથી, હવે આપણે મર્જ સૉર્ટ વાસ્તવમાં વર્તે છે કે નહીં તે નક્કી કરવા માંગીએ છીએ, ઓર્ડર એન સ્ક્વેર ઈન્ટ્યુટિવ પ્રકારો જેમ કે નિવેશ સૉર્ટ(Insertion sort) અને પસંદગી સૉર્ટ(selection sort) કરતા વધુ સારું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:18)

તેથી, મર્જ સૉર્ટનું વિશ્લેષણ કરવા માટે, આપણે સૌ પ્રથમ મર્જ ઑપરેશનને જોવાની જરૂર છે. તેથી, યાદ રાખો કે જ્યારે આપણે બે સૉર્ટ કરેલ સૂચિને મર્જ કરીએ છીએ, ત્યારે આપણે શું કરીએ છીએ, અમે બંને બાજુ બાજુ બાજુએ લઈએ છીએ અને અમે દરેક સૂચિના નાના તત્વોને અંતિમ સૂચિ સીમાં સીધી નકલ કરીએ છીએ.

(સ્લાઈડટાઈમ: 00 નો સંદર્ભ લો : 37)

તો, આ તે કોડ હતો જે આપણે છેલ્લે લખ્યું હતું. તેથી, મૂળરૂપે આપણી પાસે સૂચિ A છે, જે 0 થી mi ઓછા 1 થી ચાલે છે અને આપણી પાસે બીજી સૂચિ bi છે, જે 0 થી n ઓછા 1 થી ચાલે છે અને આપણે આઉટપુટને ત્રીજી સૂચિ સીમાં ઉત્પન્ન કરીએ છીએ, જેમાં 0 થી m વત્તા સૂચકાંક છે. n minus 1. તેથી, આપણે શું કરીએ છીએ, આપણે સૂચિ I માં j ઈન્ડેક્સ I, J ને આ સૂચિ અને ઈન્ડેક્સ કે માં મૂકીએ છીએ અને આપણે આ બે મૂલ્યોની તુલના કરીએ છીએ અને અમે બે મૂલ્યોની નાનું લઈએ છીએ અને તેને ખસેડીએ છીએ. અને પછી આ ખસેડો અને પછી આપણે આ બે સૂચકાંકોને ખસેડતા રહીએ અને જ્યારે આપણે ખસેડીએ છીએ ત્યારે. તેથી, પ્રશ્ન એ છે કે આ કેટલો સમય લે છે?

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:22)

તેથી, દરેક પુનરાવર્તિત સૂચનામાં કે આપણે સીમાં એક તત્વ ઉમેરીએ છીએ, પરંતુ સીનું કદ બરાબર એમ વત્તા n છે, કારણ કે એમાં એ તત્વો છે અને બી અને દરેકમાં n તત્વો છે. તેમાંથી છેવટે તે સીને બનાવશે. તેથી, m પ્લસ n એલિમેન્ટ્સ છે અને આ લૂપના દરેક પુનરાવર્તનમાં, લૂપ કે જે આપણે પહેલા કર્યું હતું, આ લૂપના દરેક પુનરાવર્તનમાં, એક તત્વ સીમાં ઉમેરવામાં આવે છે, કાં તો આમાં અથવા જો આમાં, કે વધારો થાય છે. તેથી, તમે જે સૂચિ જોઈ શકો તે કરશે, કે દરેક પુનરાવર્તનમાં પ્રગતિ કરશે. તેથી, આ લૂપ માટે અને લૂપમાં m m પ્લસ n પગલાંઓ છે, જો તમે ખૂબ કાળજીપૂર્વક ગણાશો, તો આપણી પાસે સરખામણી હશે, અમારી પાસે એક વધુ તુલના હોઈ શકે છે, આપણી પાસે અસાઈનમેન્ટ અને બીજું ઘણું છે. પરંતુ, તમે તપાસ કરી શકો છો કે તેમાં સાત કરતાં વધુ પગલાં શામેલ નથી, તેથી ધ્યાનમાં લીધેલા પગલાંઓની સંખ્યા સતત છે, તેથી આપણે કઈ શાખા લીધી છે, તેથી અમારી પાસે m પ્લસ એન પુનરાવર્તન છે, જેમાંના દરેકમાં સતત સંખ્યાબંધ પગલાં છે. તેથી, અમારી પાસે ઓર્ડર એમ પ્લસ એન પગલાંઓ છે, પરંતુ એમ પ્લસ n એ અલબત્ત, મહત્તમ 2 મીટર વત્તા n થી બાઉન્ડ્ડ છે. તેથી, આપણે ફક્ત કહી શકીએ કે આ વસ્તુ મહત્તમ mi પ્લસ n નો ક્રમ લે છે. જો એમ લગભગ મોટેભાગે n જેટલો જ હોય છે જે સામાન્ય રીતે મર્જ સૉર્ટમાં કેસ હશે, કારણ કે આપણે તેને અડધા ભાગમાં વિભાજિત કરીએ છીએ, તો બંને લંબાઈના સંદર્ભમાં મોટા ભાગની 1 થી અલગ હશે. પછી, અમને લાગે છે કે મર્જિંગ એક રેખીય સમય ઑપરેશન છે, તેથી અમે બે સૂચિને મર્જ કરવા માંગીએ છીએ, તે સાથે મળીને બે સૂચિની લંબાઈમાં પ્રમાણસર સમય લે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:46)

તેથી, હવે સૉર્ટને મર્જ કરવા માટે આવી રહ્યા છે, આપણે શું કરવા માંગીએ છીએ, આપણે કદ n ની સૂચિ લેવા માંગીએ છીએ અને તમે તેને કદ 2 ની બે સૂચિમાં વિભાજિત કરવા માંગો છો. અને પછી, આપણે આ અલગ રીતે સોર્ટ કરીએ અને પછી મર્જ કરીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:05)
તેથી, આ કરવા માટે, તે ખૂબ જ સ્પષ્ટ છે કે જો આપણે કદ n ના ઈનપુટ પર સૉર્ટ મર્જ કરીને લેવાયેલ સમય તરીકે tn ને જુએ, તો આ માટે આપણે બે સૂચિને સૉર્ટ કરવાની જરૂર છે કદ n દ્વારા 2 અને પછી, આપણે જોયું છે કે તે મર્જ કરવા માટે ક્રમમાં ઓર્ડર એન ટાઈમ લે છે. હવે, આ n ની સ્પષ્ટ રીતે ગણતરી કરવા માટે, આપણે ધારીશું કે n એ 2 ની શક્તિ છે. હવે, તે નિષ્કર્ષ કાઢે છે કે મર્જ સૉર્ટ કોઈપણ રીતે 2 ની શક્તિ હોવાની જરૂર નથી, પણ સંખ્યા પણ નહીં . જો તે પણ નથી, તો જ્યારે આપણે તેને બે ભાગમાં વહેંચીશું, ત્યારે તે સમાન થશે નહીં, કારણ કે આપણે એક વિચિત્ર સંખ્યાને બે સમાન ભાગોમાં વિભાજિત કરી શકતા નથી. પરંતુ, તે કોઈ વાંધો નથી, એલ્ગોરિધમ હજી પણ યોગ્ય રીતે કાર્ય કરશે, આ ધારણા છે કે n એ 2 થી k છે, તે માત્ર એક સરસ ગણતરી સાથે આવવા માટે એક સંકેત છે. તેથી, આપણી પાસે કદ 2 ની 2 સમસ્યાઓ, 2 ની 2 ઘાત ગુણ્યા 2 અને પછી આપણું મર્જ છે. તેથી, અમે કઈ રીતે હલ કરીએ છીએ જે
((સમયનોસંદર્ભ લો: 04:00)).
તેથી, અલબત્ત, અમારી પાસે બેઝ કેસ પણ છે જે કહે છે કે જ્યારે આપણી પાસે ફક્ત એક જ ઘટક હોય, ત્યારે માત્ર એક જ તત્વ હોય તે ચકાસવામાં સમય લાગશે નહીં, તેથી 1 નું ટી બિંદુ છે. તેથી, જો તમારી પાસે કોઈ ઉકેલ હોય તો મારો અર્થ એ છે કે આ બે પુનરાવર્તન જેવા અભિવ્યક્તિ છે, તો પછી તેને કેવી રીતે ઉકેલવું. તેથી, મૂળભૂત વિચાર તે કરવાનો સરળ માર્ગ છે. અલબત્ત, તમે વધુ વ્યવહારિક માર્ગથી આવી શકો છો જે આપણે હમણાં જ નહીં દાખલ કરીશું. પરંતુ, એક સરળ રીત એ છે કે તમે સમાન વિશ્લેષણનો ઉપયોગ કરીને વિસ્તરણ ચાલુ રાખો છો, તે જ અભિવ્યક્તિ, જ્યાં સુધી તમે બેઝ કેસ સુધી નહીં આવે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:31)
તેથી, આપણે બેઝ કેસ ટી 1 થી સમાન 1 સાથે પ્રારંભ કરીએ છીએ અને સામાન્ય કેસ tn એ 2 વત્તા t ના n વત્તા 2 વત્તા n છે. તેથી, હવે આપણે શું કરી શકીએ, આપણે 2 ની n ની વિસ્તરણ કરવા માટે સમાન સમીકરણનો ઉપયોગ કરી શકીએ છીએ. તેથી, આપણે n નો 2 વડે વિસ્તૃત કરીએ, તો પછી આપણે શોધી શકીએ કે n ની 2 ઘાત 2 ઘાત થાય છે અને 4 ઘાત થાય છે. પ્લસ n બાય 2, કારણ કે હું જે પણ અહીં ભાગાકાર કરું છું 2, n 2 બાય 2 એ n 4 વડે છે અને હું જે પણ અહીં લઈ રહ્યો છું અને તેને અહીં પ્લગ કરું છું, તેથી 2 વડે n બને છે. હવે, જો હું આને કાળજીપૂર્વક વિસ્તૃત કરું, તો આ 2 અને આ 2, તેને 4 લખ્યા સિવાય, અને હું તેને 2 ચોરસ તરીકે લખીશ, હું પણ અવલોકન કરીશ અને આ ઉપયોગી થશે કે આ 4 ફરી 2 વર્ગ છે અને પછી આ n બાય 2 ગુણાકાર થાય છે, તેથી આ 2 વડે 2 વડે ગુણાકાર થાય છે. તેથી, મને 2 ગુણ્યા 2 ની સંખ્યા મળે છે, તેથી તે n વત્તા n છે, તેથી મને 2 n મળે છે. તેથી, તે અકસ્માત નથી કે મારી પાસે 2 ચોરસ, 2 વર્ગમાં 2 ચોરસ છે જે આપણે જોઈશું, જો તમે એક વધુ પગલું કરશો, હવે જો હું આ સમીકરણ ટી 2 ચોરસ દ્વારા લે અને હું આ જ નિયમ લાગુ કરું છું , તો 2 ની ચોરસ દ્વારા n ની આ ટી ફેલાયેલી છે. તે 2 ચોરસ ભાગ 2 n દ્વારા 2 q દ્વારા વિભાજીત થાય છે, તેથી મને 2 ગુણ્યા n થી 2 થી વત્તા n 2 ચોરસ મળે છે. હવે, આ અને આ રદ થશે, તેથી મને અહીં 1 n મળે છે, મારી પાસે 2 n છે, તેથી મને 3 એન મળે છે અને આ 2 ને 2 ચોરસ મળે છે, હું 2 ની ક્યુબ તરીકે લખું છું, તેથી હું 2 ચોરસ ટન થી 2 વડે જઈશ. સ્ક્વેર પ્લસ 2 n થી ક્યુબ ટીન 2 ક્યુબ વત્તા 3 એન. તેથી, તે ચકાસવું સરળ છે કે જો તમે આ j વખત કરશો તો આપણી પાસે 2 ની શક્તિ j વખત t ની 2 ઘાત થશે જે power j plus j n છે. તેથી, હવે આપણે બેઝ કેસમાં ક્યારે પહોંચીએ, આપણે બેઝ કેસમાં પહોંચીએ, જ્યારે 2 થી 2 પાવર j થી 1 થાય. તો, આ ક્યારે થાય છે? ઠીક છે, 2 ની સાથે power j એ 1 ની બરાબર છે, તેનો અર્થ એ છે કે 2 થી power j બરાબર n છે, તેથી j એ બીજું નથી, n નો લોગ base 2 પર છે. તેથી, આપણે હંમેશા log n નો ઉપયોગ કરીશું. સામાન્ય રીતે નો અર્થ એ છે કે n નો આધાર 2 છે, સિવાય કે ઉલ્લેખિત રીતે. તેથી, n પગલાંઓ લોગ કર્યા પછી, આપણે આ જેવો દેખાય છે તે અંત થાય છે, તે 2 થી jtn 2 થી j સુધી છે, log n પગલાંઓ પછી, j લોગ n છે, તેથી 2 લોગ n પર, તો પછી t ની આ દ્વારા n, તેથી આ 1 નું હવે છે, તેથી આ માત્ર 1 છે, તેથી તે 1 વડે ગુણાકાર થાય છે અને તે પછી, મારી પાસે j ગુણ્યા n છે, તેથી j

એ log n છે, તેથી j લોગ n અને n બાજુ છે. તો, હું આ વિસ્તરણ લોગ n વખત કર્યા પછી હું 2 સાથે લોગ n વત્તા લોગ n વખત n સુધી અંત કરું છું, 2 વ્યાખ્યાયિત કરીને log n એ માત્ર n છે, તે લોગની વ્યાખ્યા છે જે 2 ની ઘાતક છે તમને n આપે છે. તો, આ n છે અને મારી પાસે આ n લોગ n છે અને આપણે સામાન્ય વસ્તુઓથી જાણીએ છીએ કે n એ n લોગ n છે. તેથી, આ 2 ગણી એન લોગ n દ્વારા બંધાયેલ છે. તેથી, આપણે કહી શકીએ કે બધા મર્જ સૉર્ટ x વખત O n લોગ n ઉપર. તેથી, અમે એક નોંધપાત્ર પ્રેરણા પ્રાપ્ત કરી છેવેગમેન્ટ, કારણ કે યાદ રાખો, હજી સુધી બંને દાખલ સૉર્ટ અને પસંદગી તમારા ઓ એન સ્ક્વેરને સૉર્ટ કરો અને અમે આ વિભાજન અને વ્યૂહરચનાને જીતીને O n ચોરસથી O n લોગ n નીચે આવી ગયા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:06)

તો, આ મોટો સોદો કેમ છે, આપણે શરૂઆતમાં જોયું કે ઓ n લોગ n એ O n ચોરસ કરતાં વધુ કાર્યક્ષમ છે. જો આપણે એમ માનીએ છીએ કે અમે કહ્યું છે કે વ્યાજબી ધોરણ ડેસ્કટોપ મશીન દર સેકન્ડમાં 8 ઓપરેશન્સ માટે 10 કરી શકે છે, તો પછી સૉર્ટિંગ એલ્ગોરિધમ સાથે સૉર્ટ કરી શકાય તેવા સંભવિત ઈનપુટનું કદ, 10,000 વર્ગથી એન સ્ક્વેર એલ્ગોરિધમ માટે 10 લોગ અથવા 1 કરોડ એન લોગ એન અલ્ગોરિધમ માટે. તેથી, જો તમારે ખરેખર મોટા પ્રમાણમાં ડેટાને સૉર્ટ કરવું પડશે, તો એન લોગ n એ અમારા માટે યોગ્ય સમયે તે કરવા માટે શક્ય બનાવે છે, જ્યારે n ચોરસ વર્ષ લાગી શકે છે તે પહેલાં અમે તેને હલ કરીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:43)

તેથી, આપણે નિષ્કર્ષ પૂરું કરીએ તે પહેલા કેટલીક સરસ વસ્તુઓ ધ્યાનમાં લેવાની છે. તો, આ મર્જ ઑપરેશન જેનો ઉપયોગ આપણે મર્જ સૉર્ટમાં કરીએ છીએ, તે વાસ્તવમાં કોઈપણ સૉર્ટ કરેલ સૂચિ પર ખૂબ જ ઉપયોગી કાર્ય છે. તો, એક વસ્તુ જે આપણે મર્જમાં કરીએ છીએ એ છે કે આપણે વાસ્તવમાં કંઈપણ ગુમાવ્યા વિના સૂચિને ભેગા કરીએ છીએ. તો, જો આપણે બે યાદીઓને મર્જ કરીએ, તો આપણી અંતિમ વસ્તુમાં, 1, 2, 4 અને 2, 3, 6 પછી આપણી પાસે 2 ની 2 કોપી હશે, કારણ કે ત્યાં 2 કૉપીઝ અને પછી 3, 4, 6 છે. તેથી, કેટલીકવાર આપણે આ ન મેળવવા માંગીએ છીએ, તમે ફક્ત એક જ નકલ રાખવા માંગી શકો છો, આપણે આને સેટ તરીકે ગણીશું, તેથી જો હું આને 1, 2, 4 અને સેટ 2, 3 તરીકે લઈશ, 6, ત્યારબાદ બે સેટના પરિણામી જોડાણ એ 1, 2, 3, 4, 6 છે તમારી પાસે તેની માત્ર એક જ નકલ છે. પછી, આપણે સરળતાથી મર્જ પ્રક્રિયામાં આ કરી શકીએ છીએ ફક્ત કહીને કે જ્યારે હું શોધી શકું, અગાઉ આપણી પાસે બે કેસો હતા, આપણી પાસે બી જે કરતાં ઓછું અથવા તેના બરાબર હતું અને પછી અમારે બી જે કરતા વધારે હતું. તેથી, પ્રથમ કિસ્સામાં, અમે એ, કે બી બી કેસમાંથી કૉપિ કરીયેલો બીજો કેસ કૉપિ કર્યો હતો અને કોઈ પણ કેસમાં આપણે ફક્ત કે હું જ કે સાથે જ વધારો કર્યો હતો. આ કિસ્સામાં, આપણે કહીએ છીએ કે જો બંને પક્ષો સમાન છે, તો આપણે તેની એક નકલ અંતિમ સૂચિમાં રાખીએ છીએ, પરંતુ અમે આ બંને કૉપિઝ પર છોડીએ છીએ, અમે ત્રણેય પોઈન્ટરને વધારીએ છીએ. તો, જો આપણે આ સ્થિતિ અને આ સ્થિતિને ઉમેરીએ, તો અમારું આઉટપુટ તમે 2 પર કૉપિ કરશો અને પછી આપણે આ નિર્દેશકને જમણી બાજુએ ખસેડીશું અને આ નિર્દેશકને જમણી બાજુએ ખસેડીશું, તેથી હું બે વખત કૉપિ કરી શકતો નથી, તેથી આ એક વસ્તુ છે આપણે કરી શકીએ. બીજી વસ્તુ જે આપણે કરી શકીએ છીએ તે છે બે સૂચિને છૂટા પાડવા, તેથી જો આપણી પાસે ફરીથી 1, 2, 3 2, 3, 6 હોઈ શકે તો પછી તમે ખાતરી કરો કે આપણી આઉટપુટમાં ફક્ત 2 અને 3 છે. , હવે આપણે શું કરીએ છીએ, જ્યારે આપણે આ બંને તરફ જોવું શરૂ કરીએ, જો તેઓ સમાન નથી, તો દેખીતી રીતે તે છૂટાછવાયામાં નથી. પરંતુ, ક્યા છોડવું જોઈએ, સારી 2 હજુ પણ હોઈ શકે છે, કારણ કે 1 નાનું છે, તેથી આપણે આ નિર્દેશકને જમણી બાજુએ ખસેડીએ છીએ. તેથી, જો હું બી જે કરતા નાના હોઉં તો આપણે વધારીશું, હવે જો કહીએ કે જ્યારે A, B j એ સમાન હોય, તો આપણે યુનિયન માટે કર્યું તેમ, આપણે તેને 2 બહાર ખસેડીશું અને મર્જ કરીશું, અમે બંને પોઈન્ટરને દૂર કરીશું , તેથી હવે બંને પોઈન્ટર આ સ્થિતિમાં આવે છે. હવે, ફરીથી આપણે કહીશું કે આપણી પાસે 3 છે અને આપણે ચાલુ રાખીશું અને પછી જ્યારે આપણે બીજો રસ્તો જોશું, જો બી j એ A કરતાં નાના હોય, તો આપણે j ને વધારીશું અને નહીં. તેથી, નોંધવું મહત્વપૂર્ણ છે કે મર્જ એ ખૂબ જ સામાન્ય કામગીરી છે, આપણે પહેલા જે કર્યું તે આપણે કરી શકીએ છીએ જે વાસ્તવમાં બે સૉર્ટ કરેલ સૂચિને એક મોટી સૉર્ટ કરેલી સૂચિમાં ફેરવવાનું છે, જ્યારે કોઈ ઘટકો ગુમાવતા નથી અથવા અમે ડુપ્લિકેટ્સને દૂર કરી શકીએ છીએ કરશે, એ સેટ યુનિયન ઓપરેશન તરીકે

ગણવામાં આવે છે અથવા અમે ફક્ત સામાન્ય જોડીને જ રાખી શકીએ છીએ જે ઈન્ટરસેક્શન ઓપરેશન સેટ કરે છે. તેથી, તપાસ કરવા માટે કે તમે આ સમજો છો, કદાચ તમારે નીચેની કસરતો અજમાવવા જોઈએ. તેથી, આપણે વિચારીએ છીએ કે આપણે સૂચિ તફાવત તરીકે ઓળખાવા માંગીએ છીએ, તેથી આ તફાવત થોડો સમય આ રીતે લખાયો છે. તેથી, જો હું સેટ 1, 2, 3 લઈશ અને સેટ 3, 4, 6 કહીશ અને હું સેટ તફાવતને પૂછું છું, તે કહે છે કે A માં છે, પરંતુ B માં નથી, તેથી જવાબ આ કિસ્સામાં હોવો જોઈએ અને 1 પછી 2, પરંતુ 3 નહિ, કારણ કે 3 અંતે છે. તેથી, આપણે સૂચિ માટે આ જ વસ્તુ કરી શકીએ છીએ. તેથી, ઉદાહરણ તરીકે, જો મારી પાસે 1, 2, 3 અને સૂચિ 3, 4, 6 સૂચિ હોય તો હું સૂચિ ફક્ત 1, 2 રાખવા માંગુ છું. તેથી, હમણાં જ તપાસો કે તમે જે રીતે કર્યું છે તેમાં મર્જ કરી શકો છો કે કેમ તફાવત યાદી કરવા માટે યુનિયન અને આંતરછેદ માટે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:20)

તેથી, સૉર્ટ કરો, સૉર્ટ મર્જ કરો, જોકે તે ઓર્ડર n લોગ n એ અલ્ગોરિધમનો સૉર્ટિંગ છે અને તેથી તે ઈન્સર્શન સૉર્ટ અથવા સિલેક્શન સૉર્ટ કરતાં નોંધપાત્ર રીતે ઝડપી છે, તે થોડી ટૂંકી હોય છે. મર્જ સૉર્ટ સાથેનો મુખ્ય સમસ્યા એ છે કે તેને વધારાની જગ્યાની જરૂર છે. જુઓ, જ્યારે હું A અને B લે છે અને હું તેને સીમાં મર્જ કરું છું, ત્યારે મર્જ એરે(Array)ને અલગ સ્થાનમાં સ્ટોર કર્યા વિના આ કરવાનું લગભગ અશક્ય છે. કારણ કે, જો હું જગ્યાએ મર્જિંગ સ્ટેક કરું છું, તો એ અને બીનું સૉર્ટ કરેલ ઑર્ડર મર્જ સૉર્ટ મેળવે છે અથવા મને વસ્તુઓ ખસેડવાની રહે છે અને તેથી મર્જ રેખીય સમયમાં મોટી નથી. તેથી, જો હું રેખીય સમય A અને B નો માર્ગ મર્જ કરું, તો તે ફક્ત કદની બાહ્ય સૂચિનો ઉપયોગ કરીને પ્રાપ્ત કરી શકાય છે જે તેને સ્ટોર કરવા માટે બે સૂચિની રકમ છે. આનો અર્થ એ છે કે જો તમે મોટા એરે(Array)ને સૉર્ટ કરી રહ્યાં છો અને વધારાની જગ્યા પણ વાપરી રહી છે, જેથી મર્જ ઑપરેશન માટે જગ્યાને ડુપ્લિકેટ કરવાનું ચાલુ રાખવામાં આવે. અને અલબત્ત, મર્જ સૉર્ટની બીજી વસ્તુ જેનો ઉકેલ લાવવા માટે ખૂબ જ મુશ્કેલ છે તે છે કે તે સ્વાભાવિક રૂપે પુનરાવર્તનશીલ છે અને તેનો અર્થ એ છે કે વાસ્તવમાં તેને સરળતાથી પુનરાવર્તિત કરવાનો કોઈ રસ્તો નથી, કારણ કે અમને સૉર્ટ કરેલ વસ્તુઓ બનાવવાની અને પછી મર્જ કરવાની જરૂર છે. તેથી, આપણે ભેગા કરવા માટે દરેક બિંદુએ આ પુનરાવર્તિત ઉકેલને સંગ્રહિત કરવાની જરૂર છે. તેથી, પુનરાવર્તિત કૉલ અને વળતર ખર્ચાળ પણ હોઈ શકે છે. તેથી, જો સૉર્ટ મર્જ કરવુ તે એક ખૂબ જ આકર્ષક પ્રકાર છે, તે સૈદ્ધાંતિક જટિલતા છે, આ મર્યાદાઓને કારણે કેટલીકવાર પ્રથાઓમાં તેનો ઉપયોગ કરવામાં આવતો નથી અને આપણે જોશું કે કેવી રીતે તેને બીજી રીતે દૂર કરવી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:40)

તેથી, મર્જ ઓપરેશનને લીધે એ અને બી ને એક સૂચિમાં સી સાથે જોડવા માટે મર્જ ઓપરેશનને લીધે મુખ્ય જગ્યાને વધારાની જગ્યાની જરૂર છે અને સૉર્ટ કરો. હવે, ધારી લો અને આનું કારણ બને છે કારણ કે ડાબી બાજુ જેવું કંઈક હોઈ શકે છે, જે 2, 4, 6 કહી શકે છે અને જમણી બાજુ 1, 3, 5 કહી શકે છે અને હવે આપણને બંને બાજુએ વસ્તુઓ લેવાની જરૂર છે. તેથી, કેટલીકવાર મને અહીંથી કંઈક વિચારવાની જરૂર છે, પછી મારે અહીંથી કંઈક લેવાની જરૂર છે, પછી મારે અહીંથી કંઈક લેવાની જરૂર છે. તેથી, જો કે પ્રત્યેક અડધા ભાગને સૉર્ટ કરવામાં આવે છે, તો ડાબી બાજુના તત્વો જે જમણી બાજુએ જ જોઈએ અને જમણી બાજુના તત્વો ડાબી બાજુએ જવું આવશ્યક છે. તેથી, હવે, જો આપણે આ કરવાનું ટાળવા માટે રસ્તા ઉપર આવી શકીએ, તો લાલ રંગની દરેક વસ્તુ લીલા ભાગની અંદરની દરેક વસ્તુ કરતા નાની હોય, પછી એકવાર મેં લાલ ભાગને ગોઠવ્યો અને પછી મેં હરિત ભાગ ગોઠવ્યો લાલ ભાગમાં આપમેળે બધું લીલો ભાગમાં રહેલી દરેક વસ્તુની ડાબી બાજુ રહે છે. તેથી, જો મેં આનો ત્વરિત ઉમેરો કર્યો હોય, જો મેં ડાબે ઉમેર્યું હોય તો હું 2, 1, 3 થી શરૂ કરું છું અને હાથ બાજુ 6, 4, 5 થી શરૂ થાય છે અને પછી, હું સ્થાનિક રીતે આ વસ્તુઓને સૉર્ટ કરું છું. તેથી, મને 1, 2, 3 મળ્યું અને પછી અહીં મને 4, 5, 6 મળ્યા પછી જો હું ખાતરી કરું કે ડાબી બાજુની દરેક વસ્તુમાં વિભાજન શોધવામાં આવ્યું હતું તે જમણી બાજુની બધી વસ્તુ કરતાં નાના છે, પછી વિભાજન થાય છે અને પછી સબ સમસ્યાને હલ કરવાની કોઈ સંયોજન નથી મર્જ કરવાની જરૂર છે. તેથી, આગામી સોર્ટિંગ એલ્ગોરિધમ આ વિચારને અમલમાં મૂકવાની વ્યૂહરચનાને જોશે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 02
મોડ્યુલ - 07
લેક્ચર - 15
ક્વિક સૉર્ટ(Quicksort)

તેથી, હવે આપણે ક્વિક સૉર્ટ(Quicksort) તરીકે ઓળખાતા અન્ય અલ્ગોરિધમને જોવા માટે તૈયાર છીએ. તેથી, પ્રારંભિક 1960 ના દાયકાની શરૂઆતમાં ટોની હોરે(Tony Hoare) નામના કમ્પ્યુટર વૈજ્ઞાનિક દ્વારા ક્વિક સૉર્ટ(Quicksort)ની શોધ કરવામાં આવી હતી; તે લગભગ 50 વર્ષ પહેલા છે. અને ટોની હોરે(Tony Hoare) એક જાણીતા કમ્પ્યુટર વૈજ્ઞાનિક છે અને તે એક ટ્યુરિંગ(Turing)ને અસર કરે છે કે જે શૈક્ષણિક કમ્પ્યુટર વૈજ્ઞાનિક માટે ઉચ્ચતમ સિદ્ધિઓમાંની એક છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 00:28)
તો, ક્વિક સૉર્ટ(Quicksort)નો હેતુ શું છે? સારુ, ક્વિક સૉર્ટ(Quicksort)નો હેતુ એ છે કે આપણે મર્જ સૉર્ટ(merge sort)માં જોયેલી કેટલીક ક્ષતિઓને દૂર કરવાનો છે. તેથી, મર્જ સૉર્ટમાં આપણે જે વસ્તુઓ જોયેલી તે છે તે છે કે મર્જ ઑપરેશનને કારણે, અમારે વધારાના સ્ટોરેજની જરૂર છે અને તેથી મર્જ સૉર્ટ(merge sort) થોડો ખર્ચાળ બનાવે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:42)
તેથી, અમે પણ જોયું કે અમને આ વધારાની સ્ટોરેજની આવશ્યકતા છે કારણ કે અમારી પાસે મર્જ ઑપરેશનમાં છે કે જ્યારે આપણે મર્જ કરેલ એરે(Array)ને વ્યુત્પન્ન કરી રહ્યા છીએ ત્યારે અમે બંને બાજુથી ઘટકો ખેંચી શકીએ છીએ. તેથી, મૂળભૂત રીતે જમણી બાજુના કેટલાક ઘટકો ડાબી બાજુના કેટલાક ઘટકો કરતાં નાના હોઈ શકે છે અને આ તે વપરાશકર્તા છે જે મર્જ કરવામાં પરિણમે છે. તેથી, આપણે બધું જ વિભાજીત કરી શકીએ છીએ. તેથી, તે નથી થતું, ડાબેની દરેક વસ્તુ નાની છે અને જમણી બાજુની બધી જ વસ્તુ મોટી છે, શું આ ભાગમાં વિભાજન કરવું અને જીતવું શક્ય છે?
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:16)
સારું, આ કેસ છે, તો આપણે શું કરવાની જરૂર છે, આપણે મધ્યમ મૂલ્ય કેન્દ્રમાં મૂકવાની જરૂર છે. તેથી, ધારીએ છીએ કે આપણે મધ્યમ શોધી શકીએ છીએ. તેથી, યાદ રાખો કે સરેરાશ શું છે, સરેરાશ એ મૂલ્ય છે કે એરે(Array)માં બરાબર અડધા મૂલ્યો મોટા અને અર્ધ નાના છે. તેથી, હવે આપણે એમ કરતાં ડાબા અર્ધ કરતાં નાની બધી વસ્તુ ખસેડીએ છીએ. તેથી, આપણી પાસે અહીં મૂલ્યોનો સમૂહ છે જે એમ કરતાં ઓછા અથવા બરાબર છે અને પછી અમારી પાસે જમણી તરફ છે જે કંઈક કરતાં વધુ સખત છે ... તેથી, અલબત્ત, આપણે આ સ્થળાંતર કરવું પડશે, પરંતુ દાવો એ છે કે આપણે રેખીય સમયમાં આ સ્થળાંતર કરી શકે છે અને અમે આ કરવા માટે એક માર્ગ જોશું. તેથી, ધારી રહ્યા છીએ કે આપણે આ કરી શકીએ, મૂલ્ય મી પસંદ કરો જે મધ્યમ છે અને એમથી ડાબેથી નાની બધી વસ્તુને પાછી ખસેડો, પછી આ લગભગ અડધા બિંદુ બનશે, કારણ કે એમ એ મૂલ્ય છે જે એરે(Array)ને બે ભાગમાં વિભાજિત કરે છે, તે માત્ર નાના અથવા અડધા અને તે મોટા અથવા અડધા. હવે, હું આ પુનરાવર્તિત વસ્તુ કરું છું, હું આને સૉર્ટ કરું છું અને હું આને સૉર્ટ કરું છું અને હવે તેને મર્જ કરવાની કોઈ જરૂર નથી, કારણ કે ડાબેની દરેક વસ્તુ જમણી બાજુની બધી જ વસ્તુ કરતાં નાની છે. તેથી, હું ફક્ત આગળ વધું છું અને એરે(Array) ધારણ કરી શકાય છે. તેથી, જો હું આ કરું છું, તો વિશ્લેષણ દ્વારા મર્જ સૉર્ટ(merge sort) માટે, મારી પાસે પુનરાવર્તન છે જે t ની n છે 2 times tn 2 વત્તા n અને આપણે હવે આ ઓર્ડર n લોગ n છે. તેથી, આ આપણને સૉર્ટને મર્જ કરવા જેવી જ જટિલતા આપશે, પરંતુ તે વધારાની જગ્યા બનાવવા સાથે સંકળાયેલા કેટલાક ઓવરહેડ્સને ટાળશે. કારણ કે, જ્યારે હું અહીં રીકર્સિવ(recursive) કોલ કરું છું, ત્યારે હું આ ભાગને સરળતાથી અને આ ભાગને સોર્ટ કરી શકું છું, કારણ કે જ્યારે હું આ ઉકેલ કરું છું ત્યારે મને બીજા ભાગનો સંદર્ભ લેવાની જરૂર નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:42)
તેથી, અલબત્ત, એક કેચ હોવો જોઈએ અને કેચ એ મધ્યસ્થ કેવી રીતે મેળવશે? અમારી ચર્ચાના પ્રારંભમાં, અમે કહ્યું હતું કે અમે જે કારણોને સૉર્ટ કરવા માંગીએ છીએ તે એક છે મધ્યયુગીન શોધવા જેવી આંકડાકીય બાબતો કરવી. તેથી, જો આપણે એરે(Array)ને સૉર્ટ કર્યું છે, તો સરેરાશ મૂલ્ય મધ્યમ મૂલ્ય છે, પરંતુ અલબત્ત, અમારું લક્ષ્ય એરે(Array)ને સૉર્ટ કરવાનું છે. તેથી, આપણે ધારી શકતા નથી કે આપણી મધ્યસ્થી છે, કારણ કે આપણે પહેલેથી જોયું છે કે સૉર્ટિંગ એ મધ્યસ્થ શોધવાનો એક સરળ રસ્તો છે. તેથી, તે એક પ્રકારની ચિકન અને ઈંડા સમસ્યા છે, અમે મધ્યસ્થીને સૉર્ટ કરવા માટે ઉપયોગ કરી શકતા નથી. તેથી, ટોની હોઅર(Tony Hoare) એલ્ગોરિધમ્સે ક્યા ઝડપી સ્વરૂપમાં કહ્યું છે કે, માધ્યમની જરૂર નથી, ફક્ત એમાં કેટલાક મૂલ્યને પસંદ કરે છે અને અમે જે કહ્યું છે તે કરો. તેથી, અમે આ પિવોટને પસંદ કરીએ છીએ અને પછી તમે તેને બે ભાગમાં ભંગો છો, જે પિવોટ કરતાં નાના હોય છે અને તે પીવોટ કરતા મોટા હોય છે. તેથી, પિવૉટ માત્ર થોડા મૂલ્ય અને એરે(Array) છે, તે મધ્યસ્થ હોવું જરૂરી નથી અને આપણે જોશું કે જો તે મધ્યમ નથી, તો તે ખરાબ કેસની જટીલતાના સંદર્ભમાં કેટલીક સમસ્યામાં પરિણમે છે, પરંતુ ચાલો તેને અવગણો . તો, આપણે ફક્ત એરે(Array)માં અમુક વેલ્યુ પસંદ કરીએ છીએ અને આપણે તે બધા વેલ્યુને તેના કરતા નાના લે છે, તેને ડાબી બાજુએ ખસેડો, તે બધા જે તે કરતા વધારે છે તેને જમણી તરફ ખસેડો. અને પછી, અમે તેમને વારંવાર સૉર્ટ કરીએ છીએ અને પછી અમે બાંહેધરી આપીએ છીએ કે ડાબી બાજુના કાંઈને પછીથી જમણી બાજુની કોઈપણ વસ્તુ સાથે જોડવાની જરૂર નથી. તેથી, પરિણામી એરે(Array) સૉર્ટ થયેલ છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:55)
તેથી, આ ક્વિક સૉર્ટ(Quicksort) છે, પીવોટ ઘટક પસંદ કરો. તેથી, ઉદાહરણ તરીકે, અમલમાં મૂક્યા મુજબ તમે એરે(Array)માં પહેલું મૂલ્ય જ પસંદ કરી શકો છો. આ એરે(Array)ને નીચલા અને ઉપલા ભાગમાં વહેંચો. તેથી, નીચલું ભાગ તે છે જે પીવોટ કરતા ઓછું છે, ઉપલા ભાગ તે છે જે પિવોટ કરતા વધારે છે. તો, આ પાર્ટિશનિંગનો નિર્ણાયક પગલું છે, આપણે જોઈશું કે આ ભાગલા કેવી રીતે કરવું, તે હવે આપણે જોઈશું. પછી, અમે આ પિવોટને અહીં ખસેડીએ છીએ. તેથી, તે યોગ્ય સ્થાને છે અને હવે આપણે આ ભાગ અને આ ભાગને વારંવાર સૉર્ટ કરીએ છીએ અને અમારું કાર્ય પૂર્ણ થાય છે, કારણ કે કશું ખસેડવાની જરૂર નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:35)
તેથી, અહીં ઉદાહરણ દ્વારા અલ્ગોરિધમનો એક પ્રકારનો ઉચ્ચ સ્તરનું વર્ણન છે. તેથી, આ મારી એરે(Array) છે તેવું વિચારીને, પછી હું પાઈવોટ પર પ્રથમ તત્વ પસંદ કરું છું 43. તેથી, 43 ની આદર સાથે, હવે હું આ એરે(Array)ને ભાગલા કરું છું. તેથી, 43 કરતા નાના બધા. તેથી, અહીં 43 કરતાં નાના તત્વો શું છે, આપણી પાસે 32, 22 અને 13 છે. તેથી, આ તત્વો ડાબી બાજુએ આવે છે અને તે તત્વો જે 78, 63, 57, 91 જેવા છે. જમણી બાજુએ જવું જોઈએ. તેથી, હું આ ભાગલા કરું છું અને હું આભાગલા કેવી રીતે કરી શકું, સારી રીતે હું ... તેથી, મેં બધું ડાબી બાજુ લાવ્યું અને ત્યારબાદ, હું વારંવાર ડાબે સૉર્ટ કરું છું. તેથી, હવે તેને સૉર્ટ કરવામાં આવશે નહીં, ફક્ત 43 કરતા ઓછાનું સંપાદન કરી રહ્યો છું, આ સૉર્ટ કરેલું નથી, ત્યાં બધું કરતાં મોટી છે. પછી, જો હું ધારું છું કે હું વારંવાર તેને સૉર્ટ કરી શકું છું, તો મેં આખી એરે(Array) ગોઠવી છે, કારણ કે હવે પીળી બાજુમાં કંઈ પણ લીલા બાજુ સાથે જોડવાની જરૂર નથી, લાલ પિવોટ મૂલ્ય આ બંનેને અલગ કરે છે. તેથી, પ્રથમ વસ્તુ જે આપણે સમજવાની જરૂર છે તે આ ભાગલા કેવી રીતે કરવું તે છે. તેથી, ભાગલા કરવાની બે રીત છે. તેથી, આપણે એક વિગતવાર જોઈશું અને તેના માટે થોડો કોડ બતાવીશું અને પછી આપણે ઉદાહરણ દ્વારા બીજાને જોશું અને તમને રસ હોય તો તમારે કોડ જાતે લખવો પડશે. તેથી, અહીં ભાગલા કરવાનો એક રસ્તો છે. તેથી, મેં ડાબી બાજુના પિવોટ ઘટકથી પ્રારંભ કર્યું છે અને હવે મારી પાસે આ મૂલ્યોની સંપૂર્ણ શ્રેણી જમણી બાજુએ છે જે ક્રમાંકિત નથી. તેથી, હું બે સૂચકાંકો મૂકીશ જે મેં આ ચિત્રમાં બે રંગ પોઈન્ટર, પીળી પોઈન્ટર અને લીલી પોઈન્ટર સાથે સૂચવ્યું છે, એકવાર અમે એલ્ગોરિધમમાં થોડા પગલાઓ ખસેડ્યા પછી આ મહત્વ થોડું સ્પષ્ટ બનશે. તેથી, આપણે શું કરીએ છીએ તે છે કે બધું લીલું પોઈન્ટરની જમણી બાજુએ છે. તેથી, લીલો પોઈન્ટર એ ભાગના અંતને સૂચવે છે જે પહેલાથી જ ભાગલા છે. તેથી, લીલી પોઈન્ટરની જમણી બાજુએ કંઈપણ ભાગલા વિનાનું છે અને બીજી તરફ પીળી પોઈન્ટર સૂચવે છે. તેથી, આ નીચલા ભાગની મર્યાદા સૂચવે છે. તેથી, મારે જરૂર છે. તેથી, મૂળભૂત રીતે સામાન્ય

રીતે આ ચિત્ર મારી પાસે જઈ રહ્યો છું. તો, અહીં મને પોવૉટ હશે, પછી મને અહીં નીચી ભાગ મળશે જે પહેલાથી મળી છે. પછી, હું અહીં ઉપલા ભાગ પાસે જઈ રહ્યો છું, આ ઘટકો પહેલેથી સ્કેન કરેલા છે અને વિભાજિત થયા છે અને પછી મારી પાસે તે ભાગ છે જે કરવાનું છે. તેથી, શરૂઆતમાં બધું કરવાનું છે અને ત્યાં કોઈ નાનો ભાગ નથી અને ઉપલા ભાગ નથી. તેથી, હું શું કહું છું કે આપણે આ જેવા પોઈન્ટરને રાખીશું. તેથી, આ વસ્તુ તેના અંત તરફ નિર્દેશ કરશે અને આ વસ્તુ તેના અંત તરફ નિર્દેશ કરશે. તેથી, આ તે છે જે આપણે પ્રાપ્ત કરવા માંગીએ છીએ. તેથી, અમે આ ચિત્ર સાથે કહ્યું તેમ અમે પ્રારંભ કરીએ છીએ. તો, આપણે શું કરીએ છીએ, જો આપણે કંઈક ઓછું જોઈશું, તો હું નીચલા ભાગને વિસ્તારું છું અને હું આગલા તત્વ પર જાઉં છું, ફરીથી આપણે કંઈક નીચે જોઈશું. તેથી, અમે નીચલા ભાગને વિસ્તૃત કરીએ છીએ. તેથી, જો આ બિંદુ કહેશે કે નીચલા ભાગમાં 32 અને 22 ની બે વેલ્યુ છે અને ઉપલા ભાગ ખાલી છે અને બધું આગળ 78 થશે

((સમયનોસંદર્ભ લો: 07:51)).

હવે, હું 78 તરફ જોઉં છું. તેથી, 78 43 કરતા મોટો છે. તેથી, નીચલું ભાગ અહીં રહે છે અને હવે મારી પાસે ખાલી ખાલી ઉપલા ભાગ નથી 78. હવે, હું 63 તરફ જોઉં છું, એકવાર 63 એ ઉપલા ભાગની છે. તેથી, ફરીથી હું આને અનુસરું છું, 57 ફરીથી ઉપલા ભાગ સાથે જોડાય છે. તેથી, હું અનુસરવા માટે આગળ વધું છું, 91 ફરીથી ફોલોઅન તરફ આગળ વધું છું. તેથી, જ્યારે હું 13 મા આવે ત્યારે પ્રથમ રસપ્રદ વસ્તુ થાય છે. તો, હવે, જ્યારે હું 13 મા આવે ત્યારે, મને લાગે છે કે તે નીચલા ભાગમાં જવું જોઈએ, પરંતુ નીચલું ભાગ ઘણું દૂર છે. તેથી, હું કેવી રીતે પ્રાપ્ત કરી શકુંછે તો, હું શું કરીશ, મને ખબર છે કે નીચલા પોઈન્ટરની જમણી બાજુએ આ તત્વ છે. તેથી, આ પી કરતાં મોટું છે અને આ તત્વ પી કરતા નાના છે. તેથી, મને જે જોઈએ છે તે પ્રાપ્ત કરવાની એક રીત એ છે કે આ બંને મૂલ્યોનું વિનિમય કરવું. તેથી, હું 13 અને 78 નું વિનિમય કરું છું. તેથી, હું 13 લે છે, હું તેને નીચે લેબલ કરું છું, પછી હું બંને બિંદુઓનું વિનિમય અને ખસેડો. તેથી, આ આગળનું ભાગલિંગ એલ્ગોરિધમ છે જે અ ભાગલા ભાગની લંબાઈને ઘટાડે છે, જો હું કંઈક ઉપર જે કંઈક જોઉં, તો હું ફક્ત લીલા પોઈન્ટર ખસેડો. જો મને કંઈક ઓછું દેખાય છે, તો હું તે નીચલા તત્વને ઉપલા ભાગના પહેલા ભાગ સાથે વિનિમય કરું છું અને પછી હું બંને ભાગલાને વિસ્તૃત કરું છું. પછી, આખરે, આ બિંદુએ મારી પાસે હજુ પણ અંતિમ વસ્તુ નથી, પરંતુ મને આ પિવટ ઘટક આ બંને વચ્ચે હોવા જોઈએ. તેથી, હવે મુદ્દો એ છે કે હું જાણું છું કે આ છેલ્લું છે. તેથી, પીળા પોઈન્ટરની જમણી બાજુએ શું છે તે પ્રથમ ઉપલા સીમા છે અને પીળી પોઈન્ટરની ડાબી બાજુએ જે છે તે છેલ્લી નીચલી વસ્તુ છે. તેથી, હું 43 અને 13 નું વિનિમય કરી શકું છું અને પછી મને અંતિમ એરે(Array) ભાગલા મળે છે, હું મધ્યમાં તે પીવોટ, જમણી બાજુ ડાબી અને ઉપલા ભાગના નીચેના ભાગને જોઈએ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:40)

તો, આ રીતે આપણે સામાન્ય રીતે ઝડપી ક્રમમાં કરીએ છીએ. તેથી, સામાન્ય રીતે હવે યાદ રાખો કે આ ભાગલા કર્યા પછી, આપણે આ ભાગને ઝડપથી સૉર્ટ કરવા અને આ ભાગને ઝડપથી સૉર્ટ કરવા પડશે. તેથી, રિકર્સિવ(recursive) કૉલ્સ વિવિધ સેગમેન્ટોને સૉર્ટ કરશે. તેથી, દરેક કૉલ માટે કહેવું ઉપયોગી છે કે હું કેટલીક ડાબી સીમાથી અમુક અધિકાર મર્યાદામાં સૉર્ટ કરી રહ્યો છું. તેથી, સામાન્ય રીતે ઝડપી ક્રમાંક એરે(Array) લેશે અને તે બે પોઈન્ટર લેશે, તે L થી R માઈનસ 1 થી સૉર્ટ કરશે. હવે, જો આ લંબાઈ નાની હોય, તો બીજા શબ્દોમાં કહીએ તો, મારી પાસે ફક્ત એક જ મૂલ્ય છે, જો R ઓછા છે. તે 1 થી ઓછું અથવા બરાબર છે, જો કાં તો તમે મૂલ્ય જોઈએ અથવા જો મારી પાસે કોઈ મૂલ્ય સૉર્ટ ન હોય, તો પછી હું કંઈ પણ કરતો નથી. તેથી, આ મૂળ કેસ છે. તેથી, આ પુનરાવર્તન મૂલ્ય એલ્ગોરિધમ છે, જો સૉર્ટિંગ, અરેને માત્ર એક જ તત્વ તરીકે સૉર્ટ કરવામાં આવે છે જે આપણે કશું જ નથી કરતા. નહિંતર, અગાઉના ઉદાહરણની પરિભાષાનો ઉપયોગ કરીને, અમે પીળા નિર્દેશકની સ્થિતિને સૂચવવા માટે પીળાનો ઉપયોગ કરીએ છીએ અને લીલી પોઈન્ટરની સ્થિતિને સૂચવવા માટે અમે લીલાનો ઉપયોગ કરીએ છીએ. તો, આ બે વેરીએબલો આ બે તીરની સ્થિતિ સૂચવે છે. તેથી, યાદ રાખો કે આપણે પિવોટનો જમણો ભાગ શરૂ કરીએ છીએ. તેથી, શરૂઆતમાં, આપણે પોઝિશન પર છીએ, આપણી પાસે તે પીવોટ અને આર માઈનસ 1 છેલ્લી શરતો છે. તો, આ આપણું પીવટ પી છે. તેથી, આપણી પ્રારંભિક વસ્તુ એ છે કે આ બંને પોઈન્ટરને અહીં મુકો. તેથી, શરૂઆતમાં પીળા એ પ્લસ 1 છે. તેથી, આ સ્થિતિ છે અને આપણે હવે લીલાને ખસેડવાની શરૂઆત કરીએ છીએ. તેથી, લીલો લ પ્લસ 1 શરૂ થાય છે અને ત્યાં સુધી જાય છે

... તો, જો આપણે જોયું કે લીલા મૂલ્ય પિવોટ કરતાં નાનું છે. તેથી, આ પીવોટ છે. તેથી, એ એક એ પીવોટ છે. તો, લીલો વેલ્યુ, જે વેલ્યુ હું લીલી પોઈન્ટર હેઠળ જોઈ રહ્યો છું તે નાના છે, પછી હું આ એક્સ્ચેન્જ કરું છું. તેથી, જો હું ક્યાંક પહેરીશ અને મારી પાસે આ ગ્રીન વેલ્યુ હશે, તો તે શું થાય છે તે નાના છે, પછી હું આ બે મૂલ્યોનું વિનિમય કરું છું. તેથી, આ તે જ છે, સ્વેપ એ પીળા સ્થાને એલિમેન્ટ અને પોઝિશન લીલી પર તત્વ છે અને એ સ્વેપ હોવું જોઈએ અને પછી, હું પીળા બિંદુને પણ વધારીશ. લીલો પોઈન્ટ કોઈપણ રીતે પીમાં વધારો થયો છે. તો, આખરે, આ લૂપ પછી મેં આ પાર્ટિશનિંગ હદ સુધી કર્યું છે જ્યાં મારી પાસે પીવોટ ઘટક છે, મારી પાસે નીચલું ભાગ અને ઉપલા ભાગ છે. અને હવે હું શું કરવા માંગુ છું, હું પીવટને કેન્દ્રમાં ખસેડવા માંગું છું, આ સમયે મારી પાસે અહીં પીળો પોઈન્ટર છે અને અહીં લીલા પોઈન્ટર છે. તેથી, મારે અહીં છેલ્લું તત્વ લેવાની જરૂર છે અને પિવોટમાં વિનિમય કરવો જોઈએ અને આ તે જ છે. વેલ્યુને વેલ્યુ પોઝિશન l ને વેલ્યૂ પીલ્સે ઓછા 1 પર સ્થિત કરો, હવે મેં આ કર્યું છે, હવે હું વારંવાર સૉર્ટ કરવા માંગું છું. તેથી, હું આ પીળા પોઈન્ટરની શરૂઆતથી જમણેથી ડાબેથી સૉર્ટ કરવા માંગું છું. તેથી, હું એ માંથી A થી પીળામાં સૉર્ટ કરું છું, જ્યારે હું બધું જ જમણી બાજુ સૉર્ટ કરવા માંગું છું. તેથી, હું પીળા વત્તા 1 સાથે સૉર્ટ કરો અને સૉર્ટ કરો. તેથી, આ ઈનપુટ રિકર્સિવ(recursive) કૉલ્સ છે. પરંતુ, હવે અગત્યની બાબત એ છે કે આ સમયે એલ અને પીળા વચ્ચેની દરેક વસ્તુ પિવોટ કરતાં નાની છે, પીળી વત્તા 1 થી ઉપરની દરેક વસ્તુ પિવોટ કરતાં મોટી છે. તેથી, આ બે સૉર્ટિંગ સબ રિકર્સીવ કોલ્સ ક્વિક સૉર્ટ(Quicksort) પછી, કંઈ વધુ કરવાની જરૂર નથી, અમે પૂર્ણ કરી છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:46)
તેથી, જેમ મેં કહ્યું કે આ ભાગલા વ્યૂહરચના અલગ રીતે અમલીકરણ કરી શકાય છે અને હકીકતમાં ટોની હોરે (Tony Hoare)દ્વારા રજૂ કરવામાં આવેલી આ મૂળ ભાગલા વ્યૂહરચના છે. તેથી, આ મૂળ વ્યૂહરચનામાં, આ વિચાર એક અંતથી શરૂ થતો નથી અને જ્યાં સુધી તમે બધા તત્વનો દાવો ન કરો ત્યાં સુધી સફાઈ કરવી, પરંતુ તે વિરુદ્ધ અંત શરૂ થયો. તેથી, તમે કોઈ અર્થમાં ઉભી થવાનું શરૂ કરો છો, તમે અહીંથી નીચલી બાજુનું નિર્માણ કરવાનું પ્રારંભ કરો છો અને તમે અહીંથી ઉપરના ભાગનું નિર્માણ કરવાનું પ્રારંભ કરો છો. અને ધીમે ધીમે નીચલી બાજુ વધે છે, જ્યાં સુધી તે ઉપરની તરફ જાય છે અને પછી બધું જ સ્થળે છે અને પછી, તમે પહેલા જેટલું અંતિમ સ્વેપ કરો છો. તેથી, અહીં તમે શું કરો છો, તમે ફરીથી પ્રારંભ કરો છો તે જ રંગીન વસ્તુનો ઉપયોગ કરશે. તેથી, પીળો એ નીચલા લીલાને સંદર્ભિત કરે છે. તો, હું શું કરીશ, હું પીળા પોઈન્ટર લઈશ અને સ્કેનિંગ રાખીશ જ્યાં સુધી મને તે વેલ્યુ ન મળે જ્યાં સુધી પીળા ન હોય. તેથી, 32 43 કરતા નાના છે. તેથી, યાદ રાખો કે આપણે પીળા ભાગલાને વિકસાવવાનો પ્રયાસ કરી રહ્યા છીએ તે નીચલા ભાગલા છે. તેથી, 43 કરતાં નાના અને નીચલા ભાગલામાં 43 કરતાં મોટા બધા નાના ભાગોને સમાવવાનો પ્રયાસ કરી રહ્યા છે. તેથી, હું પીળા વસ્તુને ચાલુ રાખું છું ત્યાં સુધી મને કોઈ સેટમાં પહેલી ભૂલ મળે છે. તેથી, 32 છે. તેથી, હું તેના પર છોડું છું, 22 43 કરતા પણ નાનું છે. તો, હું તેના ઉપર અવગણો અને હવે હું મૂલ્ય 72 સુધી પહોંચું છું. તેથી, આ મુદ્દો મારા ભાગને અહીં સમાપ્ત કરે છે અને 72 સમાવેલ નથી, કારણ કે તે 43 કરતા વધારે છે. હવે , હું જમણા બાજુથી શરૂ કરું છું અને હું સ્થિતિની શોધ કરું છું, જ્યાં હું ઉપલા વસ્તુમાં વસ્તુઓ શામેલ કરી શકું છું, પરંતુ 13 ની પહેલી વસ્તુ જે હું જોઈ શકું તે ત્યાં હોવી જોઈએ નહીં. તેથી, આ પોઈન્ટર ઉપર ઉપલા સીમા છે. તો, હવે, હું જે કરું છું, હું આ બંને વેલ્યુનું વિનિમય કરું છું, જો હું આ બંને વેલ્યુનું વિનિમય કરું છું, તો આ 13 થશે, આ 78 થશે અને પછી, હું આ બે સીમાઓને 1 વડે ખસેડી શકું. તેથી, આ આ ભાગલા કરવાની વ્યૂહરચનામાં મૂળભૂત પગલું છે. તેથી, હું જે કરું છું તે આગળનું પગલું છે, હું 13 અને 78 નું વિનિમય કરું છું અને ના, હું કહું છું કે મારી પાસે અહીં નીચી વસ્તુ છે અને મારી પાસે ઉપરની વસ્તુ છે. તેથી, હવે આ અતિક્રમણ છે, અમારી પાસે ડાબી બાજુ અને ઉપરની બાજુની જમણી બાજુ માટે નીચલી વસ્તુ છે અને વચ્ચેના ભાગમાં આપણી પાસે ક્રમિક તત્વો છે, પરંતુ આપણી પાસે આ બે સૂચકાંકો છે, ડાબે સૌથી વધુ ડાબી બાજુનું સૌથી વધુ નિષ્કર્ષણ ઘટક છે, સૌથી વધુ નિષ્કર્ષણ તત્વ. તેથી, હવે, હું ફરી એક જ વસ્તુ કરવાનું શરૂ કરું છું, હું પીળા જમણા હલનચલન કરી શકું છું, જ્યાં સુધી હું નીચું લંબાવું નહીં ત્યાં સુધી, હું તેને ગમે ત્યાં સુધી વિસ્તૃત કરી શકતો નથી, કારણ કે 63 ત્યાં પહેલેથી જ હોવું જોઈએ નહીં. તેથી, હું આ ભાગલાને ખસેડી શકતો નથી. બીજી તરફ, ઉપલા ભાગલા ખસેડી શકે છે, કારણ કે 91 વધારે મોટું છે. તેથી, હું તેને છોડી દઈશ, 57 હજુ પણ મોટું છે. તેથી, હું તેને ફરીથી ખસેડીશ, 63 હજુ પણ મોટું છે. તેથી, હું ફરીથી ડાબે ખસેડો. અને હવે મને લાગે છે કે યોગ્ય પાર્ટિશન સૂચક ડાબે પાર્ટિશનની ડાબે ખસેડ્યું

છે. તેથી, જ્યારે આ વિનિમય થાય છે, ત્યારે તે આ ભાગલાને સમાપ્ત કરે છે. તેથી, જ્યારે જમણી સીમા ડાબી બાજુ પાર કરે છે, જ્યારે આપણે તત્વનું વિભાજન સમાપ્ત કરીએ છીએ, કારણ કે બે ઘટકો વચ્ચે ભાગલા કરવાનું કંઈ જ નથી. તો, એક વખત આપણે સમાપ્ત થઈ ગયા પછી, હવે આપણી પાસે આ જ સમસ્યા છે જે પહેલા આપણે આ તત્વને આ કેન્દ્રમાં ખસેડવા માંગીએ છીએ, પરંતુ હવે યાદ રાખો કે આ બિંદુએ, જ્યારે આ વસ્તુ સમાપ્ત થઈ જાય છે, ત્યારે જમણી બાજુ તરફ નિર્દેશ કરે છે નિમ્ન સીમાનો અંત બિંદુ. તેથી, હું ફક્ત આ બંનેનું વિનિમય કરી શકું છું. તેથી, હું 13 અને 43 નું વિનિમય કરી શકું છું. તેથી, હું તેને અહીં ખસેડવા માટે લઈ ગયો છું અને પછી, મારો જવાબ મળી ગયો છે. તેથી, જો હું આને ખસેડે અને પછી, હું ગ્રીન પોઈન્ટરને એકસાથે ખસેડીશ, હવે મારી પાસે નિમ્ન ઘટકોની ઉપરની બાજુ છે અથવા ઉપલા તત્વોના પહેલા તરફની તરફેણ છે અને હવે હું આ ભાગમાં ઝડપથી સૉર્ટ કરી શકું છું. આ ભાગ. તેથી, અમે સ્યુડો કોડ લખીશું નહીં અથવા આનું વર્ણન કરીશું નહીં, એલ્ગોરિધમ અને વધુ વિગત, પરંતુ તમે ચોક્કસપણે આ નિર્દેશોને આગળ વધતા રાખવા જેવી જ રીતને અજમાવવા અને કાર્ય કરવાનો પ્રયાસ કરી શકો છો, જેમ કે અમે અગાઉના ભાગલિંગ માટે કર્યું હતું અને જુઓ કે તમે તેને બરાબર મેળવી શકો છો, આ ઘણી પુસ્તકોમાં પણ ચર્ચા થાય છે. તો, આ બંને ભાગલા એલ્ગોરિધમ છે જે આપણી પાસે પાઠ્યપુસ્તકોમાં છે અને તમે પસંદ કરી શકો છો કે જે ક્યારેય વધુ સરળ છે. બંને કિસ્સાઓમાં યાદ રાખો કે તેઓ મૂળભૂત ચલ સ્થિતિ છે, આ બે માર્કર્સ છે અને આ બે માર્કર્સ એ ભાગ દર્શાવે છે જે ભાગલા, નીચલા અને ઉપરના ભાગની મર્યાદા પહેલાથી જ છે. અને પછી, ત્યાં એક અવિભાજ્ય ભાગ છે અને તે પછી, તમે જે કર્યું છે તે પક્ષપાતી ભાગ ખાલી થઈ જાય છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 08
લેક્ચર - 16
ક્વિક્સૉર્ટ: એનાલિસિસ (Quicksort: Analysis)

અમે ક્વિક્સૉર્ટ(Quicksort) જોયું છે જે એક વિભાજક છે અને એલ્ગોરિધમ જીતી છે જે મર્જ સૉર્ટ(Merge sort)માં વધારાની અૅરે માટે આવશ્યકતાને વધારે છે. . તેથી, ચાલો ક્વિક્સૉર્ટ(Quicksort)ના વિશ્લેષણ કરીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:11)
તેથી, યાદ રાખો કે ક્વિક્સૉર્ટ(Quicksort) કેવી રીતે કાર્ય કરે છે, તમે પીવોટ ઘટક પસંદ કરો છો જે સામાન્ય રીતે અૅરે(Array)ના આ પ્રથમ તત્વને કહે છે. અને પછી તમે શું કરો છો, તમે આને બે ભાગમાં વિભાજિત કરો છો જેમ કે તમારી પાસે નીચલું ભાગ છે જે પી કરતાં ઓછો અથવા બરાબર છે અને તે ઉપલા ભાગ હોઈ શકે છે, તે p કરતાં મોટો છે. અને તમે આ પિવોટને વચ્ચે વચ્ચે ખસેડો અને પછી તમે આ નીચલા ભાગ અને ઉપલા ભાગને અલગથી સૉર્ટ કરો અને પછી તમારે કોઈ સંયોજન પગલું કરવાની જરૂર નથી, કારણ કે આ બે વસ્તુઓ એકબીજાને ધ્યાનમાં રાખીને યોગ્ય સ્થિતિમાં છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:45)
તો, પ્રથમ વસ્તુ જે આપણે નોંધ્યું છે તે છે કે આ ભાગલા ખરેખર કાર્યક્ષમ છે. અમે આખા અૅરે(Array)ના એક સ્કેનમાં કરી શકીએ છીએ. તેથી, આપણે n સમયે ક્રમમાં કોઈપણ પિવોટના સંદર્ભમાં ભાગલા કરી શકીએ છીએ. તેથી, પ્રશ્ન એ છે કે વારંવાર આવતી સમસ્યાઓ કેટલી મોટી છે? તેથી, જો પીવોટ મધ્યમ હોય તો તમે અપેક્ષા રાખશો કે મધ્યમાં વ્યાખ્યા પ્રમાણે આ કદ 2 ના છે. કારણ કે સરેરાશ એ તે તત્વ છે જે અૅરે(Array)ને બે ભાગમાં વિભાજિત કરે છે, તે તત્વોના અડધા કરતાં વધારે છે. મધ્યમ, મધ્યમ કરતાં અડધા નાના હોય છે. અને જો તમારી પાસે આ નસીબદાર પરિસ્થિતિ છે કે પિવોટ મધ્યમ છે, તો આપણે મર્જ સૉર્ટ(Merge sort) પુનરાવર્તન સાથે સમાપ્ત થાય છે જે કહે છે કે n ના t ને 2 ભાગો 2 બે ભાગો માટે લે છે અને આ પાર્ટિશનિંગ પગલાં છે. તેથી, પુનરાવર્તન પછી મર્જ પગલું નથી, પરંતુ પુનરાવર્તન પહેલાં ભાગલા સેટ. તેથી, આપણી પાસે મર્જ સૉર્ટ(Merge sort)માં જોયું છે, આ પુનરાવર્તન ઑર્ડર n લોગ n લે છે જો આપણે તેને વિસ્તૃત કરીએ, પરંતુ પિવૉટ કોઈ અર્થમાં શ્રેષ્ઠ કેસમાં છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:44)
આપણે શું સૌથી ખરાબ કેસ કરીએ છીએ? જ્યારે સૌથી ખરાબ કેસ જ્યારે પીવોટ ભારે મૂલ્ય હોય ત્યારે, સૌથી નાનો મૂલ્ય અથવા સૌથી મોટો મૂલ્ય. તેથી, જો તે સૌથી નાનું મૂલ્ય છે તો શું થશે તે એ છે કે બધું પીવોટ કરતા મોટું હશે. તેથી, તમારી પાસે ઉપલા તત્વ સમૂહ હશે જેની પાસે n ઓછા 1 મૂલ્યો છે, કારણ કે પીવોટ એ એક નાનું મૂલ્ય છે અને અમારી પાસે આ બાજુ કંઈ નથી. સપ્રમાણતાપૂર્વક, જો પીવૉટ તમારા અૅરે(Array)માં સૌથી મોટું મૂલ્ય છે, તો તમારી પાસે નીચલા તત્વ સમૂહમાં બધું હશે. તો, આ ફરીથી n ઓછા 1 ની સાથે છે અને પીવટ એ કંઈક છે જે એક અત્યંત અંતમાં છે અને બીજી તરફ કંઈ નથી. તેથી, હવે આપણે જોઈ શકીએ છીએ કે આ માપ ના અૅરે(Array)ને સૉર્ટ કરવા માટે, મારે એક નાનું સેગમેન્ટ સૉર્ટ કરવું પડશે જે ફક્ત n ઓછા 1 છે, તે n ઓછા 1 કરતા વધારે નાનું છે. તો, આપણું tn tn માઈનસ 1 વત્તા લે છે n, n એ ભાગલા કરવા માટેનો સમય છે અને ટીન 1 ઓછામાં ફરીથી ખરાબ કેસમાં એક પીવોટ ઘટક હશે જે અત્યંત મૂલ્ય છે. તેથી, ઉદાહરણ તરીકે, માનીએ છીએ કે આપણે પહેલેથી સૉર્ટ કરેલા અૅરે(Array)થી પ્રારંભ કરીએ છીએ જેમ કે 1, 2, 3, 4, તો તે શું થાય છે, આપણે 1 ને પિવોટ તરીકે પસંદ કરીએ છીએ અને પછી આ તે પરિણામ આપે છે જેને આપણે 2, 3, 4 અને ત્યારબાદ હું 2 પીવોટ તરીકે પસંદ કરીશ અને આ આપણા સૉર્ટમાં પરિણામરૂપ થશે 1 તે 3, 4 અને તેથી, સૉર્ટ કરશે. જો તમારી પાસે કોઈ અર્થમાં પહેલાથી સૉર્ટ થયેલ અૅરે હોય, તો પિવોટ હંમેશાં એક આત્યંતિક મૂલ્યો છે. તેથી, આગલું પગલું અૅરે(Array)ને ખૂબ ખરાબ વિભાજિત કરે છે. અને અલબત્ત, આપણે આ ટીનને વિસ્તૃત

કરીએ છીએ, tn માઈનસ 1 વત્તા એન, આપણને સારાંશ મળે છે જે આપણને પ્રથમ પસંદગી સૉર્ટ(selection sort) અને નિવેશ સૉર્ટ(insertion sort) માટે મળે છે. તેથી, આ ક્રમ n ચોરસ બને છે. તેથી, ક્વિકસૉર્ટ(Quicksort)નો સૌથી ખરાબ કેસ વાસ્તવમાં n ચોરસ છે, જે પસંદગી સૉર્ટ(selection sort) અને નિવેશ સૉર્ટ(insertion sort) માટે સૌથી ખરાબ કેસ જેવું જ છે. તેથી, આપણે આ વધુ જટિલ અલ્ગોરિધમ ક્વિકસૉર્ટ(Quicksort)થી કેમ ચિંતા કરીએ છીએ, જ્યારે આપણે પહેલેથી જ ઘણા સાહજિક અલ્ગોરિધમનો જાણ કરીએ છીએ જેની પાસે ક્રમ n ચોરસ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:36)

તેથી, તે તારણ આપે છે કે આપણે જે વાસ્તવમાં બતાવી શકીએ છીએ તે ક્વિકસૉર્ટ(Quicksort) ખરેખર આ ખરાબ કેસમાં ખૂબ જ વારંવાર વર્તે નથી. તેથી, આપણે ખરેખર ક્વિકસૉર્ટ(Quicksort)ના કિસ્સામાં ગણતરી કરી શકીએ છીએ જેને એવરેજ કેસ જટિલતા કહેવામાં આવે છે અને બતાવે છે કે આ n લોગ n છે. તેથી, આપણે વાસ્તવમાં બતાવીશું નહીં કે તે n લોગ n છે, પરંતુ અમે ક્વિકસૉર્ટ(Quicksort)ના સરેરાશ કેસ વિશ્લેષણની ગણતરી કરવા માટે તેનો અર્થ શું છે તે સમજાવવાનો પ્રયાસ કરીશું. જેમ આપણે શરૂઆતમાં કહ્યું હતું તેમ ગણતરી કરવા માટે સરેરાશ કેસ ખૂબ મુશ્કેલ છે. તેથી, ચાલો જોઈએ કે આ કરવાનું શું છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:09)

તેથી, સરેરાશ કેસની ગણતરી કરવી મુશ્કેલ છે તે પ્રથમ કારણ છે, કારણ કે અમને તમામ સંભવિત ઈનપુટ્સનું વર્ણન કરવાનો માર્ગ હોવો જરૂરી છે. હવે, સૉર્ટિંગ એલ્ગોરિધમ માટે પણ તમામ સંભવિત ઈનપુટ્સ એક અનંત જગ્યા છે, ધારી રહ્યા છીએ કે હું માત્ર નિશ્ચિત રેખાના એરે(Array) લઈ જાઉં છું, હું ધારું છું કે હું 4 લંબાઈની એરે(Array) લઈશ. તેથી, મારી પાસે એરે(Array) હોઈ શકે જે 43, 12, 38 અને પછીની જેમ દેખાય તો, આ 4 ઘટકો સાથે એક એરે(Array) છે, મારી પાસે 4 તત્વોનું બીજું એરે(Array) હોઈ શકે છે જે 72, 21, 63 અને 95 કહે છે. પરંતુ, આ રીતે આપણે ચાલુ રાખી શકીએ છીએ અને કોઈપણ ઘટકો મૂકી શકીએ છીએ અને તેમાં અસંખ્ય ભૂલો છે કદ 4. પરંતુ, આ વચ્ચે એક સામાન્યતા છે, જે કહે છે કે પ્રથમ તત્વ બીજા તત્વ કરતાં મોટો છે. હકીકતમાં, બીજો તત્વ એ સૌથી નાનો તત્વ છે અને તેથી. તેથી, જો તમે આ જુઓ તો અમે કહી શકીએ કે ત્યાં 4 તત્વો છે અને આપણે તેમના વિશે વિચારીએ છીએ, પછી સૌથી નાનો તત્વ અહીં છે, બીજો સૌથી નાનો તત્વ અહીં છે, ત્રીજો સૌથી નાનો તત્વ અહીં છે અને ચોથા નાનો તત્વ અહીં છે . તો, હું ખરેખર આને એરે(Array) 3, 1, 2, 4 તરીકે વિચારી શકું છું, કારણ કે 4 તત્વો અને 4 તત્વો આ રીતે આદેશિત છે. તેથી, વાસ્તવિક મૂલ્યો ફક્ત સંબંધિત સંબંધિત બાબતો માટે મહત્વપૂર્ણ નથી. તેથી, આપણે વાસ્તવમાં કદના ઈનપુટ્સ વિશે વિચાર કરી શકીએ છીએ અને 1 થી એન અથવા ક્રમ 1 ના ક્રમચયના આ પ્રકારના ક્રમમાં હોઈ શકીએ છીએ. હવે, આ ક્રમચયોમાં અમારી પાસે કોઈ પસંદગી નથી, તેમાંથી કોઈ પણ આપણા ઈનપુટ તરીકે આવશે. તેથી, આપણે બધા જાણીએ છીએ કે ત્યાં n ફેક્ટરીયલ આવી ક્રમચયો છે અને આપણે કહીએ છીએ કે તેમાંના દરેક સમાન સમાન છે. તેથી, પ્રત્યેક પ્રત્યેક n એ ફેક્ટોરિયલ દ્વારા સંભવિત છે. હવે, આપણે કદ n ના આ બધા n ફેક્ટોરિઅલ ઈનપુટ્સને જોઈએ છીએ અને જુઓ કે આપણું એલ્ગોરિધમ કેવી રીતે વર્તે છે. તેથી, આપણે વાસ્તવિક ગણતરી કરીશું નહીં, પરંતુ જો તમે સરેરાશ જુઓ છો, તો તમે બધા n ફેક્ટોરિઅલ ઈનપુટ્સ માટે જે વાસ્તવિક સમય લે છે તે જોશો, n ફેક્ટોરિઅલ દ્વારા ઉમેરવામાં અને વિભાજીત કરવું, જે સંભવિતતામાં છે જે સિવાયના રનિંગની ગણતરી તરીકે ઓળખાય છે. સમય. પછી, તમે બતાવી શકો છો કે આ ખરેખર ક્રમ n લોગ n છે. તો, આપણે તેને બતાવ્યું નથી, આપણે ફક્ત બતાવવા માટે ગણિત જરૂરી છે તે સમજાવીશું. પરંતુ, ક્વિકસૉર્ટ(Quicksort)માં તમે સાબિત કરી શકો છો કે સંભવિત ચાલી રહેલ સમય બધા સંભવિત રેન્ડમ ઈનપુટ્સમાં સમાન ઈનપુટ્સ, આ ખરેખર ઑર્ડર n લોગ n છે. તેથી, જો કે ક્વિકસૉર્ટ(Quicksort)માં ઓ એન સ્ક્વેર્ડ સૌથી ખરાબ કેસ છે અને સરેરાશ તે મર્જ સૉર્ટ(Merge sort) જેવા હોય છે અને મર્જ સૉર્ટ(Merge sort)ના કેટલાક ખાડો વિના તે વર્તે છે, તે ખાસ કરીને મર્જ એરે(Array) બનાવવા માટે વધારાની જગ્યાની આવશ્યકતા નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:41)

હવે, તમે ખરેખર આ સરેરાશ કેસ વર્તણૂંકનો ખૂબ જ સરળ રીતે ઉપયોગ કરી શકો છો. તેથી, આ ખરાબ કેસ શા માટે થાય છે? સૌથી ખરાબ કેસ થાય છે, કારણ કે અમે જે પીવોટ પસંદ કરીએ છીએ તે ખરાબ પિવોટ હોઈ શકે છે, કારણ કે જોયું કે

જો તમે પહેલો તત્વ તમારા પિવોટ તરીકે મૂક્યો છે, તો સૉર્ટ કરેલ એરે(Array) એક ખરાબ કેસ બની જાય છે, કારણ કે દર વખતે જ્યારે પિવોટ એ અત્યંત તત્વ છે. બીજી બાજુ, તમે છેલ્લો તત્વ લઈ શકો છો અને તમને સમાન સમસ્યા હશે, જો તમે મિડપોઈન્ટ ફરીથી પસંદ કરો છો, તો તમે એરે(Array)ના મધ્ય બિંદુને બનાવી શકો છો જે તમે અત્યંત તત્વથી પ્રારંભ કરો છો અને પછી તમે પાછળથી કામ કરી શકો છો અને હંમેશાં રચના કરી શકો છો. સૌથી ખરાબ કેસ જે ક્રમાંક n ચોરસ લે છે. તેથી, આપણે જે કહી રહ્યા છે તે છે કે કોઈપણ નિયત વ્યૂહરચના માટે, જો હું તમને અગાઉથી જણાવીશ કે હું હંમેશાં પિવૉટની સ્થિતિને ચોક્કસ રીતે ગણતરી કરવા જાઉં છું, તો પાછળથી કામ કરીને તમે હંમેશાં ખાતરી કરી શકો છો કે વર્તમાન સ્થિતિ સમસ્યા, તમારી પાસે સૌથી ખરાબ કેસ છે જે અત્યંત ઈનપુટ છે અને તે કંઈક ફરીથી બનાવવું જે તે વ્યૂહરચના માટે O n ચોરસ લેશે. તેથી, ઉકેલ એ છે કે, જ્યારે હું રિકર્સિવ(recursive) સબ સમસ્યામાં ક્વિક્સૉર્ટ(Quicksort)ને લાગુ કરવા માગતો હોઉં ત્યારે દરેક વખતે વ્યૂહરચનાને ઠીક કરવી નહીં, મારી પાસે 0 થી n ઓછા 1 પોઝિશન હોય છે જે મને પીવોટ તરીકે પસંદ કરવાની જરૂર છે. પરંતુ, તમને 0 અથવા n ઓછા 1 અથવા 0 અને n ઓછા 1 વચ્ચેના મધ્ય-માર્ગ હોવાનું કહેવા કરતાં, હું કહું છું કે હું સમાન સંભાવના સાથે આમાંથી કોઈપણ મૂલ્ય પસંદ કરીશ. તેથી, આનો વિચાર કરો, હું 0 અને એન 1 ઓછા 1 ની વચ્ચે રેન્ડમ નંબર પસંદ કરી રહ્યો છું અથવા જો તમે ગ્રાફિકલી રીતે વિચારી શકો છો તો તે મૃત્યુ પર પસાર થવું અથવા ફેંકવું જેવી છે. તેથી, મરી સામાન્ય રીતે છ ચહેરાઓ કહે છે. તેથી, જો તમે ભાડેથી મૃત્યુ પામે, તો તમને 1 થી 6 ની વચ્ચેની કોઈ પણ સંખ્યા સંભવિત હશે. તેથી, હવે આપણી પાસે એક બાજુનો મૃત્યુ છે. તેથી, અમારી પાસે એક જટિલ વસ્તુ છે, આપણે તેને ફેંકીએ છીએ અને જે પણ સંખ્યા આવે છે, આપણે તેને એક પીવોટ તરીકે બનાવ્યો છે. તેથી, હવે આ અલ્ગોરિધમનો વર્તણૂક સુધારાઈ નથી, તે કેવી રીતે આ મૃત્યુ પામે છે તેના પર નિર્ભર છે. તેથી, આ એક અલગ પ્રકારનો અલ્ગોરિધમનો છે જે રેન્ડમડાઈઝ્ડ અલ્ગોરિધમ(Randomized Algorithm) કહેવાય છે. તેથી, હવે તમે ક્વિક્સૉર્ટ(Quicksort)ને રેન્ડમડાઈઝ્ડ સ્પીડમાં ખૂબ સરળ રેન્ડમલાઈઝેશન પગલા સાથે અમલમાં મૂકી શકો છો, એટલે કે દરેક ક્વિક્સૉર્ટ(Quicksort) પર રેન્ડમ પર ફક્ત પિવોટ પસંદ કરો. અને તે તારણ આપે છે કે ફરીથી તમે સમાન ગણતરી કરી શકો છો, કહીને કે શક્ય બધી રેન્ડમ પસંદગીઓમાં હું પાઈવોટ બનાવવા માંગું છું, અપેક્ષિત રનિંગ સમય ઓર્ડર n લોગ n છે. તેથી, આ ખૂબ જ સરળ છે, આ એવરેજ કેસો n લોગ n ની હકીકતનું એક દ્વિ પરિણામ છે, તમે સારા સંભાવના સાથે આ n લોગ n વસ્તુને પ્રાપ્ત કરવા માટે ખૂબ જ સરળ રેન્ડમાઈઝ્ડ વ્યૂહરચના બનાવીને તેનું શોષણ કરી શકો છો. .

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:05)

આપણે જે પ્રકારનું મર્જ કર્યું છે તેના વિશેનું બીજું પાસું, જે થોડી સીમિત છે, એ છે કે તે સ્વાભાવિક રૂપે રિકર્સિવ(recursive) છે. હવે, ક્વિક્સૉર્ટ(Quicksort)નો અમારો ઉકેલ જગ્યાના આ ડુપ્લિકેશનને અવગણે છે, પરંતુ તે રિકર્સિવ(recursive) છે. હવે, તે ક્વિક્સૉર્ટ(Quicksort)માં જોવા મળે છે તમે વાસ્તવમાં રિકર્સિવ એલ્ગોરિધમ પુનરાવર્તન કરી શકો છો. તેથી, મુદ્દો એ છે કે રિકર્સિવ(recursive) કૉલ્સ વિવાદિત સેગમેન્ટ્સ પર કાર્ય કરે છે. તેથી, તમારે રિકર્સિવ(recursive) કૉલમાં યાદ રાખવાની જરૂર છે તે સંપૂર્ણ સેગમેન્ટ નથી, પરંતુ ફક્ત કયા સેગમેન્ટ પર તમારે કામ કરવાની જરૂર છે, તમારે પરિણામોને જોડવાની જરૂર નથી. તેથી, અમે ખૂબ વિગતવાર ચર્ચા કરીશું નહીં, પરંતુ તે તારણ આપે છે કે તમે સ્ટેકનો ઉપયોગ કરી શકો છો, તમે વાસ્તવમાં તમારા પોતાના સ્ટેકને જાળવી શકો છો અને જ્યારે પણ તમે રિકર્સિવ(recursive) કૉલ કરો છો, ત્યારે તમે ફક્ત સ્ટેક, ડાબે અને જમણો અંત પોઈન્ટ સંગ્રહિત કરો છો કયા સેગમેન્ટને સૉર્ટ કરવાની જરૂર છે. અને આ રીતે તમે ખરેખર રિકર્સિવ એલ્ગોરિધમનો ઉપયોગ કરી શકો છો જે આપણે પહેલા લખ્યું હતું અને તેને રિકર્સિવ(recursive) અલ્ગોરિધમનો રૂપાંતરિત કરીશું. હવે, તમે આને સામાન્ય સૂચિમાં શા માટે કરવા માંગો છો, કારણ કે તમારી પ્રોગ્રામિંગ ભાષાના આધારે પુનરાવર્તન વિરુદ્ધ પુનરાવર્તનમાં તમારી પાસે વેપાર બંધ છે. કારણ કે, જ્યારે તમે રિકર્સિવ કૉલ કરો છો, જ્યારે તમે સામાન્ય રીતે પ્રોગ્રામિંગ ભાષામાં કોઈ ફંક્શન કૉલ કરો છો, ત્યારે શું થાય છે તે તમે જે કમ્પ્યુટિંગની ગણતરી કરો છો તે સસ્પેન્ડ કરવાની છે. તેથી, તમારે સસ્પેન્ડ કરવાની અને પ્રાપ્ત કરવાની જરૂર છે. તેથી, જ્યારે તમે રિકર્સિવ કૉલ કરો છો ત્યારે તમારે તમારી પાસે જે કંઈપણ મૂકી દો અને પછી તમારે સ્થાનિક વેરિયેબલનો એક નવો સેટ લેવો પડશે. તેથી, પ્રોગ્રામની યાદમાં તમારે કેટલાક નવા ડેટાને લોડ કરવું પડશે, પછી તમારે ફંક્શનને એક્ઝેક્યુટ કરવું પડશે, તમે સમાપ્ત કરવા માંગો છો, તમારે તેને ફેંકવું અને સંદર્ભને પુનર્સ્થાપિત કરવું પડશે, તમારે

ફરીથી શરૂ કરવું પડશે. તેથી, આ થોડો સમય લે છે અને તે કેટલાક સંસાધનો લે છે અને તેથી, સામાન્ય રીતે ફંક્શન કૉલ કરવાની કિંમત, પછી ભલે આપણે તેની જટિલતામાં પાયાની કામગીરી તરીકે ગણતરી કરી શકીએ, તે ખરેખર કેટલાક અંકગણિત ક્રિયા જેવા કે ઉમેરણ અથવા કંઈક . તેથી, જ્યારે તમે વારંવાર રિકર્સિવ કૉલ કરો છો ત્યારે, તમે મૂળભૂત રીતે સ્ટેક પર કંઈક બદલી રહ્યા છો અને બદલી રહ્યા છો તે કેટલીક નવી ફ્રેમ છે અને પછી તેને પાછું મૂકીને આ સમય લે છે. તેથી, તે સામાન્ય રીતે કાર્યક્ષમ હેતુઓ માટે સામાન્ય છે, પુનરાવર્તનને રિકર્સિવ(recursive) કરવા માટે સારું છે. બીજી બાજુ, આ પ્રક્રિયા એલ્ગોરિધમને વધુ અસ્પષ્ટ બનાવી શકે છે અને અસંખ્ય પ્રોગ્રામિંગ ભાષાઓ વાસ્તવમાં ઑપ્ટિમાઈઝ કરવામાં આવે છે, કમ્પાઈલર્સ(compilers) આ આપમેળે કરવા માટે પ્રયાસ કરી શકે છે. તેથી, રિકર્ઝન વચ્ચે આ તફાવત હોઈ શકે છે અને પુનરાવર્તન હંમેશાં હંમેશાં મદદ કરતું નથી. પરંતુ, તે જાણવું ઉપયોગી છે કે અમુક એલ્ગોરિધમ્સ બંને રીતે કરી શકાય છે અને ચોક્કસ અલ્ગોરિધમનો અહીં એક કરવાનું મુશ્કેલ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:26)

તેથી, હવે આપણે ક્વિક્સૉર્ટ(Quicksort) છોડીએ તે પહેલાં અમારી અંતિમ ટિપ્પણી. તેથી, વ્યવહારમાં ક્વિક્સૉર્ટ(Quicksort) ખૂબ ઝડપી છે, કારણ કે આપણે કહ્યું છે કે સૌથી ખરાબ કેસ ખૂબ ભાગ્યે જ થાય છે. આ કારણોસર, સામાન્ય રીતે ક્વિક્સૉર્ટ(Quicksort) એ ડિફૉલ્ટ એલ્ગોરિધમ છે જે તમે જુઓ છો કે લોકો જ્યારે બિલ્ટ ઈન સૉર્ટ ફંક્શન ધરાવે છે ત્યારે તેનો ઉપયોગ કરે છે. તેથી, જો તમારી પાસે કોઈ સ્પ્રેડ શીટ હોય અને તમને કૉલમને સૉર્ટ કરવાની મંજૂરી આપે, તો સામાન્ય રીતે આ કૉલમને સૉર્ટ કરવા માટે તમારી પૃષ્ઠભૂમિમાં આ અલ્ગોરિધમનો ઉપયોગ ચાલી રહ્યો છે અથવા જો તમે સૉર્ટ ફંક્શનમાં બિલ્ટ કર્યું છે. ઉદાહરણ તરીકે, સી, સી વત્તા પ્લસ, જાવા તમને બધાને સૉર્ટ કરવા માટે પરવાનગી આપે છે, પણ પાયથોન ફક્ત તમને સૉર્ટ કરવા માટે પરવાનગી આપે છે. લગભગ કોઈપણ પ્રોગ્રામિંગ ભાષામાં, આ પ્રકારનું કાર્ય પ્રોગ્રામરને ફક્ત એક સરળ કૉલ દ્વારા ઉપલબ્ધ છે તે સામાન્ય રીતે ક્વિક્સૉર્ટ(Quicksort)નું અમલીકરણ છે. અલબત્ત, આ અમલીકરણ વિવિધ ઑપ્ટિમાઈઝેશનનો ઉપયોગ કરી શકે છે જેમ કે રેન્ડમાઈઝેશન અને અન્ય વસ્તુઓ તેને ઝડપી બનાવવા માટે, પરંતુ અંડરલાઈન કરેલ અલ્ગોરિધમ પર તેનું હૃદય સામાન્ય રીતે ક્વિક્સૉર્ટ(Quicksort) છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 09
લેક્ચર - 17
સૉર્ટિંગ: સમાપનની રીમાર્કસ

ચાલો આપણે વિચારીએ છીએ કે વિવિધ સૉર્ટિંગ એલ્ગોરિધમ્સ પર પાછા જુઓ.
(સ્લાઈડ્સમયનો સંદર્ભ લો: 00:07)
તેથી, એક મહત્વપૂર્ણ માપદંડ કે જે આપણે ભૂલશો નહીં તે એ છે કે, સૉર્ટિંગ વારંવાર તબક્કામાં થાય છે. તેથી, અમારી પાસે અશ્વિન, ભારતી, ચંદર અને દીપા (Aswin, Bharathi, Chander and Deepa) લોકોના નામની સૂચિ હોઈ શકે છે. તેથી, અમે આ ક્રમમાં ગોઠવ્યા છે અને તેમને પરીક્ષામાં કેટલાક ગુણ મળી શકે છે. તેથી, તેમને 60, 43, 60 અને 95 મળી શકે છે, હવે આપણે તેને ગુણ દ્વારા સૉર્ટ કરવા માંગીએ છીએ. તેથી, જ્યારે આપણે અલબત્ત ગુણ દ્વારા આને સોર્ટ કરીએ છીએ, ત્યારે આપણે ભારતી અને અશ્વિનની સ્થિતિનું વિનિમય કરવાની જરૂર છે, પરંતુ આપણે અશ્વિન અને ચંદરની સ્થિતિનું વિનિમય કરવા નથી માંગતા. તેથી, બીજા શબ્દોમાં કહીએ તો અમે મૂળાક્ષરોના હુકમોના સૉર્ટિંગને વિક્ષેપિત કરવા નથી માંગતા. તેથી, અંતિમ જવાબમાં ભારતી પાસે 43, અશ્વિન 60, ચંદર 60 અને દીપાની પાસે હોવું જોઈએ, પરંતુ જો તમે ચંદ્રર અને અશ્વિન ખોટું બોલ્યા હોત તો તે ખોટું હશે, પરંતુ અનિચ્છનીય છે. તેથી, જો આપણી પાસે આલ્ફાબેટિકલ ક્રમમાં વિદ્યાર્થીઓની સૂચિ હોય અને પછી અમે ગુણ દ્વારા સૉર્ટ કરીએ, તો અમને આશા છે કે સમાન ગુણવાળા લોકો હજી પણ આલ્ફાબેટિક ક્રમમાં ગોઠવવામાં આવે છે. તેથી, સ્પ્રેડશીટ(spreadsheet)માં જ્યાં આપણી પાસે કોલમ સાથે એક કોષ્ટક હોય, તેથી આપણે આ સ્તંભ દ્વારા સૉર્ટ કરીએ છીએ અને પછી આપણે આ કૉલમ દ્વારા સૉર્ટ કરીએ છીએ, બીજી સૉર્ટિંગ પહેલા સૉર્ટિંગના ક્રમમાં વિક્ષેપ ન લેવી જોઈએ. તેથી, આને સ્થિર સૉર્ટિંગ કહેવામાં આવે છે જે ઘણી વખત જ્યારે તમે એક એટ્રીબ્યુટ(Attribute) પર સૉર્ટિંગને સૉર્ટ કરી રહ્યા હોય, પરંતુ અન્ય લક્ષણો હોઈ શકે છે. તેથી, ડેટા આઈટમ સામાન્ય રીતે એક અનન્ય મૂલ્યો નથી, તે માત્ર એક સંખ્યા નથી, પરંતુ તે છે ... તેથી, તમે નામ, ગુણ અને અન્ય સરનામાંઓ લઈ રહ્યા છો તે ફોન નંબર હોઈ શકે છે અને તમે તફાવત પર સૉર્ટ કરી શકો છો. તો, સ્પ્રેડશીટ વિશે વિચારો જ્યાં આપણી પાસે વિવિધ સ્તંભ હોઈ શકે છે. તેથી, દરેક અનુગામી સૉર્ટ પહેલાનાં સૉર્ટને વિક્ષેપિત ન કરે, તેથી તેને સ્થિર સૉર્ટિંગ કહેવામાં આવે છે. તેથી, આ દેખીતી રીતે ઈચ્છનીય છે, કારણ કે આપણે તે હંમેશાં કરીએ છીએ, જો સૉર્ટિંગના આગલા રાઉન્ડમાં સૉર્ટિંગના પ્રથમ રાઉન્ડને વિક્ષેપિત કરવામાં આવે તો આપણે ખરેખર ખૂબ નાખુશ હોઈશું. તેથી, જે પણ એલ્ગોરિધમ સ્થિર છે.
(સ્લાઈડ્ટાઈમ નો સંદર્ભ લો: 02:10)
તેથી, બતાવ્યા પ્રમાણે ઝડપી સૉર્ટ(Quick Sort) સ્થિર સ્થાયી નથી, તેથી યાદ રાખો કે આપણે તે થોડીક વાર લીધું છે અને ઓછામાં ઓછા આપણા અમલીકરણમાં આપણે શું કર્યું છે તે અમે આ નીચલા સેટને બનાવ્યું હતું અને પછી તેના ઉપર જમણે સેટ, તેથી તે સાચું છે, આ તે ચિત્ર હતું અને પછી આખરે, અને આ નિર્ણાયક બિંદુ છે, આપણે આ સ્વેપ કરીએ છીએ. તેથી, કલ્પના કરો કે અમારી પાસે એક એવી સ્થિતિ છે જ્યાં આપણે અહીં ઉદાહરણ તરીકે હતા, આ મુદ્દે અશ્વિન 60 અંક સાથે અને ચંદર સાથે 60 ગુણ સાથે પણ છે. તેથી, નીચલા સમૂહના નિર્માણ સમયે, તેઓ મૂળ ઈનપુટમાં યોગ્ય ક્રમમાં હતા. પરંતુ, હવે શું થઈ રહ્યું છે તેનો સ્વેપ કર્યા પછી તે અહીં આવેલો છે અને ચંદર 60 ગુણ સાથે અહીં ગયો છે. તેથી, આ સ્વેપિંગ ઑપરેશનએ મૂળ રીતે એ અને સીના હુકમને આ હડતાળથી ઉલટાવી દીધો છે જેથી તે ઈનપુટમાં હતું અને તેથી હવે આ ઓર્ડર ઝંપલાવશે. તેથી, ભાગલા દરમ્યાન આ સ્વેપ ઑપરેશન મૂળ ક્રમમાં વિક્ષેપ પાડે છે અને તે પસંદગી સૉર્ટ જેવી વસ્તુઓમાં પણ થાય છે. કારણ કે, પસંદગીના સૉર્ટમાં આપણે જ્યાં પણ આ લાંબા અંતરની વસ્તુઓ કરીએ છીએ, તેથી પસંદગીના પ્રકારમાં યાદ રાખો કે અમે લઘુત્તમ પસંદ કરીએ છીએ અને પછી અમે તેને શરૂઆતમાં બદલીએ છીએ. હવે, જો શરૂઆતમાં તત્વ સમાન મૂળાક્ષર ક્રમમાં હોય, તો મારો મતલબ એ છે કે તે નાની છે, બીજામાં તે

હવે એરે(Array)માં પોઈન્ટ તરફ આગળ વધે છે અને ઓર્ડર વિક્ષેપિત થાય છે. તેથી, કોઈપણ પ્રકારની લાંબા અંતરની સ્વેપિંગ સ્થિરતા માટે ખૂબ જોખમી છે. મર્જ સોર્ટ વિશે શું? સારું, સૉર્ટ કરો, સૉર્ટ કરો સામાન્ય રીતે અમે સ્થિર પ્રદાન કર્યું છે ... બધું જ જોઈએ કે જમણી બાજુએ કંઈક મૂળ એરે(Array)માં કંઈક ડાબી બાજુએ ન જવું જોઈએ. તેથી, અમને કંઈક જોઈએ નથી ... જો તમારી પાસે અહીં અને અહીં બે ઘટકો છે જે આ ક્રમમાં રહેવું જોઈએ, તો અમે તેમને વિનિમય કરવા માંગતા નથી. હવે, શા માટે તેઓ વિનિમય કરશે? તેઓ માત્ર સમાન હોય તો જ તેઓ વિનિમય પામશે, તે એકમાત્ર કેસ છે જ્યાં સ્થિરતા નાશ પામે છે, જ્યાં સમાન તત્વો વિક્ષેપિત થાય છે

(સમયનોસંદર્ભ લો: 03:57)

અગાઉના સૉર્ટિંગમાં. તેથી, આપણે ખાતરી કરવાની જરૂર છે કે પહેલાની એરે(Array)માંથી કંઈક ડાબી બાજુના તત્વ પર જમણી બાજુનો કોઈ તત્વ આવે અને જ્યારે તમે મર્જ કરી રહ્યા હો, ત્યારે મૂળભૂત રીતે આપણે કહ્યું કે આ કિસ્સામાં, હું એ કરતાં ઓછું છે અથવા બી જે સમાન, અમે એ અને ઓનલે પસંદ કરીએ છીએ y ... તેથી, જો તમે ઓછા લખવાની ભૂલ કરી હોય અને પછી આપણે બી કરતાં પસંદ કરીએ, બીમાંથી પસંદ કરીએ, પછી જ્યારે તેઓ સમાન હોય, ત્યારે આપણે પ્રથમ બીમાંથી પસંદ કરીએ અને એમાંથી નહીં પણ તે અસ્થિરતા બનાવશે. તેથી, અમારા મર્જ ઑપરેશનમાં નિર્ણાયક એ છે કે જ્યારે હું A બી કરતાં ઓછું અથવા સમાન હોય ત્યારે આપણે A માંથી પ્રાધાન્યતા B માં તત્વ પસંદ કરીએ છીએ, તેથી આ મૂળ ઓર્ડરના સંદર્ભમાં સમાન તત્વોના ડાબા જમણા હુકમને સાચવે છે અને સંવેદના મર્જ સૉર્ટ(Merge Sort) કરે છે. સ્થિર છે. તમે પણ તે જ રીતે તપાસ કરી શકો છો, જો તમે નિવેશ સૉર્ટ(Insertion Sort) કાળજીપૂર્વક કરો છો, તો નિવેશ સૉર્ટ(Insertion Sort) પણ સ્થિર છે અને આ કહેવાનું નથી કે ઝડપી સૉર્ટ(Quick Sort) સ્થિર બનાવી શકાતું નથી, તે જ રીતે અમે તેને અમલમાં મૂક્યું છે તે ઝડપી સૉર્ટ(Quick Sort) સ્થિર નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:52)

અને બીજું માપદંડ, જે આપણે હમણાં જ જોયું છે તે એ છે કે આપણે ફક્ત બે ઘટકોની તુલના કરવા અને તેના બદલામાં ક્રમમાં જરૂરી પગલાંઓની માત્રા પર ધ્યાન કેન્દ્રિત કર્યું છે. પરંતુ, તમે અન્ય વસ્તુઓ કરવા માંગી શકો છો. હમણાં પૂરતું, તમે હલનચલનને દંડ કરવા માંગી શકો છો જે તમને એરે(Array)ના છેલ્લા ભાગોમાં લઈ જાય છે. તેથી, તમે કહી શકો છો કે વસ્તુઓને દૂર કરવા માટે આટલું સારું નથી, એકબીજા પાસે વસ્તુઓનું વિનિમય કરવાનું વધુ સારું છે. તેથી, કંઈક જે બબલ સૉર્ટ(Bubble Sort) જેવું ન હતું જેને આપણે ન જોઈ શકીએ, જે ફક્ત અડીને ઘટકોનું વિનિમય કરશે, જે પસંદગી સૉર્ટ જે મોટા પ્રમાણમાં એરે(Array)માં વિનિમય કરે છે તેના કરતાં વધુ સારું હશે. હવે, ડેટા હોઈ શકે છે અને તમે એરે(Array)ને સૉર્ટ કરી રહ્યાં છો એવું માનતા હો તો આ થઈ શકે છે, પરંતુ આ એરે(Array) ભિન્ન ભિન્ન છે, તે એટલું મોટું છે કે તે એક સર્વર પર સંગ્રહિત નથી. જ્યારે તમે દર વખતે એક સર્વરથી બીજામાં સ્વેપ કરવા માંગો છો, ત્યારે તમે વધારાની કિંમત ચૂકવી શકો છો જે અલગ છે. તેથી, બીજા શબ્દોમાં કહીએ તો તે દ્રશ્યો પાછળના વિવિધ માપદંડ હોઈ શકે છે જે આપણે જોઈ શકતા નથી. તેથી, કલ્પના કરો કે જ્યારે તમે કોઈ વસ્તુને હાથથી સોર્ટ કરો છો, એવું માનતા હોવ કે તમે ભારે કાર્ટૂન્સનો સમૂહ ગોઠવી રહ્યા છો, તો તમે કલ્પના કરી શકો છો કે ભારે દોરડું ખસેડવું તે લાંબા અંતર કરતાં થોડું વધારે ખર્ચાળ છે. તેથી, ત્યાં અન્ય માપદંડ હોઈ શકે છે જે સૉર્ટિંગની તમારી કિંમતને નિયંત્રિત કરે છે જેમાં આપણે આમાં બિલકુલ વિચાર કર્યો નથી અને લોકોએ આ વસ્તુઓને સાહિત્યમાં જોયા છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:05)

તેથી, ઘણીવાર એક પ્રશ્ન પૂછવામાં આવે છે કે કઈ સૉર્ટ શ્રેષ્ઠ છે. તેથી, કમનસીબે તે તારણ આપે છે કે કોઈ એક સૉર્ટિંગ એલ્ગોરિધમ હંમેશાં શ્રેષ્ઠ હોવાની ખાતરી આપે છે, તે ખરેખર ખરેખર નિર્ભર છે, અમે ઊંડાણ, આંદોલન અને અન્ય માપદંડોને સૉર્ટ કરવા જેવા કહ્યું છે. કેટલાક સૉર્ટિંગ એલ્ગોરિધમ કેટલાક સંદર્ભમાં સારી કામગીરી કરી શકે છે, કેટલાક અન્ય સંદર્ભોમાં. મોટા ભાગનામાં, સરળ એરે(Array) માટે મેમરી આધારિત સંદર્ભ, ઝડપી સૉર્ટ(Quick Sort) શ્રેષ્ઠ છે. આ માટે આપણે કહ્યું છે કે, ઝડપી સૉર્ટ(Quick Sort) એ સામાન્ય રીતે એલ્ગોરિધમ્સને સૉર્ટ કરવા માટે પસંદગીની ડિફૉલ્ટ અમલીકરણ છે. જો કે આપણે પિવોટ અને અન્ય ઘણા ઘટકો પસંદ કરવા વિશે કંઈક બુદ્ધિશાળી કરીએ છીએ. બીજી તરફ જ્યારે આપણે ઘણું બધું ડેટા ખસેડવું પડે છે, તો તમે ડેટાબેઝને સૉર્ટ કરી રહ્યા હોવ તો કેટલીકવાર અમે આખી વસ્તુ સંગ્રહિત કરી શકતા નથી. અમે સમગ્ર વસ્તુને મેમરીમાં સૉર્ટ કરી શકતા નથી, તેથી વાસ્તવમાં અમે મર્જ

સૉર્ટ(Merge Sort)માં વિવિધતાનો ઉપયોગ કરીએ છીએ જેને બાહ્ય મર્જ સૉર્ટ(Merge Sort) કહેવાય છે. તેથી, તે વાસ્તવમાં સંદર્ભો પર આધાર રાખે છે, કેટલીકવાર અમે એકનો ઉપયોગ કરીશું, કેટલીકવાર અમે બીજાનો ઉપયોગ કરીશું. જો તેઓ શ્રેષ્ઠ અલ્ગોરિધમ હતા, તો અન્ય એલ્ગોરિધમ્સનો અભ્યાસ કરવાની જરૂર રહેશે નહીં. તેથી, એ હકીકત છે કે આમાંના ઘણાં ઍલ્ગોરિધમ્સ જ્યાં અભ્યાસ કરે છે, બતાવે છે કે વસ્તુઓને બહેતર બનાવવા માટે તમને એક અથવા બીજા વિચારોનો ઉપયોગ કરવાની જરૂર પડી શકે છે.

(સ્લાઇડટાઇમ નો સંદર્ભ લો: 07:10)

આપણે મર્જ સૉર્ટ(Merge Sort) ઓર્ડર એન લોગ એન એલ્ગોરિધમ તરીકે જોયું છે, અન્ય એન લોગ એન એલ્ગોરિધમ છે જે આપણે આ કોર્સ પર એક પછીથી હેપ સૉર્ટ માટે જોઈશું. યાદ રાખવાની મુખ્ય વસ્તુ એ છે કે જો તમારી પાસે નિષ્ક્રીય એન સ્ક્વેર્ડ એલ્ગોરિધમ હોય તો જે સામાન્ય કિસ્સામાં સચોટ રીતે સૉર્ટ એન લોગ એન જેવી નથી, તો તે ફક્ત નાના ડેટા માટે જ કાર્ય કરશે. જો કે, ક્યારેક તે નાના ડેટા, નિષ્કપટ અલ્ગોરિધમનો સરળતા જટીલ મૂલ્યની જટીલતાને ધારણ કરે છે. તેથી, તમે હાયબ્રીડ એલ્ગોરિધમ્સ પણ ધરાવી શકો છો, તમે ભાગલાનો ઉપયોગ કરી શકો છો અને n ક્યાં મોટો છે તે જીતી શકો છો અને પછી n નાનું બને છે, તો તમે દાખલ થવા માટે સ્વિચ કરી શકો છો. તેથી, ત્યાં ઘણી બધી વ્યૂહરચનાઓ છે જે લોકો ઉપયોગ કરે છે અને પ્રયોગોના કેટલાક સંયોજનો સાથે વારંવાર પ્રાયોગિક રીતે અરે(Array) છે. તેથી, તે કહેવું પૂરતું નથી કે એક ઍલ્ગોરિધમ શ્રેષ્ઠ છે, તમારે કયા કાર્યો અને શું કાર્ય કરતું નથી તેના વિશે સારી વિચાર કરવાની જરૂર છે, જેથી તમે તમારી આવશ્યકતાઓને અનુરૂપ તમારી સૉર્ટિંગ પ્રક્રિયાને અનુરૂપ કરી શકો.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 01
લેક્ચર - 18
ગ્રાફ્સનો પરિચય

તેથી, જ્યારે આપણે કોઈ સમસ્યા માટે ઍલ્ગોરિધમનો ડિઝાઈન કરીએ છીએ, ત્યારે સમસ્યાની માહિતીને અમે એવી રીતે રજૂ કરવાની જરૂર છે કે જેનો અમે ઉપયોગ કરી શકીએ. તેથી, આ મોડેલિંગ કહેવામાં આવે છે. તેથી, અમને પ્રોપર્ટીનું મોડલ કરવા માટે કોઈ પ્રકારની સંકેત અને માળખાંની જરૂર છે. તેથી, આ મોડ્યુલમાં આપણે ગ્રાફ્સ તરીકે ઓળખાતા માળખાના એક અત્યંત મહત્વપૂર્ણ વર્ગને જોઈશું જે ઘણા વિવિધ સંદર્ભોમાં અત્યંત ઉપયોગી છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 00:23)
તેથી, ચાલો આપણે રાજકીય નકશાને રંગીન કરવાની સમસ્યાથી પ્રારંભ કરીએ. તેથી, અહીં ભારતના રાજ્યોનો નકશો છે, કેમ કે તમે જુદા જુદા રાજ્યો જુદા જુદા રંગો જોઈ શકો છો. હવે, આ રંગ પાછળ સિદ્ધાંત શું છે? જ્યારે આપણે કોઈ રાજ્ય રંગીશું, ત્યારે અમારે ખાતરી કરવી જોઈએ કે તેની નજીકના કોઈ પણ રાજ્ય સમાન રંગ ધરાવતું નથી, કોઈ પણ રાજ્ય કે જે બાજુની બાજુએ હોય છે તે સમાન સરહદ વહેંચે તે જ રંગ હોવું જોઈએ, કારણ કે અન્યથા તે એકને અલગ પાડવું મુશ્કેલ છે. બીજું રાજ્ય. તેથી, આપણે જોઈ શકીએ છીએ કે રાજસ્થાન ગુજરાતથી અલગ રંગ છે, કારણ કે રાજસ્થાન અને ગુજરાત આ બિંદુએ એક સામાન્ય સીમા વહેંચે છે. એ જ રીતે, રાજસ્થાન અને મધ્યપ્રદેશની સીમા છે, તેથી રાજસ્થાન અને મધ્યપ્રદેશમાં વિવિધ રંગ છે. બીજી તરફ, રાજસ્થાન અને તેલંગણા એક સરહદ વહેંચી શકતા નથી, તેથી અમે રાજસ્થાન અને તેલંગણા માટે સમાન રંગનો ઉપયોગ કરી શકીએ છીએ. તેથી, આપણે જે પ્રશ્નો પૂછી શકીએ છીએ તે આવા નકશાને આપવામાં આવે છે, આ માપદંડને સંતોષવા માટે આપણે કેટલા રંગો રંગવાની જરૂર છે. હવે, સ્પષ્ટપણે આપણે દરેક રાજ્યને એક અલગ રંગ સોંપી શકીએ છીએ અને આમ આપણે ખાતરી કરી શકીએ છીએ કે કોઈ પણ બે રાજ્યો જેની સીમા સામાન્ય નથી, વાસ્તવમાં નકશામાં ક્યાંય પણ બે રાજ્યો સમાન રંગ ધરાવતા નથી. પરંતુ, ઘણી પરિસ્થિતિઓમાં આપણે ઘણા રંગો કરતાં ઘણું ઓછું અપેક્ષા રાખી શકીએ છીએ. તેથી, અમારો પ્રશ્ન એ છે કે, આપણને કેટલા રંગોની જરૂર છે?
(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:37)
તેથી, આ સમસ્યાને વધુ અમૂર્ત રીતે રજૂ કરવાનો માર્ગ અહીં છે. તેથી, આપણે જે પ્રથમ વસ્તુ કરીએ છીએ તે છે કે આપણે બદલીએ છીએ, આપણે બદલીશું નહીં, અમે વાસ્તવમાં નકશામાં દરેક રાજ્યને એક કાળો ડોટ અસાઈન કરીશું. તેથી, લગભગ રાજ્ય જ્યાં રાજ્યનું કેન્દ્ર છે ત્યાં આપણે આ કાળો બિંદુ મૂકીએ છીએ. હવે, આપણે એકબીજાના પડોશીઓ કેવી રીતે છે તે વિશે માહિતી રેકોર્ડ કરવી પડશે. તેથી, આપણે શું કરીએ છીએ તે અમે દરેક રાજ્યના જોડીને જોડીએ છીએ જે સીમાને વહેંચે છે. તેથી, જેમ આપણે જોયું કે રાજસ્થાન અને ગુજરાતમાં સરહદ સામાન્ય છે, સરહદ છે અને તેથી અમે રાજસ્થાન અને ગુજરાત વચ્ચે આ પ્રકારની ધાર મૂકીએ છીએ, હું માત્ર ચિલ્લ્ન છું. હવે, અમારી કલરની સમસ્યા રાજ્યોને રંગો અસાઈન કરવાની છે, જેથી કોઈ પણ બે રાજ્યો જે ધારની વિરુદ્ધ કદ પર ધાર ધરાવતા હોય તે સમાન રંગ ધરાવતા હોય. તેથી, રાજ્યનું રંગ રાજ્ય સાથે સંકળાયેલા ડોટને રંગવા જેવું જ છે. તેથી, આપણે ઉદાહરણ તરીકે, ઉત્તર પ્રદેશ જેવા રાજ્ય સાથે પ્રારંભ કરી શકીએ અને તેને લાલ રંગ આપી શકીએ અને હવે આ લાલ છે, તેનો અર્થ એ કે ઉત્તરાખંડ, અથવા હરિયાણા, અથવા રાજસ્થાન, અથવા મધ્યપ્રદેશ, અથવા છત્તીસગઢ જેવા નજીકના રાજ્ય છે, અથવા ઝારખંડ, અથવા બિહાર. આમાંથી કોઈ પણ હવે રંગીન લાલ હોઈ શકે નહીં કારણ કે તે બધા પડોશી રાજ્યો છે. તેથી, આગામી પગલામાં ઉત્તર પ્રદેશ લાલ રંગ ધરાવતું, હવે રંગીન હરિયાણા રંગીન છે, જેમાં વાદળી અને ઉત્તરાખંડ છે, જે હરિયાણા અને ઉત્તરપ્રદેશ બંને સાથે રંગીન લીલો છે, તેથી આપણે ત્રણ રંગોનો ઉપયોગ કર્યો છે. જો આપણે રાજસ્થાન સાથે આગળ વધીએ, તો અમને લાગે છે કે રાજસ્થાન ઉત્તરાખંડ સાથે સીમા વહેંચી શકતું નથી. તેથી, લીલા રંગનો ફરીથી ઉપયોગ કરવો

શક્ય છે, તેથી અમે ત્રણ રંગો સાથે સ્ટિકિંગ કરી રહ્યા છીએ, અમે તે રંગને રાજસ્થાન સુધી વિસ્તૃત કરી શકીએ છીએ. પછી, અમે પંજાબ પહોંચ્યા, હવે પંજાબ ઉત્તર પ્રદેશ સાથે જોડાયેલું નથી, તેથી અમે પંજાબ માટે ઉત્તર પ્રદેશના લાલ રંગનો ફરી ઉપયોગ કરી શકીએ છીએ. જો કે, અમે હિમાચલ પ્રદેશમાં આવી શકીએ છીએ, અમે શોધી કાઢીએ છીએ કે હિમાચલ પ્રદેશ પંજાબ અને હરિયાણા અને ઉત્તરાખંડ સાથે જોડાયેલું છે અને તેથી, આપણે આ ત્રણ રંગોનો ઉપયોગ કર્યો છે અને આપણે હિમાચલ પ્રદેશ માટે ચોથા રંગનો ઉપયોગ કરવો જોઈએ. તેથી, જો તમે આ રીતે ચાલુ રાખો છો, તો આખરે દેશના તમામ રાજ્યોને રંગો અસાઈન કરી શકો છો જે આ ચોક્કસ સંપત્તિને સંતોષે છે. જ્યારે પણ, હું બે બિંદુઓ જોઉં છું, જો તેઓ એક લીટીથી જોડાયેલા હોય, તો પછી બે બિંદુઓ જુદા જુદા રંગો હોય છે, આ અપૂર્ણપણે મિલકતને કેપ્ચર કરે છે જે તમે બે સરહદો સાથે સામાન્ય સરહદ સાથે, નકશા પર વિવિધ રંગો ધરાવતા હોય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:07)

તેથી, જો તમે આ નકશા રંગની સમસ્યાને જુઓ છો, તો આ નકશાને વાસ્તવિક રેખા નીચે હવે જરૂરી નથી, કારણ કે બિંદુઓએ નકશા વિશે બધું જ મેળવ્યું છે. તેથી, આપણે અંડરલાઈન ચિત્રને ફેંકી શકીએ છીએ, હું આ બિંદુઓની માત્રા અને તેમની વચ્ચેના જોડાણો રાખું છું અને જો તમે આ બિંદુઓને રંગ આપો છો, તો ધ્યાનમાં રાખો કે આ રેખાઓ સામાન્ય સીમાઓનું પ્રતિનિધિત્વ કરે છે. અમે મૂળ નકશા રંગીન મિલકતને હલ કરવા સમાન સમસ્યાઓને હલ કરી રહ્યાં છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:35)

તેથી, આ પ્રકારનું આકૃતિ, આ આપણે ગ્રાફને શું કહીએ છીએ, તેથી આ બિંદુઓને શિરોલંબ કહેવામાં આવે છે, તેથી આ શિરોલંબ છે, તેને નોડ પણ કહેવામાં આવે છે. તેથી, આ, નોડ્સ અને શિરોબિંદુઓનું વર્ણન કરવા માટે બે શબ્દો છે. તે જાણવું ઉપયોગી છે કે શિરોબિંદુનું બહુવચન, શિરોબિંદુ એક ગાંઠ છે, ઘણા ગાંઠોને શિરચ્છેદ કહેવામાં આવે છે અને આ જોડાણો એ ધાર છે. તેથી, આપણી પાસે શિરોબિંદુઓ વચ્ચે ધાર છે, તેથી તે ખૂબ જ સરળ છે. ગ્રાફ એ ફક્ત એક ચિત્ર છે, તેમાં કેટલાક બિંદુઓનો સમાવેશ થાય છે જે શિરોબિંદુઓના ગાંઠો અને તેમની વચ્ચેના કેટલાક જોડાણો જેને ધાર તરીકે ઓળખવામાં આવે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:15)

તો, સમસ્યા જે આપણે ઉકેલી છે તેને ગ્રાફ કલર કહેવામાં આવે છે. તેથી, અમે ચાર રંગનો ઉપયોગ કર્યો છે, તમે ચકાસી શકો છો કે આપણે ચાર રંગનો ઉપયોગ કર્યો છે, આપણે વાદળી, લીલો, લાલ અને પીળો ઉપયોગ કર્યો છે અને આ આખા ગ્રાફને ફક્ત આ ચાર રંગોથી રંગવામાં અમને સંચાલિત કરવામાં આવે છે. હવે, તમે કહી શકો છો કે તે આ ગ્રાફની મિલકત છે અથવા તે આખા ગ્રાફની સંપત્તિ છે. તેથી, વાસ્તવમાં તે તારણ આપે છે કે જો તમે જે પ્રકારના નકશાને દોર્યું અને અમે જે ગ્રાફ બનાવ્યું તે રૂપાંતરિત કરીએ, તો ચાર રંગો હંમેશાં પૂરતા હોય છે. હવે, આ ગ્રાફ વિશેના ગાણિતિક તથ્ય છે. તેથી, તમે ચોક્કસ નકશાને રંગ આપવા વિશે કોઈ ચોક્કસ સમસ્યા લઈ શકો છો અને પછી અમે બધા નકશા વિશે એક સામાન્ય પ્રશ્ન પૂછી શકીએ છીએઆ પ્રકારની અને આપણે વાસ્તવમાં તેના વિશે અથવા પ્રણાલી વિશેના ગાણિતિક તથ્યને સાબિત કરી શકીએ છીએ, તે કહે છે કે કોઈપણ નકશામાં જે આ પ્રકારના માંથી લેવામાં આવે છે, કોઈપણ ગ્રાફ આ પ્રકારના નકશામાંથી મેળવે છે, ચાર રંગો ગ્રાફને હલ કરવા માટે પૂરતા હોય છે રંગીન મિલકત. હવે, આ ઉકેલવાની એક સરળ સમસ્યા નથી, તે ઘણા વર્ષો સુધી ખુલ્લી સમસ્યા હતી અને તે ખરેખર પ્રખ્યાત થિયરી હતી, જ્યારે તે ખરેખર સાબિત થયું હતું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:19)

હવે, તમે અવલોકન કરી શકો છો કે જો આપણે આ ગ્રાફનો ઉપયોગ ફક્ત ચાર રંગોથી જ અમારા ગ્રાફનો રંગ શોધી કાઢ્યો છે, તો મૂળ નકશા જેની સાથે આપણે શરૂઆત કરી હતી તેમાં ઘણા વધુ અને ચાર રંગ હતાં. તો, અમારી પાસે આ છે, જો તમે આજુબાજુ જોશો, તો તમને 1, 2, 3, 4 અને પછી 5 નકશા પર ઓછામાં ઓછા પાંચ રંગ મળશે, જો 6 તમે આ સફેદ શામેલ કરો છો અને બીજું. તેથી, ભલે આપણામાં પ્રણાલી છે કે આપણે ચાર રંગોનો ઉપયોગ કરી શકીએ છીએ, વાસ્તવમાં તે વ્યક્તિ જેણે આ નકશા રંગી લીધેલ છે તે ચાર રંગો કરતા વધુ ઉપયોગ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:47)

તેથી, ગ્રાફ તરીકે અમારા રજૂઆત તરફ જવાનું એક ફાયદો એ છે કે આપણે સમસ્યાની બધી આવશ્યક સુવિધાઓને દૂર ફેંકી દીધી છે, આપણે આ રાજ્યના આકારને જાણવાની જરૂર નથી , તે જાણવાની જરૂર નથી કે તે કદ છે. આપણે જાણવાની જરૂર છે કે કયા રાજ્ય સાથે કયા રાજ્ય જોડાયેલ છે, તે રાજ્ય કયા રાજ્યની સરહદ છે. તેથી, હવે એક વખત ગ્રાફ પાસે આવી ગયા પછી, આપણે ગ્રાફને ફરીથી ક્રમાંક આપી શકીએ છીએ. પરંતુ, અર્થમાં આપણે જોડાણોને વધુ સ્પષ્ટ બનાવવા માટે આ પ્રકારનાં ભીડવાળા ભાગોને વિસ્તૃત કરી શકીએ છીએ અને અમે આ સુધારેલા ગ્રાફ સાથે કામ કરી શકીએ છીએ અને આ સુધારેલા ગ્રાફ માટેનું સોલ્યુશન મૂળ ગ્રાફના ઉકેલ જેવું જ છે, સમસ્યા નથી બદલો. તેથી, સમસ્યાનું મોડેલિંગ કરવાની સૌથી મહત્વપૂર્ણ લાક્ષણિકતાઓમાંની એક છે સમસ્યાના આવશ્યક ભાગોને રાખવા અને અનિવાર્ય ભાગને ફેંકવું, જેથી તમે તે સમસ્યા પર ધ્યાન કેન્દ્રિત કરી શકો જે ખરેખર હલ કરવાની જરૂર છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:38)

તેથી, આપણે જોઈયેલી બીજી સમસ્યા જે ગ્રાફ તરીકે સહેલાઈથી રજૂ કરી શકાય તે એ એરલાઈન રૂટીંગની છે. તેથી, અમારી પાસે એરલાઈન રૂટીંગ છે, તમે દર્શાવેલ એરલાઈન્સના રૂટને પૂછી શકો છો અને એક દિશામાં ફ્લાઈટ્સ સૂચવતી વચ્ચેના આ પ્રકારના ફોર્મેટ અને તીર દ્વારા બંધારણ, તમે કનેક્ટિવિટી વિશેના પ્રશ્નો પૂછી શકો છો. ત્યારબાદ, હું નવી દિલ્હીથી ટ્રીવંડ્રમથી એરલાઈન્સ બદલ્યા વગર જઈશ. તેથી, આ ફરીથી એક સમસ્યા છે, જ્યાં આપણે કનેક્ટિવિટી પર ધ્યાન કેન્દ્રિત કરી શકીએ છીએ અને વાસ્તવિક ગ્રાફને ફેંકી શકીએ છીએ, તે ખરેખર વાસ્તવિક નકશા માટે વાંધો નથી, તે આપણા માટે કોઈ વાંધો નથી

((સમયનોસંદર્ભ: 08: 14))

શહેરો ત્યાં છે, તેઓ એકબીજાને માન આપે છે. આપણે જાણીએ છીએ કે આ શહેર કયા શહેરથી ફ્લાઈટ દ્વારા જોડાયેલું છે અને શહેરોનાં જોડીઓ વચ્ચે મને કયા પ્રકારનાં માર્ગો મળી શકે છે તે જાણવામાં રસ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:27)

તેથી, ઔપચારિક રીતે ગ્રાફમાં ફક્ત બે ભાગ છે, તે શિરોબિંદુ અથવા ગાંઠોનો સમૂહ છે જે સામાન્ય રીતે વી લખાય છે અને ત્યાં કિનારોનો સમૂહ છે જે શિરોબિંદુઓના જોડી છે. તેથી, દરેક ધાર જોડી વી કોમા વી જોડી છે. હવે, જો તમારી પાસે એવા પ્રકારનો ગ્રાફ છે કે જે નકશા માટે છે, તો આપણે તેને રંગી રહ્યા છીએ, પછી આપણે v એ પહેલા તફાવત નથી કે v એ v પ્રાઈમ પહેલા અથવા v પ્રાઈમ પહેલા v છે. જ્યારે, આપણે કહીએ છીએ કે બે રાજ્યો એક સામાન્ય છે સીમા, તે કોઈ વાંધો નથી કે આપણે ત્યાં જે ઓર્ડરનો ઉલ્લેખ કરીએ છીએ તેનો અર્થ છે. તેથી, વી અને વી પ્રાઈમ વચ્ચે એક ધાર છે, જો વી અને પ્રાઈમ વી વચ્ચેની ધાર હોય તો જ, અને તેથી શિરચ્છેદની કોઈપણ જોડ વચ્ચે ફક્ત એક ધાર છે. બીજી બાજુ, એરલાઈન ગ્રાફમાં અમારી પાસે દિશાઓ હતી, અમારી પાસે એક શહેરથી બીજા શહેરની ફ્લાઈટ હોઈ શકે છે, પરંતુ ખરાબ નહીં. અમારી પાસે આ ત્રિકોણીય પ્રકારની વસ્તુઓ છે, જ્યાં આપણે એક શહેરથી બીજામાં અને પાછળ જઈ શકીએ છીએ. તેથી, આનો અર્થ એ નથી કે તમે સીધા જ આ દિશા પર પાછા જઈ શકતા નથી, તેથી આ નિર્દેશિત ગ્રાફ નથી. તેથી, જો તમારી પાસે નિર્દેશિત ગ્રાફમાં વી થી વી પ્રાઈમની ધાર હોય, તો અમે ગેરેંટી આપતા નથી કે ત્યાં v પ્રાઈમ થી વિરુદ્ધની ધાર છે. અને જેમ આપણે જોયું તેમ, આપણે હવે આ પ્રકારના ગ્રાફને સરળતાથી ચિત્ર દ્વારા ચિત્ર તરીકે વર્ણવી શકીએ છીએ નોડ્સ અને ત્યારબાદ કિનારીઓને કિનારીઓ તરીકે જોડે છે. તેથી, ક્યાં તો તેને દિશામાન કરવું અથવા અમે ગ્રાફને નિર્દેશિત કરી શકીએ જેમાં સુધારણાને સૂચવવા માટે અમે તીર સાથે રેખાઓ દોરીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:48)

તેથી, ગ્રાફ કલરિંગ સમસ્યા અણધાર્યા ગ્રાફ પર એક સમસ્યા છે અને આપણે ઔપચારિક રીતે કહી શકીએ છીએ કે રંગ એ કલર સી શોધવાનું છે, કલર સી એક ફંકશન છે જે દરેક વર્ટેક્સ વી રંગને સોંપી દે છે. V ની C અને ગ્રાફના સંદર્ભમાં, જ્યારે આપણી ધાર સમૂહમાં ધાર વી અને વી પ્રાઈમ હોય, ત્યારે C ની v એ C ના v સમયથી અલગ હોવું જોઈએ. કાયદેસર રંગ હોવાનો આ અર્થ છે. તેથી, હવે આપણે આ ગ્રાફને ગાણિતિક પદાર્થ તરીકે લઈ શકીએ છીએ અને સમયો અને કિનારીઓના સંદર્ભમાં ગાણિતિક સમસ્યા તરીકે સંપૂર્ણ રીતે ઉદ્દેશિત રીતે સમસ્યાનું વર્ણન કરીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:31)

એ જ રીતે, આપણે ગાણિતિક વિજ્ઞાનમાં રુટ શોધવાની સમસ્યા વ્યક્ત કરી શકીએ છીએ, શરૂઆતમાં આપણી સમસ્યા એક નિર્દેશિત ગ્રાફ છે. પછી, અમે શહેરોને અનુરૂપ શિરોબિંદુ ઓળખીએ છીએ, જ્યાં આપણે ગાંઠો શોધવા માંગીએ છીએ. તેથી, હું આપું છું કે આપણી પાસે નવી દિલ્હીનું પ્રતિનિધિત્વ કરે છે અને વી 5 ત્રિવેન્દ્રમનું પ્રતિનિધિત્વ કરે છે અને અમારું લક્ષ્ય નવી દિલ્હીથી ત્રિવેન્દ્રમ સુધીની વિરુદ્ધ 0 થી 5 માં મૂળ શોધવાનું હતું. તેથી, આ રુટને પાથ તરીકે વર્ણવી શકાય છે, તેથી પાથ એ ધાર દ્વારા જોડાયેલા શિરોલંબનો ક્રમ છે. તેથી, તમે શિર્ષકોનું અનુક્રમણિકા શોધવા માંગો છો જેમ કે પ્રથમ વર્ટેક્સ વી 0 છે જ્યાં આપણે પ્રારંભ કરવા માંગીએ છીએ, છેલ્લું શ્વેત વી કે જ્યાં આપણે સમાપ્ત કરવા માંગીએ છીએ અને ભાગ સાથે દરેક શિરોબિંદુઓ vi, v પ્લસ પ્લસ એ ફ્લાઈટ છે આગ્રામાં એક ધાર.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:17)
હવે, આ સમસ્યા માટે કોઈ ગ્રાફને દિશા નિર્દેશિત કરવાની જરૂર નથી હોતી, તેથી અમે તે જ ગ્રાફ લઈ શકીએ છીએ અને અમે માની શકીએ કે તેઓ અણધાર્યા છે, તે એરલાઈન ખરેખર તમામને સેવા આપે છે બંને દિશામાં શહેરની જોડી, સમસ્યામાં હજુ પણ સમજાણ છે. તેથી, હવે, આપણી પાસે અણધાર્યા ગ્રાફ છે, પરંતુ આપણે કહીએ છીએ કે v 0, v 1 એ ધાર છે, એટલે કે v 0, v 1 એ પણ ધાર છે. આપણે પાછળ અને આગળ જઈ શકીએ છીએ અને હવે આ જ પ્રશ્ન છે, આપણે શિરોલંબનો ક્રમ શોધી શકીએ છીએ, આપણે વી 0 થી શરુ કરીએ છીએ અને v5 માં જઈએ છીએ કે આ પાથ પર દરેક જોડી, દરેક વી, વી, પ્લસ 1 એ એક ધાર છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 02
લેક્ચર - 19
ગ્રાફ્ ની રજૂઆત

તેથી, આપણે જોયું છે કે મોડેલિંગ સમસ્યાઓ માટે આલેખ ખૂબ જ ઉપયોગી ગણિતશાસ્ત્રીય માળખાં છે. પરંતુ, જ્યારે આપણે ગ્રાફ્ સૈદ્ધાંતિક સમસ્યાને હલ કરવા માટે ઍલ્ગોરિધમ લખીએ છીએ, ત્યારે અમને અમારા એલ્ગોરિધમના ગ્રાફ્નું પ્રતિનિધિત્વ કરવા અને તેમાં ફેરફાર કરવા માટેની રીતની જરૂર છે. તો, આપણે આ લેક્ચરમાં તે જોઈશું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:15)
તેથી, યાદ રાખો કે ગ્રાફ્ એ શિરોબિંદુનો સમૂહ છે અથવા નોડ્સ V એ કિનારીઓના સમૂહ દ્વારા જોડાયેલ છે. અમે બે પ્રકારના ધાર, અધીરા કિનારીઓ અને નિર્દેશિત કિનારીઓ ધરાવી શકીએ છીએ. અણધાર્યા ધારને બે શિરોબિંદુઓ વચ્ચેની રેખા તરીકે દોરવામાં આવે છે અને તે હકીકતને રજૂ કરે છે કે વી અને વી પ્રાઈમ જોડાયેલા છે. તે કોઈ વાંધો નથી કે આપણે આ એજ વી કોમા વિરુદ્ધ પ્રાઈમ અથવા વી પ્રાઈમ અલ્પવિરામ વી કહીએ છીએ, ત્યાં ફક્ત એક જ ધાર છે. બીજી તરફ્ નિર્દેશિત ગ્રાફ્માં, અમે ખરેખર ધાર સાથેની દિશા સાથે સંકળાયેલા છીએ. તેથી, આપણે વી થી વિરુદ્ધ પ્રાઈમ તરફ્ની ધાર દોરી શકીએ છીએ અને આ અમે અમારા ધાર સેટમાં એક જોડી વી કોમા વી પ્રિમ્સ તરીકે લખીશું કે કહીએ તો શરુઆતનો ભાગ વી છે અને ધારનો અંત વી મુખ્ય છે. અને આ વી પ્રાઈમ થી વી ની ધાર સમાન નથી જે V પ્રાઈમ અલ્પવિરામ વી તરીકે લખવામાં આવશે. તેથી, જ્યારે તમે ધારની બાબતોનો ઉલ્લેખ કરો છો ત્યારે શિર્ષકોની હરોળને નિર્દેશિત કરે છે, પછી અણધાર્યા ગ્રાફ્ એ જોડી છે શિરોબિંદુ, તે કોઈ વાંધો નથી કે આપણે તેના વિશે વિચારીએ છીએ, તે વિરુદ્ધ v અથવા v ની વિરુદ્ધ છે, તે ફક્ત આ બે જોડીના શિરોબિંદુ વચ્ચેનું જોડાણ છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:22)
તેથી, અમે રૂટ શોધવા સહિત બે લાક્ષણિક સમસ્યાઓ જોયા, જે બંને દિશા નિર્દેશિત અને નિર્દેશિત ગ્રાફ્ માટે રજૂ કરી શકાય છે. તેથી, નિર્દેશિત ગ્રાફ્માં, આપણે વી 0 થી v5 સુધીનો માર્ગ શોધીશું, જેમ કે દરેક જોડી v, v1; વી 1, v2 વગેરે, આ અમારા ગ્રાફ્માં કિનારે દિશામાન છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:40)
તે જ સમયે, આપણે સમાન ગ્રાફ્ને અણધાર્યા કરી શકીએ છીએ અને ફરીથી આપણે પાથ શોધી રહ્યા છીએ, જ્યાં આપણે v 0 અને દરેક નજીકના જોડી v થી v1, v1 થી v ની શરૂઆત કરીએ છીએ. 2 એક ધાર છે, જેમ કે આપણે છેલ્લે લક્ષ્ય વેર્ટેક્સ v 5 માં સમાપ્ત થાય છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:53)
તો, સમસ્યા જે આપણી પાસે છે, ચાલો આપણે અણધાર્યા ગ્રાફ્સને વળગી રહીએ, કારણ કે આપણને આ આપવામાં આવે છે નિર્દેશિત ગ્રાફ્ને નિર્દેશિત કરો અને અમને સ્રોત વર્ટેક્સ વિ અને લક્ષ્ય વર્ટેક્સ વીએટી આપવામાં આવે છે અને અમે પૂછ્યું છે કે, આ ગ્રાફ્માં વિરુદ્ધથી વીએટીમાં જવાનો કોઈ રસ્તો છે. હવે, પાછલા ગ્રાફ્માં આપણે શું કર્યું છે અને આપણે શું કરી શકીએ તે આપણે ગ્રાફ્ પર નજર રાખવું જોઈએ અને જુઓ, જો તમે આવા પાથને દૃષ્ટિથી ઓળખી શકો છો, તો ફક્ત વિ અને વીટી જોડાયેલ છે. પરંતુ, જ્યારે આપણે ગ્રાફ્ને હેન્ડલ કરવા માટે ઍલ્ગોરિધમ લખીએ છીએ, ત્યારે ચિત્રને જોવા માટે અમે કેવી રીતે ઍલ્ગોરિધમ મેળવી શકીએ છીએ. તેથી, આપણા માટે ગ્રાફ્ એ એક ચિત્ર છે અને જો ગ્રાફ્ ખૂબ જટિલ ન હોય અને પરિસ્થિતિને સમજવાનો પ્રયાસ કરીએ તો અમે ચિત્રને સરળતાથી જોઈ શકીએ છીએ. પરંતુ, અમે આ ચિત્રને ઍલ્ગોરિધમ પર કેવી રીતે ઉપલબ્ધ કરશું? અમને આ ચિત્રને રજૂ કરવાની રીતની જરૂર છે જે ગ્રાફ્ને અમારા અલ્ગોરિધમનો ઈનપુટ તરીકે આપે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:42)

તેથી, ચાલો આપણે ધારણા કરીએ કે કોઈ ગ્રાફમાં આપણે વિચારીએ છીએ, ત્યાં ફક્ત શિરોબિંદુનો મર્યાદિત સમૂહ છે. તેથી, જો જીવન સરળ બનાવવા માટે n શિરોલંબ હોય, તો ચાલો આપણે આ શિરોબિંદુઓ 1, 2 ને n સુધી નામ આપીએ. તેથી, અમારા શિરોબિંદુઓ હંમેશા 1, 2, 3, 4 સુધી n સુધી જતા હોય છે. તેથી, તેથી હવે એજ એ સંખ્યાઓની જોડી છે જે હું કોમા જે. તેથી, આપણે જે રજૂઆત કરી શકીએ તે એ છે કે ફક્ત હું અને જે જોડી જોડાયેલા છે તે રેકોર્ડ કરવું. તો, આને અડજાનસી (adjacency) મેટ્રિક્સ કહેવામાં આવે છે, આપણે આ મેટ્રિક્સમાં કહીએ છીએ કે ij એ 1 છે, જો અને ij એજ હોય તો જ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:24)

તેથી, જ્યારે આપણે આવા મેટ્રિક્સ લખીએ છીએ, તો જો આપણે પહેલા ગ્રાફ જેવા ગ્રાફ લઈશું, તો હવે આપણે શિરોલંબ 1 થી 3 સુધી 10 નું નામ બદલીશું, કારણ કે આમાં વાસ્તવમાં 10 શિરોબિંદુઓ છે. ગ્રાફ અને પછી આપણે આ મેટ્રિક્સ લખીશું જે ઈન્સ્ટન્ટ માટે 1 થી 3 ની ધાર છે અને તેથી એન્ટ્રી એ 1 3 છે. 1 થી 5 ની કોઈ ધાર નથી અને તેથી એન્ટ્રી એ 1 5 0 છે. તેથી, આ રીતે આપણે ગ્રાફમાં જે દરેક ધાર શોધીએ છીએ, એ, ઉદાહરણ તરીકે, એ 4 5, અમને ગ્રાફમાં એન્ટ્રી મળશે જે કહે છે કે એ 4 5 એ 1 છે. હવે, યાદ રાખો કે આ અવ્યવસ્થિત છે. તેથી, જો ત્યાં ધાર 5 5 5 હોય, તો એક ધાર 5 4 પણ છે અને તેથી આપણે જોશું કે ખરેખર મેચિંગ ધાર 5 છે. તેથી, આ ગ્રાફ વાસ્તવમાં આ સમપ્રમાણતા છે, તેથી તે વાસ્તવમાં સમપ્રમાણતામાં છે આ ત્રાંસા તો, જો હું લાઈન ઉપર 1 જોઉં, તો હું લાઈનની નીચે 1 જોશ, કારણ કે 8 9 9 8 જેટલું જ છે જે આપણા અધીરા કિનારીઓ છે. તેથી, આ અડજાનસી (adjacency) મેટ્રિક્સ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:20)

તેથી, હવે અડજાનસી (adjacency) મેટ્રિક્સ સાથે આપણે શું કરી શકીએ? ઠીક છે, ઉદાહરણ તરીકે આપણે જે કરી શકીએ છીએ તે બધા પાડોશીઓને એક શિરચ્છેદ છે. ધારો કે, જો આપણે સીડીના પાડોશીઓને જોઈએ તો, પછી આપણે પંક્તિ પર જોવું જોઈએ અને તે પંક્તિમાંની બધી એન્ટ્રીઝ 1 જોઈશું. ઉદાહરણ તરીકે, આપણે વર્ટેક્સ 4 ના પાડોશીઓને જોવા માંગીએ છીએ, આપણે પંક્તિ 4 તરફ જોઈશું, પછી 4, એન્ટ્રી 4 કોમા 1 સૂચવે છે કે 4 1 એક ધાર છે. તે છે, તેથી આપણી પાસે 1 ભાગ છે, પછી આપણે આગળ વધીએ છીએ ... અને ત્યારબાદ આગલું એક 4 કોમા 5 છે, તેથી 5 એક પાડોશી છે. અને પછી, આગળનું એક 8 છે, તેથી 8 પાડોશી છે, તેથી એક શંકુના પાડોશીઓને શોધવા માટે, અમે હું લેબલ કરેલી પંક્તિ પર જાઓ અને અમે પંક્તિને ડાબેથી જમણી તરફ સ્કેન કરીએ છીએ અને દરેક જેને આપણે શોધીએ છીએ , અનુરૂપ કૉલમ પાડોશી છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:09)

હવે, આપણે પાથ કેવી રીતે શોધી શકીએ? પછી, હવે આપણે પડોશીઓ અને પડોશીઓના પડોશીઓ તરફ જોઈ શકીએ છીએ. તેથી, આપણે આપણી સમસ્યામાં સ્રોત વર્ટેક્સથી પ્રારંભ કરીએ છીએ, જો તમે ક્રમાંકનને જુઓ છો, તો મને આપો, નવી દિલ્હી 1 લેબલવાળી જોડણી સાથે સંબંધિત છે અને ત્રિવેન્દ્રમ લેબલવાળી લેબલવાળી 10 લેબલ થયેલ છે. અને આપણે જાણીએ છીએ કે, આપણે શું મેળવી શકીએ વર્ટીક્સ 1 થી નવી દિલ્હીથી વર્ટેક્સ 10 ત્રિવેન્દ્રમ. તેથી, આપણે શું કરીએ છીએ, આપણે 1 લીટી પર શરુ કરીએ છીએ અને આપણે જાણીએ છીએ કે પાડોશીઓને કેવી રીતે શોધી શકાય છે. તેથી, આપણે કહીશું કે પ્રત્યેક શંકુ 1 નું કોઈ પાડોશી એકથી એક સુધી પહોંચી શકે છે. તેથી, આપણે શૂન્ય 1 થી પ્રારંભ કરીએ છીએ જે આપણે પહેલેથી શરૂ કરી દીધી છે, આપણે પહોંચી શકીએ છીએ, કારણ કે આપણે ત્યાંથી શરૂ કરીએ છીએ. હવે, આપણે સ્કેન કરીએ છીએ કે તે પાડોશીઓ છે અને પછી અમને લાગે છે કે આ 3 પાડોશીઓ, 2, 3 અને 4 છે. તેથી, હવે આપણે નિષ્કર્ષ કરી શકીએ કે જો તમે 1 થી પ્રારંભ કરો છો, તો આપણે ચોક્કસપણે 2, 3 અને 4 સુધી પહોંચી શકીએ છીએ. આપણે આ પંક્તિઓ પણ લીલા રંગીશું, તેથી હવે આપણે સૂચવ્યું છે કે 1 થી એક પગલામાં, આપણે 2, 3 અને 4 સુધી પહોંચી શકીએ છીએ. હવે, જે કંઈપણ આપણે 2, 3 અને 4 થી મેળવી શકીએ તે પણ 1 થી પહોંચી શકાય છે. તેથી, હવે આપણે એકબીજાને પડોશીઓ તરફ ધ્યાન કેન્દ્રિત કરીએ છીએ કે આપણે પહેલાથી જ શોધી કાઢ્યું છે અને તેમના પાડોશીઓ તરફ ધ્યાન આપીએ છીએ. તેથી, અમે 2 ના પડોશીઓ તરફ ગયા. તેથી, 2 પાસે ફક્ત 2 પડોશીઓ, 1 અને 3 છે અને તે તારણ આપે છે કે 1 અને 3 ને પહેલેથી જ 1 થી પહોંચવા માટે ચિલ્નિત કરવામાં આવ્યા છે, અમે આ વિશે કંઈ જ નથી. તેથી, 2 થી પહોંચતા શહેરો અમારી સમસ્યામાં કોઈ માહિતી ઉમેરતા નથી. તેથી, તમે 3 સુધી જઈ શકો

છો, તેવી જ રીતે 3 1 અને 2 સુધી પહોંચી શકે છે, જેમાંથી બંને માં મુલાકાત લીધેલ તરીકે પહેલેથી જ સૂચિબદ્ધ કર્યા છે અથવા જેમાંથી 1 સુધી પહોંચી શકાય છે. તેથી, અમે ફરીથી 4 સુધી જઈ શકીએ છીએ, કારણ કે 3 પાસે નવું કંઈ નથી અમને જણાવો. હવે, જ્યારે આપણે **વર્ટેક્સ** 4 સુધી પહોંચીએ છીએ, ત્યારે તમને કહેવા માટે કંઈક નવું છેઓ, કારણ કે હવે 4 થી, તે કહે છે કે તમે અલબત્ત 1 સુધી પહોંચી શકો છો, આપણે જાણીશું, પછી આપણે 5 અને 8 સુધી પહોંચી શકીશું. તો હવે, આપણે હવે અમારા પ્રતિનિધિત્વમાં સૂચવીશું કે 5 અને 8 રંગીન લીલા છે સૂચવે છે કે આ 1 થી પરોક્ષ પાથ દ્વારા પણ પહોંચી શકાય છે, હું 1 થી 4 અને પછી 4 થી 5 અને 8 સુધી જઈ શકું છું. તેથી, મેં પહેલેથી જ 1, 2, 3 અને 4 પર પ્રક્રિયા કરી છે, તેથી પછી હું આગળ જોઈશ 5 જોવા માટે વર્ટેક્સ 5 અમને કોઈ નવી પાડોશી આપી શકે છે. તેથી, તમે 5 ને જુઓ છો અને હવે 5 પાસે પાડોશી છે જે આપણે પહેલેથી જ 4 જોયો છે, પરંતુ તેની પાસે એક નવો પાડોશી 6 અને અન્ય નવો પાડોશી 7 છે, તેથી જો હું 5 પર પ્રક્રિયા કરું તો મને 6 અને 7 મળશે. હવે, હું કોઈપણ જોઈ શકું છું 6, 7 અથવા 8 ની, તેથી ચાલો આપણે 8 તરફ જોઈએ. તેથી, 8 પાસે પડોશીઓ 4 છે, જે મેં પહેલાથી જોઈ લીધેલ છે, 6 જે મેં પહેલાથી જોઈ લીધું છે, પરંતુ તેમાં નવું પાડોશી 9 છે, જે મેં જોયું નથી. તેથી, હું પાડોશીઓની સૂચિમાં 9 ઉમેરો છું જે 1 થી આખરે પહોંચી શકાય છે. અને હવે જો હું 6 ને જોઉં, તો મને લાગે છે કે 6 5, 7, 8 અને 9 સુધી પહોંચી શકે છે; જે બધા પહેલેથી પડોશીઓ તરીકે જાણીતા છે 1 થી પહોંચી શકાય છે. તેથી, મારી પાસે કોઈ નવી માહિતી નથી, તેવી જ રીતે જ્યારે હું 7 જાઉં છું, ત્યારે હું 5 અને 6 સુધી પહોંચી શકું છું, જે મને પહેલાથી ખબર છે તે પહોંચી શકાય તેવું છે. તેથી, ત્યાં કોઈ નવી માહિતી નથી. અને આખરે, જે વર્ટેક્સ હજુ સુધી તપાસવામાં આવ્યું નથી તે 9 છે, તેથી 9 થી 9 સુધી હું 6 સુધી પહોંચી શકું છું જે મને ખબર છે, 8 જે હું જાણું છું અને ત્યાં એક નવું વર્ટેક્સ 10 છે, તેથી હું 9 માંથી 10 સુધી પહોંચી શકું છું, તેથી હું 10 ને ચિલ્લિનત કરું છું અને એક વખત મેં 10 ચિલ્લિનત કર્યા છે, મારી સમસ્યા હલ થઈ ગઈ છે. મને જાણવા મળ્યું છે કે 1 થી 10 સુધી જવાનો માર્ગ છે, પરંતુ પડોશીઓના સેટને વ્યવસ્થિત રીતે વિસ્તૃત કરવાનો હું એક સમયે એક સ્તર સુધી પહોંચી શકું છું. તેથી, આ આપણને અમુક અલ્ગોરિધમ આપે છે, આપણે આ ગાણિતીક નિયમોને વધુ ચોક્કસ બનાવીશું કારણ કે આપણે લાંબા સમય સુધી જઈએ છીએ, પરંતુ તમે જોઈ શકો છો કે આ રજૂઆતને એડજસ્ટિંગ સી મેટ્રિક્સ તરીકે ઉપયોગ કરીને, આપણે આ મેટ્રિક્સ લઈ શકીએ છીએ અને વાસ્તવમાં તે સમસ્યાને અન્વેષણ કરવા માટે તેનો ઉપયોગ કરી શકીએ છીએ હાથમાં અને તેને યોગ્ય કાર્યવાહી તરીકે અસાઈન કરવામાં આવે છે જે અમે અસરકારક રીતે અમલમાં મૂકી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:50)

તેથી, આ અલ્ગોરિધમનો વધુ ચોક્કસ બનાવવા માટે, આપણને આ ઍલ્ગોરિધમને વધુ વ્યવસ્થિત બનાવવાની જરૂર છે, આપણે જે શિરોબિંદુઓની મુલાકાત લીધી છે તેનો ટ્રૅક રાખવો પડશે, જેથી આપણે સમાન વર્ટેક્સ નું સંશોધન ચાલુ રાખીશું નહીં ફરી. તેથી, આપણે વર્તુળમાં ફરતા જવું અને ફરીથી અને ફરીથી તે જ સમસ્યા કરવાનું ચાલુ રાખવા નથી માંગતા. તેથી, તે ચાલુ કરશે કે આ વિશિષ્ટ સમસ્યાને ઉકેલવા માટે બે મૂળભૂત વ્યૂહરચનાઓ છે જે ગ્રાફમાં સૌથી મૂળભૂત સમસ્યા છે, જે શોધી કાઢશે, શું જોડાયેલું છે. તેથી, આપણે જે વ્યૂહરચનાને અમલમાં મુક્યા તે સૌથી પહેલા પહોળાઈ કહેવામાં આવે છે, તે છે કે આપણે સૌ પ્રથમ આ તમામ ચીજોની શરૂઆતના પડોશીઓમાંથી આ પડોશીઓની શોધ કરીએ છીએ જે એક પગથિયાં દૂર છે, બધા પડોશીઓ તે વસ્તુ જે બે પગથિયાં દૂર છે અને તેથી ચાલુ અન્ય દિશામાં જ્યાં સુધી શક્ય હોય ત્યાં સુધી જવાનું છે, તેથી તમે શરુઆતમાં એક પાડોશીને પસંદ કરો છો, પછી તમે નવી શંકુની એક પાડોશીને પસંદ કરો છો, પછી તમે તે નવી શંકુની એક પાડોશીને પસંદ કરો છો. અને આપણે નવી કડી શોધી શકતા નથી, પછી તમે પાછા જાઓ અને અન્વેષણ કરો અને પાછલા શિર્ષકોના અન્ય પાડોશીઓ અને તેથી આગળ અને આને પહોળાઈ કહેવામાં આવે છે. તેથી, આપણે આ એલ્ગોરિધમ્સ પછીથી લેક્ચરમાં વિગતવાર જોઈશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:50)

તેથી, તમે જે વસ્તુને જોઈ શકો તે એક છે કે આ મેટ્રિક્સમાંની મોટાભાગની એન્ટ્રીઓ વાસ્તવમાં 0 છે. તેથી, યાદ રાખો કે જો તમારી પાસે n શિરોબિંદુ હોય, તો આ મેટ્રિક્સ કદ n ચોરસ છે, કારણ કે હું એન પંક્તિઓ અને એન કૉલમ્સ છે. હવે, જો તમે નિર્દેશિત ગ્રાફમાં ધારની સંખ્યાને ગણતરી કરો છો, તો શિરચ્છેદનો દરેક જોડી ધાર હોઈ શકે છે, અમે સામાન્ય રીતે સ્વ-લૂપ્સને મંજૂરી આપીએ છીએ, અમે સામાન્ય રીતે વર્ટેક્સ i માટે i થી i ની ધારને ધ્યાનમાં લઈએ નહીં. અને જુદા

જુદા જોડીઓની સંખ્યા 2 ને પસંદ કરે છે, આ બે શિરોબિંદુઓ પસંદ કરી શકે તે રીતે આ છે. તેથી, તમે બધા જોડીઓને પસંદ કરો છો, પછી તે n માં minus 1 થી 2 થાય છે, તેથી આ મૂળભૂત સામાન્ય લક્ષિત હકીકત છે. તેથી, અમારી પાસે n ચોરસ કિનારીઓ વિશે વધુ હોઈ શકે છે અને જો તમે n ચોરસ કિનારીઓ વિશે સાંભળ્યું છે, તો મેટ્રિક્સમાંની ઘણી એન્ટ્રીઓ 1 હશે, પરંતુ તે મોટાભાગની પરિસ્થિતિઓમાં કિનારીઓની સંખ્યા n ચોરસ કરતા ઓછી છે. તેથી, આ કોઈ અર્થમાં નકામું પ્રતિનિધિત્વ છે, કારણ કે 1 ની સંખ્યામાં હકારાત્મક માહિતીને પ્રાપ્ત કરવા માટે અમે ઘણાં 0 નો રેકોર્ડિંગ કરી રહ્યા છીએ. અને તે પણ યાદ રાખો કે આ 1 નું સમપ્રમાણિત છે, તેથી મારી પાસે લીટી વિશે દરેક 1 માટે, તેથી જો હું આ ત્રિકોણાકાર દોરું, કે જે પ્રત્યેક 1 ત્રિભુણા ઉપર છે, મારી પાસે ત્રાંસા નીચે એક સમપ્રમાણ 1 છે. તેથી, આ મેટ્રિક્સનો અડધો ભાગ નકામું છે, જો હું જાણું કે આ ટોચનો અડધો ભાગ પુરતો છે અને આ ટોચના ભાગમાં મારી પાસે મોટે ભાગે 0. હશે. તેથી, આપણે બીજા રજૂઆત વિશે વિચારી શકીએ છીએ, જ્યાં આપણે ફક્ત સંબંધિત માહિતીને હાથમાં રાખીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:20)

તેથી, આને અડજાનસી (adjacency)ને સૂચિ કહેવામાં આવે છે, તેથી આપણે અડજાનસી (adjacency)ને સૂચિમાં જે કરીએ છીએ તે એ છે કે આપણે સ્પષ્ટપણે દરેક પાર્ટિકલ્સ, તેના પડોશીઓની સૂચિ માટે જાળવી રાખીએ છીએ. તેથી, આપણે જોયું કે એક 2, 3 અને 4 થી જોડાયેલ છે. તેથી, આપણે કહીએ છીએ કે 1 થી 3, 3, 4 સાથે જોડાયેલ સૂચિ એ જ છે, 2, 1 અને 3 જોડાયેલ છે, તેથી આપણી પાસે જોડાયેલ સૂચિ છે 1 અને 3, તેથી આ નોડ છે અને આ પડોશીઓ છે. તેથી, હવે, આ પ્રતિનિધિત્વમાં, જ્યારે પણ મને નોડ હોય ત્યારે, હું તે નોડ માટે તે એન્ટ્રી પર જાઉં છું અને હું સૂચિને જોઉં છું અને હું તેને પાડોશીઓને સ્કેન કરી શકું છું અને હું તેને પડોશીઓની સંખ્યાના પ્રમાણમાં મેળવી શકું છું. તેથી, ત્યાં કોઈ ફાયદો છે કે મને કોઈ નકામું માહિતી જોઈતી નથી, કારણ કે હું અહીં 0 ની નજીકના મેટ્રિક્સ સંગ્રહિત નથી કરતો, હું સામાન્ય માહિતીને જાળવી રહ્યો નથી જે વર્ટેક્સ કનેક્ટ નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:12)

તેથી, આ બે રજૂઆતમાં તેમના ફાયદા અને ગેરફાયદા છે, અડજાનસી (adjacency)ને મેટ્રિક્સમાં, અમને વધુ જગ્યા અને પછી અડજાનસી (adjacency)ને સૂચિની જરૂર છે, પરંતુ કેટલાક પ્રશ્નો આસાનીથી કરતા અડજાનસી (adjacency)ને મેટ્રિક્સનો જવાબ આપવા માટે સરળ છે. યાદી. જો તમે જાણવા માંગતા હો કે, વેરટેક્સ જે વર્ટીક્સ બાજુનો પાડોશી છે, આપણે ફક્ત મેટ્રિક્સમાં એક એન્ટ્રી સાબિત કરવી પડશે, જો આપણે એજે જો 1 હોય તો આપણે તેને તપાસો. બીજી તરફ અડજાનસી (adjacency)ને સૂચિમાં, જો તમે ઇચ્છો શોધવા માટે j એ મારો પાડોશી છે, અમને એન્ટ્રી પર જવાની અને સમગ્ર સૂચિને સ્કેન કરવાની જરૂર છે. તેથી, આ મેટ્રિક્સ અને આ વચ્ચેના તફાવત પર સૉર્ટિંગ મોડ્યુલની શરૂઆતમાં અમારી પાસે મૂળ ચર્ચા સમાન છે. તેથી, અહીં અમે એકમ સમયે આ પ્રવેશની તપાસ કરી શકીએ છીએ, જ્યારે i અથવા પાડોશી લિસ્ટ શોધવા માટે, i ના પાડોશીઓની સંખ્યા માટે પ્રાયોગિકતાની જરૂર છે. બીજી તરફ, જો તમે વાસ્તવમાં મારા બધા પડોશીઓને શોધી કાઢો છો, તો પછી અસંબંધિત મેટ્રિક્સમાં કેટલા પડોશીઓની જરૂર છે તેના પર ધ્યાન આપવું જોઈએ, અમને સમગ્ર પંક્તિને સ્કેન કરવી પડશે. તેથી, ગ્રાફમાં n વર્ટેક્સ છે, જ્યારે નોડમાં ફક્ત 2 અથવા 3 પાડોશીઓ હતા, ત્યારે પણ નિર્દેશિત કરવામાં આવશે કે આમાંની એન એન્ટ્રીઓ ખરેખર વાસ્તવિક પડોશીઓ છે. બીજી તરફ, આપણે જોયેલી અડજાનસી (adjacency)ને સૂચિમાં, અમે તે બરાબર રેકોર્ડ કરીએ છીએ પરંતુ તે એક પડોશીઓ છે. તેથી, પડોશીઓને સ્કેન કરવાનો સમય સીધી રીતે પાડોશીઓને પૂર્વગામી છે. તેથી, જો પ્રત્યેક શિરચ્છેદ ફક્ત નાના સંખ્યામાં પડોશીઓ હોય, તો અડજાનસી (adjacency)ને સૂચિ તે પડોશીઓ તરીકે ઝડપથી આપવા માટે, જ્યારે અડજાનસી (adjacency)ને મેટ્રિક્સને તમારે નાના નંબરને શોધવા માટે સ્કેન ઑર્ડર એન એન્ટ્રીઝની આવશ્યકતા હોય.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 03 મોડ્યુલ - 03 લેક્ચર - 20
બ્રેડથ ફર્સ્ટ સર્ચ(બીએફએસ(BFS)) (Breadth First Search (BFS))

(સ્લાઈડટાઈમનો
સંદર્ભ લો: 00:05)

તેથી, ચાલો ગ્રાફનું અન્વેષણ કરવા માટે ઔપચારિક એલ્ગોરિધમ જોઈએ. તેથી, યાદ રાખો કે ગ્રાફમાં શિરોલંબનો સમૂહ અને ધારનો સમૂહ છે, કિનારીઓ વચ્ચેના જોડાણ છે, તે દિશા નિર્દેશિત અથવા નિર્દેશિત હોઈ શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:15)

તેથી, જો આપણે તેને અણધાર્યા ગ્રાફ જુઓ, તો જે સમસ્યા આપણે જોઈ રહ્યા છીએ તે શોધવા માટે છે કે શું સ્રોત વર્ટેક્સ લક્ષ્ય વર્ટેક્સ સાથે જોડાયેલું છે. અને અમે કહ્યું કે આ સ્ત્રોત વર્ટીક્સ વિરુદ્ધ સ્ક્રોલ વર્ટેક્સ વિરુદ્ધ વી પાથ શોધવા માટેની આ રકમ, જ્યાં પાથ પરના દરેક જોડી ગ્રાફમાં ધાર દ્વારા જોડાયેલ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:35)

તેથી, અમે એવી દલીલ કરી હતી કે આપણે નાના ગ્રાફ્સ માટે દૃષ્ટિથી આ કરી શકીએ છીએ, પરંતુ જો તમે એલ્ગોરિધમ લખવા માંગો છો, તો ગ્રાફને રજૂ કરવાની રીતની જરૂર છે. તેથી, આપણે જે પ્રથમ વસ્તુ નક્કી કરી હતી તે છે, કે આપણે શિરોબિંદુઓ 1 થી n ને નામ આપીએ. તેથી, આપણી પાસે શિર્ષકો છે, આપણે તેને 1, 2, 3, 4 ને n સુધી બોલાવીશું. પછી આપણે અડજાનસી (adjacency) મેટ્રિક્સ દ્વારા ગ્રાફના માળખાને રજૂ કરી શકીએ છીએ. આ અડજાનસી (adjacency) મેટ્રિક્સમાં, હું જે એન્ટ્રી વર્ટીક્સ i થી વર્ટીક્સ j થી ધારની ભેટ અથવા ગેરહાજરી સૂચવે છે. તેથી, Aij 1 છે, ત્યાં ધાર છે, aij 0 છે, જો કોઈ વિષય નથી. આ મેટ્રિક્સમાં, જો તમે શોધવા માંગતા હો કે, શિરચ્છાનો જોડી સીધો કનેક્ટ થઈ રહ્યો છે, તો અમારે ફક્ત તેમની યોગ્ય એન્ટ્રી સાબિત કરવી પડશે, અમે તે સતત સમયમાં કરી શકીએ છીએ. શૂન્યાવકાશના તમામ આઉટગોઇંગ પાડોશીઓને શોધવા માટે, અમને તે શિરચ્છેદ માટે પંક્તિને સ્કેન કરવી પડશે. તેથી, શિરોબિંદુની સંખ્યાના સંદર્ભમાં તે રેખીય સમય લે છે. હવે, એક વસ્તુ જે આપણે જોયેલી તે છે કે, સામાન્ય રીતે ઘણાં ગ્રાફ્સ, આ મેટ્રિક્સ મોટે ભાગે 0 છે, મોટાભાગના જોડી જોડાયેલા નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:36)

તેથી, અમે દરેક સૂચિના પાડોશીઓને આ સૂચિ દ્વારા વધુ કોમ્પેક્ટ પ્રતિનિધિત્વ મેળવી શકીએ છીએ. તેથી, 1 અને 0 ની પંક્તિને રાખવાને બદલે, અમે તે શિર્ષકો રેકોર્ડ કરીએ છીએ, જેની એન્ટ્રીઓ 1 છે. તેથી, આ ગ્રાફ્સ માટે વધુ કાર્યક્ષમ રજૂઆત તરીકે રાખે છે, જ્યાં ધારની સંખ્યા શિરોબિંદુઓની સંખ્યાની નજીક છે. પરંતુ, આ કિસ્સામાં શોધવા માટે, આપેલ જોડી ij જોડાયેલ છે કે કેમ, જો તેમાં જોવામાં આવે તો, હું સૂચિ તરફ જોવું જોઈએ અને જોવું જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:00)

હવે, લક્ષ્ય વર્ટેક્સમાં સોર્સ વર્ટેક્સને જોડતા પાથ શોધવા માટેની અમારી વ્યૂહરચના, જે નીચે મુજબ છે, અમે સ્રોત વર્ટેક્સથી પ્રારંભ કરીશું અને પદ્ધતિસર રીતે ગ્રાફને અન્વેષણ કરીશું. દર વખતે, અમે એક શિરચ્છેદ પર ગયા, તમે તેને મુલાકાત તરીકે ચિલ્નિત કરશો, અને પછી અમને ખાતરી કરવી પડશે, કે જ્યારે તમે શિરોબિંદુઓની શોધ કરી રહ્યા હો, ત્યારે અમે તે જ કર્ણને ફરીથી બે વાર શોધી શક્યા નહીં. તેથી, અમે કહ્યું હતું કે તેઓ બે મૂળભૂત વ્યૂહરચનાઓ પ્રથમ અને ઊંડાઈ પહેલા પહોળા થશે. તેથી, આજે આપણે પહોળાઈ પ્રથમ શોધ તરફ જોશો.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:28)

તેથી, વિચારની પહોળાઈ પ્રથમ શોધ સ્તર દ્વારા ગ્રાફ સ્તરે અન્વેષણ કરવાનો છે. તેથી, આપણે સ્રોત વર્ટેક્સ પર પ્રારંભ કરીએ છીએ અને હવે આપણે બધા શિરોબિંદુઓનું અન્વેષણ કરીએ છીએ, જે એક પગથિયું દૂર છે, જે સ્રોત વર્ટેક્સ તરફ સીધા ધાર દ્વારા જોડાયેલ છે. પછી, આપણે બધા શિરોબિંદુને બહાર કાઢી નાખીએ છીએ જે નવા સ્તરે 1 લેવલથી દૂર છે. તેથી, આ સ્રોત વર્ટેક્સથી બે પગલા દૂર છે, પછી ત્રણ પગલાઓ અને તેથી આગળ. હવે, જ્યારે આપણે આ સેટ

સમયગાળો કરી રહ્યા છીએ, ત્યારે આપણે બે જથ્થાઓનો ટ્રૅક રાખવો પડશે, આપણી પાસે પ્રથમ કોર્સ નોંધવું પડશે, આપેલ શિરચ્છેદની મુલાકાત લીધી છે કે કેમ. શું તે શોધવામાં આવ્યું છે, તે સ્રોત વર્ટેક્સ સાથે જોડાયેલું છે, તે પ્રત્યેક આવા શૂન્ય માટે પણ, આપણે તેને પાડોશીઓની શોધ કરવી જોઈએ. તેથી, દરેક શિખર માટે, આપણે જાણવાની જરૂર છે કે, તે મુલાકાત લેવામાં આવી છે અને અમે મુલાકાત લઈએ તે પછી અમે તેને શોધી શકતા નથી. તેથી, આપણે યાદ રાખવાની જરૂર છે કે અમે તેને શોધ્યું છે અથવા તેનું સંશોધન કર્યું નથી અને જે લોકો અન્વેષણ નથી કરી રહ્યાં છે, આપણે ખાતરી કરવી પડશે કે અમે એકવાર તેમને શોધીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:26)

તેથી, આનો માર્ગ આ જથ્થોને ટ્રૅક રાખવા માટે કેટલાક ડેટા માળખાંનો ઉપયોગ કરવો છે. જેમ આપણે કહ્યું હતું કે શિરોબિંદુનો સમૂહ 1 થી n ની સંખ્યા છે. તેથી, આપણે એરે(Array) 1 થી n એ એન્ટ્રીની મુલાકાત લઈ શકીએ છીએ, જે અમને કહે છે કે શું વર્ટેક્સની મુલાકાત લીધી નથી કે નહીં. હવે, જ્યારે હું એક શિરચ્છેદની મુલાકાત લઈશ, ત્યારે આપણે તેને પછીથી શોધવાની જરૂર છે. ત્યારથી, અમે તેને તાત્કાલિક અન્વેષણ કરી શકતા નથી, અમને થોડીવાર માટે બાકી રહેવું પડશે. તેથી, આપણે આ શિરોબિંદુઓનું સંગ્રહ રાખવું પડશે, જે મુલાકાત લેવામાં આવી છે, પરંતુ હજી સુધી શોધવામાં આવી નથી. આ કરવા માટે ઘણાં રસ્તાઓ છે, પરંતુ તેઓ જે રીતે મુલાકાત લેશે તેના આધારે અન્વેષણ કરવાનો એક કુદરતી રસ્તો છે. તેથી, આપણે જે ક્રમમાં મુલાકાત લઈએ છીએ તે ક્રમમાં અન્વેષણ કરવા માંગીએ છીએ, એક અજ્ઞાત ડેટા સ્ટ્રક્ચરની અવગણના કરવામાં આવે છે, પરંતુ મુલાકાત લેવાયેલા શિરોબિંદુઓ એક કતાર છે. તેથી, આપણે દરેક મુલાકાત લીધેલ કતારને કતારમાં મૂકીએ છીએ, પ્રથમ વખત આપણે તેમાં આવીએ છીએ, અને પછી આપણે બધા શિરોબિંદુઓને કતારમાં પ્રક્રિયા કરીએ ત્યાં સુધી, બધા શિરોબિંદુઓની શોધ થઈ જાય નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:30)

તો, બીજું ઉચ્ચ સ્તરનું સ્યુડો કોડ, પહોળાઈની પ્રથમ શોધ માટે, જ્યારે આપણે પહેલી વખત જે વર્ટીક્સમાં પહોંચ્યા છીએ તે પહોંચે ત્યારે, આપણે આખરે તેનો અન્વેષણ કરીશું. શંકુદ્રષ્ટિની શોધ કરવાનો અર્થ શું છે, અમે આઉટગોઇંગ કિનારીઓ પર ધ્યાન આપીએ છીએ, જો જે પહેલેથી મુલાકાત લીધી હોય, તો આપણે કંઈ કરવાનું નથી. બીજી તરફ, જો j ની મુલાકાત લીધી ન હોય, તો અમે તેને મુલાકાત તરીકે ચિલ્લિનત કરીશું અને પછીથી અન્વેષણ કરવા માટે કતાર ઉમેરીશું. તેથી, જો સ્રોત વર્ટેક્સને ચિલ્લિનત કરીને અને તેને કતારમાં અને દરેક તબક્કે ઉમેરીને પ્રારંભ કરીએ, તો અમે કતારના મુખ્ય ભાગની શોધ કરીશું. જો કોઈ તબક્કે, કતારમાં કોઈ શિરોબિંદુ નથી, તો તેનો અર્થ એ છે કે દરેક શિર્ષક, આપણે મુલાકાત લીધી છે તેનું સંશોધન કરવામાં આવ્યું છે અને તેથી ત્યાં કોઈ નવી માહિતી બાકી નથી અને અમે સંશોધનને રોકી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:10)

તેથી, વાસ્તવિક કોડને પહોળાઈ માટે પ્રથમ શોધ આપતા પહેલા, ચાલો આપણે જે ઉદાહરણ આપીએ છીએ અને જુઓ, આ માહિતી માળખાં કેવી રીતે અપડેટ કરવામાં આવે છે. તેથી, આપણી પાસે આ 10 શિરોબિંદુઓ 1 થી 10 છે અને આપણે તેને 1 થી શરૂ કરવાની રીત શોધી છે. તેથી, કતારમાં, આપણી પાસે બે પોઈન્ટર હોય છે, આપણી પાસે એક પોઈન્ટર હોય છે જે આપણે લીલા રંગમાં સૂચવીએ છીએ, જે કતારનો મુખ્ય ભાગ છે. , નિર્દેશક લાલ, કે જે કતારની પૂંછડી છે. તેથી, આપણે કતારને પ્રારંભિક શૂન્ય 1 નો પ્રારંભ કરવા માટે પ્રારંભ કરીએ છીએ અને આપણે પણ આ શંકુની મુલાકાત લેવાનું ચિલ્લિનત કરીએ છીએ. હવે, આપણે 1 અન્વેષણ કરવું પડશે, તેથી આપણે શું કરીએ છીએ, આપણે કતારમાંથી 1 ને દૂર કરીએ છીએ અને વ્યવસ્થિત રીતે તપાસ કરીએ છીએ કે તે દરેક આઉટગોઇંગ ધાર છે અને જો તે લક્ષ્ય શિરોબિંદુ પહેલેથી જ મુલાકાત લીધેલા નથી, તો અમે તે ઉમેરીશું. તો, 1 થી 10 ના ક્રમમાં પ્રથમ બે છે, 1 નું પ્રથમ પાડોશી, તે મુલાકાત લીધી નથી, તેથી અમે તેને કતારમાં ઉમેરીએ છીએ અને મુલાકાત લીધેલા એરે(Array)માં મુલાકાત લેવા માટે ચિલ્લિનત કરીએ છીએ. તેવી જ રીતે, અમે 3 ને ચિલ્લિનત કરીએ છીએ, અને પછી આપણે 4 ચિલ્લિનત કર્યું છે, આ સ્થિતિમાં, અમે શૂન્ય 1 નું સંશોધન કર્યું છે. તેથી, હવે, આપણે કતાર પર પાછા જઈએ છીએ, જો ત્યાં હજુ પણ શોધખોળ કરવાનું બાકી છે, જેની પાસે છે મુલાકાત લીધી છે અને ત્યાં અમે પ્રથમ શોધ એટલે કે 2 પસંદ કરીએ છીએ. તેથી, હું કતારમાંથી 2 ને દૂર કરું છું અને અમે શોધ્યું છે કે તે બહારના પાડોશીઓ છે. હવે, આ કિસ્સામાં 2 ના

આઉટગોઈંગ પાડોશર્સ 1 અને 3 છે. પરંતુ, 1 અને 3 બંને મુલાકાત લીધેલા હોવાથી, અમારે કોઈ કાર્ય નથી અને અમે કતાર પર પાછા જઈએ છીએ. આગળનાં તબક્કામાં, આપણે કડી 3 પસંદ કરીએ છીએ અને આપણે તે બહાર જનારા પાડોશીઓ તરફ ધ્યાન આપીએ છીએ. ફરી, તે બહાર જનારા પાડોશીઓ 1 અને 2 છે, બંને સુવિધા પહેલાથી જ મુલાકાત લીધી છે, તેથી કંઇ કરવાનું બાકી નથી. અમે કતારમાં પાછા આવીએ અને 4 પીકઅપ લઈએ અને તેનો અન્વેષણ કરવાનો પ્રયાસ કરીએ. તેથી, 4 પાસે પાડોશી 1 છે, જેનો પહેલેથી જ મુલાકાત લેવામાં આવ્યો છે, ત્યારબાદનો પાડોશી 5 છે, કારણ કે 5 કતારમાં આપણે ઉમેરાયેલી નવી છે અને આપણે તેને મુલાકાત તરીકે ચિલ્નિત કરીએ છીએ, તો તેની પાસે પડોશીઓ છે. તેથી, આપણે તેને પણ ઉમેરીશું. કતાર અને મુલાકાત તરીકે તેને ચિલ્નિત કરો. હવે, આપણે 4 અન્વેષણ કરવાનું સમાપ્ત કરી લીધું છે. તેથી, આપણે કતાર પર પાછા જઈએ છીએ અને આગામી વસ્તુની શોધ કરીશું, જે આ કિસ્સામાં 5 છે. તેથી, 5 ને પડોશીઓ 4, 6 અને 7 મળ્યાં; 4 પહેલાથી જ મુલાકાત લેવામાં આવી છે, પરંતુ 6 નવું છે અને તેથી 7 છે. હવે, અમે પાછા જાઓ અને 8 નું અન્વેષણ કરીએ, તેથી 8 પાસે પડોશીઓ 4, 6 અને 9 છે; 4 અને 6 જૂના છે, પરંતુ 9 નવું છે, અમે કતારમાં 9 ઉમેરીએ છીએ. હવે, આપણે પાછા જાવ અને 6 નું અન્વેષણ કરીએ, કારણ કે 6 એ કતારના માથાના શિર છે. તેથી, હવે, આપણે 6 ના પાડોશીઓને જોઈએ છીએ, 6 ના પડોશીઓ 5, 7, 8 અને 9 છે. તેથી, જ્યારે આપણે કતારમાંથી 6 ને દૂર કરીએ છીએ, ત્યારે આપણે કંઈ કરવાનું નથી, આપણે પાછા જઈએ અને શોધી શકીએ, ત્યાં કંઈ નથી કરવામાં આવે છે, કારણ કે 6 ના બધા પડોશીઓ પહેલેથી જ મુલાકાત લીધી છે. પછી, અમે 7 પસંદ કરીએ છીએ; ફરી એકવાર આપણે શોધી કાઢીએ છીએ કે 7 ના બધા પડોશીઓ પહેલેથી જ મુલાકાત લીધાં છે. છેવટે, અમે 9 પસંદ કરીએ છીએ અને અમે 9 પસંદ કરીએ છીએ, અમને લાગે છે કે તેની પાસે પાડોશી 6, 8 અને 10, 10 નવું છે, તેથી 10 માર્ક કરે છે અને તેને કતારમાં ઉમેરો. છેલ્લે, અમે 10 પર જાઓ, અમને લાગે છે કે તેમાં માત્ર એક પાડોશી 9 છે, જેનો પહેલેથી જ મુલાકાત લેવામાં આવી છે, આ તબક્કે કતાર ખાલી થઈ ગઈ છે અને અલ્ગોરિધમ સમાપ્ત થઈ ગયું છે. હવે, આ કિસ્સામાં, આપણે બધા શિરોબિંદુઓની મુલાકાત લીધી છે, જે સૂચવે છે, કે જે સૂચવે છે કે આ લેબમાં પ્રત્યેક શિરોબિંદુ વાસ્તવમાં તેનાથી પહોંચી શકાય તેવું છે. જો આપણે વધુ શિરોબિંદુ ઉમેર્યા હોય, ઉદાહરણ તરીકે, જો આપણે આ ગ્રાફમાં ઘટક ઉમેર્યું હોય; તે ગ્રાફને હોઈ શકે તેવા ઘણાં ઘટકોથી અમને રોકવા માટે કંઈ નથી, તે વધુ 11, 12, 13 અને 14 ની શિખરો છે. ત્યારબાદ 1 થી શરૂ થતી આ પ્રક્રિયા 11, 12, 13 અને 14 છોડી દેશે અને અવ્યાખ્યાયિત તરીકે ચિલ્નિત થશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:54)

તેથી, અહીં બીએફએસ(BFS) માટે વધુ ઔપચારિક કોડ છે, તેથી બીએફએસ(BFS) એ એક કાર્ય છે જે તે દલીલ કરે છે, જ્યાંથી આપણે ગ્રાફને અન્વેષણ કરવા માંગીએ છીએ. તેથી, જેમ આપણે કહ્યું હતું કે શિરોબિંદુઓ 1 થી n તરીકે ઓળખાય છે, આપણી પાસે આ એરે(Array)ની મુલાકાત લીધી છે, જે 0 થી શરૂ થાય છે અને આપણી પાસે શિર્ષકોને હજી પણ શોધવા માટે ચિલ્નિત કરવા માટે એક કતાર છે, આ ખાલી છે. તેથી, અમે ચિલ્નિત કરીને પ્રારંભ કરીએ છીએ કે જે 1 ની બરાબર મુલાકાત લીધી. તેથી, વર્ટીક્સ હું ચિલ્ન છે જે મુલાકાત લીધી અને અમે કતારમાં ઉમેર્યા. હવે, જ્યાં સુધી કતાર ખાલી નથી, આપણે નીચે આપેલ લૂપ માં આગળ વધીએ છીએ, આપણે કતારના માથાને બહાર કાઢીએ છીએ, તે શિરચ્છેદ છે કે તે પછી તે શોધવામાં આવશે. તે દરેક માટે આઉટગોઈંગ કિનારીઓ છે, જો તે એવી વસ્તુ છે જે પહેલાં મુલાકાત લીધી ન હોય, તો અમે તેને મુલાકાત તરીકે ચિલ્નિત કરીએ છીએ અને અમે તેને કતારમાં ઉમેરીએ છીએ. તેથી, આ બરાબર અલ્ગોરિધમ માટે કોડ છે, અમે ફક્ત હાથ દ્વારા ચલાવીએ છીએ અને તે સીધા જ શાંત છે. મુલાકાત લીધેલા શિરોબિંદુઓ અને કતારના ટ્રેક રાખવા માટે બે ડેટા સ્ટ્રક્ચર્સ અને એરે(Array) છે જે શોધવામાં ફાઈલની જરૂર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:00)

તેથી, અમે કેવી રીતે ઍલ્ગોરિધમની જટિલતાનું વિશ્લેષણ કરીએ છીએ

((સમયનોસંદર્ભ: 10:05)),

લૂપને જોવાનું એક રીત છે. તેથી, યાદ રાખો કે, જ્યારે આપણી પાસે આ જેવી ઍલ્ગોરિધમ હોય

((રેફરટાઈમ: 10:10))

જે લૂપ ધરાવે છે, લૂપ સામાન્ય રીતે તે સ્થાન છે જ્યાં આપણે તેને જોવાનું છે. તેથી, આપણી પાસે એક પુનરાવર્તન છે, જ્યાં આપણી પાસે 1 થી n નો લૂપ છે; જ્યાં આપણે સોંપીએ છીએ, આ તે j બરાબર 0. તેથી, આ કંઈક છે જે ક્રમમાં n સમય લે છે. અને હવે, આપણી પાસે અહીં લૂપ છે, જ્યાં આપણે કતારમાંથી વસ્તુઓ કાઢીએ છીએ અને આપણે અહીં વસ્તુઓ કરીએ છીએ. તેથી, આપણે સમજવું પડશે કે, આ લૂપ એક્ઝેક્યુટ કેવી રીતે ચાલશે અને આ આંતરિક લૂપ કેવી રીતે એક્ઝેક્યુટ થશે. તેથી, દરેક શિરોબિંદુ બરાબર એકવાર કતારમાં પ્રવેશે છે. તેથી, તે એક જ સમયે દાખલ થાય છે,
((સમયનોસંદર્ભ લો: 10:47))
પછી તે કહે છે કે આ બાહ્ય લૂપ અહીં ક્રમમાં સમય લેશે. કારણ કે, આલેખને જોડે છે તેવું માનવું, દરેક કર્ણ એક વાર મુલાકાત લેવામાં આવશે અને જ્યારે તે કાંકરા કતારમાં પ્રવેશવા માટે જશે અને તે બરાબર એકવાર કતારના માથામાં આવશે. હવે, દરેક શિરચ્છેદ માટે, આપણે તે બધાને પડોશીઓને સ્કેન કરવું પડશે. તેથી, આપણે જે કહ્યું તે હતું કે, આ ક્રમમાં n હશે, જો તમારી પાસે અડજાનસી (adjacency) મેટ્રિક્સ હોય. કારણ કે તમારે j માટે પહોળાઈની પ્રથમ એન્ટ્રીમાંથી જવાની જરૂર છે અને આ માટે આખી પંક્તિ જોવી જોઈએ, તેથી આ ઓર્ડર n છે. તેથી, તેથી, આ ગ્રાફ જોડાયેલ છે, દરેક કર્ણમાં પહેલા કતારમાં આવશે. તેથી, આપણે કતાર લૂપ બરાબર n વખત કરીશું. અને દરેક જે જે કાઢવામાં આવે છે તે માટે, આપણે તે બધાને પડોશીઓની તપાસ કરવાની જરૂર છે. તેથી, તે કદ n નું સ્કેન હશે અને તેથી, તે એકંદર n ચોરસ છે, આ બે નેસ્ટેડ લૂપ્સ કદ n છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:43)
તેથી, આ સ્પષ્ટપણે એક પરિસ્થિતિ છે, જ્યાં જો અમારી પાસે ખૂબ જ ધાર હોય, તો અમે અડજાનસી (adjacency) સૂચિ રજૂઆતનો ઉપયોગ કરીને જટિલતાને સુધારી શકીએ છીએ જે અડજાનસી (adjacency) મેટ્રિક્સ રજૂઆતમાં છે. તેથી, કિનારીઓની સંખ્યા m અને m એ n સ્ક્વેર્ડ કરતા ઘણી નાની છે. તેથી, જો એમ તે પોતે જ પ્રમાણમાં કહે છે, તો આપણે j સાથે સંકળાયેલ સૂચિમાં શું કરી શકીએ, જે જેના પાડોશીઓને જોવાનું છે. તમારે તમારા અડજાનસી (adjacency) મેટ્રિક્સમાં બધી ઓર્ડર એન એન્ટ્રીઝના બધા શિરોબિંદુઓ જોવાની જરૂર નથી, આને ડિગ્રી કહેવામાં આવે છે. તેથી, શિરોબિંદુની ડીગ્રી તેનાથી જોડાયેલા ઘણા શિરોબિંદુઓ છે. તેથી, જો આપણે આ કરીએ, તો આપણે જે શોધીશું તે છે કે, આંતરિક લૂપના પુનરાવર્તનની અંદર, પડોશીઓ માટે સ્કેનીંગના દરેક પગલાંમાં સમયનો પ્રમાણસર ડિગ્રી j લે છે. હવે, અલબત્ત j ની ડિગ્રી, એક નોડથી બીજામાં બદલાય છે, તેથી આપણે આ કેવી રીતે યોગ્ય રીતે ગણીશું. તેથી, જો તમે તેને જોડી કરેલ લૂપ ગણતા નથી, પરંતુ સમગ્ર ચક્ર શિરોલંબની સમગ્ર પ્રક્રિયામાં તેને ગણાશો, તો આપણે જે જોશું તે ગ્રાફમાં દરેક ધાર માટે, ફોર્મ હું કોમા જે. ત્યાં એક સમય હશે અને હું પ્રક્રિયા કરીશ અને હું જા સૂચિ જોઈશ અને ત્યાં એક સમય હશે અને પ્રક્રિયા થશે અને હું જોશ કે હું આમાં છું. બીજા શબ્દોમાં કહીએ તો, જો હું બધા શિરોબિંદુઓની બધી બાજુની સૂચિને લઈશ, તો દરેક ધારને બે વખત રજૂ કરવામાં આવશે અને ધાર આઈજે એન્ટ્રી જે તરીકે દેખાશે અને હું અને એન્ટ્રી I માટેની સૂચિ અને જે માટેની સૂચિ. તેથી, જ્યારે, પુનરાવર્તન દીઠ ગણતરી કરવી સરળ નથી, હું તમામ પુનરાવર્તનોમાં ગણું છું, હું પ્રત્યેક ધારને બરાબર બે વાર અન્વેષણ કરીશ. તેથી, 2 મીટરને એમ આદેશ આપ્યો છે, તેથી સમગ્ર પુનરાવર્તન દરમિયાન પડોશીઓની શોધખોળ એક વખત થાય છે. તેથી, આનો અર્થ શું છે કે, મુલાકાત લેવાયેલા દરેક સ્થળની મુલાકાત લેવાનું પ્રથમ પુનરાવર્તન સમય વધુ વધુ સમય લે છે, જ્યારે આપણી પાસે ઓ એન લૂપ હોય છે, જેના પર આપણે એમ પગલાં લઈએ છીએ, તે પૂર્ણપણે એન પગલાંઓ તરફ જાય છે, તેથી તે સંપૂર્ણ એન પ્લસ મી છે. . તેથી, પહોળાઈની પ્રથમ શોધની જટિલતા, જો તમે અડજાનસી (adjacency) સૂચિનો ઉપયોગ કરી રહ્યાં છો, તો તે અડજાનસી (adjacency) મેટ્રિક્સ માટે ઓ n પ્લસ મી વિરુદ્ધ ઓર્ડર n સ્ક્વેર્ડ પર જાય છે. તેથી, જો આપણી પાસે થોડા કિનારીઓ હોય તો તે પહોળાઈ પ્રથમ શોધ માટે અડજાનસી (adjacency) સૂચિનો ઉપયોગ કરવા માટે ઘણું અર્થપૂર્ણ બનાવે છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 14:08)
હવે, ગ્રાફમાં ઈનપુટ કદ સામાન્ય રીતે એમ અને એન પર લઈ જાય છે. બીજા શબ્દોમાં, બંને પરિમાણો એકબીજાથી સ્વતંત્ર છે; કારણ કે આપણે શિરોલ્સનો સમૂહ ઘણા કિનારીઓ સાથે ખૂબ જ થોડા કિનારીઓ બનાવી શક્યા હોત. તેથી, સામાન્ય રીતે એમ અને એન એકસાથે ઈનપુટ કદ લેવામાં આવે છે. તેથી, ઓર્ડર એમ પ્લસ એન ખરેખર રેખીય અલ્ગોરિધમનો છે.

તેથી, આ શ્રેષ્ઠ શક્ય છે, કારણ કે આપણે સામાન્ય રીતે પ્રક્રિયા કરવા માટે ઈનપુટ વાંચવું પડશે નહીં. તેથી, રેખીય સમયમાં, પહોળાઈ પ્રથમ શોધ કરવામાં સક્ષમ હતા.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:40)

તેથી, આપણે બધાં પહોળા પ્રથમ શોધમાં બીજું કંઈક કરી શકીએ છીએ, જે આપણે ઓળખી લીધું છે તે છે કે જે શિરોબિંદુઓ સામાન્ય સ્ત્રોત સાથે જોડાયેલા છે તે સામાન્ય રીતે સ્રોત વર્ટીક્સમાંથી એક છે જ્યાંથી તમે પહોળાઈ પ્રથમ શોધ શરૂ કરો છો. તો, આપણે કેવી રીતે ઓળખી શકીએ, આપેલ શિર્ષક માટે સ્ત્રોત શાખામાંથી કેવી રીતે જવું. તો, જો આપણે મેં અને બીએફએસ(BFS) શરૂ કર્યું હોય તો હું જે 1 બરાબર 1 ની મુલાકાત લીધી, આપણે જાણીએ છીએ કે હું અને j જોડાયેલ છે, પરંતુ આપણે પાથને જાણતા નથી. તેથી, આપણે યાદ રાખી શકીએ કે આપણે સરળતાથી શું કરી શકીએ, જ્યાં આપણે દરેક શિરચ્છેદ ચિલ્નિત કર્યું છે. તેથી, જ્યારે આપણે મુલાકાત લીધેલ કતારના ચિલ્ન તરીકે ચિલ્નિત કરીએ છીએ, ત્યારે તે ચિલ્નિત કરવામાં આવે છે કારણ કે તે પહેલાથી જ મુલાકાત લેવાયેલા પાડોશી જેનો પાડોશી છે. તેથી, જ્યારે આપણે કહી શકીએ કે k એ આઈજે ચિલ્નિત થયેલ છે, તો અમે પિતૃ શબ્દનો ઉપયોગ કરીશું. તેથી, આપણે કહીશું કે k નું માતાપિતા j છે, જ્યારે માતાપિતા લિંક્સને અનુસરે છે; આપણે k થી મૂળ સ્ત્રોત વર્ટેક્સ તરફ પાછળના પાથને ફરીથી બનાવી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:38)

તેથી, આ એપ કોડ કરવું ખૂબ જ સરળ છે, તેથી આપણને ફક્ત એક લીટી ઉમેરવી પડશે, તેથી આપણે બે લાઈન કરીશું. તેથી, શરૂઆતમાં આપણે કહીએ છીએ કે પ્રત્યેક શિરોબિંદુ માટે માતાપિતા અનિશ્ચિત છે. તેથી, આપણે ઓછા 1 જેવા મૂલ્યને કહ્યું હતું, કારણ કે આપણે નોંધ્યું છે કે શિરોબિંદુઓનું નામ 1 થી n છે, અને પછી જ્યારે આપણે એક અક્ષાંશ ચિલ્નિત કરીએ છીએ, કારણ કે આપણે ધારક જે.કે. ની શોધ કરી રહ્યા છીએ, પિતૃ k એ j હશે. તો, આપણે કે = બરાબર j ની પિતૃને સોંપીએ છીએ, બાકીનો કોડ પહેલાની પહોળાઈની પ્રથમ શોધ સમાન છે, તમે શિરોલંબોને કેવી રીતે ચિલ્નિત કરીએ તેના ટ્રેક રાખવા માટે તમે તેને ફક્ત આ નવો એરે(Array) પેરેંટ ઉમેરો. તેથી, આપણે પાથની રચના કરી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:16)

તેથી, આપણે પાછળ જઈ શકીએ છીએ, તેથી કે જે જે દ્વારા ચિલ્નિત થયેલ છે, જે બીજા કોઈ વસ્તુ દ્વારા ચિલ્નિત થશે અને બીજું. તેથી, આ દ્વારા નીચેની કડીઓને અનુસરીને, આપણે કોઈપણ K થી આ પાથ વર્ટેક્સ પર પાછા જઈ શકીએ છીએ અને તેથી, આપણે પાથને ફરીથી ગોઠવી શકીએ છીએ. બીજી વસ્તુ જે આપણે પહોળાઈ પ્રથમ શોધમાં કરી શકીએ છીએ તે સ્તરનો ટ્રેક રાખે છે. તેથી, યાદ રાખો, અમે કહ્યું હતું કે પહોળાઈની પ્રથમ શોધ ખરેખર ગ્રાફિક સ્તરને સ્તર દ્વારા શોધે છે. સ્રોતથી એક પગથિયાં સુધી બધા શિરોબિંદુઓ તરફ જુએ છે, તો તે શિરોલંબ આમાંથી એક પગથિયું દૂર છે, તેથી સ્તર 2 પર, સ્ત્રોતથી બે પગથિયાં દૂર છે અને બીજું. તેથી, આ વાક્યની માત્ર મુલાકાત લેવાને બદલે, આપણે કહી શકીએ કે તે કયા સ્તરની મુલાકાત લે છે. તો, શરૂઆતમાં આપણે કહીએ છીએ કે બધા સ્તરો અનિશ્ચિત છે, અને પછી દર વખતે જ્યારે આપણે નવી શિરચ્છેદની મુલાકાત લેતા હોય છે, ત્યારે તે જે વર્ટીક્સની મુલાકાત લે છે તેના કરતા 1 સ્તર વધારે છે. તો, છેવટે આપણે કહીએ કે, જો લેવલ જે કોઈ સંખ્યા પૃષ્ઠને અસાઈન કરે છે, તો તે મૂળ શ્વેત રંગથી ટી સ્ટેપ્સ દૂર હોવું આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:15)

તેથી, આને પિતૃ કોડ સાથે સરળતાથી કોડમાં ઉમેરી શકાય છે. તેથી, છેલ્લા સમય, અમે આ સોંપણી માતાપિતા માટે ઉમેરી. હવે, આપણે લેવલ માટે એસાઈનમેન્ટ પણ ઉમેરીએ છીએ, તેથી આપણે શરૂઆતમાં કહીએ છીએ કે પ્રત્યેક શિરોબિંદુનું સ્તર અનિશ્ચિત છે. તેથી, તે ઓછા 1 જેવા અવિવેકી મૂલ્ય છે. જ્યારે આપણે કહીએ છીએ કે શરુઆતની શરુઆતનો સ્તર 0 છે, અને હવે જ્યારે પણ આપણે કોઈ નવી શ્રેણીની મુલાકાત લઈએ છીએ, જો તે સ્તર છે, જો તે મુલાકાત લેવામાં આવે છે, તો તે હોવું જોઈએ સ્તર. તેથી, આ સ્તર ઓછા 1 છે, તેથી તે સ્તર 1 ઓછા છે, તેનો અર્થ એ થાય કે મુલાકાત નથી, તેથી સ્તર ઓછા 1 નો અર્થ નથી. અગાઉની વસ્તુની જેમ જ, મુલાકાત લીધેલ K ની જેમ જ 0 છે

અને જો આમ હોય તો આપણે કે જે સ્તરની મુલાકાત લીધી હતી તેના કરતા સ્તર એક કરતાં વધારે હોવું જોઈએ અને અલબત્ત, અમે કહ્યું હતું કે માતાપિતા પહેલા હતા.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 18:17)
તેથી, આપણે આ બધા કામો આપણા અગાઉના ઉદાહરણમાં જોઈ શકીએ છીએ, સામાન્ય રીતે આપણે 1 થી પ્રારંભ કરીએ છીએ અને આપણે ચિલ્ડિનત કરીએ છીએ કે તે 0 જેટલું સ્તર છે અને તેમાં જોડી નથી. હવે, જ્યારે આપણે 2, 3 અને 4 ઉમેરીશું ત્યારે તે બધા સ્તર 1 પર હશે અને તે બધાને તે 1 હશે, કારણ કે આપણી પાસે 1 થી અન્વેષણ છે. પછી, પહેલાની જેમ, આપણી સૂચિમાં નવું કંઈપણ ઉમેરેલું નથી. , પૃષ્ઠ સૂચિમાં કંઈપણ ઉમેરે નથી. હવે, 4 5 અને પૂર્ણાંક ઉમેરશે, અત્યાર સુધી આ 2, સ્તર 2 થશે અને માતાપિતા 4 હશે, જે સૂચવે છે કે મેં મારી શોધમાં 4 માંથી 5 અને 8 ખરીદ્યા છે. પછી, 5 થી, મને 6 અને 7 મળશે. તેથી, હવે સ્તર 3 પર છે, કારણ કે 5 પોતે સ્તર 2 પર હતું, તેથી, 6 અને 7 સ્તર 3 પર છે અને તેમના માતાપિતા 5 છે, 8 માંથી, હું જઈશ 9, 9 એ લેવલ 3 પર પણ છે, કારણ કે 8 પહેલાથી સ્તર 2 પર હતું, તેથી 9 લેવલ 3 છે અને તે 8 મા છે. અને અંતે, આપણે શોધીશું કે 10 સ્તર 4 પર છે અને તે માતાપિતા 9 છે તેથી, હવે, અમે જોડિયા શોધી શકો છો. તેથી, આપણે કહી શકીએ કે, મને 10 થી મળ્યું, હું 9 માંથી આવ્યો, તેથી 9 8 માંથી આવ્યા, તેથી 8, 4 માંથી 4, 4 થી આવ્યા. તેથી, પાથ ખરેખર 10 થી 9 થી 8 થી 4 સુધી 1 છે. તમે ખરેખર જોઈ શકો છો, અમારી પાસે 10 થી 9 થી 8 થી 4 સુધીનો પાથ છે. તેથી, તમે પાથ માહિતી અને તે જ સમયે અંતર માહિતીને ફરીથી બનાવી શકો છો, જ્યારે તમે પહોળાઈ પ્રથમ શોધની ગણતરી કરી રહ્યા છો.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 19:57)
તેથી, પહોળાઈ પ્રથમ શોધ જો આપણે આ સ્તરની પૂર્તિ ઉમેરતા હોઈએ, તો તે ખરેખર દરેક નોડને કિનારીઓની સંખ્યાના સંદર્ભમાં સૌથી ટૂંકા માર્ગ આપે છે. હવે, સામાન્ય રીતે, આપણે જોશું કે, જો તમારી પાસે સમાન કિંમત નથી, તો અમારી પાસે ધાર પર વિવિધ ખર્ચ છે, પછી સૌથી ટૂંકા રસ્તાને ધારની સંખ્યા માટે ફક્ત ટૂંકાતમ શબ્દની જરૂર નથી. આપણે દરેક પાથ સાથે સંકળાયેલ ખર્ચ ઉમેરવો પડશે જેને ભારાંકવાળા ગ્રાફ કહેવામાં આવે છે. પરંતુ, એક વજનવાળા ગ્રાફમાં જે ગ્રાફ છે તે અમારી પાસે હમણાં જ સરળ ધાર છે. બ્રેડથ ફર્સ્ટ સર્ચ(બીએફએસ(BFS)) (Breadth First Search (BFS)) પહેલી શોધમાં સ્રોત વર્ટેક્સના કિનારીઓની સંખ્યાના સંદર્ભમાં ટૂંકા પાથની ગણતરી કરવાના ઉમેરવામાં લાભ છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 03
મોડ્યુલ - 04
લેક્ચર - 21
ડેપ્થ ફ્સ્ર્ટ સર્ચ(ડીએફ્એસ(DFS)) (Depth first search (DFS))

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:09)

તેથી, આપણે જોયું છે કે બ્રેઅડથ ફ્સ્ર્ટ સર્ચ (breadth first search)નો ઉપયોગ કરીને કનેક્ટિવિટી કેવી રીતે અન્વેષણ કરવી. તેથી, ચાલો હવે બીજી વ્યૂહરચના જુઓ જે સામાન્ય રીતે ડેપ્થ ફ્સ્ર્ટ સર્ચ (Depth first search) તરીકે ઓળખાય છે. તેથી, સ્તર દ્વારા બધા શિરોબિંદુઓનું સ્તર શોધવાની જગ્યાએ ડેપ્થ ફ્સ્ર્ટ સર્ચ (Depth first search). દર વખતે જ્યારે આપણે નવી કડી શોધીએ છીએ, ત્યારે આપણે તરત જ અન્વેષણ કરીએ છીએ કે તે પગલાં છે. તેથી, અમે હવે વર્ટીક્સ 1 શરૂ કરીએ છીએ અને પહેલાથી મારા પાડોશી જેની મુલાકાત લઈએ છીએ જે હજુ સુધી અન્વેષણ નથી કરતું. જ્યારે અમે સમજૂતીને સ્થગિત કરીએ છીએ અને તેના બદલે j ને શા માટે અન્વેષણ કરીએ છીએ, અને તમે આગળ વધતા નહીં ત્યાં સુધી તમે આ કરવાનું ચાલુ રાખશો. તેથી, જ્યારે તમે પાછા ફરવાનું સંશોધન કરવા માટે કશું નવું ન હોય ત્યારે અટવાઈ જાય છે, કારણ કે તમે પહેલાથી જ કેટલાક શિરોબિંદુ છોડી દીધી છે, કારણ કે સસ્પેન્ડ કરવામાં આવી છે. તેથી, તમે પાછા નજીકના નિલંબિત શિરચ્છેદ પર પાછા જાઓ છો, જેમ કે હજુ પણ એક અજાણ્યા પાડોશી છે, અને પછી તમે તે શિરચ્છેદની આગળના અજાણ્યા પાડોશીને અન્વેષણ કરો. તેથી, રાહ જોવી તે વિશે વિચારીએ છે કે જે બધા શિરોબિંદુ બાકી છે જે હવે સ્થગિત છે તે હવે સ્ટેક(stack)માં છે. તેથી, જ્યારે તમે ગ્રાફ્માં ઊંડા અને ઊંડાણમાં જાઓ છો ત્યારે સ્ટેક(stack) સુધી બિલ્ડ કરો, અને પછી જ્યારે પણ તમે મૃત સમાપ્તિ પર સ્ટેક(stack) મેળવો ત્યારે બેક અપ લો અને તમે સ્ટેક(stack)માં વસ્તુઓને પહેલા પ્રક્રિયા કરો.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:08)
તો, પહેલા ચાલો આપણે આપણા આલેખ પર હાથ દ્વારા આ અલ્ગોરિધમનો અમલ કરીએ, તમે જુઓ સ્યુડો કોડ લખો તે પહેલાં તે કેવી રીતે કાર્ય કરે છે. તેથી, આ વખતે આપણે 4 થી શરૂ કરી રહ્યા છીએ. તેથી, પ્રથમ પગલું એ મુલાકાત તરીકે 4 ચિલ્ડિનત કરવું છે. તેથી, અમે મુલાકાત લીધેલ તરીકે ચિલ્ડ્ન 4 ને તારાંકિત કરીએ છીએ અને અમે ઓળખીએ છીએ કે તેની પાસે પાડોશીઓ 1, 3, 5, અને 6 પહેલા 3 પડોશીઓ છે. તેથી, અમે તેમને એક ઑર્ડર આપીએ છીએ જે 1 થી શરૂ થાય છે; 1 ની મુલાકાત લીધી નથી. તેથી, અમે તેને મુલાકાત તરીકે ચિલ્ડિનત કરીએ છીએ. અને અમે સ્ટેક(stack) પર 4 મૂક્યા, અને 4 નું અમલીકરણ સસ્પેન્ડ કર્યું, કારણ કે હવે અમે પગલું 1 માં અન્વેષણ કરવા જઈ રહ્યા છીએ. તેથી, હવે આપણે 1 માં જઈએ, તો આપણે જોયું કે 1 માં આમાંના 2 પડોશીઓ 2, 3, અને 4 છે. 2, જો સ્ટેક(stack) પર 2 હોઈ શકે, કારણ કે તે 2 ન હોઈ શકે, તો ટિપ્પણી 2 ની મુલાકાત લીધી. તેથી, તમે પહેલા મુલાકાત લીધી ન હતી, અમે સ્ટેક(stack) પર 1 મૂકીએ છીએ, કારણ કે હવે અમે સસ્પેન્ડ કરી રહ્યા છીએ. તેથી, હવે આપણે 2 ના પાડોશીઓ તરફ નજર કરીએ છીએ. તેથી, 2 ની પડોશીઓ 1 અને 3 છે, 1 ની મુલાકાત લીધી છે , 3 નથી. તેથી, હવે આપણે 3 ની મુલાકાત લીધી અને નિલંબિત તરીકે ચિલ્ડિનત કરીશું. હવે જ્યારે આપણે 3 ની સંખ્યામાં આવીશું, ત્યારે તેમાં 2 પડોશીઓ 1 અને 2 હશે, પરંતુ તે બંને પહેલેથી જ મુલાકાત લીધેલ છે. તેથી, આમ બનવા માટે કંઈ નથી. તેથી, આપણે જવું જોઈએ અને આપણે પાછલા શિર્ષક પર પાછા જવું જોઈએ જે આપણે 2 એટલે કે સસ્પેન્ડ કર્યું છે, અને અન્વેષણ કરવું તે પડોશીઓ છે. 2 ની પાછળ જવા માટે, તેથી અમે પહેલેથી જ જાહેર કર્યું હતું કે 1 કરવું ન હતું, અમે પહેલાથી પ્રક્રિયા કરી લીધી છે. તેથી, આપણે 4 તરફ્ જોવું પડશે, પરંતુ 4 પહેલાથી જ મુલાકાત લીધેલ છે. તેથી, મેં 2 સાથે કર્યું છે. તેથી, હવે આપણે પાછા જવું જોઈએ અને સ્ટેક(stack)માં આગળના ભાગમાં સંગ્રહ કરવો પડશે જે સ્ટેક(stack)માં હતો જે 1

નંબરનો હતો. હવે આપણે 1 પાછા જઈએ છીએ, અને આપણે શોધી કાઢીએ છીએ કે 1 થી આપણે પહેલાથી જ મુલાકાત લીધી હતી 2, 3 કે જેનો આપણે 1 ની મુલાકાત લીધી નથી, તે પહેલાથી કરવામાં આવી છે તે માટે ચિલ્લિનત થઈ ગઈ છે. અને 4 તે પણ છે જ્યાં આપણે 1 થી આવ્યા. તેથી, આ થઈ ગયું. તેથી, 1 પણ નકામું છે. તેથી, હવે આપણે પાછા જઈએ અને 4 થી શરુઆતમાં પહેલા શૃંખલા પર પાછા ફરીશું, અને બીજું કંઈક કરવાનું છે તે જુઓ. તેથી, 4 થી પાછા આવીને, આપણે શોધી કાઢીએ છીએ કે, તેની પાસે પહેલા પડોશીઓની શોધ કરતાં વધુ પાડોશીઓ છે. તેથી, હવે અમે ચાલુ રાખીએ છીએ કે ત્યાં છે
((સમયનોસંદર્ભ લો: 03:18))
5, 6, અને 8. તેથી, અમે 5 પસંદ કરીએ છીએ અને જોવા મળતા નથી, અમે 5 ચિલ્લિનત કરીએ છીએ
((સમયનોસંદર્ભ લો: 03:27))
ફરીથી સસ્પેન્ડ કરો. તેથી, 5 થી, હવે આપણી પાસે 4, 6 અને 7 છે. તેથી, હવે આપણે 6 પસંદ કરીશું અને સસ્પેંડ કરીશું. 6 થી, આપણી પાસે 7, 5, 7 અને 8 હશે; 5 એ પહેલાથી થઈ ગયું છે, પરંતુ 7 થઈ ગયું નથી. તેથી, આપણે 7 પસંદ કરીશું અને સસ્પેંડ કરીશું. હવે 7 થી, અમને લાગે છે કે તેની પાસે પડોશીઓ 5 અને 6 છે, જે બંને પહેલાથી ચિલ્લિનત છે. તેથી, 7 છે
((સમયનોસંદર્ભ લો: 03:56)).
તો, આપણે પાછા આવીશું અને 6 ને ફરીથી જોશું, અને જે કંઈપણ કરવા જવાનું છે તે જોવાનું છે, ત્યાં 8 છે. તેથી, હવે આપણે 8 ઉમેરીશું, અને ફરી સસ્પેન્ડ કરીશું 8. હવે આપણે 8 ની વયે છીએ અને હવે આપણે પૂછો કે 8 માંથી શું શોધી શકાય છે, 8 4 માંથી પહેલેથી જ થઈ ગયું છે - 4, 6, અને 9 છે. તો, 4 પહેલાથી જ થઈ ગયું છે, 6 પહેલાથી થઈ ગયું છે, પરંતુ 9 ની જરૂર છે. તો, આપણે 9 ઉમેરીશું અને 8 ને સ્થગિત કરીશું, પછી આપણે 9 તરફ જઈશું. તો, 9 માંથી આપણે શોધી શકીએ કે તે એક નવો પાડોશી 10 છે. તો, આપણે 10 ઉમેરીશું પછી આપણે કરી શકીએ, કારણ કે 10 ને કરવાનું કંઈ નથી. તેથી, અમે પાછા જાઓ અને 9 પ્રક્રિયા કરો; 9 માં બોલવાની કોઈ નવી વાત નથી, કારણ કે 9 અમે બીજા પડોશીઓ 6 અને 8 પહેલાથી જ થઈ ગયા છે, તમે 8 ની પાછળ પાછા જાઓ છો, 8 ની પાસે કહેવું વધુ કંઈ નથી, 6 થી 6 પાછા જશે, 6 કહેવું વધુ કંઈ નથી 5, 5 માં કંઈક કહેવું વધુ નથી, 4 થી પાછા જશે, હવે જ્યારે આપણે આ સંપૂર્ણ સમાપ્તિ શરૂ કરીશું ત્યારે તમને 4 થી 5 મળશે. તો, હવે આપણે જોઈએ છીએ કે 4 થી 6 પહેલાથી થઈ ગયું છે, તે 2 થી 6 ની બરાબર છે. 4 થી 8 પહેલાથી થઈ ગયું છે. તેથી, આ બીજું કંઈ આપણે 4 થી કરી શકતા નથી અને તેથી આ સમાપ્તિ થઈ ગઈ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:00)
તેથી, આપણે સ્ટેક(stack)થી સ્પષ્ટપણે જે કર્યું તે વધુ ચપળતાપૂર્વક કરી શકાય છે, અમે ફક્ત પુનરાવર્તન કર્યું. તેથી, જ્યારે પણ આપણે નવી વર્ટેક્સ જેની મુલાકાત લઈએ છીએ, ત્યારે આપણે જેએફના ડીએફએસ(DFS)ને વર્તમાન કડીના સ્થગિત ડીએફએસ(DFS)ને બોલાવીએ છીએ. તેથી, અમે આ સ્ટેક(stack)ને સ્પષ્ટ રીતે જાળવી રાખ્યા નથી જે અમે સિમ્યુલેટ કરી રહ્યા છીએ, કારણ કે તે પુનરાવર્તિત ફંક્શન કૉલ્સ દ્વારા નિશ્ચિત રૂપે જાળવી રાખશે.
(સ્લાઈડસમયનો સંદર્ભ લો: 05:27)
તેથી, અમલમાં મૂકવા માટે ડીએફએસ(DFS) એ ખૂબ જ સરળ અલ્ગોરિધમનો છે. તેથી, શરૂઆતમાં આપણે કહ્યું કે પ્રત્યેક શિર્ષકની મુલાકાત લેવા માટે 0 છે, અને યાદ રાખો કે આપણે ડીએફએસ(DFS) ની પહોળાઈ પહેલી શોધમાં છીએ જ્યાં આપણે ક્યાંથી આવ્યા તે ટ્રેક રાખવા માંગીએ છીએ, તેથી આપણે કહીશું કે બધું ત્યાં અચોક્કસ છે. હવે આપણે આરંભ કરીએ છીએ કે આપણે શરૂઆતના વાક્યના ડીએફએસ(DFS) ને કૉલ કરીએ છીએ. તેથી, જ્યારે આપણે ડીએફએસ(DFS)ને શંકુચૂંકોમાંથી બોલાવીએ છીએ, ત્યારે અમે તેને ચિલ્લિનત કર્યું છે તે માર્ક કરીએ છીએ. તેથી, અમે તેને ચિલ્લિનત કરવા માટે આ પ્રથમ વસ્તુની મુલાકાત લઈએ છીએ. અને હવે દરેક અન્ય શિરચ્છેદ માટે કે જે તે સામાન્ય વિચાર સાથે જોડાયેલ છે, આપણે તપાસ કરીએ છીએ કે જે વાક્ય આપણે પહેલેથી જ મુલાકાત લીધી છે કે નહીં; જો તે મુલાકાત લેતી નથી, તો આપણે તેની મુલાકાત લેવા માંગીએ છીએ. તેથી, આપણે કેવી રીતે મુલાકાત લઈશું, અમે તેના માતાપિતાને ચિલ્લિનત કરીશું. અને પછી અમે આ ડીએફએસ(DFS) ને સ્થગિત કરીએ છીએ અને તે ડીએફએસ(DFS) ને કૉલ કરીએ છીએ. તેથી, આ આપણે સ્ટેક(stack) સાથે સ્પષ્ટપણે કરી રહ્યા છીએ, પરંતુ તે પુનરાવર્તન કરવું વધુ સરળ છે. તેથી,

ડીએફએસ(DFS) એ ખૂબ, ખૂબ જ સરળ પુનરાવર્તિત એલ્ગોરિધમ છે, હું દરેક અનપેક્ષિત પાડોશીને જુએ છે અને તે પુનરાવર્તિત રીતે દરેક અનપેક્ષિત પર ડીએફએસ(DFS) છે

(સ્લાઈડસમયનો સંદર્ભ લો: 06:30)

તો, ઊંડાણની પ્રથમ જટિલતા શું છે? સારુ, દરેક શિરચ્છેદ બરાબર એકવાર ચિલ્લિત અને શોધવામાં આવે છે. તેથી, આપણે જે નું ડીએફએસ(DFS) કરીએ છીએ, એકવાર દરેક જે માટે. તેથી, આ ઓર્ડર n છે

((સમયનોસંદર્ભ: 06:42)),

વાસ્તવમાં બરાબર n કોલ્સ છે જો બધું સળંગ પહોંચે છે. હવે જ્યારે આપણે j ની ડીએફએસ(DFS)ને ચોક્કસ j માટે બોલાવીએ છીએ, ત્યારે આપણે j ના બધા પાડોશીઓની તપાસ કરવાની જરૂર છે. આપણે જોયું કે જો આપણે અડજાનસી (adjacency) મેટ્રિક્સ ધરાવતા હોઈએ; તેનો અર્થ છે કે, આપણે પંક્તિ j તરફ જોવું જોઈએ, અને આપણને આ પંક્તિમાં દરેક એન્ટ્રી જોવાની જરૂર છે. તેથી, આ ક્રમમાં એન સમય લે છે. તેથી, અમારી પાસે ઓર્ડર n કોલ્સ છે અને દરેક કૉલ ઓર્ડર n વખત લે છે. તો, આપણી પાસે ઓર્ડર n ચોરસ સમય છે. બીજી તરફ, જો તમે ઉપયોગ કરો છો અને અડજાનસી (adjacency)ને સૂચિ છો, તો જ્યારે આપણે j ના પાડોશીઓ તરફ નજર કરીએ છીએ, ત્યારે અમારે ફક્ત જોડાયેલા ચોક્કસ શિરોબિંદુઓ પર જ જોવું જોઈએ અને આપણે પહેલા કહ્યું છે કે જો આપણે દરેક ધારની ગણતરી કરીશું તો બે વાર; એક વાર હું j થી અને એકવાર j થી i થી. તેથી, કોલ્સની કુલ સંખ્યા, પડોશીઓને સ્કેન કરવા માટેના પગલાંઓની કુલ સંખ્યા અમે ધારની સંખ્યાનો ક્રમ બનીશું. તેથી, એકંદર સમય રેખીય એમ પ્લસ n હશે જેમ કે ડેપ્થ ફર્સ્ટ સર્ચ (Depth first search). તેથી, જો આપણે અડજાનસી (adjacency)ને સૂચિનો ઉપયોગ કરીએ તો ઈનપુટના કદમાં પ્રથમ ઊંડાઈ બંને શોધ અને ડેપ્થ ફર્સ્ટ સર્ચ (Depth first search) રેખીય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:45)

તેથી, ઊંડાણમાં પ્રથમ શોધ અને ડેપ્થ ફર્સ્ટ સર્ચ (Depth first search) વચ્ચેનો એક મોટો તફાવત તે છે કે જે પાથ પ્રથમ શોધ શોધમાં પહોળાયેલા હોય તે ટૂંકા રસ્તાઓ નથી. તો, જો આપણી પાસે એક ત્રિકોણ હોય તે પહેલાં આપણે ગ્રાફ જોઈશું તો આપણી પાસે 1, 2 અને 3 હશે; પછી 1, 2, 3 ના પાથને શોધવા માટે કઈ ઊંડાણ પ્રથમ શોધ કરશે જે 2 થી પસાર થાય છે, જો આપણે આ નાનો પાડોશી લઈએ તો તે અન્વેષણ કરવાનો સમય છે. તેથી, જ્યારે ખરેખર 1 ભાગમાં આવે છે, ત્યારે મને તે મળશે 3 પહેલાથી જ મુલાકાત લે છે

((સમયનોસંદર્ભ: 08.14))

તેથી, એવું લાગે છે કે ઊંડાણમાં પહેલી શોધ કંઈક ઉપયોગી ન હોઈ શકે, પરંતુ વાસ્તવમાં આ શોધખોળનો રસ્તો આપણને ઘણી બધી માહિતી આપે છે. તેથી ગ્રાફ વિશે ઘણી ઉપયોગી લાક્ષણિકતાઓ વાસ્તવમાં રેકોર્ડીંગ કરી શકાય છે જેમાં ઓર્ડર રેકોર્ડ કરીને ડીએફએસ(DFS) છે જે શિરોબિંદુ છે. તેથી, આ માટે આપણે ડીએફએસ(DFS) દલીલ કરીએ છીએ, આપણે કંઈક નંબરિંગ કહીએ છીએ. તેથી, અમે એક કાઉન્ટર જાળવી રાખીએ છીએ જે દર વખતે જ્યારે આપણે કાંકરા દાખલ કરીએ છીએ ત્યારે ડીએફએસ(DFS) પ્રારંભ થાય છે અને જ્યારે તે કૂદશે. તેથી, આપણે દરેક વર્ટીક્સ સાથે ગ્રાફ 2 મૂલ્યોમાં જોડાયેલા છીએ; જ્યારે મેં પહેલી વખત જેએફના ડીએફએસ(DFS) કર્યું ત્યારે કાઉન્ટરનું મૂલ્ય, અને જેએફની ડીએફએસ(DFS) ચલાવતી વખતે કાઉન્ટરનું મૂલ્ય.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:00)

તો, આ રીતે આપણે આપણા એલ્ગોરિધમમાં આ કરીશું. તેથી, અમે આ કાઉન્ટરને 0 થી પ્રારંભ કરીને પ્રારંભ કરીએ છીએ, હવે જ્યારે પણ હું ડીએફએસ(DFS) નો ઉપયોગ કરું છું, ત્યારે હું જે કરું છું તે પ્રથમ વસ્તુ એ છે કે હું કાઉન્ટર વેલ્યુ અસાઈન કરું છું જે કંઈક 3 ની સંખ્યાને બોલાવે છે. તેથી, વાસ્તવમાં મેં શરૂ કરેલા ડીએફએસ(DFS) પહેલાં કાઉન્ટરની સંખ્યા છે, અને પછી હું ગણતરીમાં વધારો કરું છું. તેવી જ રીતે જ્યારે હું બહાર નીકળવા જઈ રહ્યો છુ, હું પોસ્ટની સમાન સંખ્યા પોસ્ટ કરું છું, અને ફરીથી હું ગણતરીમાં વધારો કરું છું. તેથી, આ તે છે ... તેથી, વચ્ચેના ઘણાં બધા યાદગાર કૉલ

(સમયનોઉલ્લેખ કરો: 09:33)).

તેથી, આ મુદ્દો એ જ નથી રહ્યો કે હું વધતી જતી સંખ્યા સાથે ગણું છું

((સમયનોસંદર્ભ: 09:38))
વચ્ચે થઈ રહ્યું છે, અને તે આ ક્રમમાં આ ક્રમમાં છે કે જેમાં આ 2 નંબરો પૂર્વ છે હું અને પોસ્ટ કરું છું, જો તમે શિર્ષકોમાં છો તો હું ખરેખર ઘણી બધી માહિતી મેળવી શકું છું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:48)
તેથી, ચાલો આપણે આ ઉદાહરણ જોઈએ જે આપણે પહેલા કર્યું હતું. તેથી, આપણે પહેલા 4 થી શરુઆત કરીએ તેવું વિચારીએ છીએ. તેથી, આપણે કહીશું કે તેનો પૂર્વ નંબર 0 છે, કારણ કે ત્યાં શરૂ થશે અને પછી આપણે વધારો કરીશું અને પછી આપણે 1 માં જઈશું. તો, જ્યારે આપણે 1 દાખલ કરીએ, તો સંખ્યા ગણાય 1 છે. તેથી, તેની પૂર્વ સંખ્યા 1 છે, અને પછી આપણે 2 પર જઈએ, તેનો પૂર્વ નંબર 2 છે; પછી આપણે 3 જઈએ, તે પૂર્વ સંખ્યા 3 છે, અને પછી તરત જ 3 જો, કારણ કે અમારી પાસે બહાર જવા માટે કોઈ નવા પડોશીઓ નથી. તેથી, તે પોસ્ટ નંબર હવે 4 છે. તેથી, હું પહેલાની સંખ્યા અને નીચે પોસ્ટ નંબર પર લખું છું. જ્યારે હું 2 થી પાછો આવું છું અને મને લાગે છે કે આ કરવાનું બીજું કંઈ નથી, તો તે બેમાંથી નીકળી જાય છે. તેથી, તેનું પોસ્ટ નંબર 5 થાય. તો, ધ્યાન રાખો કે અહીં, હું પગલું 2 પર દાખલ છું અને ડાબી બાજુ 5 ની વચ્ચે મેં 3 અને 4 ક્યાંક કર્યું છે. તેવી જ રીતે હું 1 થી પાછા આવીશ, અને હું
((સમયનોસંદર્ભ: 10:36))
તેમાં કહેવા માટે નવું કંઈ નથી. તેથી, હવે હું પગલું 6 પર 1 છોડીશ, પછી હું 4 માં પાછો આવીશ, પણ હું 4 થી સમાપ્ત થતો નથી, કારણ કે હું 5 કરી શકું છું. તેથી, હું પગલું 7 પર 5 દાખલ કરું છું, 5 થી હું 6 માં પગલું 6 તરીકે દાખલ કરું છું, 6 થી હું પગલું 9 પર 7 દાખલ કરું છું, હવે 7 થી હું બીજા ક્યાંય જઈ શકતો નથી, તેથી હું 7 ની કૂદકો લઉં છું અને હું 6 મા પાછું આવે છે, જ્યારે 6 થી હું આગળ જોઉં છું અને મને 8 મળે છે. તેથી, 8 દાખલ કરો. તેથી, હું હતો તે સમયે 10. તેથી, હવે હું 8 થી 8 પગથિયું દાખલ કરું છું, 8 થી હું 12 મા ક્રમાંકમાં 9 દાખલ કરું છું, 9 થી 9 મા ક્રમાંક 13 માં પગલું 10 દાખલ કરું છું, જ્યારે હું 10 મા ક્રમાંક કરું છું અને હું 9 થી 9 છોડી દઉં છું, તો હું 8 થી 16 છોડી દઉં છું, 7 7 , હું 18 થી 5 છોડી દઈશ, અને હું 4 ની વયે પાછો આવીશ અને અંતે હું સમાપ્ત થઈશ, આભાર. તેથી, મારી પાસે દરેક સાથે, દરેક નોડ હવે 3 મૂલ્યોનાં આ 2 મૂલ્યો અને 4 પગલાં છે. તેથી, હું પૂર્વ અને પોસ્ટ વેલ્યૂ કરીસ, આ પૂર્વ અને પોસ્ટ મૂલ્ય ખૂબ મદદરૂપ થઈ શકે છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 11:46)
તેથી, ગ્રાફ્સ ચક્ર છે કે નહીં તે વિચારીને આપણે શોધી શકીએ છીએ. તેથી, આ એક ચક્ર છે. તેથી, ગ્રાફમાં આ જેવું લૂપ છે કે નહીં તે આપણે શોધી શકીએ છીએ કે જોવા માટે જવું છે કે નહીં. તેથી, આ વિશિષ્ટ અક્ષાંશ છે, કારણ કે જો હું ગ્રાફમાંથી આ 4 ને દૂર કરું છું, તે ગ્રાફમાં જે પહેલા બધું જોડાયેલું છે તે બધું સુધી પહોંચવા માટે હવે જોડાયેલું નથી. તેથી, હવે 1 2 3 થી આમાં મેળવવા માટે આનો કોઈ રસ્તો નથી
((સમયનોસંદર્ભ: 12:06)).
તેથી, તેઓ આવા કટ શિર્ષકો છે. તેથી, આ વિવિધ વસ્તુઓ છે જેનો ઉપયોગ આ ડીએફએસ(DFS) નંબરોની મદદથી કરી શકાય છે, આપણે આમાંના કેટલાક પછીના વ્યાખ્યાનમાં ગણતરીઓમાંથી જોશું, પરંતુ આ ડીએફએસ(DFS) ને વાસ્તવમાં વધુ ઉપયોગી સંશોધન વ્યૂહરચના બનાવે છે, બીએફએસ (BFS)પર આપણને ટૂંકા પાથ આપે છે, પરંતુ બીજી બાજુ, ડીએફએસ(DFS) માહિતીની સંપત્તિ તરીકે આપે છે જે આ પુનરાવર્તન
((સંદર્ભઆપો: 12:31)
માં ગુપ્ત રીતે છુપાવેલી છે)
સંશોધન જે ગ્રાફના વિવિધ માળખાકીય ગુણધર્મોને વધુ અથવા ઓછા મુક્ત કરવા માટે વિસ્ફોટ કરશે. ડીએફએસ(DFS) કરી રહ્યા છીએ, આપણે તેના વિશે ઘણી બધી રસપ્રદ વસ્તુઓ શોધી શકીએ છીએ.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક -03
મૉડ્યૂલ - 05
લેક્ચર - 22
બીએફ્એસ (BFS) અને ડીએફ્એસ (DFS) ની એપ્લિકેશનો

અમે બ્રેડથ ફ્રસ્ટ સર્ચ (Breadth First Search) અને ડેપ્થ ફ્રસ્ટ સર્ચ (Depth first search) નો ઉપયોગ કેવી રીતે કરવો તે જોયું છે, સ્રોતથી લઈને એક તરફ્નો માર્ગ છે કે નહીં તે શોધવાની પ્રથમ શોધ. લક્ષ્ય વર્ટેક્સ. પરંતુ આ બે કાર્યવાહીમાં એક વધુ ઘણું કરી શકે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:10)
તેથી, યાદ રાખો કે ગ્રાફ્ એ શિરોબિંદુનો સમૂહ છે અને કિનારોનો સમૂહ છે જે શિરોબિંદુઓ વચ્ચે જોડાણ ધરાવે છે, અને આ કદાચ નિર્દેશિત અથવા નિર્દેશિત હોઈ શકે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:20)
હવે, જ્યારે આપણે પહોળાઈ પ્રથમ શોધ કરીએ છીએ, ત્યારે આપણે એક શિખરથી શરૂ થતા લેવલ એક્સ્પ્લોરેશન્સનું સ્તર કરીએ છીએ, જ્યારે આપણે નવી કર્વેક્સ પર દર વખતે ઊંડાણપૂર્વક પ્રથમ શોધ કરીએ છીએ, ત્યારે અમે સંશોધનને સ્વિચ કરીએ છીએ તે શિરચ્છેદ અને જ્યારે પણ આપણે કોઈ અંતિમ અંત સુધી પહોંચીએ છીએ ત્યારે અમે બૅકઅપ લઈએ છીએ. અને ઊંડાણની લાક્ષણિકતાઓમાંની એક પહેલી શોધ કરે છે કે અમે આ ક્રમિક પ્રક્રિયામાં બહાર નીકળો શિરોલંબમાં દાખલ થતા ક્રમને ટ્રૅક રાખી શકીએ છીએ. તેથી, ચાલો હવે જોઈએ કે નીચેનાં ગ્રાફ્ના આ માળખા વિશે વધુ જાણવા માટે આપણે કેવી રીતે બીએફ્એસ (BFS) અને ડીએફ્એસ (DFS) નો ઉપયોગ કરી શકીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:48)
તેથી, દિશાવિહીન ગ્રાફ્નો એક મૂળભૂત ગુણધર્મ વણજોડાયેલ છે કે કેમ, શું આપણે દરેક શિર્ષકમાંથી દરેક બીજા ભાગમાં જઈ શકીએ છીએ. તેથી, તમે આ બે ચિત્રોમાં જોઈ શકો છો કે ડાબી બાજુનો ગ્રાફ્ જોડાયેલ છે, કારણ કે તમે પ્રત્યેક શિરચ્છેદમાંથી દરેક બીજા ભાગમાં જઈ શકો છો. બીજી તરફ્, જમણે બાજુ પર કેટલાક શિરોબિંદુઓ અન્ય શિરોબિંદુઓથી પહોંચી શકતા નથી. ઉદાહરણ તરીકે, કોઈ 2 થી 7 અથવા 6 થી બીજા ક્યાંય જઈ શકતું નથી. હવે, જ્યારે આપણી પાસે અણધાર્યા ગ્રાફ્ છે જે વણજોડાયેલ થઈ ગયું છે, ત્યારે આપણે જોડાયેલા ઘટકો શું છે તે શોધવા માટે પણ રસ ધરાવો છો .... તેથી, આપણે તે શિરોબિંદુઓને એકત્રિત કરવા માટે જૂથ બનાવવું છે જે એકબીજા સાથે એક એકમમાં જોડાયેલા છે અને શોધી કાઢે છે. આમાંના કયા એકમો છે અને તેમાં કેટલી એકમો છે, જેમાં શિરોલંબ એક જ યુનિટના છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 01:32)
તેથી, પ્રથમ લક્ષ્ય પર જોડાયેલ ઘટકોને ઓળખવા માટે બીએફ્એસ (BFS) અથવા ડીએફ્એસ (DFS) નો ઉપયોગ કરવો છે. તેથી, આ કરવું ખૂબ સરળ છે જે નોડ લેબલ 1 અથવા કોઈપણ અન્ય નોડથી પ્રારંભ થાય છે. હવે, અમે આ નોડમાંથી બીએફ્એસ(BFS) અથવા ડીએફ્એસ(DFS) ચલાવીએ છીએ અને આ પ્રક્રિયામાં આપણે મુલાકાત લીધેલ સંખ્યાબંધ નોડ્સને ચિલ્નિત કરીશું, જો આ બિંદુએ કોઈ શિરોલંબ હોય તો પ્રથમ નોડથી શરૂ થતા બીએફ્એસ (BFS) અથવા ડીએફ્એસ (DFS) દ્વારા મુલાકાત લેવામાં આવતી નથી, તેનો અર્થ એ કે તેઓ તે જ કનેક્ટ થયેલા ઘટકથી સંબંધિત નથી. તેથી, બતાવવાનું સરળ છે કે જે ચિલ્નિત થયેલ ચિલ્ન છે તે જોડાયેલ ઘટકની બરાબર છે. તેથી, જોડાયેલ ઘટક, જેમાં શરૂઆતનો નોડ છે, તેથી હવે આપણે પાછા જઈએ છીએ અને અમે સૂચિમાં પ્રથમ નોડને જોઈએ છીએ જે ચિલ્નિત થયું નથી અને તે નોડમાંથી બીએફ્એસ (BFS) અથવા ડીએફ્એસ (DFS) ને ખૂબ પ્રારંભ કરે છે. તેથી, આપણે એક નવો જોડાયેલ ઘટક મેળવીશું, જેમાં તે શિરોબિંદુઓ હશે જે પ્રથમ નોડમાંથી પહોંચી શકાય તેવું છે જે ચિલ્નિત અને જોડાયેલ છે. હવે, ત્યાં નોડ્સ છે જે અવલોકનિત છે, અમે તેમાંના એકમાંથી ફરીથી પ્રારંભ કરીએ છીએ અને આગળ વધીએ છીએ.

તેથી, આપણે હવે શું કરી શકીએ તે છે, આપણે દરેક ડીએફએસ (DFS) ને અલગ નંબરો સાથે લેબલ કરી શકીએ છીએ, તેથી અંતે આપણે દરેક ભાગ સાથે સંકળાયેલી હોઈ શકીએ છીએ, તે ઘટકની સંખ્યા જેને આપણે શોધ્યું છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:51)
તેથી, ચાલો આપણે આનો એક ઉદાહરણ જોઈએ, તેથી ધારો કે આપણી પાસે આ ગ્રાફ છે, ઉદાહરણ તરીકે ઉદાહરણ તરીકે, આપણી બીએફએસ (BFS) માં અતિરિક્ત વેરીએબલ અથવા ઘટકોની સંખ્યા કરવા માટે ડીએફએસ (DFS) કોલ કોમ્પ્યુટથી શરૂ થાય છે. તેથી, અમે શરૂઆતમાં કોમ્પ્યુટ 1 કહ્યું અને કદાચ આપણે ડીએફએસ અથવા બીએફએસ નો નોડ 1 થી શરૂ કરીએ. તેથી, આ પ્રક્રિયામાં આપણે 1, 2, 5, 9 અને 10 ની મુલાકાત લઈશું અને આ બધા માટે આપણે જે જે સમાન ભાગનું ઘટક કહીશું . હવે, આ બિંદુએ આપણે સમજીશું કે બધા નોડ્સની મુલાકાત લેવામાં આવી છે, ફક્ત 12 નોડ્સમાંથી માત્ર 5 જ મુલાકાત લેવામાં આવી છે. તેથી, આપણે નાનાં નોડ પર જઈએ જે 3 ની મુલાકાત લેતી નથી અને ફરીથી શરૂ થાય છે, પરંતુ આપણે ફરીથી શરૂ કરીએ તે પહેલાં આપણે 2 ની ગણતરી કરીએ છીએ. હવે, આપણે બીએફએસ અથવા ડીએફએસ શરૂ કરી શકીએ નોડ 3 અને આપણે જે પણ પહોંચી શકીએ તે બધુંની મુલાકાત લઈએ, અને આમાં પ્રક્રિયા અમે ઓળખીશું કે આ 6 નોડ્સ ઘટક 2 માં છે, આ સમયે નોડ 6 હજુ પણ ચિલ્નિત થયેલ નથી. તેથી, અમે બી.એફ.એસ. અથવા ડીએફએસના ત્રીજા રાઉન્ડને ફરી શરૂ કરીએ છીએ જેમાં કોમ્પ 3 જણાવે છે જે ત્રીજા ઘટકને ઓળખે છે. હવે, ત્યાં વધુ અવલોકન નોડ્સ નથી, તેથી અમે બીએફએસ (BFS) અને ડીએફએસ (DFS) ના આ પુનરાવર્તિત એપ્લિકેશનના પરિણામ રૂપે બંધ કરી શકીએ છીએ અને ત્યાંથી, અમે તમામ ઘટકો ઓળખી કાઢ્યા છે અને તેમને ક્લસ્ટર કરી છે. તેથી, તે બધા ઘટકો સમાન ઘટકમાં અથવા સમાન ઘટક નંબર સાથે સંકળાયેલા છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 04:11)
અને ગ્રાફની અન્ય રસપ્રદ માળખાકીય મિલકત તે ચક્ર ધરાવે છે કે નહીં તે છે. તેથી, એક વિશ્લેષક ગ્રાફ એ એક ડાબું ગ્રાફ છે, જેમાં તમે કોઈપણ નોડ પર પ્રારંભ કરી શકતા નથી અને કિનારીઓનું અનુક્રમણિકા અનુસરી શકો છો અને નોડ પર પાછા આવી શકો છો અને
(જમણીબાજુ: 04:29))
ચક્ર સાથે ગ્રાફ. તેથી, ઉદાહરણ તરીકે, 5, 9 અને 10 ચક્ર સ્વરૂપો છે, ત્યાં બીજા ચક્ર પણ છે, ઉદાહરણ તરીકે, ઘણા ચક્ર છે, 3, 4, 7, 8 નું એક સંપૂર્ણ ચક્ર છે, જેમાં એકમોમાં 3 સાથે નાના ચક્ર પણ છે, 4, 8 અને 3, 7, 8 અને 7, 8, 11 એ એક ચક્ર પણ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:46)
તેથી, આપણે જ્યારે બીએફએસ ચલાવતા હોઈએ ત્યારે આપણે તે કરી શકીએ છીએ તે એક એજ છે જેનો ઉપયોગ એ વર્તુળોનો ટ્રેક રાખવો છે જે વાસ્તવમાં મુલાકાત લીધેલ શિરોલંબોને ચિલ્નિત કરવા માટે વપરાય છે. હવે, જો તમે ડાઈસ પર એક જેવા વિશ્લેષણાત્મક ગ્રાફ છો, તો તમે ચકાસી શકો છો કે ગ્રાફમાંના દરેક નોડનો વાસ્તવમાં બીએફએસ (BFS) શોધના ભાગ રૂપે ઉપયોગ કરવામાં આવશે. બીજી બાજુ, જો તમે ગ્રાફ પર બીએફએસ ચલાવો છો જે ચક્ર ધરાવે છે, તો તમે જોશો કે કેટલાક ધારનો ઉપયોગ કરવામાં આવતો નથી, કારણ કે જ્યારે તમે તે કિનારીઓનું અન્વેષણ કરવાનો પ્રયાસ કરો છો, ત્યારે તમને લાગે છે કે લક્ષ્ય વર્ટેક્સ પહેલાથી જ મુલાકાત લીધેલ છે. દાખલા તરીકે, 10 થી પડોશીમાં પહેલાથી 10 ની મુલાકાત લીધી છે જ્યારે અમે 9 અન્વેષણ કરવાનું શરૂ કરીએ છીએ ત્યારે અમે ધાર 9, 10 નો ઉપયોગ કરતા નથી. તેવી જ રીતે, અમે એજ 4, 8 નો ઉપયોગ કરતા નથી, કારણ કે તે પહેલાથી પડોશીઓના સમૂહ તરીકે મુલાકાત લેવાય છે 3 ના, યાદ રાખો, પહોળાઈની પ્રથમ શોધમાં આપણે 3 ની શોધ કરીએ છીએ તે બધા એક પગલું પડોશીઓ છે. તેથી, આપણે સીધી 4, 8 અને 7 ની મુલાકાત લીધી શક્યા નથી, તેથી જ્યારે આપણે 4 સુધી આવીએ છીએ ત્યારે આપણે એજ 4, 8 નો ઉપયોગ કરવાની જરૂર નથી, જ્યારે આપણે 7 પર આવીએ છીએ ત્યારે અમને 7, 8 અને તેથી વધુ ઉપયોગ કરવાની જરૂર નથી. તેથી, આ કિનારીઓ છે જે છોડી દીધી છે. હવે, તે જોવાનું સરળ છે કે જો તમારી પાસે n શિરોલંબવાળા ગ્રાફ છે અને તે જોડાયેલ છે અને તેમાં ચક્ર નથી, તો તે બરાબર n ઓછા 1 ધાર હશે, આ પ્રકારના ગ્રાફને વૃક્ષ કહેવાય છે. તેથી, વૃક્ષોની ઘણી વ્યાખ્યાઓ છે, પરંતુ વૃક્ષો મૂળભૂત રીતે જોડાયેલા વિશ્લેષણાત્મક ગ્રાફ છે, જોડાયેલા માધ્યમો તમે દરેક જગ્યાએથી દરેક જગ્યાએ જઈ શકો છો, વિશ્લેષણાત્મક અર્થ એ છે કે ત્યાં કોઈ આંટીઓ નથી અને n

શિરોલંબ પર કોઈપણ જોડાયેલ વિશ્લેષણાત્મક ગ્રાફ બરાબર n ઓછા 1 ધાર હશે . તેથી, કોઈપણ ગ્રાફમાં જો આપણે બીએફએસ (BFS) નું અન્વેષણ કરીએ છીએ, તો જે બાફ્સ વાસ્તવમાં વાપરે છે તે એક વૃક્ષ બનાવશે અને આને બીએફએસ (BFS) વૃક્ષ કહેવામાં આવશે. હવે, બાકીના કિનારીઓ વિશે શું થશે તે આને વૃક્ષોના કિનારીઓ કહેવામાં આવશે, જે તમે ખૂબ જ સરળતાથી તપાસ કરી શકો છો તે છે કે કોઈપણ નોન ટ્રી એજ એ ચક્ર રચવા માટે પહેલાથી જ વૃક્ષની ધાર સાથે જોડાઈ જશે. બીજા શબ્દોમાં કહીએ તો, જ્યારે આપણે બીએફએસ ચલાવીએ છીએ ત્યારે જો આપણે શોધી શકીએ કે કેટલાક શિરોલંબ અથવા કેટલાક કિનારીઓ છે જેનો ઉપયોગ ન થાય છે, અન્ય શબ્દોમાં કોઈ પણ વૃક્ષો નથી, તો આ ગ્રાફમાં ચોક્કસપણે એક ચક્ર હશે. તેથી, એક ચક્ર હોવાનું બીએફએસ કરતી વખતે નોન ટ્રી એજને શોધવા સમાન છે, તો નો ટ્રી એજ ધાર એ છે કે આપણે અલ્પવિરામ શોધી શકીશું અને હું શોધી શકું છું કે j એ પહેલેથી જ મુલાકાત લેવાય છે. તેથી, અમે મારી પાસે નથી જતા, હું બીએફએસ (BFS) માં અલ્પવિરામનો ઉપયોગ કરતો નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:11)
તેથી, ચાલો આપણે એજ વસ્તુ કરીએ, પણ ડીએફએસ (DFS) અને ચાલો પ્રી અને પોસ્ટ નંબર્સની ગણતરી કરીએ. તો, આપણે કેટલાક પ્રેક્ટિસ સેટ મેળવીશું, તેથી આપણે ડીએફએસને વર્ટેક્સ 1 પર સૉર્ટ કર્યું છે અને આપણે ચિલ્નિત કરીએ છીએ કે તે કાઉન્ટર છે, તેથી આપણે 0 વડે વેરટેક્સ 0, વર્ટેક્સ 1 દાખલ કરીએ, તો આ 0 ની પહેલી સંખ્યા છે. પ્રથમ પાડોશી 1 થી 2 નું અન્વેષણ કરો, તેથી તેની પાસે પહેલું નંબર 1 છે, પરંતુ 2 ને વધુ સફળતાઓ નથી. તેથી, અમે 2 થી બહાર નીકળીએ છીએ, તેથી તેની પાસે પોસ્ટ નંબર 2 છે, તેથી અમે દર વખતે દાખલ કરીએ ત્યારે યાદ રાખો કે અમે કાઉન્ટરને વધારીએ છીએ, દર વખતે જ્યારે આપણે બહાર નીકળીએ છીએ ત્યારે અમે કાઉન્ટરને વધારીએ છીએ. તો, આપણે એક ઈન્ક્રીમેન્ટ એક્ઝિટ વર્ટેક્સ 2 દાખલ કરીએ છીએ એક પગથિયાં પર પાછા આવે છે અને તે અન્વેષણ કરે છે તે આગામી પાડોશી છે જે 5 છે જે આપણે સમય 3 પર દાખલ કરીએ છીએ, પછી આપણે 4 વડે વેરટેક્સ 9 ખસેડીએ છીએ ત્યારબાદ 9 વડે આપણે 10 વડે જઈએ. , હવે 10 અજાણ્યા કોઈ પાડોશીઓ નથી. તેથી, આપણે 6 ના સમયે 10 થી બહાર નીકળીએ છીએ, 9 માં આગળ કોઈ પડોશીઓ નથી, તેથી 7 સમયે 7 ની બહાર નીકળો 5 ની વયે પાછા આવે છે, ત્યારબાદ 5 પાસે કોઈ પડોશીઓ નથી, તેથી અમે 5 થી બહાર નીકળીએ છીએ અને અંતે આપણે 1 થી બહાર નીકળીએ છીએ. તેથી, પગલું 9 આપણે પ્રોસેસિંગ 1 પૂર્ણ કરી દીધી છે. હવે, આપણે પ્રથમ વર્ટીક્સ પર જઈએ છીએ જે નામ નથી એટલે કે 3, આપણે ત્યાંથી નવા ડીએફએસ (DFS) માં ફરીથી પ્રારંભ કરીએ છીએ. તેથી, આપણે 3 માંથી 10 સમયે 10 દાખલ કરીશું, 11 સમયે 4 તરફ જશે, હવે 4 આપણે 12 સમયે 8 પર જઈએ છીએ, 8 થી આપણે તેના સૌથી નાના પાડોશીઓ તરફ નજર કરીએ છીએ જે 7 ની છે અને 7 થી 13 આપણે 11 વખત જઈએ છીએ. 14, હવે 11 અને 7 બંનેની શોધ કરવા માટે કોઈ નવા પડોશીઓ નથી. તેથી, આપણે 11 થી 7 ની બહાર નીકળી શકીએ છીએ, હવે પડોશીઓ અન્વેષણ નથી કરતા, તેથી અમે 8 દૃશ્યથી બહાર નીકળી જવું જોઈએ 12. તેથી, આપણે 12 મા ક્રમે 12 દાખલ કરીએ, પછી 12 થી બહાર નીકળીશું અને હવે 8 સમાપ્ત થઈ ગયું છે, તેથી અમે 8 થી બહાર નીકળીશું. હવે, અમે પાછા 4, 4; દેખીતી રીતે, તેમાં કોઈ અન્ય શિરોલંબ નથી, તેથી આપણે 3 પાછા આવીએ અને છેવટે, આપણે 21 ના સમયે 3 થી બહાર નીકળીશું, આ 0.6 પર હજુ પણ ચિલ્નિત નથી. તેથી, અમે 6 થી નવા ડીએફએસ શરૂ કરીએ છીએ, તેથી અમે 6 સમયે 6 દાખલ કરીએ છીએ, પ્રત્યેક 6 પાસે કોઈ પડોશીઓ નથી, તેથી અમે 23 વાગ્યે બહાર નીકળીએ છીએ. તેથી, આ પ્રકારના બીએફએસ (BFS) વૃક્ષોની સંગ્રહ પેદા કરે છે. તેથી, જ્યારે આપણે વણજોડાયેલેડ ગ્રાફ પર ડીએફએસ (DFS) કરીએ છીએ, ત્યારે દરેક જોડાયેલ ઘટક ઝાડ પેદા કરશે. હવે, જો આપણે કિનારીઓ તરફ નજર કરીએ કે જે આપણે અન્વેષણ કર્યું ન હતું, તો આ ફરીથી વૃક્ષની બહારના કિનારીઓ બનાવશે. તેથી, આપણે તેને અલગ રંગ પર દોરી શકીએ છીએ, તેથી અમારી પાસે 5 અને 10 ની વચ્ચેની ધાર છે જેને અમે અન્વેષણ કરી શક્યા નથી, કારણ કે અમે 5 નું સંશોધન કર્યું છે કે કેમ 10 સીધીથી 9 અને તેથી આગળ. તેથી, ફરી એક વાર બીએફએસની જેમ, જો આપણે વૃક્ષો ન હોય તો ડીએફએસ સમાપ્ત થઈ જાય, પછી આપણી પાસે ચક્ર હોય છે. તેથી, નિર્દેશિત ગ્રાફ પર બીએફએસ અને ડીએફએસ બંને બિન-વૃક્ષ ધારની હાજરી દ્વારા એક ચક્ર જાહેર કરી શકે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:07)

તેથી, નિર્દેશિત ગ્રાફ્સ સાથેની આ સ્થિતિ થોડી વધુ પૂરક છે, તેથી ચાલો જોઈએ કે જ્યારે આપણે નિર્દેશિત ગ્રાફમાં ચક્ર હોય ત્યારે શું થાય છે. તેથી, એક નિર્દેશિત ગ્રાફમાં આપણને કિનારીઓ સાથેના તીરો, તીરોને અનુસરવાની જરૂર છે. તેથી, ઉદાહરણ તરીકે, 1, 3, 4, 1 જે ચક્ર છે, કારણ કે આપણે બદલાવ વગર દિશામાં જઈ શકીએ છીએ, જ્યાં 1, 6, 2, 1 એ કોઈ ચક્ર નથી, કારણ કે અને 2 થી 2 તરફ દિશા બદલવાની રીત 1 ને જે હું જઈ શકતો નથી. તેથી, ચાલો આપણે એક ડીએફએસ (DFS) કરીએ અને આ ગ્રાફમાં સપોર્ટ સાઈકલ્સને શું જણાવી શકીએ તે જુઓ. તેથી, આપણે હંમેશાં શ્વેત 1 સાથે પ્રારંભ કરીએ છીએ, તેથી 1 પાસે ફ્રી નંબર 0 છે, તે નાનો પાડોશી 2 છે, 2 ના નાના પાડોશીમાં 5 છે, 5 ના નાના પાડોશીમાં 6 છે, 6 ના નાના પાડોશીમાં 7 છે, હવે 7 માંથી કોઈ બહાર નીકળતી ધાર નથી. તેથી, આપણે 6 માંથી 6 નો ટ્રેક શોધી શકીએ છીએ જે આપણે 2 ની ઉપર જઈ શકીએ જે પહેલા જોઈ છે, તેથી આપણે 6 છીયે. 6. હવે, આપણે 5 ની વાણીએ છીએ, 5 હજુ પણ આઉટગોઈંગ એજ છે જે 8 છે, તેથી આપણે 8. 8. હવે, 8 થી આપણે કંઈ પણ કરી શકતા નથી, તેથી અમે 8 થી પાછા 5 પર પાછા ફરો, હવે 5 ની શોધ કરવા માટે કશું બાકી નથી. તેથી, આપણે 5 છોડીશું તેવી જ રીતે આપણે 2 ને છોડી દઈશું, આપણે સહકાર કરીશુંમને 1 થી પાછા, હવે 1 વાગ્યે આપણે આ ડાબે પાથ શોધી કાઢીએ છીએ. તેથી, હવે આપણે જોઈ શકીએ છીએ કે આપણે જોઈ શકીએ છીએ અને અન્ય ઘટાડો 3 થઈ જાય છે. તેથી, અમે 3 અન્વેષણ કરીએ છીએ, 3 એ 4 નું અન્વેષણ કરશે, પરંતુ 4 8 અથવા 1 ના જઈ શકે છે, કારણ કે 1 પહેલાથી જોયું છે અને સ્રોત 8, તેથી 4 બહાર નીકળી જશે, તેથી 3 બહાર નીકળી જશે અને પછી 1 બહાર નીકળી જશે. તેથી, આ એક જોડાયેલ ગ્રાફ હોવાનું જણાય છે, પરંતુ તેમાં ચક્ર છે, તેથી હવે જો આ કિનારીઓ પર પ્રથમ જોવું હોય, તો ખેંચાયેલી ધાર એ પહેલાની જેમ વૃક્ષની કિનારીઓ છે. હવે, જો તમે ધારની ધારણા કરો છો કે જે આલેખનો ભાગ નથી, તો તે 3 જૂથોમાં આવી છે. તેથી, પ્રથમ પ્રકારનો ધાર જે વૃક્ષનો ભાગ નથી તે આપણે આગળનો ભાગ કહીએ છીએ, તેથી આગળનો ધાર એ એક ધાર છે જે વૃક્ષની નીચે નોડમાં નીચે નોડ બનાવે છે. તો, ઉદાહરણ તરીકે, આપણી પાસે 1 થી 6 નો નોડ છે. તેથી, આ ધાર એક વૃક્ષ ધાર છે જે વૃક્ષની ધાર નથી, પરંતુ તે આગળનો ભાગ છે કારણ કે 1 ભાગમાં આશરે 6 હતી. તેવી જ રીતે નોડ 5 થી 7 થી, કારણ કે આપણે ખરેખર આ ગ્રાફ શોધી શકીએ છીએ 5, 6, 7, તેથી 5 થી 7 વૃક્ષ નથી. તેથી, આ આગળના ધાર છે, બીજી શ્રેણીની ધાર જે ગ્રાફમાં છે જે વૃક્ષમાં નથી, પાછળની ધાર છે, તે વૃક્ષ ઉપર જાય છે. તેથી, 6 માંથી એજ 2 ની પાછળ છે જેનો આપણે ઉપયોગ ન કર્યો, કારણ કે બે પહેલાથી જ અન્વેષણ કરવામાં આવી છે. તેવી જ રીતે 4 થી 1 સુધી 1 ત્યાં ધાર છે જે અમે અન્વેષણ કરી શક્યા નથી, કારણ કે તે પહેલાથી જ ત્યાં હતું. ત્યાં બીજી કિનારીઓ છે જે વૃક્ષમાં નથી, પરંતુ તે ગ્રાફમાં છે અને આ 6 થી 2 ની વચ્ચેની કિનારીઓ છે. તેથી, આ પાછળના ભાગથી આગળનો ભાગ છે અને 4 પાછલા ભાગથી પહેલાની વર્ટિક્સ કોલ છે. 1 થી, તેથી આને એજ ધાર કહેવામાં આવે છે. તેથી, ગ્રાફમાં પાછલો ધાર જે ડીએફએસ (DFS) ટ્રીમાં નીચલા શિરોબિંદુથી ઉચ્ચ શિખર સુધી જાય છે. અને છેવટે, ત્યાં અમુક ધાર છે જે આગળ પાછળ કોઈ આગળ નથી, પરંતુ માર્ગદર્શિકાઓ છે. તેથી, આ 8 થી 7 અને 4 થી 8 ની કિનારીઓ છે, તેથી આ ક્રોસ, તેથી 4 8 ની નીચે નથી અને 4 ની નીચે 8 છે, 7 8 થી નીચે નથી કારણ કે 5 અને તેથી નીચે બંને છે. તેથી, આ આપણે ક્રોસ ધારને બોલાવીએ છીએ, હવે દલીલ કરવાનું સરળ છે કે ક્રોસ એજ જમણીથી જમણી બાજુ જશે. બીજા શબ્દોમાં કહીએ તો, તે માત્ર ઉચ્ચ સંખ્યાથી નીચલા ક્રમાંક પર જ જાય છે, કારણ કે જો તમે હમણાં આના જેવા ધારને દોરવા માંગતા હો, તો તેનો અર્થ એ થશે, પરંતુ 2 થી 4 ની ધાર હતી. તેથી, આપણે બદલે ચોથા રુટની શોધ કરીશું વજન કરતાં પાછા જાઓ અને અન્વેષણ કરો. તેથી, આપણે ધાસની કિનારીઓ ન મેળવી શકીએ જે નીચલા નંબરોથી ઉચ્ચ ક્રમાંક સુધી જાય છે, તે ઉચ્ચ ક્રમાંકથી નીચલા નંબરો સુધી જવું જોઈએ. તેથી, હવે આપણી પાસે એક નથી, પરંતુ ત્રણ પ્રકારનાં નોન ટ્રી ધાર, નિર્દેશિત કેસની જેમ જ્યારે આપણે 3 ધાર અને નોન ટ્રી ધાર વચ્ચેનો સ્પષ્ટ ભેદ ધરાવતા હતા, અહીં આપણી પાસે 3 પ્રકારના નોન વૃક્ષ ધાર છે. હવે, આમાંના એકમાં ચક્ર સાથે અનુરૂપ છે, તેથી જો હું આ 1, 3 ધાર તરફ જોઉં, તો આ 1 6 ધાર. તેથી, આ 1 6 ધાર ખરેખર કોઈ ચક્ર બનાવતું નથી, કારણ કે આપણે પહેલાથી જોયું છે કે આ એક ચક્ર નથી. તેથી, તે ચક્ર પૂર્ણ કરવા માટે, તે નિર્દેશિત ચક્રને મળતા સેટ સહિતનો પાથ હોય તો તે આવશ્યક છે. હવે, એ જોવાનું સરળ છે કે આ એકમાત્ર પરિસ્થિતિ છે જ્યાં આની સાથે પાછળનો ભાગ હશે, કારણ કે પાછળનો ભાગ છે, આપણે જાણીએ છીએ કે ત્યાં 2 થી 6 સુધીનો માર્ગ આવે છે અને પછી આ ફોર્મ પાછળના ભાગને અનુસરે છે. એક નિર્દેશિત ચક્ર. તેવી જ રીતે, આપણે જાણીએ છીએ કે નીચેના પાછળના ભાગમાં 1 થી 4 સુધીનો માર્ગ આવે છે જેનો

આપણે સીધા જ અનુસરે છે. બીજી તરફ, જો આપણે તેના માટે અન્ય પ્રકારના કિનારીઓને જોવું જોઈએ તો તેનો અર્થ છે, તો અહીં આપણી પાસે આ રસ્તો છે જે અહીં બીજા પાથની સમાંતર છે. તેથી, એક સાથે મળીને આ બંને જુદા જુદા માર્ગો 1 થી 6 માં જઈ રહ્યા છે, પરંતુ ત્યાં કોઈ છુપાવી નથી, એ જ રીતે આપણે જોઈ શકીએ છીએ કે જો અમારી પાસે આ જેવા ક્રોસ ધાર હોય, તો આપણી પાસે અહીંથી કેટલાક પાથ આવી રહ્યા છે અને કેટલાક પાથ જઈ રહ્યા છે. ત્યાંથી. પરંતુ ફરીથી, આ 2 જુદા જુદા રસ્તાઓ 5 થી 7 સુધી પહોંચ્યા અને તેઓ ચક્ર પર આધારિત નથી. તેથી, થોડો વિશ્લેષણ કરીને કે ફક્ત પાછળના ભાગો ચક્ર બનાવે છે અને આ વાસ્તવમાં કંઈક છે જે તમે સાબિત કરી શકો છો કે અમે ઔપચારિક સાબિત નહીં કરીએ, પરંતુ હમણાં જ આપણે જે રીતે કર્યું તે દલીલ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:56)

પરંતુ, નિર્દેશિત ગ્રાફમાં એક ચક્ર હોય છે, જો ફક્ત ડીએફએસ બેક ધારનો પ્રભાવ કરે છે, તો હવે તે જણાવે છે કે આ પૂર્વ અને પોસ્ટ નંબરિંગ એ કિનારીઓના પ્રકારો સાથે વર્ગીકૃત કરવા માટે ખૂબ જ ઉપયોગી છે. ત્યાં ગ્રાફમાં. તેથી, બંને વૃક્ષ અને આગળનાં કિનારીઓ માટે, તમે જોશો કે જો તમે આ ક્રમાંકન

((સમયનોસંદર્ભ: 16:19)

પર પાછા જાઓ છો કે)

આ વસ્તુઓ એક ઈન્ટિગ્રલ બનાવે છે જે તમે 0 થી 15 સુધી વિચારી શકો છો 0 થી 15 સેટ કરો જે તેમને 11 થી 14 સુધી શોધે છે, તે પગલું 2 થી 9 સુધી 3 નું અન્વેષણ કરે છે, હું 5 અને તેથી વધુની તપાસ કરીશ. તેથી, જો તમે પ્રિ અને પોસ્ટ નંબર જુઓ છો જે મેં પહેલાની સંખ્યાને શોધવાનું શરૂ કર્યું છે અને મેં પોસ્ટ અને અન્ય જે કંઈ નીચે હતું તે અંગેની તપાસ કરવાનું સમાપ્ત કર્યું છે. તેથી, ફોરવર્ડ ધાર માટે આપણે આંતરિક અંતર્ગત કરતાં મોટી હોઈશું. કારણ કે, હું આ સમયગાળા દરમિયાન નીચે ગયો હતો તે પૂરું થાય તે પહેલાં હું પાછો આવ્યો. તેથી, 0 થી 15 થી 3 અલ્પવિરામ, આગળના ધારને શામેલ કરવા માટે. તેથી, આપણે ઈન્ટિગ્રલ 3, 6 0 ની અંદર છે, 15 આ પણ છે

((સંદર્ભસમય: 17:05))

વૃક્ષની ધાર માટે સાચું છે, કારણ કે વૃક્ષમાં ધાર પણ હું આગળ જઈ રહ્યો છું પછી હું આ દાખલ કર્યા પછી નીચે નોડ દાખલ કરું છું. તેથી, તે પછીથી શરૂ થવાનું બિંદુ છે અને તે સ્પષ્ટ રીતે નિર્દેશ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:16)

તેથી, બંને વૃક્ષની ધાર અને આગળના ભાગો માટે, જો હું તમારી પાસેથી વી પર જાઉં છું તો શરૂઆતના નોડ સાથેના અંતરાલમાં તમે બીજાને સમાવશો. પણ, આ એક જ અંદર બેઠા હશે, પ્રી વી પોસ્ટ વી તમારા પહેલા પોસ્ટની અંદર બેઠા હશે, તેથી મારી પાસે આ ચિત્ર હશે. તેથી, આ તમારા માટે આંતરિક પૂર્વ હશે અને આ અંતરાલ પૂર્વ છે. તેનાથી વિપરિત બરાબર વિરુદ્ધ કિનારીઓ માટે એક અલ્પવિરામ પ્રારંભિક વિરામ છે અને મોટા અંતરાલમાં જાય છે. તેથી, નાનો ઈન્ટિગ્રલ એજ સાથે પ્રારંભ બિંદુ હશે, પછી મોટો ઈન્ટિગ્રલ એ અંતિમ બિંદુ હશે. તેથી, જો હું મારા ડીએફએસ વૃક્ષમાં ધાર તરફ જોઉં છું અને જો હું વૃક્ષ તરફ જોઉં છું અને પોસ્ટ નંબરો એ અંતિમ બિંદુ સાથે સંકળાયેલ હોય તો હું આગળની ધાર અને પાછળની ધારના શબ્દો નક્કી કરી શકું છું અને અંતે, તે ક્રોસ ધાર માટે અંતરાલો નિષ્ક્રીય છે

((સમયનોસંદર્ભ લો: 18:02)).

તેથી, આપણે અહીં જોઈ શકીએ છીએ કે આપણે સો પર પહોંચતા પહેલા 4 vertex 7 ને પ્રોસેસ કરવાનું સમાપ્ત કરી દીધું છે, અંતરાલ 7, 8 અને 4, 5 વચ્ચે કોઈ આંતરછેદ નથી. તેવી જ રીતે આપણે 4 ની સંખ્યા પહેલા જ પ્રક્રિયા 8 સમાપ્ત કરી દીધી છે તેથી જ તે છે એક ક્રોસ ધાર, તેઓ વૃક્ષની વિવિધ શાખાઓ પર છે. તેથી, ત્યાં કોઈ આંતરછેદ 7, 8 અને 12, 13 નથી, તેથી કોઈ નિર્દેશિત ગ્રાફમાં એક ચક્ર હોય છે, જો ફક્ત ડીએફએસ બેક ધારને છતી કરે છે, અને આપણે ધારણા કરી શકીએ છીએ કે કિનારીઓ તરફ આગળ વધીને ધાર, પછાત ધાર અથવા ક્રોસ ધારો છે ધારના અંતિમ બિંદુઓની પૂર્વ અને પોસ્ટ ક્રમાંકન.

(સ્લાઈડસમયનો સંદર્ભ લો: 18:35)

હવે, ચક્રની ઓળખ કરવી મહત્વપૂર્ણ છે, કારણ કે અમારી પાસે ચક્ર નથી, કારણ કે અમારી પાસે ચક્રનો ખૂબ સરસ વર્ગ છે જેને કહેવાય છે કે સાયક્લિક ગ્રાફ્સ મોડેલિંગ ડિપેન્ડન્સીઝ(dependancies) માટે ઉપયોગી છે. દાખલા તરીકે, જો તમે

ઓફર કરેલા અભ્યાસક્રમોના સમૂહની સૂચિ સૂચિબદ્ધ કરવા માંગો છો અને તેમની પાસે પૂર્વ આવશ્યકતાઓ છે, તો પછી મોડેલ કરવા માટેનો કુદરતી માર્ગ આ દિશામાં નિર્દેશિત ગ્રાફ પર ઉપયોગ કરીને પૂર્વ આવશ્યકતાઓનું પ્રતિનિધિત્વ કરે છે, ઉદાહરણ તરીકે જો તેઓ બીજગણિતથી ધાર ધરાવે છે સંકેત સાથે ગણતરી કરવા માટે બીજગણિત એ કેલ્ક્યુલેશન માટે પૂર્વ આવશ્યક છે, તેમાં ચક્ર નહીં હોય. કારણ કે આપણી પાસે બે અભ્યાસક્રમો હોઈ શકતા નથી જે એકબીજા માટે પૂર્વ આવશ્યક છે; અન્યથા, અમે ઈનપુટ લેવા માટે ઈનપુટ નહીં કરીએ. તેથી, અમે પછીના ભાષણમાં ટૂંક સમયમાં નિર્દેશિત એસાયકલિક ગ્રાફ અથવા ડીએજીસ(DAGS)ને જોશું.

(સ્લાઈડસમયનો સંદર્ભ લો: 19:16)

નિર્દેશિત ગ્રાફમાં કનેક્ટિવિટી વિશે શું? તેથી નિર્દેશિત ગ્રાફમાં કનેક્ટિવિટી એ ગ્રાફ વચ્ચેની આ ધાર હોવાનું માત્ર એક પ્રશ્ન નથી, પરંતુ તેમને યોગ્ય દિશામાં રાખવાથી. તો, આપણે કહીએ છીએ કે બે ગાંઠો એકદમ જોડાયેલા છે, જો હું પાથ દ્વારા i થી j પર જઈ શકું છું અને હું j થી i

(પાછલાસમય: 19:38)

થી પાછા આવી શકું છું. તેથી, માત્ર કેટલાક તરફના કિનારીઓ માટે એક દિશા હોય તે માટે પૂરતું નથી કે હું i થી j પર જઈ શકું છું અને j થી i થી પાછું આવું છું તે કિસ્સામાં સખત જોડાયેલું છે. તેથી, તે તારણ આપે છે કે નિર્દેશિત ગ્રાફ હંમેશાં કનેક્ટ કરેલા ઘટકોમાં કઈ રીતે ડિમપોઝ થઈ શકે છે. સખત જોડાયેલા ગ્રાફ પાસે એવી પ્રોપર્ટી છે કે જે ઘટકમાં નોડ્સની દરેક જોડ એ ઘટકમાં દરેક નોડથી સખત રીતે જોડાયેલી હોય છે જે તમે ઘટકમાં દરેક અન્ય નોડ પર જઈ શકો છો અને પાછા આવી શકો છો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 20:04)

તેથી, ઉદાહરણ તરીકે, તે આ ગ્રાફને જોશે અને પછી સખત જોડાયેલા ભાગો છે, એક ચક્ર 1, 3, 4 છે, આપણે 1, 2, 3 થી 4 સુધી જઈ શકીએ છીએ અને પાછા આવીશું. તેથી,

((સમયનોસંદર્ભ: 20:14))

થી, આ ચક્ર આપણે કોઈપણ અન્ય નોડ પર પાછા આવી શકીએ છીએ. તેવી જ રીતે 2, 5 અને 6 સ્વરૂપો સખત જોડાયેલા ઘટક છે, 7 તેના પર એક મજબૂત કનેક્ટ ઘટક દર્શાવે છે. કારણ કે, આપણે ગમે ત્યાં જઈ શકતા નથી અને 8 પણ મજબૂત કનેક્ટ ઘટક છે, કારણ કે આપણે હજી સુધી છૂટીએ છીએ કે આપણે આ ગ્રાફિક માળખાથી પાછા આવી શકીએ નહીં. તેથી, આ ગ્રાફ 4 સખત જોડાયેલા ઘટકો છે, તો તે તપાસે છે કે પૂર્વ અને પોસ્ટ નંબરોનો ઉપયોગ કરીને ડીએફએસ નંબરિંગનો ઉપયોગ કરી શકાય છે, કમ્પ્યુટર્સને એકદમ ભવ્ય આલ્ગોરિધમ આપવામાં આવે છે અને દાસગુપ્તા પ્રોપર્ટી પેપાડિમિટ્રિઓઉ અને વાઝિરાની (Dasgupta property Papadimitriou and Vazirani) દ્વારા લખાયેલ પુસ્તક અને જો તમને રસ હોય તો તે પુસ્તકમાં તે જોઈ શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 20:52)

તેથી, તમે ગણતરી કરી શકો તેના કેટલાક નક્કર ઉદાહરણો જોયા છે, ત્યાં ઘણી અન્ય ગુણધર્મો છે કે જે તમે બીએફએસ (BFS) અને ડીએફએસ (DFS) નો ઉપયોગ કરીને ગણતરી કરી શકો છો. હમણાં પૂરતું, આ વસ્તુઓને એટીક્યુલેશન પોઈન્ટ્સ કહેવામાં આવે છે, જો તમે આલેખ જેવા દેખાય છે, જ્યાં મારી પાસે કેટલાક શ્વેત છે જે નિર્ણાયક છે, જો હું આ ગ્રાફને ઘટકોને વણજોડાયેલ કરવા માટે પણ કાઢી નાખું છું, તો હું બીએફએસ (BFS) નો ઉપયોગ કરીને આવા શિર્ષકો ઓળખી શકું છું. ડીએફએસ (DFS) અને ખાસ કરીને ડીએફએસ (DFS) નો ઉપયોગ કરીને. એ જ રીતે, જો મારી પાસે એવી સ્થિતિ હોય કે જ્યાં મારી પાસે એજ ધાર હોય, જ્યાં હું આ ધારને દૂર કરીશ, તો ગ્રાફ વણજોડાયેલ થઈ જશે, પછી હું ફરીથી ડીએફએસનો ઉપયોગ કરીને એજ ધારને ઓળખી શકું છું. હવે, આ મહત્વપૂર્ણ છે, કારણ કે જો તે કોઈ પ્રકારના સંચાર નેટવર્ક અથવા કેટલાક રૂટ નેટવર્કનું પ્રતિનિધિત્વ કરે છે, આમાં સાંકડું છે, તો આ નિર્ણાયક પોઈન્ટ છે, જો આ એક છૂટાછેડા છે અને અકસ્માત કોઈ ટ્રાફિક નથી અને ડાબી બાજુના ઘણા ભાગમાંથી જાય છે. અને જમણી બાજુના ઘણા ભાગો અથવા આ એક નેટવર્ક વાયર છે જે આ કેબલ કાપી જાય છે, પછી ઘટકોને કારણે નેટવર્ક વણજોડાયેલ થઈ જશે. તેથી, આ પ્રકારના ગુણધર્મો બીએફએસ (BFS) અને ડીએફએસ (DFS) દરમિયાન પણ ગણતરી કરી શકાય છે. તેથી, તે સમજવું મહત્વપૂર્ણ છે, જ્યાં બીએફએસ અને ડીએફએસ માત્ર તે શોધવા માટે જોડાયેલા નથી કે શામેલ છે કે નહીં, તમે

સારી માહિતી મેળવી શકો છો અને આ રેખીય ટાઈમ એલ્ગોરિધમ્સ છે અને આ તે બધા ઑપરેશન છે જે કરી શકાય છે બીએફએસ અને ડીએફએસ દરમિયાન. તેથી, તમે ખૂબ અસરકારક રીતે ગ્રાફમાં વિવિધ ગુણધર્મોની ગણતરી કરી શકો છો અને આ બે શોષણનો ઉપયોગ કરી શકો છો, આ વધુ કાર્યક્ષમ પ્રક્રિયાઓ ડિઝાઈન કરશે અથવા અન્ય વસ્તુઓની ઓળખ કરવાની જરૂર પડશે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 06
લેક્ચર - 23
ડાયરેક્ટેડ એસાયક્લિક ગ્રાફ્સ(ડીએજીએસ) (Directed Acyclic Graphs (DAGs))

હવે આપણે ડાયરેક્ટેડ એસાયક્લિક ગ્રાફ્સ(Directed Acyclic Graphs) અથવા ડીએજીએસ(DAGs) તરીકે ઓળખાતા ખૂબ જ રસપ્રદ અને મહત્વપૂર્ણ વર્ગના ગ્રાફ્ તરફ્ ધ્યાન આપીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:08)
તેથી, ગ્રાફ્ના આ વર્ગને પ્રેરિત કરવા માટે, ચાલો આપણે એવી સમસ્યાની તપાસ કરીએ જ્યાં આપણી પાસે કેટલાક અવરોધ સાથે કાર્ય કરવા માટે એક કાર્ય છે. ધારો કે, અમે એક વિદેશી સફ્ર પર જઈ રહ્યા છીએ, પછી, અમને પાસપોર્ટની જરૂર છે, અમને ટિકિટ ખરીદવાની જરૂર છે, આપણે વિઝાની જરૂર છે, આપણે કેટલાક મુસાફ્રી વીમા ખરીદવા માંગીએ છીએ, કદાચ આપણને કદાચ કેટલાક વિદેશી વિનિમયની પણ જરૂર છે. તમે અમારા યજમાનો માટે કેટલાક ભેટો ખરીદવા માંગો છો.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:36)
હવે, આ કાર્યો એકબીજા પર ચોક્કસ રીતે, એક પાસપોર્ટ વગર, તમે કોઈ ટિકિટ ખરીદી શકતા નથી, કોઈ મુસાફ્રી વીમો પણ ખરીદી શકતા નથી, એકબીજા પર આધારિત છે. વિઝા માટે, તમારે ટિકિટ અને વીમા બંને ઉપલબ્ધ થવાની જરૂર છે અને વિઝા વિના, બેંક તમને વિદેશી વિનિમય આપશે નહીં. અને અંતે, તમે તમારા યજમાનો માટે ભેટો ખરીદવા માંગતા નથી, સિવાય કે સફ્ર પુષ્ટિ થાય. તેથી, જ્યાં સુધી તમારી પાસે વિઝા સહિતની આ બધી બાબતો ન હોય ત્યાં સુધી, તમે ભેટમાં રોકાણ કરવા માંગતા નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:06)
તેથી, અમારો ધ્યેય એ છે કે આપણે આ છ ક્રિયાઓ, પાસપોર્ટ મેળવવી, ટિકિટ ખરીદવી, વીમા મેળવવી, વિઝા મેળવવી, વિદેશી વિનિમય ખરીદવી અને ભેટો ખરીદવી તે આ અવરોધો આપ્યા છે. અમારા યજમાનો. આપણે કયું અનુક્રમ કરવું જોઈએ, જેથી જ્યારે પણ આપણે કોઈ કાર્ય પર સંપર્ક કરવો હોય, ત્યારે કાર્ય માટે જરૂરી અવરોધ સંતોષાય છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:31)
તેથી, તમે અપેક્ષા રાખશો કે અમે આલેખનો ઉપયોગ કરીને મોડેલ કરીશું. આ ગ્રાફ્માં, શિરોબિંદુઓ કાર્યો હશે અને પછી તમારી પાસે ટી 1 થી ટી 2 તરફ્ સંકેત થશે, જો ટી 1 ટી 2 પહેલાં આવવું આવશ્યક છે, તો બીજા શબ્દોમાં ટી 2 ટી પર આધાર રાખે છે, તમે ટી 2 ન કરી શકો ત્યાં સુધી ટી 1 પૂર્ણ થઈ ગયું છે. તેથી, ટિકિટ ખરીદતા પહેલાં પાસપોર્ટ મેળવવું ઉદાહરણ તરીકે, તેથી ટી 1 પાસપોર્ટ મેળવવામાં આવે છે, ટી 2 ટિકિટ મેળવી શકે છે. એ જ રીતે, તમારે એક વિદેશી વિનિમય ખરીદતા પહેલાં વિઝા ખરીદવો જ જોઈએ. તેથી, વિદેશી વિનિમય ખરીદવા માટે વિઝા મેળવવાની ધારણા હશે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:13)
તેથી, જો આપણે લખેલી અવરોધોને જોયેલી હોય તો આ તે ગ્રાફ્ છે જે આપણને મળ્યું હતું, તેથી અમને સેટ્સ સાથેની મર્યાદાઓ હતી, અમને ટિકિટ ખરીદવા માટે પાસપોર્ટની જરૂર છે, અમને ખરીદવા માટે પાસપોર્ટની જરૂર છે વીમા મેળવવા માટે, અમને ટિકિટ અને વીમાની જરૂર છે. તેથી, વિઝા તરફ્ નિર્દેશ કરતી બે અવરોધ છે. પછી, તમારે વિદેશી વિનિમય ખરીદવા માટે વિઝાની જરૂર છે અને અંતે, તમે કહ્યું હતું કે ટ્રિપની પુષ્ટિ થાય છે અને આ તબક્કે આ તબક્કે જ્યારે આપણે આ તબક્કે કાર્ય કરીએ છીએ ત્યારે અમે એક ભેટ ખરીદીશું, તો આપણે ધારી શકીએ કે આ સફ્ર પુષ્ટિ થયેલ છે, કારણ કે પ્લેન પર જતા અટકાવવું કંઈ અવરોધિત નથી. તેથી, આ એક ગ્રાફ્ છે જેનો અમારો ધ્યેય છે અને હવે આ છ ઓપરેશન્સને અનુસરવાનો છે, આ રીતે આપણે જ્યારે પણ કાર્ય કરવા માંગીએ છીએ, તે જે પણ છે તેના પર આધાર રાખે છે. તેથી, અમે જોઈ શકીએ છીએ કે તમારે કંઈપણ કરવા માટે પાસપોર્ટની જરૂર છે, તેથી અમારે હંમેશાં પાસવર્ડ

મેળવવાની જરૂર છે. હવે, ત્યાં સુધી ટિકિટ ખરીદવાની અને વીમો ખરીદવાની વચ્ચે કોઈ નિર્ભરતા નથી, જે અત્યાર સુધીમાં અવરોધ છે. તેથી, પાસવર્ડ પછી તમે પ્રથમ ટિકિટ ખરીદી શકો છો અને પછી વીમા ખરીદી શકો છો અથવા તમે પ્રથમ વીમા ખરીદી શકો છો અને પછી ટિકિટ ખરીદી શકો છો. તેથી, ત્યાં જુદી જુદી ઓર્ડરિંગ શક્ય છે કે જે અવરોધનું ઉલ્લંઘન કરતી નથી, બીજી તરફ વિઝા માટે અમને બન્નેની જરૂર છે. તેથી, ટિકિટ અને વીમા બંને પછી વીઝા આવવું આવશ્યક છે, પરંતુ ફરીથી વિઝા પૂર્ણ કર્યા પછી, વિદેશી વિનિમય ખરીદવા અને ભેટો ખરીદવાની વચ્ચે કોઈ મર્યાદા નથી. તેથી, તમે વિદેશી વિનિમય પહેલાં ભેટ અથવા ભેટ પહેલાં વિદેશી વિનિમય કરી શકો છો, તેથી આ વિશિષ્ટ ઉદાહરણમાં ત્યાં રિકરિંગ ટિકિટ અને વીમાના સંભવિત રૂપે બે સંભવિત રસ્તાઓ છે અને ભેટો અને વિદેશીઓને ફરીથી ગોઠવવાના બે સંભવિત રસ્તાઓ છે. વિનિમય. તેથી, એકંદર ચાર અલગ અલગ અનુક્રમ છે જે આ અવરોધો સાથે સુસંગત છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:51)
તેથી, ગ્રાફનો આ વર્ગ એક મહત્વપૂર્ણ વર્ગ છે અને તેમાં બે મહત્વપૂર્ણ સુવિધાઓ છે, તે એક છે, તે નિર્દેશિત છે. કારણ કે, આ નિર્ભરતા એક કાર્યથી બીજા કાર્યમાં હોય છે, તે સમપ્રમાણતા આધારિત નથી અને ત્યાં કોઈ ચક્ર નથી. જુઓ, જો તમારી પાસે ચક્ર હોય તો તે કાર્યોનું જૂથ એકબીજા પર નિર્ભર રહેશે, તેથી પ્રારંભ કરવાનો કોઈ રસ્તો નથી, કારણ કે દરેક કાર્ય ચક્રમાં બીજું કંઈક પર આધારિત છે. તેથી, પ્રારંભ કરવા માટે તમારે ક્યાંક ચક્રને તોડી નાખવું પડશે, પરંતુ તમે તેને તોડી શકતા નથી, કારણ કે દરેક કાર્ય ચક્રમાં બીજું કંઈક પર આધારિત છે. તેથી, આ ગ્રાફમાં ધાર પર દિશાઓ હશે અને તે આ ગ્રાફમાં કોઈપણ ચક્ર હોઈ શકશે નહીં.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:28)
તેથી, અમે આલેખને નિર્દેશિત એસાયકલિક ગ્રાફ તરીકે ઓળખાવીએ છીએ, તેથી નિર્દેશિત એસાયકલિક ગ્રાફ એ માત્ર એક નિર્દેશિત ગ્રાફ છે, જેમાં કોઈ પણ શિરોબિંદુથી કોઈ દિશામાં કોઈ પાથ નથી. તેથી, જો મેં કોઈ વર્ટીક્સ વી શરૂ કર્યું હોય, તો તે કેસ ન હોવું જોઈએ કે હું એક જ દિશામાં નિર્દેશિત કિનારીઓની અનુક્રમણિકાને અનુસરી શકું અને કોઈક રીતે ડી પર પાછા આવી શકું. તેથી, આ ત્યાં હોવું જોઈએ નહીં, તેથી આ ચક્ર ન હોવું જોઈએ, અમે ડાયરેક્ટેડ એસાયક્લિક ગ્રાફ્સ(Directed Acyclic Graphs) નામનું ટૂંકું નામ ડીએજી તરીકે લખીએ છીએ. તેથી, ઘણી વાર સરળતા આપણે આ ગ્રાફને ડીએજીએસ(DAGs) તરીકે બોલાવીશું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:02)
તેથી, સમસ્યા જે આપણે આપણા ઉદાહરણમાં ચર્ચા કરી હતી તે છે કે આપણે કાર્યોનો સમૂહ આપ્યો છે અને આપણે અવરોધોને ધ્યાનમાં રાખીને ક્રમમાં તેને લખવા માંગીએ છીએ, અવરોધ કંઈ નથી પરંતુ , ધાર. તેથી, સામાન્ય રીતે આપણને શિરોબિંદુઓનો સમૂહ આપવામાં આવે છે, આ આપણા કાર્યો અમૂર્ત રીતે 1 થી n છે અને આપણે વાંચવું છે, આપણા 1 ને n ને આ રીતે લખીએ કે અવરોધનું આદર છે. આનો મતલબ એ છે કે, આપણે સંખ્યાઓની શ્રેણી લખીશું જે 1 થી n નો ક્રમચય છે. આ રીતે, જ્યારે પણ જેકની અવરોધ હોય ત્યારે ધાર જેકે રજૂ કરે છે, તે પછી આપણે જે ક્રમાંકન કરીએ છીએ તે j એ પહેલા જ આવવું આવશ્યક છે. તેથી, તે ન હોઈ શકે કે આપણે પહેલા અમારી મર્યાદા મુજબ k પહેલાં j કરવું પડશે, પરંતુ જે ક્રમમાં આપણે ઉત્પન્ન કર્યું તે k માં j પહેલા થાય છે. તેથી, અંતિમ અનુક્રમમાં શિર્ષકોના હુકમને ડી.એ.એ.જી. દ્વારા આપવામાં આવેલી અવરોધોનો આદર કરવો જ જોઈએ, તેથી વિવિધ કારણોસર આને ટોપોલોજિકલી(topologically) ડીએજીને સૉર્ટ કરવામાં આવે છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 06:02)
તેથી, પ્રથમ નિરીક્ષણ એ છે કે જો નિર્દેશિત ગ્રાફમાં એક ચક્ર હોય, તો તમે તેને સ્થાનિક સ્તરે ઑર્ડર કરી શકશો નહીં. કારણ કે, જો તેની પાસે ચક્ર હોય તો ઉદાહરણ તરીકે, જે અને કે જે વિચારે છે તે ચક્ર પર શિરોબિંદુ છે, તો તમારી પાસે j થી k થી પાથ અને k થી j નો પાથ હશે. હવે, તે જોવાનું સરળ છે કે ટોપોલોજિકલ ઓર્ડરિંગ અવરોધ એ પાથ સુધી વિસ્તૃત છે કે જો મારી પાસે ધાર પહેલા કિ પહેલાં j છે, તો હું જાણું છું કે j એ અંતિમ અનુક્રમમાં k પહેલા જ દેખાય છે, તે પણ j થી k નો પાથ ધરાવે છે, પછી j થી k થી નિર્ભરતા ક્રમ છે. તેથી, j એ પહેલા જ દેખાય છે. હવે, જો મારી પાસે ચક્ર હોય તો તે કહે છે કે j એ પહેલા આવવું આવશ્યક છે અને કે જે પહેલા જ જોઈએ. તેથી, આને તોડી નાખવાનો કોઈ રસ્તો નથી
((સમયનોસંદર્ભ: 06:45)),

તેથી આપણે આ પરિસ્થિતિ સાથે અંત લાવીશું જ્યાં આપણે આ કામના સમૂહને અવરોધના આદર માટે ઓર્ડર આપી શકતા નથી. તેથી, આ ગ્રાફમાં ચક્ર છે, તો તે સ્પષ્ટ છે કે ત્યાં કોઈ સ્થાનિક ક્રમમાં શક્ય નથી.
(સ્લાઈડસમયનો સંદર્ભ લો: 06:58)
તેથી, આપણે શું માનીએ છીએ; જો કે, ડીએજે માટે ત્યાં કોઈ ચક્ર નથી, ગ્રાફ વાસ્તવમાં એસાયકલિક છે ત્યારબાદ આપણે તેને ટોપોલોજિકલી(topologically) રીતે ઓર્ડર આપી શકીએ છીએ. તેથી, આ વ્યૂહરચના નીચે પ્રમાણે શિરોબિંદુઓને ક્રમમાં ગોઠવવાનું છે, તમે પહેલા બધા શિરોબિંદુઓની સૂચિ કરો જેની કોઈ નિર્ભરતા નથી. અમારા પહેલાના ઉદાહરણમાં, જે વર્ટેક્સની કોઈ નિર્ભરતા નથી તે પાસપોર્ટ મેળવવામાં આવી હતી, પાસપોર્ટ મેળવવા પહેલાં અમને કાંઈ કરવાની જરૂર નહોતી, તેથી અમે તે પહેલા કરી શકીએ છીએ. હવે, એકવાર અમે એકદમ ટોચ પર હોઈએ છીએ કે નિર્ભરતા તમે કોઈપણ શ્વેતતા જુઓ છો જે તે બધી જ નિર્ભરતા છે જે હવે સંતોષકારક છે અને પછી અમે તેને અંશાંકિત કરી શકીએ છીએ. તેથી, આપણે કોઈપણ ઈનકમિંગ કિનારીઓ સાથે વ્યવસ્થિત રીતે શિરોબિંદુને સૂચિબદ્ધ કરી શકીએ છીએ, પછી તે બધાને શિરોલંબિત કરે છે જેની આવનારી ધાર પહેલેથી જ સંખ્યા માટે જવાબદાર છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:45)
તેથી, આ વિચારને ઔપચારિક બનાવવા માટે આપણે કેટલીક પરિભાષા રજૂ કરીએ છીએ, તેથી યાદ રાખો કે અણધાર્યા ગ્રાફ માટે, આપણે વી ની ડીગ્રીનો ઉપયોગ વી સાથે જોડાયેલા શિરોબિંદુઓની સંખ્યાને સંદર્ભવા માટે કરીએ છીએ. તેથી, વી હતી શિરોબિંદુઓ આગળ ધાર દ્વારા જોડાયેલું છે, તો પછી આપણે કહીશું કે વી ની ડિગ્રી 4 છે. હવે, કારણ કે આપણી પાસે નિર્દેશિત ગ્રાફ છે, તેની પાસે કિનારીઓ પર દિશાઓ છે, આપણી પાસે કેટલીક ધાર છે જે આવી રહી છે અને અમુક ધાર જે બહાર આવી રહી છે. . તેથી, આપણે ડિગ્રીને ઈન્ડેગ્રી અને આઉટડેગ્રીમાં અલગ કરીએ છીએ. તેથી, ઈન્ડેગ્રી એ વીમાં નિર્દેશિત વીમાં નિર્દેશિત ધારની સંખ્યા છે, v ની આઉટડેગ્રી વીમાંથી નિર્દેશ કરતી ઘણી ધાર છે.
(સ્લાઈડસમયનો સંદર્ભ: 08:24)
તેથી, અમારો પ્રથમ દાવો એ છે કે દરેક ડીએજી પાસે છે ડિગ્રી 0 ની સાથે ઓછામાં ઓછા એક શિરોબિંદુ, દાખલા તરીકે, ડિગ્રી 0 માં એક શંકુદ્રષ્ટા છે જે કંઈ નિર્ભરતા ધરાવતું નથી, તે કંઈ પણ પર આધાર રાખે છે, આમાં જે કંઈક નિર્દેશ કરે છે તે કશું જ નથી. હવે, આપણે આનો સાબિતી કેવી રીતે આપીએ છીએ કે આપણે કોઈ પણ વર્ટીક્સ વી સાથે પ્રારંભ કરીએ છીએ જેમ કે 0 કરતા વધારે ડિગ્રી ધરાવે છે, કારણ કે તેનામાં તે કંઈક નિર્દેશ કરે છે, તો તેમાં અમુક ધાર આવવા જ જોઈએ, તો ચાલો આપણે બી 2 માં બોલાવીએ હવે, ધારો કે આ ડિગ્રી 0 માં નથી, તો તે તેના પર નિર્દેશ કરતી વસ્તુ પણ હોવી આવશ્યક છે. તેથી, પછી મને ત્રીજો ભાગ મળે છે, તેથી આ રીતે જો હું શોધી રહ્યો છુ કે શિરચ્છેદનો સામનો 0 ડિગ્રી કરતા વધુ છે, તો અંતે, મારે મારા ગ્રાફમાંના બધા શિરોબિંદુઓને ગણતરી કરવી પડશે. હવે, જો ત્યાં હજી પણ કોઈ કેસ નથી કે નવમા ભાગમાં nth vertex માં n શિરોબિંદુ હોય તો પણ 0 નો વધારો થતો નથી, તો તેમાં આવતી ધાર હોવી આવશ્યક છે, પરંતુ તે નવી કર્ટેકથી ન હોઈ શકે. તેથી, તે અસ્તિત્વમાંના શિરોબિંદુઓમાંથી એક તરફ નિર્દેશ કરે છે જે પહેલાંથી જોયેલી છે. તો, જો મારી પાસે સતત અનુક્રમ અથવા શિરોબિંદુ હોય તો જે પ્રત્યેક પાછલા એક તરફ ડિગ્રી સાથે 0 ની સમકક્ષ નથી, તે પછી હું ચક્ર સાથે અંત કરીશ, પરંતુ આ વિરોધાભાસ છે, કારણ કે આપણી પાસે જ્ઞાનકોશ છે ગ્રાફ. તેથી, કોઈપણ નિર્દેશિત એસાયકલિક ગ્રાફમાં, તે ડિગ્રી 0 સાથે ઓછામાં ઓછા એક શિખર હોવું આવશ્યક છે, જે કાર્યની શરૂઆત અથવા કાર્યની સંખ્યામાંથી વધુ નિર્ભરતા સાથે કાર્ય સાથે સંબંધિત છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 09:58)
તેથી, આ એલ્ગોરિધમનો વધુ વિસ્તૃત સંસ્કરણ છે જે તે અગાઉ વર્ણવેલો છે. તેથી, અમે એક પસંદ કરોઅક્ષાંશ 0 સાથે, આપણે કહીએ છીએ કે આવા શંકુદ્રુપ કોઈ નિર્ભરતા નથી, હવે આપણે ગણતરી કરી છે કારણ કે હવે તે ગણતરી માટે ઉપલબ્ધ છે અને પછી અમે ગ્રાફમાંથી કાઢી નાખ્યું છે. તેથી, જ્યારે આપણે ગ્રાફમાંથી ડિગ્રી 0 માં કોઈ શૂન્યતા કાઢી નાખીએ છીએ ત્યારે ધારે છે કે જ્યારે અમારી પાસે આ જેવી ડીએજી છે. તેથી, ધારો કે આપણે આને પસંદ કરીએ છીએ અને અમે કાઢી નાખીએ છીએ, પછી સ્પષ્ટપણે ડીએજી શું છે. કારણ કે, તે હજી પણ નિર્દેશિત છે અને અમે એક ચક્ર રજૂ કર્યું નથી, તેથી તે પહેલેથી જ જ્ઞાનકોશી છે અને ધારને કાઢી નાખીને આપણે ચક્ર રજૂ કરી શકતા નથી. તેથી, સ્પષ્ટપણે તે એક ડીએજી છે, તેથી આપણે સમાન માપદંડ લાગુ કરી શકીએ છીએ, આ નવા ડીએજીમાં ડિગ્રી 0 સાથે ઓછામાં ઓછા

એક ડિગ્રી હોવા જોઈએ. તેથી, આપણે તેને અંકુશિત કરી શકીએ છીએ અને ચાલુ રાખીએ છીએ, તેથી આપણે અંદરથી શિરોબિંદુને અંકુશમાં રાખીએ છીએ ડિગ્રી 0 અને ડી.એ.જી. દ્વારા ખાલી થઈ જાય છે, દરેક એન વર્ટીક્સની ગણતરી કરવા માટે આપણે ડી.એ.જી.માંથી કાઢી નાખીશું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:00)
તેથી, ચાલો આપણે આ ડીએજીની વ્યૂહરચનાને લાગુ કરીએ, તેથી આપણે પહેલા દરેક શિર્ષકને લેબલ કરીને પ્રારંભ કરીએ છીએ તે ડિગ્રીમાં છે. તેથી, વાંચીશું કે આપણે દરેક શિખરની ડીગ્રીમાં સૂચન કર્યું છે. તેથી, ઉદાહરણ તરીકે 1 અને 2 માં કોઈ આવનારી કિનારીઓ નથી. તેથી, તેઓ ડિગ્રી 0 માં છે, વેરટેક્સ 3, 2 ડીગ્રીમાં 2 ધાર આવે છે, વર્ટેક્સ 8 ની અંદર 4 કિનારીઓ આવે છે અને 4 થી 4 ડીગ્રીમાં છે અને તેથી, હવે આપણે 0 અંશમાં ક્રમાંકિત ક્રમાંક લેવાનું અને દૂર કરવું પડશે. . તેથી, આપણી પાસે 1 અને 2 વચ્ચેની પસંદગી છે, તેથી ધારીએ કે આપણે 1 થી પ્રારંભ કરીએ છીએ, તેથી અમે 1 થી શરૂ કરીએ છીએ જે આપણે દૂર કરી દીધી છે અને હવે જ્યારે આપણે દૂર કરીએ છીએ ત્યારે અમે તેની બહારની ધારને પણ દૂર કરીએ છીએ. તેથી, 3, 4 અને 5 માં આવતા ધાર 1 થી ઘટાડે છે, કારણ કે 1 લીટી 1 ગઈ છે, તેથી તેમાં આવતા ધાર 1 દ્વારા ઘટાડે છે ત્યાં ડિગ્રીમાં 1 પણ ઘટાડો થાય છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 11:57)
તેથી, 1 ના સમાપ્તિ પર શું થાય છે તે આપણે નોંધ્યું છે અને આપણે 3, 4 અને 5 ની 2, 1 1 થી 1 0 0 ની ડિગ્રી ઘટાડીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:04)
તેથી, યાદ કરો કે તે પહેલા ઘટાડો અથવા 2 1 અને 1, હવે આ ધારો જે તેમની અંદર આવી રહી છે તે કાઢી નાખવામાં આવી છે. તેથી, જ્યારે આપણે આને રદ્ કરીએ છીએ, ત્યારે આપણે ઈનકમિંગ કિનારીઓને પણ ડીલીટ કરીએ છીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:14)
તો હવે આપણી પાસે 1 0 0 છે, હવે આપણી પાસે ત્રણ પસંદગીઓ છે, બે મૂળ છે જે ડિગ્રી 0 માં છે અને આપણી પાસે બે નવા શિરોબિંદુઓ 4 અને 5 છે જે જો તમે કરવા માંગો છો તેમને કૉલ કરો, જેની પૂર્વશરત પૂર્ણ થઈ ગઈ છે. તેથી, કાર્યો 1 એ માત્ર 4 માટે પૂર્વ વિનંતી કરાઈ હતી, ટાસ્ક 1 એ ફક્ત 5 માટે પૂર્ણ આવશ્યક ઈડી છે જે તે પૂર્ણ થઈ ગયું છે. તેથી, 4 અને 5 અથવા હવે ઉપલબ્ધ છે, તેથી આપણે કોઈ પણ 2, 4 અને 5 પસંદ કરી શકીએ તે કોઈ વાંધો નથી. તો, ચાલો ધારીએ કે આપણે 4 પસંદ કરીએ તો ફરીથી આ બે ધારો જશે. તેથી, આ 1 થી ઘટાડે છે અને આ ઘટાડે છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:48)
તો, આપણે તે કરી શકીએ છીએ 4 અને 4 ની 6 થી 6 ની ડીગ્રી ઘટાડીએ, 8 થી 4 થી 4 ની 3 ઘાત. , હવે કદાચ ટાસ્ક 2 ને દૂર કરવાનો નિર્ણય છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:00)
તેથી, જ્યારે તમે પાસ 2 કરો છો, ત્યારે 3 ની ડિગ્રી ઓછી થાય છે અને 8 ડિગ્રીમાં ઘટાડો થાય છે. તેથી, નોંધ લો કે 8 ને હજુ જ 5 અને 7 ની આવશ્યકતાઓમાં બાકી છે, તેથી તે હજી કરી શકાતું નથી, પરંતુ 3 અને 5 ઉપલબ્ધ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:14)
તેથી, કદાચ હું 5 કરું, તેથી હવે 8 વત્તા 1 અને હવે કોઈ વિકલ્પ નથી, 0 ડીગ્રી સાથેનો એકમાત્ર કાર્ય છે 3. તેથી, હું 3 કરું છું, તેથી હવે હું તે જોઈ શકું છું આ વાસ્તવમાં જે રીતે દોરવામાં આવે છે તે એ છે કે તે ક્રમ છે, મારે 6 પહેલા 6, 6 પહેલા 6, 7 થી 7 પહેલા જ હોવું જોઈએ. તેથી, મને આ બિંદુ પર કોઈ વિકલ્પ નથી. 3.
(સ્લાઈડટાઈમ: 13 નો સંદર્ભ લો)
પછી, 6.
(સ્લાઈડસમયનો સંદર્ભ લો: 13:33)
પછી, 7.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:34)
પછી છેલ્લે, 8.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:36)
આ સમયે ગ્રાફ્ ગ્રાફ્ ખાલી છે અને મને શિરોલંબનો અનુક્રમ પ્રાપ્ત કરવો પડશે જે માન્ય ટોપોલોજિકલ ઓર્ડરિંગ છે, કારણ કે શિરોબિંદુ પ્રત્યેક જોડી જે મારા મૂળ ગ્રાફ્માં ધાર છે તે ક્રમ છે. તેથી, ધારનો તે સ્ત્રોત લક્ષ્યની ધારની આગળ દેખાય છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:52)
તેથી, ચાલો આપણે કેટલાક સ્યુડો કોડને એલ્ગોરિધમ માટે જોઈએ જે આપણે હમણાં જ ડીએજી દ્વારા અમલમાં મૂક્યા છે. તેથી, આ ચોક્કસ અલ્ગોરિધમમાં આપણે પ્રથમ ડિગ્રીની ગણતરી કરીને પ્રારંભ કરીએ છીએ, તેથી ડિગ્રીની ગણતરી કરવા માટે, આપણે શોધવાની જરૂર છે કે કેટલા લોકોને જે જીની સંપત્તિ મળે છે તે શોધવા માટે અમારે કેટલી જરૂર છે. 1 ની બરાબર, કારણ કે તે j થી i ની ધાર સાથે અનુલક્ષે છે. તેથી, જ્યારે અડજાનસી (adjacency) મેટ્રિક્સ આ વાક્ય ધરાવતા સ્તંભને જોવામાં આવે છે. કારણ કે, મારી પાસેના સ્તંભમાં, તમારી પાસે એ 1, આઈ 2, એ 2 અને એમના સ્વરૂપમાં પ્રવેશો હશે. તો, આપણી પાસે આ એન્ટ્રી હશે અને આપણે આ બધાને સ્કેન કરવા માંગીએ છીએ અને પછી બધા 1s ઉમેરીશું. તેથી, આપણે 1 ની બરાબર ડિગ્રી સુયોજિત કરીને શરૂ કરીએ છીએ, એક ડિગ્રી હું સમાન 0 અને પછી દરેક હરોળ માટે આપણે j j ઉમેરો. તેથી, તે ક્યાં તો 0 અથવા 1 છે અને તેથી આપણે બધા આવનારા કિનારીઓ એકત્રિત કરીએ છીએ જે મને i ની ડિગ્રી તરીકે સૂચવે છે, હવે આપણે ગણતરી કરવાનું શરૂ કરીએ છીએ. તેથી, આપણે જાણીએ છીએ કે આ એક ડીએગ છે, તેથી આપણે જાણીએ છીએ કે દરેક બિંદુએ ડિગ્રી 0 ની સાથે ઓછામાં ઓછા 1 j છે. તેથી, આપણે આવી કોઈ j પસંદ કરીએ, aj જે ડિગ્રી 0 માં છે તે પસંદ કરો, હવે જ્યારે આપણે ગણતરી કરીએ છીએ કે આપણે ગ્રાફ્માંથી દૂર કરવા ઈચ્છીએ છીએ. વાસ્તવમાં ગ્રાફ્ને વાસ્તવમાં સંશોધિત કરવાને બદલે, અમે માત્ર તે સંશોધિત ગ્રાફ્ના એક અંદાજિત સંસ્કરણ તરીકે ડિગ્રી સાથે કાર્ય કરીશું. તેથી, આપણે પહેલા આ ચોક્કસ શિરચ્છેદ માટે ડીગ્રીમાં ઓછા 1 માં સુયોજિત કર્યું છે. તેથી, બાદબાકી 1 નો અર્થ એ છે કે તે ગ્રાફ્માં હોઈ શકતું નથી, કારણ કે તમારી પાસે ઓછા 1 કિનારી પોઈન્ટ્સ ન હોઈ શકે તે ઓછામાં ઓછા 0 કિનારી હોઈ શકે છે. તેથી, આ અસરકારક રીતે અર્થ છે કે જે j થી 4 અને હવે દરેક આઉટગોઇંગ ધાર માટે j તરીકે ગણવામાં આવશે નહીં. તેથી, જ્યાં પણ આપણે j તરફ્ ધ્યાન આપીએ છીએ, આપણે તેને ઘટાડવા માંગીએ છીએ, કારણ કે આપણે આ ધારને દૂર કરવા જઈ રહ્યા છીએ, આપણે આ ધારને દૂર કરીએ છીએ. તેથી, 1 થી એન સુધીના પ્રત્યેક k માટે આપણે j ના જવાના પાડોશીઓને સ્કેન કરીએ છીએ અને જો જેક એ એજ છે, તો જેકે 1 છે 1 આપણે 1 દ્વારા કે. માં ડિગ્રી ઘટાડીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:48)
તો, આ શું છે એલ્ગોરિધમની જટિલતા એ જોવા માટે એકદમ સરળ છે કે આ અડજાનસી (adjacency) મેટ્રિક્સ અથવા રજૂઆત માટે તે n ચોરસ છે, કેમ કે આપણે ડિગ્રીમાં પ્રારંભ કરવાનું જોયું ત્યારે સમય n ચોરસ
((સમયનોસંદર્ભ: 16:01))
લે છે. કારણ કે, આપણી પાસે 1 થી n ની એક બાહ્ય લૂપ છે અને પછી દરેક બાહ્ય લૂપ માટે આપણી પાસે 1 થી n સુધી આંતરિક લૂપ છે, તેથી આ એક સ્ક્વેર લૂપ છે. અને પછી, જ્યારે આપણે શિરોબિંદુઓને ફરીથી ગણતરી કરીએ છીએ ત્યારે આપણી પાસે એક બાહ્ય લૂપ હોય છે જે એક વાર દરેક શિરોબિંદુની ગણતરી કરશે. અને પછી આંતરિક લૂપ માટે, આપણે બધાને પડોશીઓ અને ઘટાડોની તપાસ કરવાની જરૂર છે. તો, આપણી પાસે 2 એન સ્ક્વેર લૂપ્સ છે અને તેથી, આ સમગ્ર વસ્તુ ઑર્ડર n સ્ક્વેર લે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 16:29)
હવે, આપણે બૅફ્એસ(BFS) અને ડીએફ્એસ(DFS) સાથે જોયું છે, જો આપણે અડજાનસી (adjacency) સૂચિનો ઉપયોગ કરીએ તો આપણે થોડું વધુ હોંશિયાર હોઈ શકીએ છીએ અને આપણે આ વર્ગને એન ચોરસથી રેખીય બનાવવા નીચે લાવી શકીએ છીએ. ક્રમમાં n વત્તા મી. તો, આપણે આ કેવી રીતે કરીએ? ઠીક છે, અમારી પાસે આ સૂચિ હશે, તેથી અમારી પાસે સૂચિ 1, 2 છે અને આમાંના દરેક માટે અમારી પાસે તે સૂચિ છે પડોશીઓ. તેથી, જો તમે આમાંથી પસાર થાઓ, કારણ કે આપણે આ દરેક ધારને કહ્યું છે, હવે આ એક નિર્દેશિત ગ્રાફ્ છે. તેથી, દરેક ધાર ફક્ત એક જ વખત રજૂ કરે છે જો આઈ થી આઈ તરફ્નો ધાર તે i માટે સૂચિમાં એન્ટ્રી j તરીકે દેખાશે. તેથી, જો તમે આ સૂચિને દર વખતે સ્કેન કરો

છો, ત્યારે અમે AJ ને જુએ છે, આપણે જાણીએ છીએ કે j માં એક ધાર છે અને આપણે વધારો કરીશું. તેથી, આપણે બધા ડિગ્રીને 0 થી સુયોજિત કરીને પ્રારંભ કરીએ છીએ, અમે બધી સૂચિને સ્કેન કરીએ છીએ અને જ્યારે પણ અમે કોઈ સૂચિમાં એન્ટ્રી જોતા હોય ત્યારે, અમે તેને ડિગ્રીમાં વધારો કરીએ છીએ. તેથી, સૂચિના એક સ્કેન જે સમય ક્રમમાં છે, આપણે ડિગ્રી મેળવી શકીએ છીએ. હવે, આપણી પાસે ડિગ્રીમાં સૂચિ છે, તેથી આપણે બધા ડિગ્રી શિરોબિંદુઓને કતારમાં મૂકી શકીએ છીએ, જેનાથી આગળનું વાક્ય શું છે તે શોધવાનું સરળ બને છે. તેથી, આ સ્કેનના અંતમાં આપણે બધા ડિગ્રી શોધવા માટે ઑર્ડર એમ સ્કેન કર્યું છે, હવે અને ઑર્ડર n સ્કેન અમે બધા 0 ડિગ્રી શિરોબિંદુઓને કતારમાં મૂકી શકીએ છીએ. હવે, આપણે બાકીના ઘણા બધા કરી શકીએ છીએ જેમ આપણે પહેલા કર્યું છે, આપણે કતારમાં પ્રથમ કર્ણને ગણતરી કરીએ છીએ અને પછી આપણે તેની સૂચિ પર જઈએ છીએ જે હવે અડજાનસી (adjacency) સૂચિમાં સ્પષ્ટપણે ઉપલબ્ધ છે, તે બહારના પાડોશીઓ છે, તેના ડિગ્રીમાં ઘટાડો કરે છે અને જો કોઈ હોય ડિગ્રીમાં તે 0 થાય છે, આપણે કતારમાં ઉમેરી શકીએ છીએ. તેથી, આપણે જાણીએ છીએ કે તે હવે પ્રક્રિયા કરી શકાય છે, તેથી આ એકંદરે બને છે તે સૂચિને સ્કેન કરવા માટે ઓર્ડર એમ સમય લે છે અને કતાર શરૂ કરવા માટે ઓર્ડર n સમય લે છે અને પછી આ ક્રમમાં n નો લૂપ છે જ્યાં બધા અપડેટ્સ પર અમે એકંદરે કરીશું ડિગ્રી ઓર્ડર n વખત અપડેટ કરો, તેથી આ ઓર્ડર n વત્તા એમ છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:26)
તેથી, અહીં અનુરૂપ સ્યુડો કોડ છે, તેથી પ્રથમ ગ્રાફ એ આપણા ગ્રાફમાં પ્રત્યેક શિર્ષક માટે 0 થી અંશની શરૂઆત કરવાનો છે, પછી આપણે બધા ધારમાંથી પસાર થાય છે. તેથી, અમે દરેક અડજાનસી (adjacency) સૂચિને જોઈને આ કરીએ છીએ. પ્રત્યેક શિરોબિંદુ માટે આપણે દરેક પાડોશી i, j અને અડજાનસી (adjacency) સૂચિ તરફ ધ્યાન આપીએ છીએ અને આમાંના પ્રત્યેક માટે આપણે j ની ડિગ્રી વધારીએ છીએ, કારણ કે આપણે j તરફ નિર્દેશ કરતી કિનારીઓ તરફ જોઈ રહ્યા છીએ, જેનો સંકેત આપતાં નથી. તેથી, j માં આ નિર્દેશ જુદી જુદી સૂચિમાં આવશે. પરંતુ, જ્યારે આપણે તેમને પ્રત્યેક માટે તેમનો સામનો કરીશું ત્યારે અમે તેના માટે ખાતર કરીશું અને તેમાં એક ઉમેરીશું. હવે, આપણે સૂચિ દ્વારા એક વાર વધુ શિર્ષકોની સૂચિમાં જઈએ છીએ અને દરેક વખતે ડિગ્રી 1, 0 માં જોશું, હું કતારમાં ઉમેરું છું. અને હવે આપણે આ લૂપ કરીએ છીએ ત્યાં સુધી, કતાર ખાલી ન થાય ત્યાં સુધી આપણે ઓછામાં ઓછા એક ગ્રાફને યાદ રાખીએ છીએ કે એક ગ્રાફ એ ખાલી નથી, આપણે જાણીએ છીએ કે ડિગ્રી 0 ની સાથે ડિગ્રી વર્ટેક્સમાં ઓછામાં ઓછું એક છે. તેથી , તેઓ કતારમાં હોવા જ જોઈએ, કારણ કે અમે તેમને બધાને કતારથી ઉદ્ભવ્યું છે અને દરેક જે આપણે ઉત્પન્ન કરીએ છીએ તે અમે કતારમાં ઉમેરીશું. તેથી, કતાર સાથે ખાલી નથી, આપણે કતારના પ્રથમ તત્વને ધ્યાનમાં લઈએ છીએ, પછી આપણે તેના અડજાનસી (adjacency) સૂચિ તરફ ધ્યાન આપીએ છીએ, તે બધા શિરોબિંદુઓની ડિગ્રી ઘટાડે છે અને જો તેમાંની કોઈપણ હવે 0 ડિગ્રીમાં થાય છે, તો આપણે ઉમેરતાં કતારમાં. તેથી, આ એલિમેન્ટસ સૂચિ અને તત્વોને પ્રક્રિયા કરવા માટે કતારનો ઉપયોગ કરીને હવે ટોપોલોજિકલ સૉર્ટનું રેખાત્મક અમલીકરણ બની ગયું છે. કારણ કે, આપણને આ કતારની જરૂર શા માટે છે કે અન્યથા આપણે દરેક શિરોલંબને સ્કેન કરવું પડશે કે નહીં તે નિર્ધારિત કરવા માટે કે પછી ડિગ્રી 0 માં હોવું જોઈએ, તે પછી તે ઓર્ડર n સ્કેન બની જાય છે, આ ક્રમમાં n ચોરસ બને છે. તેથી, અમને ખાતરી કરવા માટે કતારની જરૂર છે કે આપણે ગણતરી કરવા માટે આગલા શ્વેતલેખને ઓળખવાનો પ્રયાસ કરતા સમય વિતાવીશું નહીં. આપણે બધા શિરોબિંદુઓથી પસાર થવાની જરૂર નથી અને ઈન ડિગ્રી તપાસો, જ્યારે આપણે 0 ડીગ્રીમાં જોઈશું, ત્યારે આપણે તેને કતારમાં મુકીશું. તેથી, આપમેળે તે કોઈ વધુ ચેક કર્યા વિના બહાર આવશે. તેથી, આ ઓર્ડર એમ અહીં જોવા મળે છે કે જે હું અહીં કરું છું.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 07
લેક્ચર - 24
ડીએજીએસ(DAGs): સૌથી લાંબી પાથ્સ

ચાલો આપણે ડીએજી(DAG) પર ધ્યાન આપીએ અને આ વિભાગમાં આપણે ડીએજી(DAG)માં સૌથી લાંબી પાથ ઓળખવા માટે જાણીતી વિવિધ સમસ્યાને જોશું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:08)
તેથી, યાદ રાખો કે નિર્દેશિત એસાયકલિક ગ્રાફ્ એ માત્ર એક નિર્દેશિત ગ્રાફ્ છે જેમાં કોઈ પણ શિરોબિંદુથી કોઈ નિર્દેશિત પાથ નથી, તેથી તે સીધી છે અને તે એસાયકલિક છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:20)
કોઈપણ નિર્દેશિત એસાયકલિક ગ્રાફ્કિ ટોપોલોજિકલ(topological)ી ઓર્ડર કરી શકાય છે. જો તમને શિરોબિંદુઓ 1 થી n ની લાગણી હોય, તો તમે 1 ને અનુક્રમ તરીકે આ રીતે લખી શકો છો કે જે દરેક જોડી j, k જે ધાર છે, જો j, k એ મારા ગ્રાફ્માં એક ધાર છે, તો j અનુક્રમમાં કે પહેલા દેખાશે. તેથી, જો તમે આને કાર્યો તરીકે વિચારો છો તો તે નિર્ભરતા છે એટલે કે હું કાર્યને આ રીતે કરી શકું છું કે હું k કરતા પહેલા, હું તે પૂર્ણ કરી શકું છું, તે નિર્ભરતા કાર્ય છે. તેથી, આને ટોપોલોજિકલ(topological) સોર્ટિંગ(topological sorting) કહેવામાં આવે છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 00:51)
તેથી, ચાલો આપણે ડીએજી(DAG)એસ. વિશે અલગ પ્રશ્ન જોઈએ. તેથી, અમને લાગે છે કે અમારી પાસે આ ડીએજી(DAG) છે અને અમે કલ્પના કરીએ છીએ કે શિરોબિંદુઓ અભ્યાસક્રમો રજૂ કરે છે અને ધાર એ પૂર્વજરૂરી છે. હવે, આ અભ્યાસક્રમો છે જે આપણે કરવાનું છે તે ડિગ્રી પૂર્ણ કરી શકે છે, દરેક કોર્સમાં સત્રની જરૂર છે, પરંતુ અમે સત્રમાં એકથી વધુ કોર્સ કરી શકીએ છીએ. તેથી, પ્રશ્ન એ છે કે, આ આવશ્યકતાઓ સાથે, આ 8 અભ્યાસક્રમો સહિત આ પ્રોગ્રામ પૂર્ણ કરવા માટે મને ઓછામાં ઓછા સેમેસ્ટરની જરૂર છે. તેથી, સ્પષ્ટરૂપે હું પ્રથમ સત્રમાં અભ્યાસક્રમો 1 અને 2 મૂકવાનું શરૂ કરી શકું છું, કારણ કે તેમની પાસે કોઈ પૂર્વશરત નથી, તેથી તેઓ તરત જ કરી શકાય છે. હવે, 1 અને 2 કરવાથી હું 3 કરી શકું છું, કારણ કે 3 ફક્ત 1 અને 2 પર આધાર રાખે છે. એ જ રીતે, હું 4 અને 5 કરી શકું છું કારણ કે તે માત્ર 1 પર આધાર રાખે છે, હવે 8 એ 1 અને 2 પર આધાર રાખે છે મને માલની જરૂર છે અથવા ઓછામાં ઓછું 2 પર આધાર રાખે છે. મારે 2 કરવાની જરૂર છે, 2 કરવા માટે 2, પણ હું હજી 8 કરી શકતો નથી, કારણ કે તે 5, 4 અને 7 ની પણ જરૂર છે. તેથી, આ બિંદુએ આપણે ફક્ત 3, 4 અને 5. તેથી, મારા બીજા સત્રમાં હું 3, 4 અને 5 કરી શકું છું. હવે, એકમાત્ર એવો કોર્સ કે જેના માટે બધી મફત પૂર્વજીઓ સંતુષ્ટ છે 6, 7 માટે 6 ની જરૂર નથી અને 8 ની આવશ્યકતા 7 થઈ નથી. . તેથી, ત્રીજા સત્રમાં હું ફક્ત એક જ અભ્યાસક્રમ કરું છું. 6. ચોથા સત્રમાં, હું 7 કરી શકું છું અને અંતે, 5 સત્ર પછી હું આ પૂર્ણ કરી શકું છું
(સમયનોસંદર્ભ લો: 02:12).
તેથી, સામાન્ય રીતે આપણે આ પ્રશ્નનો પૂછી શકીએ, જો હું આ વિશે વિચારીશ કે જેમ કે ડીએજી(DAG) અભ્યાસક્રમોનું પ્રતિનિધિત્વ કરે છે, તો ઓછામાં ઓછા સેમેસ્ટર શું છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:24)
હવે, આ સમસ્યા ડીએજી(DAG)
((સમય:02:28))
માં સૌથી લાંબી પાથ માટે પૂછવાની અનુરૂપ છે. આપણે જે કહી રહ્યા છીએ તે છે કે 8 પૂર્ણ કરવા માટે 5 સેમેસ્ટર લાગે છે, કારણ કે ત્યાં છે નિર્ભરતાની સાંકળ, જ્યાં 8, 7 પર આધાર રાખે છે, 6 6 પર નિર્ભર છે, 3 અથવા 4 પર આધાર રાખે

છે તે કોઈ વાંધો નથી કે આપણે કઈ પસંદ કરીએ છીએ અને 3 એ 1 પર નિર્ભર છે અને આ સાંકળ આપણને 4 સેમસ્ટરનો ખર્ચ કરવા દબાણ કરે છે, કારણ કે તે લંબાઈ 4 ની સાંકળ, હું 4 કરી શકે તે પહેલાં 4 સેમિસ્ટર. નોંધ લો કે તે સૌથી ટૂંકી સાંકળ નથી, ઉદાહરણ તરીકે ટૂંકા સાંકળો છે, કારણ કે 8 થી 2 એ શંકુચક્રમાં પાછું આવે છે જે 0 અંશ ધરાવે છે, પરંતુ તે આપણને મદદ કરતું નથી કારણ કે 2 પછી હું તે કરી શકતો નથી. તેથી, જોબ મેળવવા માટે 8 કરતા પહેલાં જે બનવું છે તે બધું માટે રાહ જોવી આવશ્યક છે. તેથી, અન્ય સમસ્યાઓથી વિપરીત, જ્યાં આપણે ટૂંકા માર્ગો શોધી શકીએ છીએ, અહીં આપણે ખરેખર સૌથી લાંબી માર્ગમાં રસ ધરાવો છો. તો, આપણે આ સમસ્યાને નીચે પ્રમાણે સુયોજિત કરી શકીએ છીએ, તેથી આપણે કહી શકીએ કે કોઈ પણ વાક્ય માટે જે 0 શામેલ છે, તે કર્ણનો સૌથી લાંબો માર્ગ લંબાઈ 0 છે, કારણ કે હું તે તરત જ કરી શકું છું. અને બીજી બાજુ, જો મારી પાસે કોઈ શંકુદ્રષ્ટા હોય, જેની ઈન્ડિગ્રી 0 નથી, તો તેમાં કેટલાક ઈનકમિંગ ધાર છે. તેથી, મારી પાસે વર્ટેક્સ k છે, તેથી મારે આને સમાપ્ત કરવા માટે રાહ જોવી જોઈએ અને પછી તે કરવું. તેથી, જો હું આ બધાને લઈશ, તો આમાંના બધામાં હું મહત્તમ લંબાઈ લઈશ, કારણ કે તે સમાપ્ત થવાની છેલ્લી વસ્તુ હશે અને તે પછી મારે માટે વત્તા 1 નું ખાતું કરવું પડશે. તેથી, જો ઈન્ડિગ્રી 0 ન હોય તો, k સુધીના સૌથી લાંબી પાથની લંબાઈ 1 વત્તા મહત્તમ આવનારા પાડોશીઓ માટે સૌથી લાંબી પાથ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:01)

તેથી, તેથી, k સુધીના સૌથી લાંબી પાથની ગણતરી કરવા માટે, મને આવનારા પાડોશીઓ માટેના સૌથી લાંબી પાથની ગણતરી કરવાની જરૂર છે. હવે, જો આપણે ટોપોલોજિકલ(topological) ઓર્ડરમાં શિરોબિંદુ ગોઠવીશું અને અમે તે ક્રમમાં સૌથી લાંબી પાથની ગણતરી કરીશું, તો જ્યારે આપણે દરેક આવનારા પાડોશીને મળશે ત્યારે તે બાકી રહેશે. તેથી, આપણે પહેલાથી જ ગણતરી કરીશું કે તે સૌથી લાંબો માર્ગ છે, તેથી અમે આમાંની મહત્તમ સંખ્યાને લેવા અને તેમને ઉમેરવા સક્ષમ થઈશું. તેથી, આ શિરોબિંદુઓને ટોપોલોજિકલ(topological) ઓર્ડરમાં સૉર્ટ કરીને, હું બાંયધરી સાથે તે જ ક્રમમાં સૌથી લાંબી પાથની ગણતરી કરી શકું છું કે જ્યારે હું શરુઆત કરવા માટે સૌથી લાંબી પાથની ગણતરી કરવા માંગું છું, ત્યારે મારી પાસે તે કરવાની બધી માહિતી ઉપલબ્ધ છે, એટલે કે તે માટે સૌથી લાંબો માર્ગ પાડોશી આવનારા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:48)

તેથી, જો હું નૈતિક રીતે કરું તો હું કેટલાક શિષ્ટાચાર ક્રમમાં મારા શિર્ષકો લખીશ અને હવે જ્યારે હું આ વાક્ય પર આવીશ અને હું ગણતરી કરવા માંગું છું તે સૌથી લાંબો માર્ગ છે, હું આમાં જોઈશ મારા ગ્રાફ તમામ આવનારી કિનારીઓ અને તેઓ બધા શિરોબિંદુઓમાંથી હશે જે પહેલાં દેખાય છે. તેથી, હું અહીં વેલ્યુ લઈ શકું છું, અહીં વેલ્યુ, અહીં વેલ્યુ અને પછી તે મહત્તમ છે અને પછી અહીં ઉમેરો. તેથી, વાસ્તવમાં આપણે જોશું કે તમારે આ પછાત કરવાની જરૂર નથી, તમે વાસ્તવમાં તે આગળ કરી શકો છો. તેથી, તમે ખરેખર આગળ વધી રહ્યા છો તે રીતે ik પર મૂલ્યની ગણતરી કરી શકો છો. કારણ કે, પાછળની તરફ જવા માટે તમારે આ સૂચિને સ્કેન કરવાની અને આવનારા પાડોશીઓને શોધવાની જરૂર છે. તો, આપણે આ પ્રકારની સૌથી લાંબી પાથની ગણતરીઓ સાથે મળીને ટૉપોલોજિકલ સૉર્ટ બાજુ સાથે કરીશું, કેમ કે આપણે આગલા ઉદાહરણમાં જોશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:42)

તેથી, અહીં એક ઉદાહરણ છે જેમાં આપણે સૌથી લાંબી પાથની ગણતરી કરવા જઈ રહ્યા છીએ કારણ કે આપણે ટોપોલોજિકલ(topological) ઓર્ડરની ગણતરી કરી રહ્યા છીએ. તેથી, જેમ હું શિરોલંબ તરીકે આપવામાં આવેલ લાલ નંબરો પહેલા ટોપોલોજિકલ(topological) સૉર્ટ માટે વપરાય છે અને તે ઈન્જેગ્રીને સૂચવે છે. તેથી, પ્રારંભિક ઈન્ડેગ્રીઝ આપવામાં આવે છે જ્યાં અમારા ગ્રાફમાં પ્રારંભિક ધાર અને આપણે શું કરીએ છીએ તે આપણે આ ધારીને શરૂ કરીને સૌથી લાંબી પાથની ગણતરી કરીએ છીએ કે દરેક શિરોબિંદુનો સૌથી લાંબો માર્ગ ખરેખર 0 છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:15)

હવે , જ્યારે આપણે ટોપોલોજિકલ(topological) સૉર્ટમાં શંકુનું વર્ગીકરણ કરીએ છીએ જે આપણે અગાઉ કર્યું હતું તે ઈન્ડેગ્રીસને અપડેટ કરવું હતું, તેથી આ તે છે જે આપણે પહેલા કર્યું છે. પરંતુ, હવે આપણે જે વધારાની વસ્તુ કરીએ છીએ તે આપણે કહીએ છીએ કે જો વેરટેક્સ 1 3, 4 અને 5 ની પહેલા ગણી શકાય, તો પછી 3, 4 અને 5, 0 ની આવનારી

કિનારીઓ માટે હું જે મૂલ્યો જાણું છું તે વચ્ચે મહત્તમ લંબાઈ. તેથી, મારે 1 વત્તા તે કરવું જ પડશે, તેથી આ પાથ ઓછામાં ઓછા લંબાઈ 1 છે. તેથી, 3, 4 અને 5 નો સૌથી લાંબો રસ્તો ઓછામાં ઓછો છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 06:47)
હવે, જો હું વર્ટેક્સ ને ક્રમાંકિત કરું 4, તો તે લાંબો માર્ગ છે ઓછામાં ઓછો 1, તેથી, 6 સુધીના સૌથી લાંબો માર્ગ ઓછામાં ઓછો 1 વત્તા 1 હોવો આવશ્યક છે. એ જ રીતે, 8 સુધીનો સૌથી લાંબો માર્ગ ઓછામાં ઓછો 1 વત્તા 1 હોવો આવશ્યક છે અનેઅલબત્ત, ઈન્જેગ્રી પણ પહેલાંની જેમ ઘટાડે છે. તેથી, 6 નું અવતરણ 1 ની સંખ્યામાં જાય છે, 8 નું અવતરણ 3 થાય છે, પરંતુ 6 નું સૌથી લાંબું પાથ હવે 4 વત્તા 1 નું સૌથી લાંબું પાથ છે, તેથી તે 2 છે, તે 8 માટે સમાન છે. હવે, ધારો હું vertex 2 ની ગણતરી કરું છું, હવે 2 ફરીથી કહેશે કે સૌથી લાંબો માર્ગ, 2 ની સૌથી લાંબી પાથ 1 છે, પણ તે પહેલેથી જ 1 છે, તેથી આપણે તેને બદલી શકતા નથી. આ કહેશે કારણ કે 2 ની સૌથી લાંબી પાથ ઓછામાં ઓછી 1 છે, પરંતુ આપણે જાણીએ છીએ કે તે ઓછામાં ઓછી 2 છે, તેથી ફરીથી આપણે બદલાતા નથી. તેથી, જ્યારે આપણે નોડને કાઢી નાખીએ છીએ, ત્યારે આપણે મૂળભૂત રીતે સૌથી લાંબી પાથની વર્તમાન કિંમત લઈએ છીએ તે બહારની વસ્તુ વત્તા 1 છે, કાઢી નાખેલ નોડ વત્તા 1 નું વર્તમાન મૂલ્ય અને નોડ પર પહેલેથી જ જાણીતું છે અને અમે મહત્તમ રાખીએ છીએ. તેથી, આ કિસ્સામાં આ 1 0 થશે, આ 3 થશે 2, પણ અહીં કોઈ ફેરફાર નથી અને અહીં કોઈ ફેરફાર નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:51)
અને જ્યારે હું આ 5 ને રદ્દ કરીશ, આ 2 એક થશે, પણ કારણ કે 1 વત્તા 1 એ 2 છે અને આપણે પહેલેથી જ જાણીએ છીએ કે 8 ઓછામાં ઓછા 2 નો સૌથી લાંબો માર્ગ છે, આપણે કોઈપણ 2 માં બદલો.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:05)
હવે, જ્યારે આપણે 3 પર જઈએ, 3 કહે છે કે તે 1 નું સૌથી લાંબો રસ્તો છે, તેથી 3 ની સૌથી લાંબી પાથ 6 ઓછામાં ઓછા 2 છે, પરંતુ આપણે પહેલાથી જ જાણીએ છીએ તે 2 છે, તેથી ફરીથી કોઈ ફેરફાર નથી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:17)
હવે, આપણે કંઈક રસપ્રદ કરીશું, તેથી આપણે કહીશું કે 6 એ 2 નો સૌથી લાંબો રસ્તો છે, 7 આપણે અત્યાર સુધી માનીએ છીએ કે તેનો 0 નો સૌથી લાંબો માર્ગ છે, પરંતુ 6 થી વધુ તે લાંબો છે માર્ગ તેથી, 7 થી પાથ હવે 0 થી 3 થી અપડેટ થવું આવશ્યક છે
(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:34)
અને હવે આ કારણે, જ્યારે આપણે 7 થી 8 સુધી જઈએ છીએ, 8 માટેનો સૌથી લાંબો માર્ગ 2 થી 4 થી અપડેટ થવો આવશ્યક છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 08:46)
અને હવે છેલ્લે, આ મારી છેલ્લી કડી છે, તેથી મેં હમણાં જ ગણતરી કરી છે અને હું તેનો સૌથી લાંબો માર્ગ 4 ગણું છું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:52)
તો, આ શું કહેવામાં આવે છે સૌથી લાંબો માર્ગ 4 છે, હવે આપણે કહ્યું છે કે અમે 5 સત્રમાં કર્યું છે, આ જ ઉદાહરણનો મૂળ અર્થ એ છે કે પ્રથમ સત્રમાં જે લોકોનો સૌથી લાંબો રસ્તો 0 છે, તે બીજા સત્રમાં જેનો સૌથી લાંબો રસ્તો 1 છે અને તેથી તે જ છે અનુક્રમ તમે પહેલાં હતી. તેથી, તમે શરૂઆતમાં પ્રથમ સત્રમાં 1 અને 2 કરો છો, પછી અમે બીજા સત્રમાં 3, 4 અને 5 કરીએ છીએ, પછી અમે ત્રીજા સત્રમાં 6, ચોથા સત્રમાં 7 અને પાંચમા સત્રમાં 8 કરીએ છીએ. તેથી, આ તે જ ઉકેલ છે જેનો પ્રારંભિક ઉદાહરણમાં અમે અનૌપચારિક રીતે મેળવ્યા.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:22)
તેથી, આ સ્યુડો કોડ જે આપણે જોયો તે લાંબા માર્ગ માટે ખૂબ જ સમાન છે, આપણે ટોપોલોજિકલ(topological) સૉર્ટ માટે જે કર્યું છે તેના જેવું જ છે, અમે આ વધારાની લાંબી પાથ વેલ્યુનો ટ્રૅક રાખીએ છીએ. તેથી, જ્યારે આપણે ઈન્ડેગ્રીની શરૂઆત કરી ત્યારે અમે દરેક વાક્ય માટે 0 હોવાનો સૌથી લાંબો માર્ગ પણ પ્રારંભ કરીએ છીએ. તો, આ એ છે કે આપણે અડજાનસી લિસ્ટ વર્ઝન પહેલા ફરીથી કરી રહ્યા છીએ. તેથી, આપણે દરેક વર્ટેક્સ માટે n સ્ક્વેર્ડ કાર્ય કરીએ છીએ, અમે એન્ટ્રી કૉલમ એન્ટ્રી i સાથે અડજાનસી (adjacency) મેટ્રિક્સના સ્તંભને જોઈને ઈન્ડેગ્રીની ગણતરી કરીએ છીએ. પરંતુ,

અમે પણ અપડેટ કરીએ છીએ, હું દરેક 0 નું સૌથી લાંબું પાથ પ્રારંભ કરું છું, હવે જ્યારે આપણે આ ગણતરી કરી રહ્યા છીએ તે પહેલા આપણે કોઈપણ વેરટેક્સ ઈન્ડેગ્રી 0 પસંદ કરીશું, આપણે ગણતરી કરીશું અને આપણે આ વર્ટીક્સની ઈન્ડેગ્રીને બાદબાકી 1 થી સુધારીશું. તેથી, તે છે અમે ફરીથી પસંદ કરવા માટે વિવાદમાં લાંબા સમય સુધી ન હતા. હવે, તે બધા માટે પડોશીઓ જવાનું છે, અમે ઈન્ડેગ્રીને અપડેટ કરીએ છીએ અને અમે સૌથી લાંબો માર્ગ પણ અપડેટ કરીએ છીએ. તેથી, આપણે સૌથી લાંબો માર્ગ લઈએ છીએ જે આપણે પહેલાથી જ તે પડોશીને, કેપીની એલપીટી(LPT)ને જાણીએ છીએ અને આપણે આ નોડ માટે 1 લાંબો સૌથી લાંબો રસ્તો લઈએ છીએ અને જે પણ મોટો છે તે આપણે કે જે પીટી(PT)ની પેટર્નને બદલીએ છીએ. તેથી, તે મૂળ ટોપોલોજિકલ(topological) ઓર્ડરિંગ વસ્તુની ખૂબ જ સરળ વિવિધતા છે જે સૌથી લાંબી પાથની ગણતરી પણ કરી શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:34)

તેથી, આ જ કારણસર જટિલતા ક્રમાંક n સ્ક્વેર્ડ છે જેને આપણે નજીકના મેટ્રિક્સ અથવા ઑર્ડર n સ્ક્વેર્ડ સાથે તે ટોપોલોજિકલ(topological) શોર્ટ મળ્યું છે. કારણ કે, અમારી પાસે બધા પાડોશીઓને સ્કેન કરવા માટે નેસ્ટેડ આંટીઓ છે, તેથી પહેલા આપણે આખી વસ્તુને અડીને સૂચિ અને કતાર દ્વારા આને એક રેખીય બનાવવા માટે ઍલ્ગોરિધમનો ઉપયોગ કરીને અપડેટ કરી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:56)

તેથી, જો તમે પાછા જાઓ અને ટોપોલોજિકલ(topological) ઓર્ડરિંગ એલ્ગોરિધમ જુઓ, તો તમને મળશે કે તે જ ફેરફાર સૌથી લાંબી પાથની વધારાની ગણતરી કરવા માટે જરૂરી છે. તેથી, આપણે જે કરીએ છીએ તે છે તે ફરી શરૂ કરવું તે જ છે જે આપણે દરેક કર્ણ માટે કરીએ છીએ. તેથી, આ એક ક્રમ n પગલું છે, અમે ઈન્ડેગ્રી અને લાંબો માર્ગ બંને પ્રારંભ કરીએ છીએ. પછી આપણે બધા અડીને સૂચિમાંથી પસાર થાય છે અને પ્રારંભિક ઈન ડિગ્રીની ગણતરી કરવા માટે આ ક્રમમાં એમ પગલું લઈએ છીએ. અને હવે આપણી પાસે આ કતાર સુયોજિત કરવા માટે ઓર્ડર n પગલું છે, જ્યાં આપણે ટોપોલોજિકલ(topological) ક્રમમાં પ્રક્રિયા કરીશું. અને પછી, આપણે ક્રમમાં અને બાહ્ય લૂપ ક્રમમાં કરીએ છીએ, તેથી આ એક ઓર્ડર એન લૂપ છે, કારણ કે બધું જ રહ્યું છે અને કતારમાં એકવાર આવી ગયું છે. તેથી, આપણે કતાર એન ટાઈમ્સમાંથી દૂર કરવા જઈ રહ્યા છીએ અને હવે કારણ કે આપણે બધા એન પુનરાવર્તનની સૂચિને સ્કેન કરી રહ્યાં છીએ જે અમે એકવાર દરેક ધાર પર પ્રક્રિયા કરવા જઈ રહ્યા છીએ. તેથી, આ લૂપમાં કરાયેલ કુલ કાર્ય ઓર્ડર n બનશે. તેથી, આપણે ઈન્ડેગ્રીને અપડેટ કરતા પહેલા બરાબર કરીએ છીએ, આપણે k નો સૌથી લાંબો પાથ, વર્તમાન વેલ્યુ અને 1 વત્તા વર્તમાન નોડના વેલ્યુને અપડેટ કરીએ છીએ અને જો આપણને લાગે છે કે k હવે ઈન ડિગ્રી 0 માટે વર્ટેક્સ બની ગયું છે. અમે કીમાં ઉમેર્યા.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:13)

તેથી, ડીએજી(DAG)એસ ખૂબ જ ઉપયોગી છે કારણ કે ત્યાં ઘણી પરિસ્થિતિઓ છે, જ્યાં આપણે વિવિધ પદાર્થો વચ્ચે નિર્ભરતાને મોડલ કરવા માંગીએ છીએ. અને ટૉપોલોજિકલ(topological) ઓર્ડરિંગ એ કેનોનિકલ(canonical) એલ્ગોરિધમ છે જે નિર્ભરતાને ઉલ્લંઘન કર્યા વિના શિરોબિંદુઓની સૂચિબદ્ધ કરવા માટે છે, જે આજે આપણે જોયેલી છે તે છે કે, તમે ડીએજી(DAG)માં સૌથી લાંબી પાથની ગણતરી કરી શકો છો અને કેટલાક અર્થમાં ડીએજી(DAG)માં સૌથી લાંબો માર્ગ પાઠવી શકો છો જો આપણે જૂથોમાં શિર્ષકોની સૂચિ બનાવી શકીએ , ડીએજી(DAG)માં સૌથી લાંબો પાથ બધા શિરોબિંદુઓને ક્રમાંકિત કરવા માટે ઓછામાં ઓછા પગલાઓનું પ્રતિનિધિત્વ કરે છે. તેથી, જો આપણે જૂથોમાં અભ્યાસક્રમો જતા હોઈએ, તો જો તમે તેને એક બાજુએ કરવા અને એક જ સમયે સમાન સ્તર કરવા માંગતા હો, તો પછી અભ્યાસક્રમોના સેટ અથવા ન્યૂનતમ સંખ્યાને પૂર્ણ કરવા માટે ઓછામાં ઓછા સેમેસ્ટરની જરૂર છે કાર્યોના સેટને પૂર્ણ કરવાના દિવસો સૌથી લાંબા માર્ગ દ્વારા આપવામાં આવે છે. હવે, તે મહત્વપૂર્ણ છે કે આપણે સૌથી લાંબી પાથ માટે અસરકારક રીતે રેખીય ટાઈમ એલ્ગોરિધમ શોધવામાં સક્ષમ છીએ કારણ કે પ્રતિબંધિત

((સમયનોસંદર્ભ: 13:12))

ડીએજી(DAG)એસ. જો આપણે મનસ્વી ગ્રાફ્સ જોઈશું જે જરૂરી છે કે ડીએજી(DAG)એસ નથી અને આપણે કોર્સનો સૌથી લાંબો માર્ગ શોધવા માંગીએ છીએ, જો આપણી પાસે લૂપ્સ હોય તો ત્યાં સૌથી લાંબો રસ્તો અનિશ્ચિત રહેશે,

કારણ કે આસપાસ અને આસપાસ લૂપ જઈ શકે છે. તેથી, જો આપણે એક બનવાનો સૌથી લાંબો રસ્તો વ્યાખ્યાયિત કરીએ છીએ જેમાં આપણી પાસે કોઈ ડુપ્લિકેટ્સ વિના શિર્ષકોનું અનુક્રમ છે. તેથી, લંબાઈ વધુમાં વધુ 10 છે, પછી મનસ્વી ગ્રાફ આ એક ખૂબ જ પડકારરૂપ સમસ્યા છે અને ત્યાં કોઈ કાર્યક્ષમ અલ્ગોરિધમનો જાણ નથી. હકીકતમાં, તે અત્યંત ક્રિયાપ્રતિક્રિયાત્મક સમસ્યાઓના સમૂહનો ભાગ છે જે બધા એકબીજાના સમકક્ષ છે અને બધા ખૂબ જ મુશ્કેલ હોવાનું માનતા હોય છે. તેથી, ડીએજી(DAG) ગ્રાફના ખૂબ જ મહત્વપૂર્ણ સબક્લાસ છે જેમાં ઘણા વ્યવહારુ એપ્લિકેશનો છે અને તે ખૂબ જ મહત્વપૂર્ણ સમસ્યાઓ માટે કાર્યક્ષમ ઉકેલો સ્વીકારે છે, જેમાં સામાન્ય રીતે ગ્રાફ અને ગ્રાફ્સની પૂર્ણ વર્ગમાં એલ્ગોરિધમ નથી.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 01
લેક્ચર - 25
વેઈટેડ(Weighted) ગ્રાફ્માં લઘુતમ પાથ

ચાલો હવે અમારું ધ્યાન વેઈટેડ(Weighted) આલેખ તરફ્ ફેરવીએ, ગ્રાફ્સ જેમાં તેમની સાથે સંકળાયેલા ખર્ચનો સમાવેશ થાય છે. વેઈટેડ(Weighted) ગ્રાફ્સ સાથે અમે કરી શકીએ તે સૌથી અગત્યની વસ્તુ છે ટૂંકા પાથની ગણતરી કરવી.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:14)

તેથી, આપણે અત્યાર સુધી જે જોયું છે તેની સમીક્ષા કરીએ. તેથી, અમે વેઈટેડ(Weighted) ગ્રાફ્ને જોઈ રહ્યા છીએ જે શિરોબિંદુઓ અને કિનારીઓના સેટ્સનો સેટ છે, જ્યાં દરેક ધાર ફ્ક્ત બે શિરોબિંદુઓ વચ્ચેનો જોડાણ છે. અમે આલેખની શોધ કરવા માટે બે વ્યવસ્થિત વ્યૂહરચનાઓ જોઈ છે, પહોળાઈ પ્રથમ શોધ અને ઊંડાઈ પ્રથમ શોધ. હવે, આ બંને ગ્રાફ્ના કદમાં રેખીય હશે, જો આપણે કિનારીઓ માટે અડજાનસી(adjacency) સૂચિ રજૂઆતનો ઉપયોગ કરીએ છીએ. હવે, આપણે જોયું કે જ્યારે આપણે આપેલ શિર્ષકમાંથી બીએફ્એસ(BFS) અથવા ડીએફ્એસ(DFS)નો ઉપયોગ કરીને ગ્રાફ્ શોધીએ છીએ ત્યારે, આપણે ફ્ક્ત તે જ શોધતા નથી કે આ શિરોલંબ આ શિર્ષક સાથે જોડાયેલા છે. અમે દરેક નોડના માતાપિતા, જેમની મુલાકાત લઈએ છીએ, વધારાની માહિતી રાખીને શરૂઆતના શિર્ષક તરફ્ પાથને પાછો મેળવી શકીએ છીએ. આપણે અવલોકન કર્યું છે કે પહોળાઈ પ્રથમ શોધ, કારણ કે જો તે લેયર સ્ટ્રેટેજી દ્વારા સ્તર છે, તો વાસ્તવમાં શરૂઆતના ભાગથી ધારની સંખ્યાના સંદર્ભમાં પ્રત્યેક શિરોબિંદુની સૌથી ટૂંકી અંતરને દૂર કરે છે. અને ઊંડાણ પ્રથમ શોધો, જો કે તે ટૂંકા પાથને શોધી શકતું નથી, તે આપણને જે ક્રમમાં આપણે મુલાકાત લીધી તેના સંદર્ભમાં અમને ઘણી બધી માહિતી આપે છે. તેથી, પૂર્વ અને પોસ્ટ ડીએફ્એસ(DFS) નંબરિંગનો ટ્રૅક રાખીને, અમે વાસ્તવિક ગ્રાફ્ના ગુણધર્મોને પુનઃપ્રાપ્ત કરી શકીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 01:22)

હવે, જો આપણે કિનારીઓ પર ખર્ચ ઉમેરીશું તો શું થશે તે જુઓ. હવે, એપ્લિકેશન પર આધાર રાખીને, કેટલાક કુદરતી અર્થઘટનની આ કિંમત, ઉદાહરણ તરીકે, ગ્રાફ્ના પ્રારંભિક ઉદાહરણમાં, અમે એરલાઈન રૂટીંગ નકશા પર ધ્યાન આપીએ છીએ. તેથી, આલેખ પર, ધાર વજન, ધાર સાથે સંકળાયેલ ખર્ચ ફ્લાઈટ પર ટિકિટ ઈનામ હોઈ શકે છે અથવા જો ધાર ધોરીમાર્ગોના નેટવર્કમાં રસ્તાઓનું પ્રતિનિધિત્વ કરે છે, તો અમારી પાસે ચૂકવણી કરવા માટેના ટૉલનું પ્રતિનિધિત્વ કરતી કિંમત હોઈ શકે છે. રસ્તાના ચોક્કસ ભાગ પર. અથવા, જો તમારી પાસે રેલવે નેટવર્ક હોય, તો તે બે સ્ટેશન વચ્ચે અથવા શહેરના એક સરળ ટ્રાફ્રિક રોડ નકશા વચ્ચેની અંતરનું પ્રતિનિધિત્વ કરી શકે છે, તે વ્યસ્ત સમયે બે આંતરછેદ વચ્ચેનો કેટલો સમય લઈ શકે તે દર્શાવશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:12)

તેથી, સામાન્ય રીતે, વેઈટેડ(Weighted) ગ્રાફ્ એ એક વધારાનો ફંકશન સાથેનો એક સામાન્ય ગ્રાફ્ છે જે અમને દરેક ધારની કિંમત કહે છે, આ એક વજન કાર્ય કહેવા માટે સામાન્ય છે. તેથી, વેઈટ ફંક્શન ફ્ક્ત E માં દરેક ધારને અમુક નંબર અસાઈન કરે છે, તેથી આ કેટલાક વાસ્તવિક સંખ્યા વિશે વિચારે છે. તેથી, હવે, જો તમારી પાસે v 0 થી vn તરફ્નો પાથ હોય, તો આપણી પાસે v 0 થી vn નો પાથ છે, આપણી પાસે n એ એજન્સનો ક્રમ છે, v 0, v1, v 1 v 2, આમાંના દરેક આપણી વેઈટેડ(Weighted) આલેખનો વજન હશે અને કુદરતી વસ્તુ કરવાથી આ વજનની કિંમત ઉમેરવામાં આવશે. તેથી, આ વિચારીને અંતર છે, પછી કુલ અંતર આ પાથને અનુસરવું જોઈએ અને આપણે બનવું છે, તે અંતરનો સરવાળો હશે. જો તે ફ્લાઈટ્સનું અનુક્રમ છે, તો અમે ટિકિટ માટે ચૂકવણી કરતો કુલ ખર્ચ ખર્ચની રકમ હશે. તેથી, અમે દરેક પાથ સાથે વજનના સરવાળાને ઉમેરીને કિનારીથી પાથ સુધીના વજનને વિસ્તૃત કરીશું. અને હવે, કુદરતી સમસ્યા જે આપણે

ઉકેલવા માગીએ છીએ તે છે મેટ્રીસેસની આપેલ જોડી વચ્ચેનું સૌથી ટૂંકું પાથ; તે જ લઘુત્તમ ખર્ચ છે જે હું વી 0 થી v ના જવા માટે લઈ શકું છું.
(સ્લાઈડસમયનો સંદર્ભ લો: 03:19)
તેથી, આપણે જોયું છે કે તે પહોળાઈ પ્રથમ શોધ આ સમસ્યાને હલ કરશે, જો અમારી કિંમત એ ધારની સંખ્યાના સંદર્ભમાં છે. કહેવાનો બીજો રસ્તો એ છે કે આપણે ધારવું જોઈએ કે દરેક ધારની કિંમત એક જ છે, 1 અથવા કોઈ નિયત સંખ્યા. તેથી, જ્યારે હું વધુ ધાર તરફ આગળ વધું છું, ત્યારે ખર્ચ અમારા માટે ધારની સંખ્યા તરફ પ્રાયોગિક છે. પરંતુ, જો તમારી પાસે આ મિલકત નથી, તો કિંમત મનસ્વી હોઈ શકે છે, પછી પહોળાઈ પ્રથમ શોધ કામ કરતું નથી. ઉદાહરણ તરીકે, આપણે જોઈ શકીએ છીએ કે 1 અને 3 વચ્ચે, સીધી ધાર છે, પરંતુ તે વજન 80 છે અને આપણે 2 ઉમેરીને જઈને 1, 2, 3 ના ટૂંકા ખર્ચ પાથ કરી શકીએ છીએ. તેથી, તેની પાસે બે ધાર, તમને 10 વત્તા 6, જે 16 ની જરૂર છે તે કુલ ખર્ચ. તેથી, ભારાંકવાળા ગ્રાફમાં ટૂંકાતમ પાથને ધારની સંખ્યાના સંદર્ભમાં સૌથી નાનો માર્ગ હોવો જોઈએ નહીં, અમે કિનારીઓ સાથેના ખર્ચની રકમ વિશે વાત કરી રહ્યાં છીએ . તેથી, ધારની સંખ્યાના સંદર્ભમાં અમારી પાસે લાંબું પાથ હોઈ શકે છે, પરંતુ ટૂંકા કુલ ખર્ચ. અને અમારો ધ્યેય આ ટૂંકા માર્ગની ખરેખર લંબાઈને ધ્યાનમાં લીધા વિના આવા પાથોની ગણતરી કરવાનો છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:18)
તેથી, આપણે ટૂંકા પાથને લગતા બે પ્રકારના પ્રશ્નો પૂછી શકીએ છીએ, એક કહેવાતો સિંગલ સ્રોત પાથ છે. એક સ્ત્રોત ટૂંકા પાથ સમસ્યામાં, અમારી પાસે એક ડિજિટલ સ્થાન છે જ્યાંથી આપણે આપણા બધા પાથ શરૂ કરીએ છીએ અને અમે દરેક અન્ય પાથની સૌથી ટૂંકી અંતર શોધવા માંગીએ છીએ. હવે, આ
(સમયનોસંદર્ભ લો: 04:37)
ખૂબ કુદરતી એપ્લિકેશન છે, તેથી તમે કલ્પના કરો છો કે તમે એક ઉત્પાદક છો અને તમારી પાસે એક ફેક્ટરી છે જ્યાં તમે તમારી બધી વસ્તુઓ બનાવો છો, હવે તમારે આ વસ્તુઓને દેશભરમાં વહેંચવી પડશે. તેથી, તમે તમારા ઉત્પાદનોને ફેક્ટરીમાં પરિવહન કરવાનો સૌથી ટૂંકો રસ્તો કેવી રીતે શોધી શકો છો જે આ તમામ સ્થાનો માટે એક બિંદુ છે, જ્યાં તે વેચી શકાય છે અથવા તમામ છૂટક આઉટલેટ્સ અથવા તમે કુરિયર કંપની હોઈ શકો છો. હવે, કુરિયર કંપની બધી વસ્તુઓને આપેલ શેરીમાં ખસેડશે, તે એક કેન્દ્રિત કાર્યાલય હતું અને આ કેન્દ્રીયકૃત કાર્યાલયથી, તે શહેરનાં તમામ ભાગોમાં વિતરણ કરવાની રહેશે. તેથી, તમે વિતરણ કચેરીમાંથી સૌથી નાનો માર્ગ કેવી રીતે કુરિયર આઈટમ્સને વિતરિત કરવા માટેના તમામ જુદા જુદા રસ્તાઓ પર ગણતરી કરો છો?
(સ્લાઈડસમયનો સંદર્ભ લો: 05:15)
અને બીજી સમસ્યા કેસોની કોઈપણ જોડી અથવા શિર્ષકોની વચ્ચેની સૌથી ટૂંકી અંતરની ગણતરી કરવી. તેથી, જો તમારી પાસે એરલાઈન અથવા ટ્રેન નેટવર્કનો રૂટીંગ નેટવર્ક હોય તો, આ એક કુદરતી સમસ્યા હશે. હવે, જ્યારે તમે કોઈ શહેરમાંથી કોઈ શહેરમાં મુસાફરી કરવા માંગો છો, તો તમે A અને B અથવા ઉદાહરણ વચ્ચે ટૂંકા પાથની ગણતરી કરવા માંગો છો, જો તમે ગૂગલ નકશા જેવી સેવાનો ઉપયોગ કરો છો અને તમે કહો છો કે તમે કોઈ સ્રોતથી લઈ જવા માંગતા હો ગંતવ્ય, તે ટૂંકા પાથની ગણતરી કરવાનો અને તમને વૉકિંગ અથવા ડ્રાઇવિંગના સંદર્ભમાં સમય આપે છે, વગેરે. તેથી, અમારી પાસે બે અલગ અલગ પ્રકારની સમસ્યાઓ છે, એકમાં એક જ સ્રોત હોઈ શકે છે અને તે સ્રોત અને અન્ય સંસ્કરણ વચ્ચેના દરેક ગંતવ્ય માટેનો સૌથી નાનો માર્ગ છે જ્યાં આપણે શિરચ્છેદના દરેક જોડી વચ્ચે ટૂંકા પાથ શોધવા માંગીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:00)
તેથી, આપણે આ એકલ સ્રોતની ટૂંકી પાથ સમસ્યાને જોઈને પ્રારંભ કરીશું. તેથી, આપણે ડિઝાઈન કરવું પડશે, આપણે આ પ્રકારની સમસ્યામાં નિયુક્ત કરવું પડશે, પ્રારંભિક શ્લોક શું છે. તેથી, કદાચ આપણું પ્રારંભિક વાક્ય એ છે કે આપણે ધારી લઈએ છીએ, તે શિરચ્છેદ છે. યાદ રાખો કે આપણે સામાન્ય રીતે 1, 2, 3 ને n એ ઉપર આપીએ છીએ. તેથી, અહીં આપણી પાસે 7 શિરોલંબવાળા ગ્રાફ છે અને ધારની કિનારીઓ લાલ રંગની બાજુએ લખેલી છે અને આપણે 1 થી પ્રારંભ

કરવા અને શોધવા માંગીએ છીએ કે, ટૂંકી સંભવિત કિંમતમાં દરેક અન્ય શૂન્યથી 1 થી કેવી રીતે મેળવી શકાય છે અને આ ગણતરી કરવા માટે વ્યવસ્થિત માર્ગ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:32)
તો અહીં એક સમાનતા છે જે અલ્ગોરિધમનો સમજાવે છે. તેથી, ચાલો ધારીએ કે દરેક શિરોબિંદુ એક ઓઈલ ડિપોટ છે અને તમામ ધાર પાઈપલાઈન્સ છે અને પાઈપલાઈન્સ પાઈપલાઈનની લંબાઈની કિંમત છે. હવે, જ્યારે આપણે આ મૂળ શિરચ્છેદમાં આગ લગાવીશું, ત્યારે શ્વેત 1 શરૂ કરો અને શંકુ 1 થી જોડાયેલ તમામ પાઈપ પર આગ લાગી જશે. ધારી રહ્યા છીએ કે પાઈપ એક સમાન ગતિએ મુસાફરી કરે છે, તે સૌથી ટૂંકી, નજીકના શ્વેત ભાગમાં પહોંચશે. તેથી, પ્રથમ શ્વેત પ્રથમ ઓઈલ ડિપોટ કે જે પ્રથમ શ્વેત પછી આગને પકડી લે છે તે નજીકના શિરચ્છેદ છે. પછી, બીજા ઓઈલ ડિપોટ જે આગને પકડી રાખે છે તે બીજા નજીકના શિરોબિંદુ પાથ છે અને ધારી રહ્યું છે કે આ આગ સમાન ગતિએ મુસાફરી કરી રહી છે. જે ઝડપે આગ દરેક શિખર પર પહોંચે છે, તે ખરેખર ટૂંકા માર્ગને શાંત કરવામાં આવશે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:31)
તેથી, ચાલો આપણે આ વિશિષ્ટતાને આ વિશિષ્ટ રીતે અમલમાં મૂકવાનો પ્રયાસ કરીએ, તેથી આપણે આ સ્ત્રોત શૂન્યમાં 0 સેટ પર આગ સેટ કરવાનું શરૂ કરીએ અને હવે આગ 1 ની સાથે આગ ચાલુ થઈ જશે, 3 અને ધાર 1,2. હવે, કારણ કે ધાર 1, 2 ટૂંકા છે, 10 યુનિટ ટાઈમ વર્ટેક્સ 2 બર્ન કરશે. આ બિંદુએ, આપણે એ હકીકત દ્વારા સૂચવ્યું છે કે અહીં એક નાનો બર્ન ભાગ છે, ત્યાં આંશિક આગ છે જે 1 થી 3 સુધી જાય છે, પરંતુ તે માત્ર 80 એકમોમાંથી 10 મુસાફરી કરે છે, તે જે રીતે છે તે માત્ર આઠમા ભાગ છે. 1 થી 3 ની મુસાફરી કરી. તેથી, ટી પર 10 વર્ટેક્સ 2 બર્ન થાય છે અને આપણે કહી શકીએ કે આ 2 ભાગથી ટૂંકી અંતરની સૌથી ટૂંકી કિંમત છે. હવે, આગ 2 થી 2 દિશાઓમાં ફેલાયેલી છે, 2 થી 3 અને 2 થી 5 સુધી અને આપણે ચોક્કસપણે, 1 થી 3 ની આગ પણ તે જ દર કરતા પહેલાં સતત ચાલુ રહેશે. હવે, આપણે જોઈ શકીએ છીએ કે 6 યુનિટ પછી, વર્ટેક્સ 3 બર્ન થશે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:34)
તેથી, 16 વર્ટેક્સ 3 બળી શકે છે, હવે આમાં 1 થી 3 ની આગ છે જે 80 થી 16 ની મુસાફરી કરે છે તે અંતર છે, પરંતુ 2 થી 5 ની આગ દ્વારા 6 મુસાફરી કરી છે. 20 આનો અર્થ છે. હવે, હેવન બળીને 3 અને આ દિશામાં નવી આગ ચાલુ રહેશે, આ જૂની આગ પહેલાથી અને અલબત્ત 1 થી 3 ની સતત ચાલુ છે, આગ 2 થી 5 નો અર્થ પહેલાનો છે. તેથી, હવે, આપણે જોવું જોઈએ કે, આમાંથી કયુ પહોંચશે તે પ્રથમ લક્ષ્ય છે. તેથી, આપણે જોઈ શકીએ છીએ કે 2 થી 5 ની આગ 14 એકમોમાં છે; તમે 5 સુધી પહોંચશો જે સમયે તે ન 3 અથવા 4 રસ્તો નથી. તેથી, જો તમે અન્ય 14 અને અન્ય 14 એકમના સમયને ટી 30 થી ચાલુ રાખો છો, તો આપણને તે કર્યું 5 બળે મળે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:15)
અને અહીં આપણે 3 થી 4 માંથી 70 માંથી 14 કર્યું છે અને આપણે 80 માંથી 30 બરાબર 1 થી 3 સુધી કર્યું છે. તેથી, આ આગ સ્રોતમાંથી હજી પણ ચાલુ છે. કિનારીનો પ્રારંભ બિંદુ ધારના અંતિમ બિંદુ સુધી છે. હવે, આ રીતે ચાલુ રાખીને, અગ્નિ હવે 5 થી 6 અને 7 ની વચ્ચે પ્રચાર કરે છે અને આપણે જોઈ શકીએ છીએ કે 10 એકમોમાં, આપણે કર્યું 7 બળતણ શોધીશું.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:42)
અને હવે, 7 થી નવી આગ 6 ની તરફ શરૂ થાય છે, તે જૂની આગ 5 થી 6 સુધી આવે છે, પરંતુ તે આગળ નીકળી જશે, કારણ કે આ માત્ર 10 દ્વારા 50 થાય છે. તેથી, ત્યાં હજી પણ છે 5 માંથી 6 સુધી પહોંચે તે પહેલાં 40 યુનિટનો સમય જાય છે, પરંતુ 5 યુનિટમાં, તે 7 થી પહોંચશે, તેથી 45 ની બરાબર 45 ની બરાબર, તમને તે કળણ 6 બર્ન મળે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:02)
અને હવે ફક્ત એક જ શિરોબિંદુ છે, જે
(સમયનોસંદર્ભ લો: 10:08)
બર્ન કરવા માટે, ખરેખર 3 અને 4 છે. તેથી, આ બિંદુએ આપણે ચાલુ રાખીએ છીએ, આપણને તે બરાબર મળશે 86, ખરેખર માફ કરજો, કર્ટેક્સ 3 એ પહેલાથી જ બાળી દેવામાં આવી છે, તે આગ 3 ની પહોંચ સુધી પહોંચતી નથી. પણ,

ફક્ત બર્ન કરવા માટે બાકી રહેલા તમારી જ માત્રા 4 હતી અને તે સમયે તે 86 છે, આગ પછી આગ પછી 70 એકમો છે. ધારની ભીખ, તે છેલ્લે પહોંચે છે અને હવે બધું બળી ગયું છે. હવે, પ્રારંભિક અગ્નિ સુધી પહોંચવા માટેના દરેક ટૂંકા સમય દ્વારા અમે દરેક શિરોબિંદુને લેબલ કરીએ છીએ અને આ, અમે તે પ્રારંભિક શિર્ષક પાથ પરથી શૂન્ય સુધી પહોંચવાનો સૌથી ટૂંકા માર્ગનો દાવો કરીએ છીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:50)
તેથી, ચાલો જોઈએ, આપણે આ બર્નિંગ પ્રક્રિયાને ખરેખર કેવી રીતે ગણતરી કરીશું જે આપણે વર્ણન કરીએ છીએ. તો, આપણે શું કરીએ છીએ તે આપણે સમયને ટૂંકા અંતર સુધી જોડીએ છીએ અને દરેક શિરચ્છેદ સાથે જથ્થો છે અને શરૂઆતમાં, આપણે કંઈપણ જાણીએ છીએ. તેથી, આપણે માનીએ છીએ કે કર્ણબિંદુ પહોંચી શકાય તેવું નથી, તે દરેક શિરોબિંદુની સૌથી ટૂંકી કિંમત અનંત છે. હવે, આપણે જાણીએ છીએ કે કર્ણલેખ 0, શ્વેત 1, શરુઆતનો શ્વેત કોઈ પણ સમયે પહોંચી શકાય તેમ નથી. તેથી, અમે તેની કિંમત 0 હોવાનો નિર્દેશ આપ્યો છે અને તેને 0 હોવાનો ખર્ચ આપ્યો છે, અમે હવે ફરી વિવાદિત કરી શકીએ છીએ, તે ઓછામાં ઓછા ખર્ચની તુલનામાં પાડોશીઓનો ખર્ચ છે અને તમે પહેલેથી જ ખર્ચ કરી શકો છો
(સમયનોસંદર્ભ લો)
11:24).
તેથી, 1 થી 3 સુધી, આપણે જઈ શકીએ છીએ અને સમય 80 આપણે જાણીએ છીએ, આપણે માનીએ છીએ કે હમણાં અનંત છે. તેથી, આપણે જાણીએ છીએ કે આપણે 80 સુધી અનંત સુધારણા કરી શકીએ છીએ. એ જ રીતે આપણે 2 થી 10 ની અનંતતા સુધારી શકીએ છીએ, તેથી આ પ્રક્રિયામાં, આપણને બે અપડેટ્સ મળે છે, હવે બીજા બધા શિરોબિંદુઓ, આપણે હજુ પણ નથી જાણતા કે કેવી રીતે પહોંચવું. તેથી, તેઓ બદલાશે નહીં. હવે, આ 2 ની વચ્ચે, જો 10 એ 80 કરતા નાના હોય, તો 10 હવે તેઓ તેમના માટે ટૂંકા ભાગનો ભાગ બની શકશે નહીં, તેથી અમે કહીએ કે 10 બળી ગયા છે અને અમે તેને પાડોશી ફરીથી અપડેટ કરીશું. તેથી, આપણે કહીએ છીએ કે સમયાંતરે વૅટેક્સ 2 બર્ન થાય છે 10. હવે, કારણ કે તે સમય 10 પર બર્ન કરે છે, અમે કિંમત 10 ને 6 વત્તા 6, 16 અથવા 18 ની ન્યુનતમ તરીકે સુધારી શકીએ છીએ, જે 16 છે અને હવે, આપણે 5 ફરીથી સેટ કરી શકીએ છીએ અનંતથી 10 વત્તા 20 એ 30 છે. તેથી, આપણે આ રીતે ચાલુ રાખીએ છીએ. હવે આ વસ્તુનો ઉપયોગ કરીને હવે જે બર્ન કરવા જઈ રહ્યા છે, તે 16 અને 5 માં બર્ન થઈ રહ્યું છે, તે 30 બર્ન થશે, તે સ્પષ્ટ છે કે 3 પછી બર્ન થશે. તો, આપણે ફરીથી સેટ કરીએ છીએ કે તે બર્ન કરવાની સ્થિતિ છે અને આપણે તેને પાડોશી એટલે કે 4 થી 16 વત્તા 70 એ 86 છે. હવે, જે લોકો બળી ગયા નથી, એટલે કે 4 અને 5, જે મર્યાદિત મૂલ્ય ધરાવે છે, 5 પછીથી બર્ન થશે. તેથી, અમે તેની સ્થિતિ બર્ન કરી દીધી છે અને અમે અપડેટ કરીએ છીએ કે તે 80 અને 40 ના પાડોશીઓ છે, 6 મા છે, આપણી પાસે 80 છે અને 7 છે, આપણી પાસે 40 છે, હવે 7 બર્ન છે. તેથી, હવે, આપણે 7 નીકળતી કિંમત જોઈશું અને 86 થી 45 ને બદલીશું. હવે, 45 પછી બર્ન થાય છે, 45 ની આગળ 6 બર્ન થાય છે, તેથી અમે તેને બાળી નાખીએ છીએ અને અંતે, તમે 4 બર્નને 86 માર્ક કરો.
(સંદર્ભઆપો સ્લાઈડ ટાઈમ: 13:00)
તેથી, આ ઔપચારિક રીતે ઍલ્ગોરિધમ તરીકે લખવા માટે, અમે આ માહિતીને બળીને જાળવી રાખીએ છીએ, પરંતુ શિરોબિંદુઓ અને સમય, તે બે અલગ અલગ ઍરેમાં એક શિરોબિંદુને બાળવા માટે લે છે. અમારી પાસે બર્નિયન ઍરેઝ નામનું ઍરે છે જે બુલિયન વર્ટેક્સ બુલિયન ઍરે છે. તેથી, તે આપણને કહે છે કે કોઈ શિખરને શિરોબિંદુ બાળી દેવામાં આવ્યા છે અને પછી, અમારી પાસે અપેક્ષિત બર્ન ટાઈમ ઍરે છે, જે આપણને તે સમય કહે છે કે જેના પર શંકુદ્રુપ બાળી શકાય છે. તેથી, શરૂઆતમાં, આપણે કહ્યું હતું કે કોઈ શંકુદ્રવ્ય બર્ન નથી, તેથી અમે કહ્યું છે કે બધા શિરોબિંદુઓએ શૂન્ય મૂલ્યને બાળી નાખ્યું છે અને શરૂઆતમાં, અમે દરેક કર્ણ માટે અનંત સુધી અપેક્ષિત સમય જણાવ્યું હતું. હવે, અલબત્ત અનંત, એક ખ્યાલ છે જેનો અંદાજ કાઢવો મુશ્કેલ છે, પરંતુ આપણે સરળતાથી તપાસ કરી શકીએ છીએ કે કોઈ અંતર કોઈ અંતર પર હોઈ શકે નહીં, જે ગ્રાફ વત્તા પ્લસમાંના બધા ખર્ચની તુલનામાં વધારે છે. કારણ કે, તમે મોટાભાગે વાસ્તવમાં તે ટૂંકા માર્ગનો પણ ઉપયોગ કરી શકે છે જે વાસ્તવમાં ઓછા ધાર પણ લે છે, પરંતુ મોટાભાગે, તમે દરેક ધારનો ઉપયોગ કરી શકો છો, એજમાં બે વાર ઉપયોગમાં લેવાનો કોઈ મુદ્દો નથી, મોટા ભાગે તમે એકવાર દરેક ધારનો ઉપયોગ કરી શકો છો.

તેથી, જો તમે ગ્રાફમાં તમામ ખર્ચ ઉમેરો અને તેમાં એક ઉમેરો અને પછી, આને ચોક્કસપણે અનંત તરીકે ગણવામાં આવે છે, આ ખર્ચ કરતા કોઈ વાસ્તવિક કિંમત મોટી હોઈ શકે નહીં. તેથી, આપણે અનંતતાનો ઉપયોગ કરી શકીએ છીએ, પરંતુ તેઓ અનંત છે, ત્યાં એક નક્કર વ્યૂહરચના છે, તે અનંત સાથે અસાઈન કરવામાં આવી હતી. હવે, આપણે પ્રારંભ સ્રોત વર્ટેક્સથી પ્રારંભ કરીએ છીએ જે આપણે માનીએ છીએ કે 1 માં 6 કહે છે કે તે 1 સુધીનો સમય બર્ન કરે છે. અને હવે, આપણે વારંવાર અપેક્ષિત બર્ન ટાઈમ પર ધ્યાન આપીએ છીએ, તેને આગળ બાળીશું અને અપેક્ષિત બર્ન ટાઈમ ફરીથી પાડોશીઓ માટે ફરીથી સેટ કરીશું , આ પડોશના વર્તમાન બર્ન સમય અને આ લઘુતમ દ્વારા, તેમાંથી ઓછામાં ઓછું અને બર્ન ટાઈમના મૂલ્યના આધારે.

(સ્લાઈડસમયનો સંદર્ભ લો: 14:40)

તેથી, આ ખૂબ જ સરળ એલ્ગોરિધમ છે, તેથી અહીં તે સ્યુડો કોડમાં છે. તો, આપણી પાસે આ બે શિરોબિંદુઓ બાળી છે અને આપણી પાસે અપેક્ષિત બર્ન ટાઈમ એરે છે. તેથી, આપણે બળીને કાટખૂણે શરૂ કરીએ છીએ અને આશા રાખીએ છીએ કે શિરોબિંદુઓ માટે અનંત સુધી બર્ન સમય. હવે, આપણે કહ્યું છે કે સ્રોત વર્ટેક્સ 0 થી બર્ન કરવાનો સમય અપેક્ષિત છે, અમે નાનામાં ઓછા અનપ્લાઈડ વર્ટેક્સને પસંદ કરીએ છીએ. તેથી, અમે તમારા માટે શોધી રહ્યા છીએ, જે સળગાવી નથી અને બર્નનો સમય લઘુતમ કેવી રીતે લાગે છે, તે બાળી નાખે છે અને તેમાંથી પ્રત્યેક પાડોશીઓને જતા નથી, જે બળી નથી. જો તે માટેનો અંદાજ તમારા દ્વારા પસાર થવાના અંદાજ કરતા વધારે છે, તો તે તમારા માટેનો બર્ન ટાઈમ છે, વત્તા તમારી પાસેથી વી. પછી, તમે બર્ન ટાઈમને નવા વજન તરીકે અપડેટ કરો છો, તેથી વાસ્તવિક એલ્ગોરિધમ બર્ન ટાઈમમાં અને અપેક્ષિત બર્ન સમય કુદરતી અર્થઘટન છે. તેથી, બર્ન માત્ર મુલાકાત લીધી છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:35)

તેથી, જ્યારે આપણે મુલાકાતીઓ વિશે શ્વેત હોય છે, તેથી અમે તેને બાળી નાખીએ છીએ અને અંતર એ અપેક્ષિત એક જ સમય છે. તેથી, આપણે આ જ એલ્ગોરિધમને ફરીથી લખી શકીએ છીએ, તે ફક્ત બી.વી.ની મુલાકાત લઈને બદલવામાં આવી છે અને ઈબીટી અંતર દ્વારા આવશ્યક છે. તેથી, આપણે અનંત સુધી ખોટી અંતરની મુલાકાત લીધી, પ્રારંભિક અંતરથી શરૂ કરીને આ શરુઆતની શરુઆત 0 છે. અને તે પછી, તે વારંવાર દરેક અવ્યવસ્થિત શિરચ્છેદ માટે થશે, તેથી આ લઘુતમ અંતર, તેને મુલાકાત લેવા અને અપડેટ કરવા માટે સેટ કરો. સિંગલ સ્ત્રોત ટૂંકા પાથ માટે આ ડિજસ્ટ્રાનો(Dijkstra's) અલ્ગોરિધમ છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક- 04
મૉડ્યૂલ - 02
લેક્ચર – 26

તેથી, ચાલો હવે સિંગલ સ્ત્રોત ટૂંકા પાથ સમસ્યા માટે ડિજસ્ટ્રાના(Dijkstra's) અલ્ગોરિધમનો વિશ્લેષણ કરીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:06)
તેથી, યાદ રાખો કે ડિજસ્ટ્રા(Dijkstra)ના એલ્ગોરિધમ્સ અમે ઉપયોગમાં લેવાયેલી સમાનતામાં શિરોબિંદુને સંચાલિત કરીને કાર્ય કરે છે. તેથી, આપણે જે શિરોબિંદુઓને બાળી નાખ્યો છે અથવા
((સમયનોસંદર્ભ: 00:17)
ટ્રેક રાખીએ છીએ)
શરૂઆતમાં કશું જ મુલાકાત લેવામાં આવતી નથી અને શરૂઆતમાં આપણે કોઈ પણ શિરોબિંદુમાં કોઈ અંતર જાણતા નથી. તેથી, આપણે અંતરની બધી અંતર ધારી લીધી છે. જ્યારે આપણે સ્રોત વર્ટેક્સ પર પ્રારંભ કરીએ, તો તેને ધારણા દ્વારા 1 ને કૉલ કરીએ. તેથી, આપણે તેની અંતર 0 સુધી સુયોજિત કરીએ છીએ. પછી, આપણે વારંવાર જે પહેલું કર્ણ નથી બાળી નાખ્યું તે બગાડે છે અને જેની પાસે બળી શકાતી નથી તે વચ્ચે ઓછામાં ઓછી અંતર છે, મુલાકાત લીધી છે અને તેના બધા પડોશીઓને અંતર પાછું ફેરવી છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:45)
તેથી, આપણે આ અલ્ગોરિધમ્સની જટિલતાને જુએ તે પહેલા, આપણે ખરેખર પહેલા એ ચકાસવું પડશે કે તે સાચું છે. તેથી, ડિજસ્ટ્રા(Dijkstra)ના એલ્ગોરિધમ્સ હવે અપડેટ્સના આ અનુક્રમ બનાવે છે. તે બીએફ્એસ(BFS) અને ડીએફ્એસ(DFS) માટે થોડું વિશ્લેષણ છે, કારણ કે તે શિર્ષકોમાં મુલાકાત રાખે છે અને તે ફક્ત એક જ વાર દરેક શિર્ષકની મુલાકાત લે છે. હવે, તે કોઈ ચોક્કસ ક્રમમાં શિર્ષકોની મુલાકાત લે છે જે પહેલા પહોળાઈથી પહેલા અથવા ઊંડાઈથી જુદા છે, અને આપણે વાજબી ઠરે છે કે આ ઓર્ડર ખરેખર સાચો છે. તેથી, દરેક તબક્કે, દરેક બિંદુએ, પહોળાઈ પ્રથમ શોધ તેના બધા પાડોશીઓ જુએ છે અને તેમના વર્ટેક્સ નંબરના ક્રમમાં તેમની મુલાકાત લે છે. પ્રથમ ઊંડાણ પ્રથમ પાડોશીને પસંદ કરશે, અને પછી તેને વધુ અન્વેષણ કરશે. હવે, ડિજસ્ટ્રા(Dijkstra) એલ્ગોરિધમ જુદી જુદી વ્યૂહરચનાનો ઉપયોગ કરે છે જે પ્રથમ, સૌથી નાનું અંતર અને મુલાકાત પાડોશીને પસંદ કરવાનું છે. તેથી, આપણે આ વિકલ્પ પસંદ કરવો પડશે કે જે ક્યારેય નીચે નથી, તે આગળ વધવાનું ચાલુ રાખશે, અને તમે ક્યારેય પાછા નહીં જાઓ અને આશા કરશો કે કદાચ મને એક અલગ પસંદ કરવો જોઈએ કે આ પ્રકારની વ્યૂહરચના ખરેખર સાચી છે. તેથી, આ લોભી એલ્ગોરિધમ્સ નામની એલ્ગોરિધમ્સનું સામાન્ય વર્ગ છે જ્યાં તમારી પાસે સંખ્યાબંધ શક્ય ટ્રજેક્ટરીઝ છે જે તમે સમસ્યાનો ઉકેલ લાવવા માટે પસંદ કરી શકો છો. દરેક તબક્કે તમે કેટલીક સ્થાનિક માહિતીના આધારે આગલી પસંદગી કરો છો. આ બિંદુએ, આ બનાવવા માટેની શ્રેષ્ઠ પસંદગી જેવી લાગે છે અને પછી કોઈક રીતે જાદુઈ રીતે તમે સ્થાનિક માહિતી પર આ શ્રેષ્ઠ પસંદગી કરો છો તે વૈશ્વિક સ્તરે શ્રેષ્ઠ માર્ગ લેવાનું ચાલુ કરે છે. તેથી, આવા લોભી એલ્ગોરિધમ માટે, તે સ્થાપિત કરવું મહત્વપૂર્ણ છે કે આ આગલા પગલાની સ્થાનિક પસંદગી વાસ્તવમાં અમને વૈશ્વિક ઈષ્ટતમ આપે છે, કારણ કે ઘણીવાર આ પ્રકારની સ્થાનિક ન્યાયશાસ્ત્ર આપણને સાચો મૂલ્ય આપતું નથી.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:20)
તેથી, ચાલો સૌ પ્રથમ ડિજસ્ટ્રા(Dijkstra) એલ્ગોરિધમનો ચોકસાઈ જુઓ. તેથી, આ ચોક્કસ કિસ્સામાં ચોકસાઈ સ્થાપિત કરવાની ચાવી એ છે જે ઈન્વેરીઅંટ(Invarient) કહેવામાં આવે છે, જમણી. તેથી, હવે, ડિજસ્ટ્રા(Dijkstra) એલ્ગોરિધમ તમે પુનરાવર્તનના અનુક્રમમાં જોયું છે તે રીતે પ્રાપ્ત થાય છે. તેઓ પુનરાવર્તન છે. ઈન્વેરીઅંટ(Invarient) એ છે કે દરેક પુનરાવર્તન વખતે બર્ન અને અનબર્સ્ટ જેવા શિરોલંબનો આ શુદ્ધતા છે. તમે જેનો દાવો કરવા માંગો છો તે

છે કે બર્ન શિરોબિંદુઓ યોગ્ય રીતે હલ કરવામાં આવે છે, જો કોઈ પણ બિંદુએ હોય તો જો આપણે શિરોબિંદુઓને ગોઠવેલા અંતર તરફ ધ્યાન આપીએ, તો તે અંતર વાસ્તવમાં સ્રોત શિરોબિંદુઓમાં તે સૌથી નાનો અંતર છે જે તે જળાશયમાં છે . તેથી, જો આપણે માનીએ છીએ કે આ ઈન્ટરેક્ટિવ ઈન્વેન્ટિઅન્ટ સાચું છે, તો તે શરૂઆતમાં ચોક્કસપણે સાચું છે કારણ કે શરૂઆતમાં, ફક્ત એક જ શિરોબિંદુ જે આપણે બાળી દીધી છે તે પ્રારંભિક શ્વેત છે. તેથી, આપણે જે પહેલું કર્ણ બાળ્યું છે તે છે અને આપણે કહ્યું છે કે તે 0 થી અંતર છે. તેથી, ચોક્કસપણે તે સમયે આ ગુણધર્મ સાચું છે કે બર્ન શિરોબિંદુઓ વચ્ચે, અંતર અથવા હકીકતમાં સૌથી ટૂંકી અંતર. હવે, ધારે છે કે આપણે તેને થોડા શિરોબિંદુઓ માટે લંબાવ્યું છે, હવે આપણે એક નવું ઉમેરવા માંગીએ છીએ. તેથી, આપણે જે કહ્યું છે તે છે કે આપણે વેર્ટેક્સ v પસંદ કરીશું, જેમ કે જો આપણે x ના અંતર અને xv ના વજનને જોવું જોઈએ, જો તમે આ કુલ સરવાળાને જુઓ છો, જે V ના અંતરને અપડેટ કરવામાં આવશે, તો તે બધામાં સૌથી નાનો હશે શિરોબિંદુ જે બળી નથી. તેથી, દાવો એ છે કે જો આપણે આને આપણા બર્ન સેટમાં ઉમેરીશું, તો જમણી બાજુએ આપણે વી શામેલ કરીને આ રીતે બર્ન સેટ કરીએ છીએ, તો, વીનો અંત હવે આપણે જે ગણ્યો છે તેની તુલનામાં ખરેખર નાના હોઈ શકતા નથી અને અમે બદલી શક્યા નથી તે પછીથી અપડેટ. તેથી, તમે તેને કેવી રીતે બદલ્યું કારણ કે પછીનાં અપડેટમાં, તે પછીથી તે પછીનું કારણ હોઈ શકે છે કે આપણે ખરેખર બર્ન સેટને વધારીએ છીએ અને એક નવું શામેલ કરીએ છીએ. હવે, તે હોઈ શકે છે કે y થી w થી v નો રસ્તો છે જે પાથ કરતાં ટૂંકા છે જે આપણે હમણાં જ કુહાડીથી v દ્વારા દાવો કર્યો છે. તેથી, એ જોવાનું સરળ છે કે y ની અંતર અને wxy ની કિંમત. હવે, આ પહેલાની વસ્તુ કરતા મોટું અથવા તેના બરાબર હોવું જોઈએ કારણ કે આપણે v પસંદ કર્યું છે અને w નથી, કારણ કે આ બિંદુએ w સૌથી નાનું હતું, v હતી. કદાચ તેઓ સમાન હતા, પરંતુ ચોક્કસપણે ડબલ્યુ કરતા ઓછું નહોતું. તેથી, પછીના તબક્કે જો તમે ડબ્લ્યુએ, તો આપણે જાણીએ છીએ કે આપણે ઓછામાં ઓછું પહોંચવા માટે ઓછામાં ઓછો સમય લઈશું. પછી, આપણે ડબ્લ્યુ થી વિરુદ્ધ જવાની કિંમત વધારવી પડશે. તેથી, ત્યાંનો કોઈ રસ્તો નથી કે જો આપણે શોધી શકીએ કે y થી v ની વિરુદ્ધ છે, તો પછી યોગ્ય રસ્તો કરતાં વધુ યોગ્ય હોઈ શકે છે. તેથી, આ ઈન્વેરીઅંટને ફરી સ્થાપિત કરી રહ્યું છે જે આપણે આગળ બાળી નાખેલું છે તે યોગ્ય રીતે ઉકેલાઈ ગયું છે. તેથી, આ બતાવવાનો એક રસ્તો છે કે ડિજ્સ્ટ્રા(Dijkstra) લોભી વ્યૂહરચના વાસ્તવમાં આ સમસ્યાને યોગ્ય રીતે ઉકેલે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:01)

તેથી, હવે આપણે સાબિત કર્યું છે કે તે સાચું છે, ચાલો જટિલતાને જોઈએ. તેથી, તેમાં કેટલાક સ્પષ્ટ આંટીઓ છે. તેથી, ત્યાં પ્રથમ છે, પછી મુલાકાત લેવાયેલા મૂલ્યોને ખોટી રીતે પ્રારંભ કરવા અને અનંત સુધીના ઉદાહરણોની ઓર્ડર એન લૂપ છે. હવે, અહીં બીજું ઓર્ડર n લૂપ છે અને આ ઓર્ડર n લૂપ અંદર, બે લૂપ્સ છે. એક અન્ય કરતાં વધુ સ્પષ્ટ છે. તેથી, અહીં એકવાર આપણે x અને નવી શિખર વિશે મુલાકાત લીધી છે, પછી આપણે તેના તમામ આઉટગોઈંગ પડોશીઓમાંથી પસાર થવું પડશે. ત્યાં સ્કેનએફ બધી ધાર છે જે q માંથી બહાર આવી રહી છે. તો, તે એક લૂપ છે, પણ અહીં એક લૂપ પણ છે જે આપણે એક કર્ણકની મુલાકાત લે તે પહેલાં તે અસ્પષ્ટ છે, આપણે તેને પસંદ કરવું પડશે. તેથી, આપણે બધા અવલોકનિત શિરોબિંદુઓમાંથી પસાર થવું જોઈએ અને જેની અંતર ઓછામાં ઓછી છે તે પસંદ કરવું પડશે. તેથી, આ સામાન્ય રીતે આપણે જોયું છે કે જો તમારી પાસે કોઈ સૂચિ અથવા એરે છે જે કોઈ ચોક્કસ ક્રમમાં નથી, તો અમારે ન્યૂનતમ શોધવા અને સમગ્ર એરેને સ્કેન કરવું પડશે. તેથી, આ પૂર્ણપણે એક ઓર્ડર n ચોરસ છે. તેથી, આ એક ક્રમ n પગલું છે અને તે તે સમય અહીં લે છે, તે એ ધારણા કરે છે કે આપણે ધારને કેવી રીતે રજૂ કરીએ છીએ.

(ટાઈમસ્લાઈડનો સંદર્ભ લો: 06:10)

તેથી, બાહ્ય લૂપ રનનો સમય જે આપણે જોયો. દરેક પુનરાવર્તન માં, અમે એક શિરચ્છેદ બળી. આને બર્ન કરવા માટે ન્યૂનતમ શિરોબિંદુ શોધવા માટે ઑર્ડર n સ્કેનની આવશ્યકતા છે. અમે શિરચ્છેદને બાળી નાખ્યા પછી, તે શિરોબિંદુઓના બળી જવાના સમયને અપડેટ કરવા માટે આપણે તેના પાડોશીઓને સ્કેન કરવું પડશે. હવે, જો આપણે જોયું છે કે અડજાનસી (adjacency) મેટ્રિક્સ હોય તો, હવે તમે તેને બાળી નાખશો, તો હવે આપણે તેનો નવો પાડોશી શોધીશું. અમને તમારી નજીકની મેટ્રિક્સમાં પંક્તિ માટે સ્કેન કરવી પડશે. તેથી, આ ક્રમમાં એન સમય લેશે, જમાણે. તેથી, આપણે બાહ્ય ક્રમમાં છીએ અને અંદરની બાજુમાં આપણી પાસે બે ઓર્ડર n વસ્તુઓ એકબીજાથી સ્વતંત્ર છે. એકમાં ન્યૂનતમ બળી ગયેલી શંકુ, અગ્નિની અસ્થિરતાને આગળ બાળવા માટે, અને બીજું, તેના બધા પડોશીઓને અપડેટ કરવાનું છે.

તેથી, કારણ કે આપણી પાસે ઓર્ડર n લૂપ અંદર આ ઓર્ડર n વસ્તુ છે, એકંદરે તેનું ઓર્ડર n ચોરસ. તો, હવે નીચે આપેલા x માંની એક ચોક્કસપણે આ અડજાનસી (adjacency)ને મેટ્રિક્સ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:00)
તેથી, આપણે bfs અને dfs માં જોયું છે કે જો આપણે અડજાનસી (adjacency)ને મેટ્રિક્સમાંથી આગળ વધીએ, જ્યારે અડજાનસી (adjacency)નેવિશિષ્ટ શિરોબિંદુના બધા પડોશીઓના શિરોબિંદુઓને સ્કેનીંગ કરવાની સૂચિ વધુ કાર્યક્ષમ બને છે. તો, આપણે આ જ વસ્તુ અહીં કરી શકીએ, જમણે. તેથી, જો આપણે એડીસેન્સી મેટ્રિક્સમાંથી અડજાનસી (adjacency)ને સૂચિ રજૂઆતમાંથી ખસેડવું, તો દરેક પુનરાવર્તનની અંદર બીજું ઓર્ડર એન લૂપ હવે એકંદર ઓર્ડર એન ખર્ચ તરીકે ગણવામાં આવે છે કારણ કે તમામ પુનરાવર્તનોમાં, અમે ફક્ત એક જ વાર દરેક ધારની શોધ કરીશું કારણ કે જ્યારે આપણે અન્વેષણ કરીએ છીએ અમે તેના સ્રોત વર્ટેક્સ બર્ન, અધિકાર. તેથી, તેથી, ઑર્ડર n નો બીજો યોગદાન અડજાનસી (adjacency)ને સૂચિનો ઉપયોગ કરીને હલ કરવામાં આવે છે, પરંતુ કમનસીબે અમારી પાસે હજી પણ પ્રથમ ક્રમમાં n પગલું છે જે ન્યૂનતમ શોધવાનું છે. તેથી, અમારી પાસે અદ્રશ્ય અવ્યવસ્થિત શિરોબિંદુઓ સાથે સંકળાયેલ બળી સમયની સૂચિ છે, અને અમને તેમની વચ્ચે ન્યૂનતમ શોધવાની જરૂર છે અને આ ક્રમ n પગલું છે. તેથી, એકંદરે, જો આપણે અડજાનસી (adjacency)ને સૂચિમાં ખસેડ્યા છે, તો આ એકલા અમને મદદ કરશે નહીં કારણ કે હજી પણ મોટા લૂપની અંદર ક્રમ n પગલું છે. તેથી, અમે હજી પણ ક્રમમાં ચોરસ છે.
(ટાઈમસ્લાઈડનો સંદર્ભ લો: 08:05)
તેથી, આ બોટલ ગરદનની આસપાસ જવા માટે સક્રિય કરો, આપણે બર્ન ગાળાને સૌથી આધુનિક ડેટા માળખામાં જાળવવાની જરૂર છે. તેથી, તે ડેટાને બંધારણની જરૂર છે કારણ કે આપણે નિશ્ચિતપણે ન્યૂનતમ ઘટકને શોધી અને દૂર કરી શકીએ છીએ, પરંતુ એકવાર તે આપણી પાસે છે, તે પછી આપણે મૂલ્યોને ઝડપથી અપડેટ કરવામાં સમર્થ હોવા જોઈએ, જેથી સંપૂર્ણ અપડેટ અને એક્સ્ટ્રેક્ટિંગ બંને લગભગ સમાન રીતે સમાન. તેથી, આને ખાસ કરીને વૃક્ષ જેવા માળખાનો ઉપયોગ કરીને કરી શકાય છે, પછી આપણે આગળના ભાષણમાં જોશું કે હેપ નામની એક સરસ માહિતી માળખું છે જે અમને લોગ એન ટાઈમમાં આ બે ઓપરેશન્સ કરવા માટે ચોક્કસપણે પરવાનગી આપે છે. તેથી, જો તમારી પાસે n મૂલ્યો છે, તો તમે n એ લોગ એન ટાઈમ શોધી શકો છો અને ન્યૂનતમ મૂલ્યને વાંચી શકો છો અને અમે એક નવું મૂલ્ય શામેલ કરી શકીએ છીએ અથવા મૂલ્યને અપડેટ કરી શકીએ છીએ જે પહેલેથી જ ઢગલામાં છે. આ બધા ઑપરેશન ફક્ત લોગ એન ટાઈમ લે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:58)
તેથી, જો આ પછી આપણે જોશું અને જો આ કરી શકાય છે, તો તે શું કહે છે તે છે કે લઘુતમ બર્ન ટાઈમ શોધવાથી લોગ એન ટાઈમ થાય છે. હવે, જ્યારે આપણે શિરોલંબને અપડેટ કરીએ છીએ જે વર્તમાન બર્ન વર્ટેક્સની બાજુમાં છે, એકંદરે આપણે આ એમ એમ ટાઈમ્સ કરીએ છીએ, બરાબર, ઓ એમ ધારથી, પરંતુ દરેક અપડેટ્સ ફરીથી લોગ n લે છે. તો, આપણી જટિલતામાં હવે બે યોગદાન છે. તેથી, આ લઘુતમ શિરોબિંદુ પસંદ કરવાથી આવે છે. ન્યૂનતમ વર્ટેક્સને કાઢવા માટે, અમે આ એન ટાઈમ્સ કરીએ છીએ અને દર વખતે લોગ એન ટાઈમ પર પ્રક્રિયા કરીએ છીએ, અને પછી અંતરને અપડેટ કરવા માટે, આપણે એક વાર વર્ટીક્સ બર્ન કરીએ છીએ, દર વખતે તે લોગ એન ટાઈમ લે છે, પરંતુ અમે આ માટે કરીએ છીએ દરેક ધાર. તેથી, તે n લોગ n છે. તેથી, અમારી પાસે વૅટેક્સ પસંદ કરવા માટે n લોગ n છે અને આ અંતરને અપડેટ કરવા માટે છે, જમણે. તેથી, એકંદરે અલ્ગોરિધમ n પ્લસ એમ લોગ n બને છે. તેથી, યાદ રાખો કે ગ્રાફ n પ્લસ એમમાં ખરેખર ગ્રાફ એ કદ છે. તેથી, આ ખરેખર (()) On ચોરસની પોસ્ટ તરીકે લોગ n એલ્ગોરિધમ છે, અને અમે સૉર્ટિંગ અને અન્ય આવી સમસ્યાઓ જોઈ રહ્યા છીએ. તે એક વિશાળ વ્યવહારુ કૂદકો છે જે ઓ ના સ્ક્વેરથી n લોગ n સુધી જાય છે. સમસ્યાના કદમાં તે એક વિશાળ કૂદકો છે જે આપણે ઉકેલવાની આશા રાખી શકીએ છીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:13)
તેથી, ડિજ્સ્ટ્રા(Dijkstra) એલ્ગોરિધમ ખૂબ જ નિર્ણાયક ધારણા બનાવે છે, જે તેના લોભી પગલાની ચોકસાઈને આધારે છે, અને તે છે કે કુદરત સાથે સંકળાયેલ કોઈ નકારાત્મક ખર્ચ નથી, જમણે. અમારી દલીલ હતી કે જો આપણે અમારા બર્ન સેટમાં આગળના પાડોશી તરીકે ઉમેરવાનું પસંદ કરીએ, તો આપણે આ રીતે આવતો ટૂંકા માર્ગ શોધી શકીશું નહીં, પરંતુ

દેખીતી રીતે આપણી પાસે નકારાત્મક કિનારીઓ હશે, પછી આપણે એવી સ્થિતિમાં આવી શકીએ કે જ્યાંનો માર્ગ છે અહીં લંબાઈ, જમણી અને પછી મેં બીજા પાથને અનુસરવાનું પસંદ કર્યું નથી કારણ કે તે લંબાઈ 4 હોઈ શકે છે, પરંતુ જો હું ત્યાંથી પાછો આવીશ, તો કદાચ આ લંબાઈ 3 ની હશે. તેથી, જો હું આ રીતે આવીશ , પછી હું પ્લસ 1 નો ચોખ્ખો ખર્ચ કરું છું, જ્યારે હું આ રીતે જાઉં, તો મને પ્લસ 2 મળશે, જમણે. તેથી, તેથી, સ્થાનિક રૂપે નજીકના પાડોશીની શોધ કરીને, મને સૌથી ટૂંકી અંતરની જરૂર નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:15)

તેથી, અલબત્ત, તમે કદાચ પૂછો કે શા માટે તમે ગ્રાફમાં નકારાત્મક માર્ગો શા માટે મૂકવા માંગો છો. ઠીક છે, યાદ રાખો કે ગ્રાફ ખૂબ જ સામાન્ય મોડેલ છે. એવી ઘણી પરિસ્થિતિઓ છે જ્યાં તમે વાસ્તવમાં નકારાત્મક રીતે નકારાત્મક રીતે અર્થઘટન કરી શકો છો. તેથી, અમે વિચારીએ છીએ કે અમે ટેક્સી ડ્રાઈવર પર કહીએ છીએ અને અમે આલેખને સ્થાનોની સૂચિ તરીકે જોઈ રહ્યા છીએ, જ્યાં અમે લોકોને ચૂંટીએ છીએ અને લોકોને છોડીએ છીએ. હવે, દેખીતી રીતે એવા સેગમેન્ટો છે કે જે પેસેન્જર કારમાં છે અને ત્યાં એવા સેગમેન્ટ્સ છે જ્યાં અમારે પેસેન્જર ન હોવું જોઈએ, તે કદાચ કોઈ જગ્યાએ જવા માટે પાછો ફર્યો હોય. દાખલા તરીકે, એરપોર્ટ પરથી ટેક્સી ડ્રાઈવર ઓપરેશન સામાન્ય રીતે મફત સફર પછી એરપોર્ટ પર પાછો ફરે છે કારણ કે શહેરની અંદરથી ત્યાંથી લાંબા સમય સુધી ટ્રીપ્સ શોધવાની અપેક્ષા છે. તેથી, ત્યાં એવા સેગમેન્ટો છે કે જ્યાં તેઓ ખાલી મુસાફરી કરે છે, ત્યાં એવા સેગમેન્ટ્સ છે જ્યાં તેમની પાસે મુસાફરો છે. તેથી, કેટલાક ભાગો પૈસા કમાતા હોય છે, કેટલાક ભાગો પૈસાનો ઉપયોગ કરે છે. તેથી, ત્યાં કેટલાક હકારાત્મક ધાર છે અને કેટલાક નકારાત્મક કિનારીઓ છે, અને બીજી સ્થિતિ જે આપણે અત્યાર સુધી જોયેલી કંઈપણ સાથે સંપૂર્ણપણે સંબંધિત નથી. તમે રસાયણો, રાસાયણિક સંયોજનો વિશે વિચારી શકો છો અને અમે એક કમ્પાઉન્ડને અન્ય સંયોજનોમાં કેવી રીતે પરિવર્તિત કરીએ છીએ તે દર્શાવતા ગ્રાફને રજૂ કરી શકીએ છીએ અને અહીં ઉદાહરણ તરીકે, વય વજન એ ઊર્જાને રજૂ કરી શકે છે જે ક્યાં તો આ પ્રક્રિયામાં મુક્ત અથવા શોષાય છે. તેથી, ફરીથી તે હકારાત્મક અથવા નકારાત્મક હોઈ શકે છે, અધિકાર. તેથી, એવી ઘણી પરિસ્થિતિઓ છે જેમાં ધાર, નકારાત્મક ધાર જે વાસ્તવમાં અર્થપૂર્ણ બનાવે છે. તો, આપણે અહીં નકારાત્મક ધાર વજન શું કરીએ?

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:32)

તો, નોંધ લેવી પ્રથમ વસ્તુ કે જો તમારી પાસે નકારાત્મક ચક્ર હોય, તો જો મારી પાસે એક ગ્રાફ છે જે મેં આ પ્રકારના લૂપને કહ્યું છે અને બધી વસ્તુઓ માઈનસ 3, બાદબાકી 2 અને વત્તા ઉમેરો 1, તો જો હું એક વખત ચક્રની આસપાસ જાઉં, તો જમણે અને પાછો આવીશ, પછી માર્કસ 4 નો કુલ ખર્ચ થશે. તો, તેનો અર્થ એ કે જો મારી પાસે કોઈ અન્ય માર્ગ શરૂ થયો હોય તો અહીં આવ્યો અને બીજે ક્યાંક ચક્ર છોડી દીધો. ધારો કે અમારી પાસે સ્રોત અને લક્ષ્ય હતું, તો હું આ લુપને ઘણી વાર રાઉન્ડમાં ગોળ કરીને અને ગોપનીય રીતે ગોપનીય રીતે નાનાથી લક્ષ્ય સુધીના લક્ષ્યમાં અંતરને અંતર બનાવી શકું છું. દર વખતે હું લૂપ પર જાઉં છું, બાદબાકીની કિંમત ઘટાડે છે તેથી, હું મારી કિંમતમાં 4 ઓછા ઉમેરું છું. તેથી, હું તમારી આસપાસ કેટલી વાર જાઉં તે આધારે ખર્ચ એટલો ઓછો હશે. તેથી, જો મારી પાસે ગ્રાફમાં નકારાત્મક ચક્ર હોય, તો ટૂંકાતમ માર્ગનો પ્રશ્ન પણ કોઈ અર્થમાં નથી થતો. લઘુતમ પાથની કોઈ કલ્પના નથી, કારણ કે જથ્થો સારી રીતે વ્યાખ્યાયિત નથી, પરંતુ તે તારણ આપે છે કે જો હું આ ચક્રને નકારીશ, તો તે નકારાત્મક ધાર હોઈ શકે છે, પરંતુ નકારાત્મક ચક્ર નથી. તેથી, ઉદાહરણ તરીકે અહીં જો પ્લસ 1 ના બદલે, મેં કહ્યું કે આ વત્તા 7 હતું, બરાબર. જો મેં આ રીતે પ્લસ 7 બનાવ્યું છે અને ચક્રની આસપાસ જઈને મને પ્લસ 2 મળશે. તેથી, તે મને ચક્રની આસપાસ જવા માટે મદદ કરતું નથી કારણ કે તે ફક્ત મારા ખર્ચમાં ઉમેરે છે. તેથી, એવી પરિસ્થિતિમાં જ્યાં મારી પાસે કોઈ નકારાત્મક ચક્ર નથી, પણ મારી પાસે નકારાત્મક કિનારીઓ છે, તે હજી પણ ટૂંકા પાથની વાત કરવાની ભાવના ધરાવે છે, અને આપણે પછીથી જોશું કે ડિજસ્ટ્રા(Dijkstra)ના અલ્ગોરિધમનો સિવાય અન્ય એલ્ગોરિધમ્સ છે જે આને સંચાલિત કરી શકે છે. ખાસ કરીને અમે બેલમેન-ફોર્ડ એલ્ગોરિધમ(Bellman-Ford Algorithm)ને જોશું, અને આપણે એ પણ જોશું કે બધી જોડી ટૂંકા પાથ સમસ્યાઓ છે જે એક સ્ત્રોત શોર્ટ પાથ સમસ્યાને સામાન્ય બનાવે છે જેને ફ્લોઈડ માર્શલ(Floyd Marshall) એલ્ગોરિધમ કહેવાય છે. પાથ નકારાત્મક છે અથવા વજન નકારાત્મક છે કે નહીં તે ધ્યાનમાં લીધા વિના આપણે આ બાબતોને હલ કરીશું, જો કે ત્યાં કોઈ નકારાત્મક ચક્ર નથી.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 03
લેક્ચર - 27
નેગેટિવ એજ: બેલમેન-ફોર્ડ એલ્ગોરિધમ (Negative Edges: Bellman-Ford Algorithm)

હવે, ચાલો ગ્રાફમાં સૌથી ટૂંકા રસ્તાઓ જોઈએ જ્યાં અમે નેગેટિવ એજ વજનને મંજૂરી આપીએ. ખાસ કરીને ચાલો આપણે બેલમેન-ફોર્ડ એલ્ગોરિધમ (Bellman-Ford Algorithm)તરફ્ ધ્યાન આપીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:10)
તેથી, યાદ રાખો કે ડિજ્સ્ટ્રા(Dijkstra)ના એલ્ગોરિધમ માટેની શુદ્ધતા એક અરસપરસ પ્રોપર્ટી પર આધારિત છે કે જ્યારે આપણે તેને બાળીશું ત્યારે તે દરેક શિરોબિંદુને આપમેળે બર્ન કરશે.
(સ્લાઈડસમયનો સંદર્ભ લો: 00:25)
જો કે, આપણે જોયું કે આ દલીલ કામ કરતી નથી, જો આપણી પાસે નકારાત્મક ધાર વજન હોઈ શકે છે, કારણ કે આપણને પાછળથી એક પાથ મળી શકે છે જે આપણે બર્ન નથી કર્યો, જે હજી પણ ટૂંકા માર્ગ માટેનો ટૂંકા માર્ગ બની જાય છે. અમે અગાઉ બળી ગયા છે.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:38)
અમે એમ પણ કહ્યું હતું કે નકારાત્મક ધાર વજનને એક વસ્તુ આપવી એ એક વસ્તુ છે, પરંતુ નકારાત્મક વર્તુળોને એક સારો વિચાર નથી. કારણ કે, એકવાર તમારી પાસે એક નકારાત્મક ચક્ર હોય, તમે ચક્રની આસપાસ, તમે ઈચ્છો તેટલી વખત આસપાસ જઈ શકો છો, તે સ્વાભાવિક રૂપે પાથની લંબાઈ ઘટાડે છે, તેથી ટૂંકાતમ પાથ પણ સારી રીતે વ્યાખ્યાયિત જથ્થા નથી. તેથી, જ્યાં સુધી આપણી પાસે નકારાત્મક ધાર છે, પરંતુ નકારાત્મક વર્તુળો નથી, ટૂંકા રસ્તાઓ બહાર નીકળી જાય છે અને અમે તેને ગણતરી કરવાની આશા રાખી શકીએ છીએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:04)
તેથી, ચાલો સૌ પ્રથમ ટૂંકા માર્ગો વિશે બે મૂળભૂત ગુણધર્મો જોઈએ, જે ધાર નકારાત્મક હોઈ શકે છે કે નહીં તે ધ્યાનમાં લીધેલ છે, જો કે આપણી પાસે નકારાત્મક ચક્ર નથી. તેથી, પ્રથમ મિલકત એકદમ સ્પષ્ટ છે, તે એક ટૂંકો માર્ગ છે જે લૂપ દ્વારા ક્યારેય પસાર થશે નહીં. તેથી, મને લાગે છે કે હું s થી t માં જવા માંગુ છું અને માની લો કે હું ખરેખર તે જ વેરટેક્સથી બે વખત પસાર કરું છું અને પછી હું ચાલુ રાખું છું. હવે, આપણે જાણીએ છીએ કે આ લૂપ છે અને તે લૂપ હોવાથી, વજન 0 કરતા વધારે અથવા બરાબર છે, તે નકારાત્મક ન હોઈ શકે, કારણ કે આપણે નકારાત્મક લૂપ્સને નકારી કાઢ્યા છે. તેથી, તેથી, જો હું આ પાથ લઈશ અને હું આ લૂપને કાપીશ, તો મારી પાસે સી થી ટી તરફ્નો સીધો માર્ગ છે. અને જો હું સી થી સી તરફ્ સીધી પાથની કિંમત જોઉં તો તે આના કરતાં મોટું ન હોઈ શકે, કારણ કે આ લૂપમાં 0 છે અને કશું ઓછું નથી, સામાન્ય કિસ્સામાં લૂપ પાસે કેટલાક હકારાત્મક ખર્ચ છે અને વાસ્તવમાં તે ઘટાડે છે ખર્ચ તેથી, ટૂંકા પાથ ક્યારેય આંટીઓ નહીં, તેથી તે ક્યારેય બે વાર વેરટેક્સ મોકલશે નહીં, આનો અર્થ એ છે કે મારી પાસે મોટાભાગે ઓછામાં ઓછા n ઓછા 1 કિનારીઓ હોય છે. કારણ કે હું ફક્ત પરવાનગી આપી શકું છું, તેથી મારી પાસે સંપૂર્ણ રીતે n શિરોબિંદુઓ છે, તેથી જો હું કોઈ પણ શિરોબિંદુને પછીથી ગમે ત્યાં પુનરાવર્તન કરવાની મંજૂરી આપતો નથી; દેખીતી રીતે, મારી પાસે ફક્ત n ઓછા 1 શિરોલંબ હોઈ શકે છે, n ઓછા 1 કિનારીઓ. તેથી, ટૂંકા માર્ગની લંબાઈ પર બંધાયેલું છે, અન્ય મિલકત એ છે કે ટૂંકા માર્ગ કેવી રીતે સૌથી ટૂંકા માર્ગને માર્ગે બનાવે છે તેની પર ધ્યાન આપ્યા વિના, તે પોતે જ ટૂંકા માર્ગનો માર્ગ છે. તેથી, હમણાં પૂરતું મારી પાસે આ પ્રકારનો પાથ છે જે s થી t માં છે જે કેટલાક મધ્યવર્તી શિરોબિંદુઓ v1, v 2 up to v m દ્વારા જાય છે. પછી, જો હું ફક્ત વી.એમ. સુધીનો માર્ગ જોઉં, તો તે s થી v મી થી અમારું સૌથી ટૂંકું પાથ છે. ત્યાં કોઈ ટૂંકા માર્ગ હોઈ શકતા નથી, એવું લાગે છે કે બીજો ટૂંકા માર્ગ હતો. ધારો કે, s થી vm મેળવવાનો ટૂંકા માર્ગ, પછી હું આ અને આ લઈ શકું છું અને હવે લાલ પાથ લીલા પાથ કરતા ટૂંકા હશે. તો, તેથી, હકીકત એ છે કે હું

વી.એમ. મારફત જાઉં છું અને આ એક ટૂંકા માર્ગનો અર્થ છે કે આપણે માત્ર s થી vm ના આપણા ટૂંકા માર્ગનો આ જ ભાગ જ જોઈએ અને આખી વાત એ છે કે આપણે નિર્દેશ કરીએ છીએ. તેથી, ફરીથી 2 ની વિરુદ્ધ શું થાય છે તે સૌથી ટૂંકું પાથ હોવું જોઈએ, કારણ કે જો કોઈ ટૂંકા માર્ગ હોય તો હું તે ટૂંકા માર્ગને લઈશ અને પછી હું આ માર્ગ પર ઉમેરીશ જે મારી પાસે છે અને મને ટૂંકા માર્ગ મળશે વી 3 થી આગળ બધું. તેથી, ટૂંકા માર્ગનો દરેક ઉપસર્ગ પોતે જ ટૂંકા માર્ગ છે. તેથી, આ બંને ગુણધર્મો બેલ્મેન-ફોર્ડ એલ્ગોરિધમ(Bellman-Ford Algorithm)નો નકારાત્મક ધાર વજનની હાજરીમાં ટૂંકા માર્ગ માટે પહોંચવા માટે પૂરતી છે.

(સ્લાઇડટાઇમનો સંદર્ભ લો: 03:36)

તેથી, બેલમેન-ફોર્ડ ઍલ્ગોરિધમ(Bellman-Ford Algorithm) સુધી પહોંચવા માટે, આપણે ડિજ્સ્ટ્રા(Dijkstra)ના એલ્ગોરિધમમાં શું થઈ રહ્યું છે તેના વિશે થોડું વધુ વિશ્લેષણ કરવું પડશે. તેથી, જ્યારે પણ આપણે શિર્ષકો બાળીશું ત્યારે ડિજ્સ્ટ્રા(Dijkstra)ના અલ્ગોરિધમમાં, જ્યારે પણ આપણે મુલાકાત લીધી હોય, ત્યારે અમે તે બધાને પડોશીઓને અપડેટ કરીએ છીએ. તેથી, જ્યારે આપણે શિરોલંબ જે બર્ન કરીએ છીએ ત્યારે આપણે અંતરને અપડેટ કરીએ છીએ, અમે દરેક ધાર જેકે માટે અપડેટ કરીએ છીએ, આપણે k થી અંતરને અપડેટ કરીએ છીએ, આપણે j માટે અંતર સાથે j અને k થી અંતર સાથે પહેલેથી જ ક્યા અંતરથી જાણીએ છીએ. તેથી, આપણી પાસે જે દ્વારા નવી શોધી કાઢેલી અંતર છે અને આપણે તેની સરખામણી જેની પહેલાથી જાણીએ છીએ તેની સરખામણી કરીએ છીએ અને આપણે ડિજ્સ્ટ્રા(Dijkstra)ના અલ્ગોરિધમનો નાનો ભાગ રાખીએ છીએ. તેથી, ચાલો આ ઓપરેશનને એક અપડેટ કહીએ, તે જેમાંથી અપડેટ કરી રહ્યું છે. તેથી, ડિજ્સ્ક્રાના એલ્ગોરિધમમાં આપણે ફક્ત ત્યારે જ આ અપડેટ કરીએ છીએ જ્યારે આપણે j બર્ન કરીએ છીએ અને ડિજ્સ્ક્રાના એલ્ગોરિધમનો સાચીતા કહે છે કે જ્યારે આપણે j એ બાળી નાખ્યો, ત્યારે j ની અંતર j ની સાચી અંતર છે. તેથી, આપણે જે જોયું છે તે છે કે નકારાત્મક ધાર વજનના કિસ્સામાં, જો આપણે ડિજ્સ્ક્રાના એલ્ગોરિધમનો વ્યૂહ ઉપયોગ કરીએ તો તે ન હોત કે આપણે શિરોલંબને બાળીશું કારણ કે તે અણનમ શિરોબિંદુઓ વચ્ચેની સૌથી ઓછી અપેક્ષિત અંતર બાંહેધરી આપતું નથી કે આ સાચી અંતર છે જે આપણે પછીથી થોડી અંતર પર મેળવી શકીએ છીએ. પરંતુ, આ હોવા છતાં, આ અપડેટ ઑપરેશનમાં કેટલીક ઉપયોગી ગુણધર્મો છે જે અમે નિષ્ણાત કરી શકીએ છીએ.

(સ્લાઇડટાઇમ નો સંદર્ભ લો: 04:53)

તેથી, નિર્ણાયક વસ્તુ એ છે કે આપણે અપડેટ કરીએ છીએ કે આપણે અમને અપર બાઉન્ડ આપે છે કે જે k ના અંતરની પાસે છે, આપણી પાસે પહેલાથી છે ... તેથી, કોઈપણ સમયે ઈન્વેરીઅંટ કે જે આપણી પાસે છે તે મૂલ્ય છે આપણી પાસે k થી અંતર એ સ્રોતથી કે ના વાસ્તવિક ટૂંકા અંતર કરતા વધુ અથવા તેના બરાબર છે, શરૂઆતમાં આપણે એમ માની શકીએ કે તે અનંત છે. તેથી, તે ખરેખર નાના અંતર કરતાં ચોક્કસપણે વધારે છે. કોઈપણ સમયે જ્યારે આપણે તેને ઘટાડે છે, ત્યારે અમે તેને ઘટાડે છે કારણ કે અમને એક નક્કર માર્ગ મળ્યો છે જે આપણને ટૂંકા અંતર આપે છે. તેથી, જ્યારે આપણે આ સુધારા પછી આ કરીએ છીએ ત્યારે આપણે જાણીએ છીએ કે k થી અંતર એ જે દ્વારા આપણે હમણાં જ શોધેલી અંતર કરતાં વધુ નથી. કારણ કે, જો તે પહેલાથી ઓછું હતું અને આપણે તે રીતે રાખીએ છીએ, કારણ કે આપણે તેને પાછલા જ પ્રાઈમ પ્રાઈમ દ્વારા શોધી કાઢ્યા છે અથવા આપણે તેને હવે શોધી કાઢ્યું છે કે આપણે આ અપડેટ ઑપરેશન પર તેને અપડેટ કર્યું છે. આનો અર્થ એ પણ છે કે ટૂંકા માર્ગો વિશેના અમારા અગાઉના નિરીક્ષણને કારણે જો ખરેખર ડિજ્સ્ટ્રા(Dijkstra)ના એલ્ગોરિધમ પર પણ હોય તો, જો અંતર j એ ખરેખર સાચો છે અને ટૂંકા પાથમાં આ ધાર ઉમેરવાનો સમાવેશ થાય છે. અમે તેને આદર સાથે દાવો કરતા નથી, જો તે સૌથી ટૂંકા માર્ગ છે, તો પછી k નો અંતર યોગ્ય રીતે ગણવામાં આવશે. તેથી, વાસ્તવમાં આ અપડેટ ઑપરેશન એ એક સુરક્ષિત ઑપરેશન છે, કારણ કે તે ક્યારેય વાસ્તવિક મૂલ્ય કરતાં કે.ની અંતર લાવશે નહીં. તેથી, તે હંમેશાં તે મૂલ્યને સ્વીકારશે જે આપણે જોઈએ છે. તેથી, અમે ફોરિયર અપડેટ્સ કરવાનું ચાલુ રાખી શકીએ છીએ, તેથી બિનજરૂરી અપડેટ્સ અને તે અમને નુકસાન પહોંચાડે છે. તેથી, અસુરક્ષિત અપડેટ્સ આ ગણતરીને સ્વીકારી શકતા નથી, તે ફક્ત પ્રગતિ કરી શકશે નહીં, પરંતુ તે અમને એવી પરિસ્થિતિમાં મોકલશે નહીં કે જેનાથી અમે ન્યૂનતમ કિંમત પ્રાપ્ત કરી શકતા નથી. કારણ કે, અમે હંમેશાં કાર્ય કરીએ છીએ

અથવા લઘુતમ ખર્ચ અપનાવીએ છીએ, તેથી જ્યારે પણ અમે કોઈ મિનિમ કરીએ છીએ, ત્યારે અમે હંમેશાં નીચે આવીશું, પરંતુ વાસ્તવિક મૂલ્યની નીચે પ્રક્રિયા કરીશું નહીં જે અમે શોધવા માંગીએ છીએ.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:29)
તેથી, જો તમે ડિજ્સ્ટ્રા(Dijkstra)ના એલ્ગોરિધમ જુઓ તો તે લોભી અપડેટ્સનો એક ચોક્કસ અનુક્રમ છે, તે સૌથી નાનો અંતર શિખરો પસંદ કરે છે જે બળી નથી અને તેને બાળી નાખે છે અને સાબિતી તૂટી જાય છે, કારણ કે આ અનુક્રમણિકા મેળ ખાતી નથી, જો રસ્તો નકારાત્મક હોઈ શકે છે, તો અમને સૌથી ટૂંકું પાથ આપો. તેથી, કુદરતી પ્રશ્ન એ છે કે વાસ્તવમાં સૌથી ટૂંકા માર્ગ મેળવવા માટે હું કયા અનુક્રમણિકાઓનો ક્રમ માંગું છું? ડિજ્સ્ટ્રા(Dijkstra)ના એલ્ગોરિધમની 3D હેરીસ્ટિક્સ દ્વારા જવાને બદલે અપડેટ્સના ક્રમમાં ગણતરી કરવાની વધુ સારી રીત છે?
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:10)
તો, માની લો કે આપણી પાસે બે શિરોબિંદુઓ અને ટી છે, આપણે s થી t ના ટૂંકા પાથને શોધવા ઈચ્છીએ છીએ અને ધારો કે તે સૌથી ટૂંકા માર્ગ છે જે હું ઈચ્છું છું જે આપણે પહેલા જોયું છે. . હવે, જો આપણે આ ક્રમમાં અપડેટ્સ કરીએ છીએ કે આપણે સૌ પ્રથમ અપડેટ્સ 1 થી 1 ની ગણતરી કરીએ તો આપણે v 1 થી v 2 નું ગણતરી કરીએ છીએ, પછી આપણે v2 થી v 3 નું ગણતરી કરીએ છીએ વચ્ચે આપણે અન્ય વસ્તુઓ કરી શકીએ છીએ, તે છે ફક્ત આ અનુક્રમમાં આ અપડેટ્સ થાય છે, પછી v3 થી v 4 અને તેથી અને અંતે, આપણે vm થી t કરીએ છીએ. પછી, તમે શું જાણો છો કે આપણે પહેલાં ક્યાં જાણ્યું હતું કે જો તમે અંતરની અંતર્ગત યોગ્ય રીતે ગણતરી કરી હોય, તો આ અપડેટ યોગ્ય રીતે વી 1 ની અંતરની ગણતરી કરશે, કારણ કે આ આ પરનું બીજું છેલ્લું નોડ છે. તેથી, જો j ની અંતર સાચી હોય અને j એ બીજા અંતિમ નોડ હોય તો જો k ના ટૂંકાતમ પાથ, k ની અંતર અપડેટ પછી સાચું છે, તેથી કારણ કે s ની અંતર 0 ની અંદર છે.રેક્ટ, જ્યારે આપણે એસ.વી. 1 અપડેટ કરીએ છીએ ત્યારે આ અંતર સાચી બને છે. હવે, અપડેટ્સ આપણને બગડે નહીં કે નહીં તે પણ ઠીક કરી શકે છે, પરંતુ હવે જ્યારે આપણે v2 માં આવીશું ત્યારે આપણે v1 માંથી v2 ને અપડેટ કરીશું અને v1 આ ટૂંકી પાથ પરનો બીજો અંતિમ માર્ગ છે. તેથી, આ એક સાચી મૂલ્ય પણ બની જાય છે, પછી તે પણ યોગ્ય મૂલ્ય બની જાય છે. તો, આ બધા અપડેટ્સની મધ્યમાં જો આપણે આ સબ ક્રમના સબ ક્રમને એસ થી વી 1 પછી ઓળખીએ તો v 1 થી v 2 પછી v2 થી v 3 પછી આપણે અંતર શું છે. તેથી, હવે પ્રશ્ન એ છે કે આપણે કેવી રીતે ખાતરી કરીએ છીએ કે અમારી પાસે s અને t માટે અપડેટ્સનું આ અનુક્રમ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:38)
તેથી, બેલમેન-ફોર્ડ એલ્ગોરિધમ(Bellman-Ford Algorithm) મૂળભૂત રૂપે કહે છે કે તમારા અનુક્રમને શોધવા માટે લખો નહીં, સામાન્ય રીતે બધા સંભવિત અનુક્રમમાં અપડેટ્સની ગણતરી કરો. તેથી, ઉચ્ચ સ્તરે આ અલ્ગોરિધમનો કહેવામાં આવે છે કે શરૂઆતમાં તે અંતર 0 ની અંતર અને તમે બીજા અંતર માટે અનંત હોવાનું અંતર આપ્યો છે. હવે, આપણે ફક્ત દરેક અપડેટ્સ n ઓછા 1 વખત કરીએ છીએ, તેથી આપણે જે કરીએ છીએ તે છે શરૂઆતમાં તમે દરેક ધાર માટે અપડેટ કરો છો જે 0 ની અંતર તરીકે અને તમે અન્યથા અનંત છે, તે પછી આપણે દરેક વસ્તુને અપડેટ કરીએ છીએ. તેથી, આ બધા અપડેટ્સ છે જ્યાં સ્રોત વર્ટેક્સ છે અને v એ લક્ષ્ય મથાળું છે, આ હવે આપણને આ માધ્યમો માટે કેટલાક મર્યાદિત મૂલ્યો આપશે, જ્યાં તે ફક્ત અનંત રહેશે. તેથી, આ મર્યાદિત બનશે અને આ અનંત બનશે, હવે તમે અપડેટ્સનું એક સંપૂર્ણ અનુક્રમ કર્યું છે જેમાં બીજા બધા પાડોશીઓ હવે કેટલાક મર્યાદિત મૂલ્યોની ઈચ્છા રાખે છે, હવે તમે આ બધી વસ્તુ ફરીથી કરો છો. તેથી, તમે દરેક AJK ને અપડેટ કરો છો અને પછી તમે દરેક AJK ને અપડેટ કરો છો, હવે આપણે જાણીએ છીએ કે કોઈપણ બે શિરોબિંદુઓ વચ્ચેનો સૌથી નાનો પાથ મોટાભાગના n ઓછા 1 કિનારીઓનો છે. તેથી, જો આપણે આ વસ્તુ n ઓછા 1 વખત કરીએ, તો દાવા એ છે કે જો મને અપડેટ્સનો કોઈપણ અનુક્રમ જોઈએ તો મને અપડેટ્સનો ક્રમ આના જેવો હોવો જોઈએ અથવા કદાચ તે શોધી શકે કે n ઓછા 1 અપડેટ્સનો કોઈપણ અનુક્રમ કાયદેસર છે આ કિસ્સામાં પાથ રજૂ કરવામાં આવશે. તેથી, આ એક ખૂબ હોંશિયાર મેટ્રિક્સ છે જેમાં એક પુનરાવર્તન પછીના બધા અપડેટ્સ છે, બે પુનરાવર્તન પછીના બધા અપડેટ્સ અથવા અપડેટ ઈ 1 તરફ કોઈ ચોક્કસ અનુક્રમણિકા જોઈએ છે, પછી ઈ 2 અપડેટ કરો, પછી અપડેટ કરો પછી મને તે અહીં ઈ 1 મળશે, પછી હું અહીં ઈ 2 ને શોધીશ, પછી મને અહીં 3 મળશે અને પછી. તેથી, દરેક સંભવિત પાથને અહીં આપેલ સરેરાશ ઉદાહરણમાં દર્શાવવામાં આવે છે, જે

પાથમાંથી જાય છે, તે s થી v 1, v 1 થી v 2 અને તેથી આગળ જાય છે. પછી, આપણે શોધી કાઢીએ કે પ્રથમ અપડેટ્સ પુનરાવર્તન 1 માં આવે છે, પછી v 1 ને અપડેટ કર્યા પછી આગલા પુનરાવર્તન v1, v2 એ v2 ની સાચી અંતર છે, વાયરસ 3 v2 થી v 3 તરીકે અને તેથી આગળ આપશે . અને આખરે, પુનરાવર્તન n માઈનસ 1 ખરેખર તે નહી 1 ની અવમૂલ્યન કરી શકે છે તે આ પાથની લંબાઈ પર નિર્ભર છે. તેથી, જો આ પાથ વાસ્તવમાં છેલ્લા એકમાં n માઈનસ 1 પગલા તરીકે જોશે તો અમને વી.એમ. થી ટી અને ટી વિશે સાચી અંતર અપડેટ મળશે, પરંતુ વાસ્તવમાં તે ઓછું હશે, પછી n ઓછા 1, તે પાથ ઓછો હશે ઓછી લંબાઈ

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:55)

તેથી, એલ્ગોરિધમ વાસ્તવમાં નોંધપાત્ર રીતે સરળ છે, શું તમે સ્રોત વર્ટેક્સમાંથી પ્રારંભ કરો છો. તેથી, શરૂઆતમાં તમે દરેક અંતર માટે અંતર હોવાનું અંતર અસાઈન કરો છો અને તમે s ના અંતરને 0 થી પ્રારંભ કરો છો. હવે, તમારા ગ્રાફમાંના પ્રત્યેક ધાર માટે અંતર્ગત n એ 1 વખત તમે આંશિક રીતે નીચેની ક્રિયા પુનરાવર્તન કરો. તેથી, તમે તે ધારના લક્ષ્યની અંતર્ગત તે લક્ષ્યની વર્તમાન અંતર અથવા ધારના સ્રોતની અંતર અને કિનારીના વજનની તુલનામાં ન્યૂનતમ હશો. તેથી, આ અંધાંકમાં ફક્ત એક વખત અવરોધે છે અને આ ગુણધર્મને લીધે તમે દરેક માન્ય ટૂંકા પાથ માટે શોધી શકો છો, તમને અનુક્રમણિકાનો ક્રમ મળશે જે આ ક્રમમાં મેળવે છે જે તમે અહીં કમ્પ્યુટિંગ કરવા માંગો છો, તમે હંમેશાથી શરૂ થતા દરેક અન્ય શિરચ્છેદનો ટૂંકા માર્ગ મેળવો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:44)

તેથી, ચાલો એક ઉદાહરણ જોઈએ અને જુઓ કે બેલમેન ફોર્ડ એલ્ગોરિધમ(Bellman-Ford Algorithm) ખરેખર કેવી રીતે કાર્ય કરે છે. તો, અહીં એક ગ્રાફ છે જેની પાસે કેટલાક ચક્ર છે અને તેમાં કેટલાક નકારાત્મક વજન છે, પરંતુ ત્યાં કોઈ નકારાત્મક ચક્ર નથી, દાખલા તરીકે આપણે જોશું કે અહીં આપણી પાસે એક ચક્ર છે જે વજન 2 વત્તા 1 અને બાદબાકી 1 વત્તા 2 વત્તા છે. 1. એ જ રીતે, અહીં આપણી પાસે એક ચક્ર છે જેનું વજન 6 થી 3 થી 4 થી 5 ની આસપાસ છે, આપણી પાસે ઓછા 2 વત્તા 1 ઓછા છે 1 વત્તા 3 વત્તા 2 ઓછા 1 વત્તા 1 છે, તેથી આ ફરીથી ચક્ર અથવા વત્તા 1 છે. તેથી, ત્યાં નકારાત્મક વજન છે, પરંતુ કોઈ નકારાત્મક ચક્ર નથી અને હવે આમાં આપણે દરેક બીજા શિરોબિંદુઓથી 1 થી નાના પાથોની ગણતરી કરવા માંગીએ છીએ. તેથી, તે એક સ્રોત શિરોબિંદુ છે, તેથી આપણે આ પુનરાવર્તનને સુયોજિત કર્યું છે, તેથી પ્રારંભિક પગલાંમાં આપણે કહ્યું છે કે વર્ટીક્સ 1 થી 0 ની અંતર અને બીજું બધું અનંત છે. હવે, આપણે પહેલી પગલું બધા ચીજોની વસ્તુઓને અપડેટ કરવાનો પ્રયાસ કરીશું જે આપણે સાંભળીએ છીએ, હવે અનંતતાવાળા બધા મૂલ્યો પડોશીઓને અપડેટ કરશે નહીં, પરંતુ મૂલ્ય 1 પાસે તેના ફક્ત 2 પડોશીઓ 2 જ હશે. તેથી, વસ્તુઓને હવે મર્યાદિત મૂલ્યો બનવાની જરૂર છે, પછી અપેક્ષિત, હવે આપણી પાસે આ ત્રણ શિરોબિંદુ છે જે બંને મર્યાદિત મૂલ્યો છે. તેથી, તમે અપેક્ષા રાખશો કે પડોશીઓ જેમ કે 7 અને 2 પાસે પડોશીઓ 6 હોય. તેથી, એકમાત્ર બહાર જવાનું એ 2 માંથી 6 છે જે 8 થી 7 ની બહાર છે, તેથી તમે 7 અને 6 ની અપેક્ષા રાખી શકો છો અને ખરેખર આગલા પગલાને તમે 6 અને 7 ને મૂલ્યો મેળવો છો જે તે મૂલ્યોથી અપડેટ કરવામાં આવે છે. તેથી, 8 વત્તા 1 9 10 વત્તા 2 એ 12 છે. હવે, તમને આ મળ્યું છે, આપણે હવે શોધીશું કે આપણે હવે 1, 2, 8, 7 અને 6 ને અપડેટ કર્યું છે, હવે નોંધ લો કે મારી પાસે હવે એક બીજો માર્ગ છે જે 1 થી 2 સુધીનો નથી. એક માર્ગ હશે જે 7 થી 1 થી 2 સુધી આ રીતે જશે અને આ ઓછા પાથ બનશે, કારણ કે ઓછા 4. તેથી, આ પુનરાવર્તનમાં ખરેખર મને પ્રથમ બિન-તુચ્છ અપડેટ મળશે જે તમે કરી શક્યું નથી ડિજસ્ટ્રા(Dijkstra)ના ધારણાઓ કે બળીને શિરોલંબ અથવા અદ્રશ્ય. તેથી, અંતર બે જે પહેલાથી જ ગણતરીમાં લેવાયા હતા તે 10 ખરેખર બદલાશે, તે ઉપરાંત, કારણ કે અમારી પાસે 6 ના નવા પાડોશીઓ 6 હતા, હવે આપણને 3 માટે નવું મૂલ્ય મળશે. તો, આ પુનરાવર્તનમાં હું રીસેટ શોધી શકું છું, પરંતુ 2 નું મૂલ્ય ઘટાડે છે, ત્રણનું મૂલ્ય કંઈક મર્યાદિત બને છે. 6 ની વેલ્યુના મૂલ્ય પર ધ્યાન આપો, કેમ કે અગાઉ આપણે આના જેવા પાથને જોઈશું અને હવે આપણે શોધ્યું છે કે આ ખરેખર આ જેવા વધુ સારા પાથ છે, જે લાભમાં ઉમેરવા માટે નકારાત્મક ધાર ધરાવે છે. તેથી, આપણે 12 થી 8 ની વત્તા 1 ઓછા 8 થાય છે, તેથી આપણે આ પછીના પગલાને આગળ સતત આગળ વધીએ છીએ જે 3 તરફ આગળ વધે છે અને આ કારણ છે કે હવે 6 માટે વધુ સારો માર્ગ મળે છે. તેથી, પાછલા પાથમાં આપણે કહ્યું 6 વત્તા 12 અને 12 ઓછા 2 ની 10 ઘાત છે, હવે આપણે કહ્યું છે કે તે 12 નથી, તે 8 છે. તેથી, 8 ઓછા 2 બરાબર 6 છે. તેથી, તમને 3 ની સારી રીત

મળી છે જે આપણે નવામાં શોધી કાઢ્યા છે. પાથ, કારણ કે અમારી પાસે 10 થી 10 વત્તા 1 નો પાથ 11 અને તેથી વધુ છે. તેથી, આપણે એક વધુ પગલું ચાલુ રાખીએ અને પછી કારણ કે હવે આપણે જાણીએ છીએ કે 4 પહોંચી શકાય તેવું છે અને 11 પગલાં 4 વત્તા 3 14 છે. તેથી, 5 પહોંચી શકાય તેવું છે, પરંતુ હવે 4 પોતે જ છે કારણ કે 3 નું મૂલ્ય અપડેટ થયું છે. તેથી, આ પુનરાવર્તનમાં 3 ની વેલ્યુ 11 થી 7 સુધી અપડેટ થઈ જાય છે, આ પ્રચાર કરવામાં આવશે કે એક વધુ પુનરાવર્તન પછી. તેથી, હવે આ મૂલ્ય 14 થી 7 વત્તા 3 ની નીચે આવે છે, પરંતુ આ પ્રક્રિયામાં 3 ની કિંમતમાં શું થાય છે તે પોતે એક જ સમયે બંધાયેલું છે. તેથી, int નું મૂલ્ય ફરી 4 ની વેલ્યુ ઘટાડે છે અને અંતે, n ઓછા 1 પુનરાવર્તન પછી આપણને મૂલ્યોના સ્ટેબલ્સ સેટ મળે છે. તેથી, હવે આ શરુઆતની શરૂઆતથી બધા શિરોલંબનો ટૂંકા માર્ગો છે.
(સ્લાઇડટાઇમનો સંદર્ભ લો: 15:40)
તો, આ અલ્ગોરિધમનો તે જટિલતા શું છે, જ્યાં આ બાહ્ય લૂપ હશે, તેથી આપણે આ રન કરીશું n માઇનસ 1 વખત અપડેટ કરો. તેથી, તે ઓર્ડર n વખત ચાલે છે અને હવે દરેક લુપને આપણે મૂળભૂત રીતે દરેક ધારને અદ્યતન રીતે અપડેટ કરીએ છીએ, તેથી અમારી પાસે એક અડજાનસી (adjacency) મેટ્રિક્સ છે, આપણે દરેક જગ્યાએ એકમાત્ર દેખાવ જોવાની જરૂર છે જ્યાં પણ આપણે જે જે એન્ટ્રી લઈએ છીએ તે જે જોકી લે છે અને અમને અપડેટ કરવામાં આવ્યું છે, પરંતુ અડજાનસી (adjacency)ને સૂચિમાં અમારી પાસે તે ધાર છે જે આપણે જોઈએ છીએ. તો, આપણી પાસે 1, 2 સુધી n છે અને આપણી પાસે ધાર છે, તેથી આપણે આ અપડેટ અને ઓર્ડર n મોડેલ કરી શકીએ છીએ. તો, જો આપણે અડજાનસી (adjacency)ને મેટ્રિક્સનો ઉપયોગ કરીએ તો ઓર્ડર n સ્ક્વેર ટાઈમની કિનારીઓ કોણ છે તે ઓળખો. કારણ કે, આપણે અડજાનસી (adjacency) મેટ્રિક્સની દરેક એન્ટ્રી તપાસવી પડશે, અડીને સૂચિ આપણે ક્રમમાં n વખત કરી શકીએ છીએ. તેથી, જો અડજાનસી (adjacency)ને મેટ્રિક્સ રજૂઆતનો ઉપયોગ કરવામાં આવે તો એકંદરે ફોર્ડ (ford)એલ્ગોરિધમ એન ક્યુબ સમય લે છે, જ્યારે અડજાનસી (adjacency)ને સૂચિ રજૂઆતમાં આપણે તેને m ને ઘટાડી શકીએ છીએ. અને તેથી, કિનારીઓની સંખ્યા નાની છે જે n ક્યુબ કરતાં વધુ સારું સોલ્યુશન બનશે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 03
લેક્ચર - 28
બધી-જોડી ટૂંકો પાથ્સ

હવે ચાલો આપણે બધા જોડણો લઘુતમ પાથ સમસ્યાઓ તરફ ધ્યાન આપીએ, જ્યાં આપણે પાથો શોધવાનો પ્રયાસ કરીએ છીએ, દરેક વચ્ચે સૌથી નાનો માર્ગ ગ્રાફ્માં શિરોબિંદુ જોડી.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:10)
તેથી, યાદ રાખો કે આપણે વજનવાળા ગ્રાફ્ સાથે કામ કરી રહ્યા છીએ, અમે નકારાત્મક ધાર વજનને મંજૂરી આપીએ છીએ, પરંતુ નકારાત્મક વર્તુળોને મંજૂરી આપતા નથી, કારણ કે નકારાત્મક ચક્રો સાથે, આપણું સૌથી નાનો માર્ગ સારી રીતે વ્યાખ્યાયિત નથી, નકારાત્મક વજન સાથે, તે સારું છે વ્યાખ્યાયિત અમે જોયું કે બેલમેન-ફોર્ડ એલ્ગોરિધમ(Bellman-Ford Algorithm) અમને ડિજ્સ્ટ્રા(Dijkstra)ના અલ્ગોરિધમનો સામાન્ય બનાવવાની અને નકારાત્મક વજનવાળા ભારાંકવાળા ગ્રાફ્માં સિંગલ સ્ત્રોત ટૂંકા પાથની ગણતરી કરવા દે છે, પરંતુ નકારાત્મક ચક્ર વગર. તેથી, હવે, આપણે આને વધુ સામાન્ય બનાવવું છે અને માત્ર એક જ સ્રોતના ટૂંકા પાથની ગણતરી કરવી નથી, પરંતુ શિરોબિંદુઓના દરેક જોડીમાં ટૂંકાતમ પાથ. તેથી, જેમ આપણે સમજાવ્યું કે શરૂઆતનું ઉદાહરણ હશે, જો તમે કોઈ મુસાફરી વેબસાઈટ અથવા એરલાઈન વેબસાઈટ ચલાવવાનો પ્રયાસ કરો છો અને કોઈએ ઓછામાં ઓછા ખર્ચ અથવા ન્યૂનતમ સમય ફ્લાઈટ અથવા બાજુના કોઈપણ જોડી વચ્ચેની ટ્રેન શોધવા માંગે છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:54)
તેથી, તમે ટૂંકા રસ્તાઓનું નિરીક્ષણ કર્યું છે કે નકારાત્મક વજનની હાજરીમાં પણ ટૂંકાતમ પાથ ક્યારેય લૂપ કરશે નહીં, કારણ કે આપણે પાથની લંબાઈને વધાર્યા વગર હંમેશા લૂપને દૂર કરી શકીએ છીએ. તેથી, તેથી ટૂંકા માર્ગે બે વાર સમાન વર્ટેક્સની મુલાકાત લેતી નથી અને તેની મહત્તમ લંબાઈ 1 ની બરાબર 1 છે. તેથી, અમે બેલ્મેન-ફોર્ડ(Bellman-Ford) માટે એક રીતે આ આલ્ગોરિધમનો શોધપણ કર્યો છે અને અમે શોધીશું કે હવે તેનો ઉપયોગ આપણે બધા માટે એક ઈન્ટરેક્ટિવ અલ્ગોરિધમનો ઉપયોગ કરી શકીએ છીએ. જોડી ટૂંકા પાથ. તેથી, આપણે ટૂંકાતમ રસ્તાઓ કેવી રીતે બનાવવી તે અંગેના ઉદ્દેશ્યપૂર્ણ ઉકેલ સાથે આવીશું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:30)
તેથી, આપણે મૂળભૂત રીતે લંબાઈથી અસ્પષ્ટતાના સંદર્ભમાં ટૂંકા પાથ બનાવવા જઈ રહ્યા છીએ, પરંતુ મોટાભાગે ખાસ કરીને આપણે વચ્ચેના કયા શિરોબિંદુઓની મંજૂરી આપી છે. તેથી, શિરોબિંદુના જોડીમાં તમે કલ્પના કરી શકો તે સૌથી સરળ સૌથી સરળ માર્ગ તે ફક્ત એક જ ધાર છે. તેથી, અમારી પાસે એકલ ધાર છે અને આ ટૂંકા માર્ગનો બનેલો છે, પછી તમે સારા આકારમાં છો. પરંતુ, સામાન્ય રીતે આ ટૂંકા માર્ગનો માર્ગ હોઈ શકે નહીં, કારણ કે આવી ધાર હોય તો પણ, નકારાત્મક હોવાને લીધે, નકારાત્મક હોવાના કારણે, ત્યાંથી નકારાત્મક ધાર માર્ગો પણ હું થી j સુધી પહોંચું છું. કિનારોનો લાંબા માર્ગ, પરંતુ ટૂંકા એકંદરે ખર્ચ માટે. પરંતુ, આપણે શું જાણીએ છીએ, ટૂંકા માર્ગની લાક્ષણિકતાને કારણે, આઈ થી જીનો આ માર્ગ કેટલાક મધ્યવર્તી શિરોબિંદુઓથી પસાર થાય છે, જે બધા એકબીજાથી જુદા હોય છે, કોઈ શ્વેતતા બે વાર મુલાકાત લીધી નથી. અને બીજું, આમાંના કોઈ પણ શિરોલંબ કાં તો હું અથવા j, હું પાછા આવવા અને પછી j પર જવાનો કોઈ મુદ્દો નથી, તેથી આ પાથમાં કોઈ રસ્તો નથી. તેથી, આ અનિશ્ચિત વસ્તુ માટે આપણે શું કરી શકીએ તે પ્રતિબંધિત કરવાનું છે, i અને j વચ્ચે શું થઈ શકે છે, કયા શિરોબિંદુઓને મંજૂરી છે અને અમે ધીમે ધીમે સેટને વધારીશું. જો આપણે કોઈપણ શિરોબિંદુને મંજૂરી આપીએ છીએ, તો પછી, આપણે મનસ્વી ટૂંકા રસ્તાઓ મેળવીશું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:40)

તેથી, યાદ રાખો કે શિરોલંબને n થી n એ ક્રમાંકિત કરવામાં આવે છે, તેથી આપણે આઈજેની જથ્થાના ડબલ્યુકેની ગણતરી i થી j ના ટૂંકા પાથનું વજન હોઈશું, જ્યાં આપણે શિરોબિંદુને મર્યાદિત કરી શકીએ છીએ i થી j થી 1 અને k વચ્ચેનો ઉપયોગ થાય છે. બીજા શબ્દોમાં કહીએ તો, આપણી પાસે શિર્ષકો વીનો આ સમૂહ છે, તેથી આપણે કહીએ છીએ કે આપણી પાસે 1 થી k છે અને આપણી પાસે k વત્તા 1 થી n છે, આપણે કહીએ છીએ કે ફક્ત આ શિરોબિંદુ પાથમાં ઉપયોગમાં લેવાય છે. હવે, અંત પોઈન્ટ પોતાને ન હોઈ શકે, ન હોવું જોઈએ, પરંતુ બિંદુએ તે છે કે અંત બિંદુ બહાર છે, તો તે હજી પણ કરી શકે છે, તેથી મારી પાસે અહીં 1 અક્ષાંશ હોઈ શકે છે અને પછી તે જેવો માર્ગ હોઈ શકે છે આ અને પછી પાછા આવો. તેથી, ફક્ત એટલું જ કહેવામાં આવે છે કે મધ્યવર્તી શિરોબિંદુ, તેથી જ્યારે હું મારી સાથે પ્રારંભ કરું છું અને j પર જાઉં છું, તો આ સમૂહ 1 થી k માં આવેલું જૂઠાણું વચ્ચે શું થાય છે. તેથી, ખાસ કરીને જો કે 0 છે, કારણ કે આપણું ક્રમાંકન 1 થી n છે, તે કહે છે કે જે શિરોબિંદુઓ દેખાઈ શકે છે તે શામેલ હોઈ શકતા નથી. તેથી, બીજા શબ્દોમાં, જો મારી પાસે ડબલ્યુ 0 હોય, તો તે બધા શિર્ષકો 1 કહે છે. n એ પાથો પર દેખાઈ શકતા નથી, તેથી એકમાત્ર રસ્તો આપણે કરી શકીએ છીએ 3 આ પ્રકારના ટૂંકા રસ્તાઓ સીધી ધાર તરફ છે. તેથી, ડબ્લ્યુ 0, આ અધિષ્ઠાપિત વ્યાખ્યાનો મૂળ કેસ કહે છે કે i અને j વચ્ચેના ટૂંકા રસ્તાઓ જે 1 થી n સુધીના બધા શિરોબિંદુઓને બાકાત રાખે છે તે i અને j વચ્ચેના ફોર્મ ધાર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:07)

તેથી, હવે આ એક ઈન્ટરેક્ટિવ વ્યાખ્યા છે, કારણ કે આપણે કહીએ છીએ કે, આપણે જાણીએ છીએ કે ટૂંકા પાથને 1 થી 1 ની બાદબાકી 1 નો ઉપયોગ કરીએ, તો પછી હું લઘુતમ પાથોને કેવી રીતે ઘટાડી શકું 1 કે. તેથી, આપણે બધા પાથ વચ્ચે જાણીએ છીએ જે 1 થી 1 ની ઓછા 1 નો ઉપયોગ કરે છે, i થી j નો સૌથી નાનો માર્ગ શું છે, આપણે હવે કેવી રીતે ગણતરી કરી શકીએ, જો આપણે વેરટેક્સ k ને પણ ઉપયોગમાં લેવાની મંજૂરી આપીએ, તો તમે કેવી રીતે ગણતરી કરો છો હું થી જે ટૂંકા પાથ છે. તેથી, ત્યાં બે કિસ્સાઓ છે, પ્રથમ કેસ એ છે કે આ અતિરિક્ત વર્ટેક્સ કેસ ઉપયોગી નથી, હું શામેલ કરું છું તે ટૂંકા પાથ પણ વર્ટેક્સ કેનો ઉપયોગ કરતું નથી, તે 1 થી ઓછા 1 નો ઉપયોગ કરવા માટે પૂરતી છે. આ કિસ્સામાં સૌથી નાનું 1 થી કેનો ઉપયોગ કરીને ડબલ્યુ આઈ થી j ની અંતર એ કે ઓછા 1 નો ઉપયોગ કરીને i થી j ના ટૂંકા અંતર જેટલો જ છે. તેથી, હું કહી શકું છું કે આઈજેનો ડબલ્યુકે કે ડબલ્યુ કે કેન્યુ 1 આઈ જે સમાન છે. બીજી બાજુ, તે હોઈ શકે છે કે k નો ઉપયોગ કરીને અમને કેટલાક નજીવી સુધારણા મળે છે. તેથી, અમારી પાસે હવે કેટલાક પાથ છે જે i થી j સુધી જાય છે અને જે રીતે તે મુલાકાત લે છે. તેથી, જો તે રસ્તા પર k ની મુલાકાત લે છે, તો આપણે તેને i થી k ના પાથ અને k થી j તરફના પાથ તરીકે તોડી શકીએ છીએ. પરંતુ, ધ્યાન રાખો કે આપણે પહેલાથી જ કહ્યું છે કે આ પાથમાં કે જે આ પાથમાં દેખાય છે તે આ વાક્ય કે જે કોઈ ટૂંકા માર્ગ છે તે ફક્ત એક જ વાર દેખાય છે. તેથી, જો કાટ કે અહીં દેખાય છે, તો આઈ અને કે વચ્ચે કોઈ કે કે અને કે અને જે વચ્ચે કોઈ કે. આનો અર્થ એ છે કે મારી પાસે પહેલેથી જ k થી એક પાથ છે જે 1 થી 1 ની બાદબાકી 1 સુધી જાય છે અને મારી પાસે k થી j થી પહેલેથી જ એક માર્ગ છે જે 1 થી 1 ઓછા થાય છે અને હું આ બંનેને સંયોજિત કરું છું. તેથી, હું પાથને તોડી શકું છું, તે આઈ થી કે અને પાથ કે થી જેનો રસ્તો છે, જેમાંથી દરેક માત્ર 1 થી કે ઓછા 1 નો ઉપયોગ કરે છે. અને તે પાથની કિંમત કેટલી છે, તે આપણે સારી રીતે જાણીએ છીએ અમારી પાસે માત્ર 1 થી કે ઓછા 1 ઈન્ડેક્ટિવ કે માઈનસ 1 નો ઉપયોગ કરીને આઈ થી કે શ્રેષ્ઠ માર્ગની કિંમત છે, તમારી પાસે અમારા મેટ્રિક્સ ડબલ્યુ કે બાદબાકી 1 માં કે થી j નો શ્રેષ્ઠ માર્ગ પણ છે. તેથી, જો હું આ બંનેને ઉમેરીશ, હું અથવા કે જવાનો આ શ્રેષ્ઠ રસ્તો હોવો જોઈએ. તેથી, કેસનો સામનો કરવો, અમે કહીએ છીએ કે ક્યાં તો આપણે કે કે જે કિસ્સામાં આપણે કે કે જૂના મેટ્રિક્સનું મૂલ્ય લેતા નથી અથવા તમે કેનો ઉપયોગ કરો છો અને કયા કેસમાં અમે જૂના મેટ્રિક્સમાં આઈ કે કે પર જઈને બે એન્ટ્રી જોડીએ છીએ, અમે આ બે નાનાં નાનાં લો. તેથી, આમાંથી કયો એક, i થી j થી ટૂંકા અંતરનો સાચો મૂલ્ય 1 થી k સુધી જશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:27)

તેથી, આ આપણને એક તાત્કાલિક અલ્ગોરિધમ આપે છે જેને ફ્લૉઈડ-વૉર્સેલ(Floyd-Warshall) અલ્ગોરિધમ કહેવામાં આવે છે. તેથી, આપણે ફંક્શન W 0. નું પ્રતિનિધિત્વ કરતા મેટ્રિક્સ સાથે પ્રારંભ કરીએ છીએ, તેથી, W 0 એ એન્ટ્રીઝ છે જે બરાબર ધાર વજન છે, તેથી આઈ થી જજેની ધાર છે, ડબલ્યુ 0 આઈ થી થી જે કહે છે કે તે વજનનો સીધો પાથ.

કારણ કે, યાદ રાખો કે ડબલ્યુ 0 કોઈપણ મધ્યવર્તી પાથ્સથી પસાર થઈ શકતું નથી અને જો કોઈ ધાર નથી, કારણ કે હું કોઈપણ મધ્યવર્તી વર્ટેક્સમાંથી પસાર થઈ શકતો નથી, ડબલ્યુ 0 આઈજે અનંત હોવું આવશ્યક છે. હવે, 1 થી n માં k માટે, હું મૂળભૂત રીતે આ n વખત પુનરાવર્તિત કરું છું, હું પહેલા 1 ને ઉપયોગમાં લેવાની મંજૂરી આપું છું. તેથી, હું ડબલ્યુ 1 અને ડબલ્યુ 0 નું ગણતરી કરું છું અને હું તે કેવી રીતે કરી શકું છું, હું અગાઉ શામેલ કરવા માટે અપડેટનો ઉપયોગ કરું છું જે તમારી પાસે પહેલાથી જે છે તે ન્યુનત્તમ લેશે અને નવી રજૂ કરેલ વર્ટેક્સ દ્વારા થતી શક્યતા, પછી હું 1 અને 2, તો હું 1, 2 અને 3 ને પરવાનગી આપીશ. અને દેખીતી રીતે, જો હું 1, 2, 3 ને n સુધી પરવાનગી આપું છું, તો હું બધા શિરોબિંદુઓને વચ્ચેની પરવાનગી આપું છું, W n પાસે હવે કોઈ અવરોધ વિના સૌથી ટૂંકા રસ્તાઓ હશે. તેથી, મારે આ અપડેટ્સ બરાબર એન ટાઈમ્સ કરવાની જરૂર છે, જેથી કરીને n વખત પછી હું સૌથી ટૂંકા માર્ગ પાથને કેપ્ચર કરી શકું જેમાં પાથ પર શિરોબિંદુનો કોઈપણ મનસ્વી સંયોજન શામેલ હોય.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:37)
તેથી, વાસ્તવિક કોડ ફરીથી બેલમેન-ફોર્ડ (Bellman-Ford) જેવા અત્યંત આગળ છે. તમારી પાસે ફક્ત પ્રારંભિક પ્રારંભ છે જે અનંત સુધી દરેક રીતે સેટ કરે છે અને ત્યારબાદ તે શાખાઓ માટે બિન-તુચ્છ વજનને અપડેટ કરે છે જે ગ્રાફ ધરાવે છે. તેથી, આપણે ત્રિપરિમાણીય મેટ્રિક્સ દ્વારા આ ફંક્શન ડબલ્યુ 0 ની ટ્રૅક રાખીએ છીએ, તેથી હું અને જે બે શિર્ષકોનું પ્રતિનિધિત્વ કરે છે અને 0 પુનરાવર્તન ક્રમાંકને રજૂ કરે છે. તેથી, શરૂઆતમાં ડબલ્યુ 0 ની આઈજે કાં તો આઈજનું વજન છે, જો કોઈ ધાર છે અથવા તે અનંત છે અને તે આ બે પગલાંઓને સંભાળે છે. અને હવે, આપણે આ સમયને સમાપ્ત કરીએ છીએ, અમે આ પુનરાવર્તન કરીએ છીએ, તે છે કે આપણે બધા ડબલ્યુ આઈજે (IJ)ને સ્તરે k પર અપડેટ કરીએ છીએ, ડબલ્યુ આઈજે (IJ)ના ન્યૂનતમ સ્તર સ્તર કે માઈનસ 1 અથવા રુટની રકમ પર. તેથી, અહીં અસ્પષ્ટપણે હોવા જોઈએ

((સમયનોસંદર્ભ: 08:33)).
તેથી, આ માટે છે, કારણ કે હું 1 થી n બરાબર, j બરાબર 1 થી n માટે. તેથી, અમારી પાસે આ અદ્યતન નિયમ છે અને અમે આ અંધારામાં એન ટાઈમ્સ કરીએ છીએ અને n આપણે દાવો કરીએ છીએ કે એન્ટ્રી k માં મેટ્રિક્સ ડબલ્યુને શિરોલંબના બધા જોડીઓ માટે યોગ્ય ટૂંકા પાથ મળ્યા છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 08:55)
તેથી, ચાલો આપણે એ જ ઉદાહરણ જોઈએ જે આપણે બેલમેન-ફોર્ડ(Bellman-Ford) માટે કહીએ છીએ. તેથી, શરૂઆતમાં આપણે દરેકને અસાઈન કરીએ છીએ, તેથી ડબલ્યુ 0 ની ધાર વજન છે. તો, આપણી પાસે અડજાનસી (adjacency)ને મેટ્રિક્સ છે જેમ કે પ્રતિનિધિત્વ, દાખલા તરીકે, આપણે કહીએ છીએ કે 1 થી 2 એ વજન 10, 7 થી 6 ની ધાર છે, ત્યાં વજન ઓછું 1 ની ધાર છે અને તેથી આગળ. તેથી, આપણે ફક્ત આ તમામ આંગળીઓને મેટ્રિક્સમાં ડુપ્લિકેટ કરીએ છીએ અને બીજું બધું અનંત પર છોડી દીધું છે. હવે, ડબલ્યુ 0 થી, આપણે 1 થી જઈને શોધી શકીએ તે બધા નવા રસ્તાઓ ધ્યાનમાં લઈને ડબલ્યુ 1 પર જઈએ, પરંતુ હવે નોંધ લો કે આમાં કંઈ પણ નથી જે 1 માં આવે છે. તેથી, આ હકીકતમાં સૂચવાયેલ છે કે કૉલમ 1 અનંત છે, કોઈ ધાર 1 માં જાય છે. તેથી, કોઈ અન્ય વર્ટીક્સ એ હકીકતનો ઉપયોગ કરી શકે છે કે 1 2 અને 8 થી કનેક્ટ થયેલું છે. તેથી, જો તમે અપડેટ નિયમનો ઉપયોગ કરીને અપડેટ કરો છો, તો તમને લાગે છે કે W1 એ ખરેખર W ની સમાન છે. , કારણ કે બે શિરોબિંદુઓ વચ્ચે 1 નો ઉપયોગ કરવાની મંજૂરી આપણી સહાય કરતી નથી. તે કોઈ બે શિરોબિંદુઓ વચ્ચે નથી, હું ક્યાંયથી 1 સુધી જઈ શકું છું અને પછી 1 થી તે વર્ટેક્સ સુધી નથી. બીજી બાજુ, જો હું એન કરી શકું છુંow માં 2 શામેલ છે, પછી હું રસપ્રદ વસ્તુઓ કરી શકું છું, હું ઉદાહરણ તરીકે 1 થી 2 થી 6 સુધી જઈ શકું છું અને હું 7 થી 1 થી 6 સુધી જઈ શકું છું. તેથી, 2 અને 1 ને ઈન્ટરમિડિયેટ વર્ટેક્સ તરીકે ગોઠવો, તેથી 1 મદદ કરતું નથી અત્યારે, પણ 2 ને મને કંઈક આપે છે. તેથી, જો હું હવે ડબલ્યુ 1 ને જોઉં છું, તો હવે

મને ડબલ્યુ 0 ની જરૂર નથી, જ્યારે ડબલ્યુ 1, તો પછી હું કહું છું કે પાથ્સ 2 દ્વારા અન્વેષણ કરવામાં સક્ષમ છે કે નહીં તેવું કહેવામાં આવે છે. તેથી, ખાસ કરીને 2 થી 6 થાય છે, જે કંઈક 2 માં નિર્દેશ કરે છે, તેથી 7 થી 2 થી 6, 3 થી 2 થી 6 અને 1 થી 2 થી 6 સુધી, આ ત્રણ એન્ટ્રી અપડેટ થશે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:13)
તેથી, 1 થી 2 થી 6, તેથી મને નવી એન્ટ્રી 12, 3 થી 2 થી 6 મળશે, તેથી મને નવી એન્ટ્રી 3 અને 7 થી 2 થી 6 મળે છે, મને ઓછા 4 વત્તા 2 મળે છે. માઈનસ 3. હવે ડબલ્યુ 2 થી, હું ડબલ્યુ 3 નું ગણતરી કરીશ, ફરીથી ગણતરી કરું 3 ડબ્લ્યુ 3 પછી પાછા આવશે, મને ડબલ્યુ 1 ની જરૂર નથી, મને માત્ર ડબલ્યુ 2 ની જરૂર છે. તેથી, હું હવેથી W 1 ને ફેંકી શકું છું, હું ફક્ત ડબલ્યુ 2 પર જ છું. તેથી, હવે હું મારી જાતને 1, 2, 3 નો ઉપયોગ કરવાની મંજૂરી આપી રહ્યો છું, તેથી હું વસ્તુઓ ફેંકી શકું છું 3. તેથી, ઉદાહરણ તરીકે હવે 3 ફેંકી દો, હું 6 થી 4 સુધી જઈ શકું છું, ઉદાહરણ તરીકે, તેથી મને તે ફોર્મની એન્ટ્રી મળશે. તેથી, હું 6 થી 4 ની નવી વસ્તુ સાથે જઈ શકું છું, હું 6 થી 2 પણ જઈ શકું છું. તેથી, મારી પાસે 6 થી 2 ની નવી રીત છે, તેથી તે પણ અલગ થઈ જશે. અને રસપ્રદ રીતે, મેં પણ શોધ્યું કે, હવે 6 થી 6 નો માર્ગ છે, કારણ કે પહેલા મને તે ખબર નહોતી, પરંતુ હવે તે માટે, હું 2 અને 3 બંને ફેંકી દેવાની પરવાનગી આપું છું, હું 6 થી 6 સુધી જઈ શકું છું. આ લૂપ શોધી કાઢ્યું છે, જે હકારાત્મક વજન ધરાવે છે. જો મુશ્કેલીમાં પોઝિટિવ વેઈટ વ્યૂઅર ન હોય, કારણ કે તેની પાસે સારી રીતે વ્યાખ્યાયિત સોલ્યુશન નથી અને તેથી આગળ. તેથી, તમે આ કરવાનું ચાલુ રાખી શકો છો, અમે W સુધી અપડેટ કરીશું નહીં, જો તમે હવે બધી રીતે જાઓ અને W8 સુધી કરો, પછી તમે 1 થી 8 ની દરેક વસ્તુને મધ્યવર્તી વસ્તુ તરીકે મંજૂરી આપો પછી, આ મેટ્રિક્સ વાસ્તવમાં કોઈપણ જોડીમાં કોઈપણ જોડીની સૌથી ટૂંકી પાથની ગણતરી કરો.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:52)
તેથી, આ અલ્ગોરિધમ એ જોવાનું ખૂબ જ સરળ છે કે જટિલતા ઓર્ડર એન ક્યુબ છે, કારણ કે તમારી પાસે એન પુનરાવર્તન છે અને દરેક પુનરાવર્તન અમે સંપૂર્ણ મેટ્રિક્સને અપડેટ કરી રહ્યા છીએ જેમાં n ચોરસ એન્ટ્રીઝ છે. આ તમે તેને સુધારવા માટે વધુ કરી શકતા નથી, કારણ કે તે અડજાનસી (adjacency)ને મેટ્રિક્સ બેઝ એલ્ગોરિધમ છે, અમે સૂચિમાં ખસેડી શકતા નથી, અમને કોઈપણ ન્યૂનતમ મહત્તમની ગણતરી કરવાની જરૂર નથી. તેથી, તમે ધાર્યું બધું કરી શકો છો, તે એન ક્યુબ એલ્ગોરિધમ છે, નોંધ લો કે, તે નાજુક કેસ છે, બેલમેન-ફોર્ડ(Bellman-Ford), કારણ કે એકવાર તમે બધા જોડીઓની ગણતરી કરી લો, તેથી આ ઉકેલ સામાન્ય રીતે બેલમેન-ફોર્ડ(Bellman-Ford) . કારણ કે, ખાસ કરીને, જો તમે હવે આપેલા S માંથી બધા ટૂંકા પાથને જોઈએ છે, તો તમારે ફ્લોઈડ વૉરશૉલ(Floyd-Warshall) મેટ્રિક્સમાં તે ચોક્કસ પંક્તિને જોવાનું છે. તમે પંક્તિ એસ અને બધી એન્ટ્રીઓને S ના ટૂંકા પાથ તરીકે ઓળખાવે છે, પરંતુ યાદ રાખો કે, તે ચોક્કસ કિસ્સામાં, જો હું માત્ર એસ બેલ્મેન-ફોર્ડ(Bellman-Ford)થી હોંશિયાર અડજાનસી (adjacency)ને સૂચિ રજૂઆત કરતો હોઉં, તો આ આદેશ અહીં લેવાશે, પરંતુ તે અહીં છે ઓર્ડર એન ક્યુબની જરૂર પડશે. તેથી, આ સામાન્ય રીતે બેલમેન-ફોર્ડ(Bellman-Ford)નો દાખલો છે, તમને સમાન જવાબ મળે છે કે તમને બેલમેન-ફોર્ડ(Bellman-Ford) અને વધુ તરફથી કૉલ હશે. પરંતુ, તમે હંમેશાં એન ક્યુબ ટાઈમનો ઉપયોગ કરો છો, જો તમારા ગ્રાફમાં ખૂબ થોડા ધાર હોય તો તે સામાન્ય રીતે બેલમેન-ફોર્ડ (Bellman-Ford)વધુ કાર્યક્ષમ હોય છે. તેથી, જો તમે ઉપયોગ કરી રહ્યા છો, જો તમે માત્ર એક જ સ્રોત ટૂંકા પાથ કરવા માંગો છો, તો તમારે સામાન્ય રીતે ફ્લોઈડ વૉરશૉલમાં સીધી જ કૂદી જવું જોઈએ નહીં, તમારે કદાચ બેલમેન-ફોર્ડ (Bellman-Ford) ઈન્સ્ટન્ટ કરવું જોઈએ. અવકાશની જટિલતા વિશે, તેથી અમે કહ્યું કે અમે પ્રત્યેક 0 ડબલ્યુ, ડબલ્યુ 1 ઈત્યાદિનું પ્રતિનિધિત્વ કરવા જઈ રહ્યા છીએ તે એક સંકલન છે. તેથી, આપણી પાસે મૂળભૂત રીતે n વખત, n times n છે, કારણ કે આપણી પાસે n ગુણ્યા n એ વાસ્તવિક મેટ્રિક્સ છે, આપણી પાસે આ મેટ્રિક્સનો n છે, કારણ કે આપણી પાસે સ્તર 0, સ્તર 1 અને સ્તર કે કે સ્તર n છે. પરંતુ, આપણે કહીએ છીએ કે આપણા કામના ઉદાહરણમાં, જ્યારે તમારે સ્તર 1 નું ગણતરી કરવાની જરૂર છે, ત્યારે અમને માત્ર સ્તર 0 ની જરૂર છે, પછી આપણે એક સ્તર 0 ફેંકી શકીએ છીએ. જ્યારે, અમને સ્તર 2 નું ગણતરી કરવાની જરૂર છે, તમારે માત્ર સ્તર 1 ની જરૂર છે. તેથી, અમુક અર્થમાં, તમે માત્ર 2 કૉપિઓ રાખી શકો છો અને પાછા અને ફોલ્ડ સ્વિચ કરી શકો છો, તમે શૂન્ય સ્તરને બીજા સ્તર તરીકે લખો છો, તમે ત્રીજા સ્તર તરીકે પ્રથમ સ્તર લખો છો અને તેથી આગળ લખો છો. તેથી, અમને ફક્ત 2 કાપી નાંખવાની જરૂર છે, તે એક સમયે આ ત્રિપરિમાણીય મેટ્રિક્સનો કૉલ કરો. તેથી, તમે આ 2 સ્લાઈસેસની વચ્ચે માત્ર

2 એન સ્ક્વેર્ડ સ્પેસની વચ્ચે ભળી શકો છો. તેથી, આપણે સામાન્ય રીતે અવકાશ વિશે ચિંતા કરીએ છીએ. પરંતુ, ફક્ત એક નિરીક્ષણ કે આ વિશિષ્ટ વસ્તુમાં, તમારે ખરેખર n ઘન એરે હોવા જરૂરી નથી, તમારી પાસે 2 n સ્ક્વેર્ડ એરે અથવા તે બે સૂચકાંકો સાથે સ્ક્વેર્ડ એરે હોઈ શકે છે. અને 2 ની વચ્ચે સંકોચન રાખો અને સમાન અસર મેળવો, કારણ કે તમને બીજી કૉપિની ગણતરી કરવા માટે ફક્ત એક કૉપિની જરૂર છે.
(સ્લાઈડસમયનો સંદર્ભ લો: 14:20)
તેથી, ચાલો આપણે આ ચર્ચાને કેટલીક ઐતિહાસિક ટિપ્પણીઓમાં સમાપ્ત કરીએ. તેથી, તમે જોઈ શકો છો તે પ્રમાણે ફ્લોયડ વૉશશેલ(Floyd-Warshall), આ અલ્ગોરિધમનો સંકર નામ તરીકે હાઈફનેશન આવે છે અને વાસ્તવમાં ત્યાં બે અલગ અલગ એલ્ગોરિધમ્સ છે જે તેમાં શામેલ છે, જે ખૂબ સમાન માળખું ધરાવે છે. તેથી, વૉરશેલ દ્વારા પ્રસ્તાવિત મૂળ એલ્ગોરિધમ જેને ટ્રાન્ઝિટિવ ક્લોઝર કહેવાય છે તે માટે છે. તેથી, સંક્રમણ બંધ થવું એ એજ સંબંધથી પાથની ગણતરી કરવા બરાબર જ છે. તેથી, ધારો કે તમારા મિત્ર જેવા સંબંધો છે, તેથી તમે લોકોના જૂથમાં જાણો છો, કે જેનો સીધો મિત્ર છે, પછી તમે પૂછો કે, કોણ આડકતરી રીતે જાણે છે. તેથી, હું કોઈકને પરોક્ષ રીતે જાણું છું, જો મારી પાસે કોઈ મિત્ર હોય, તો તે વ્યક્તિને કોણ જાણે છે અથવા મારા મિત્ર છે કે જે મિત્ર તરીકે જાણે છે અને તે વ્યક્તિને જાણે છે. તેથી, કોઈકને પરોક્ષ રીતે જાણવું તે મિત્ર સંબંધનું સંક્રમિત બંધ છે. સામાન્ય રીતે, મારો અર્થ એ છે કે, દરેક ગ્રાફમાં, જો તમે મિત્રને મૂકો છો, ગમે તે સંબંધ, આપણે ધાર સંબંધ તરીકે જોઈએ છે, સંક્રમિત બંધ માર્ગ પાથ સંબંધ છે. તો, તમારી પાસે અડજાનસી (adjacency)ને મેટ્રિક્સ છે જે ધારને રજૂ કરે છે, તમે પાથ મેટ્રિક્સનું ગણતરી કરવા માંગો છો જે પાથને રજૂ કરે છે, જે પાથો દ્વારા જોડાયેલા શિરોબિંદુઓના જોડી છે. અને તેથી, વૉશશેલ(Warshall) એ સમાન એલ્ગોરિધમનો વર્ણન કરે છે, જે આપણે હમણાં લખ્યું છે અને અમે તે પછીની કેટલીક સ્લાઈડ્સમાં થોડું વિગતવાર કરીશું, આ ગણતરી એ P માંથી A અને Floyd જોવા મળે છે તેવું તમે એ જ ઍલ્ગોરિધમ તરીકે સ્વીકાર કરી શકો છો. તેથી, જો વૉર્સલ(Warshall)નું એલ્ગોરિધમ છે, તો અમે ફક્ત પાથ ત્યાં જ તપાસીએ છીએ અને ફ્લૉઈડ્સ(Floyd) એલ્ગોરિધમ કહે છે કે, તમે વાસ્તવમાં ટૂંકા પાથની ગણતરી કરવા માટે તેને અનુકૂલિત કરી શકો છો.
(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:50)
તેથી, તમે જે જોયું છે તેના કરતાં આ વિચાર ખૂબ જ સમાન છે, ચાલો આપણે તેને ઝડપથી પસાર કરીએ. તેથી, આપણી પાસે અડજાનસી (adjacency)ને મેટ્રિક્સ એ છે, જે ધારને એક આઈજે 1 છે, જો ધાર હોય અને આપણે પાથ મેટ્રિક્સ P ij એ 1 જોઈએ, તો i થી j નો પાથ છે. તેથી, આપણે ફરીથી આ જથ્થામાં પી કજને ગણતરી કરીશું, તે કહે છે કે ત્યાં પાથ છે અને આ પાથ ફક્ત 1 થી કે શિરનો ઉપયોગ કરે છે. તેથી, કે પ્લસ 1 થી એન પાથમાં દેખાશે નહીં અને ફરીથી એન પોઈન્ટ સમાવેલ નથી. તેથી, હું અને j મનસ્વી છે, તે આ સુપર સેલિબ્રિટી દ્વારા પ્રતિબંધિત આઈ અને જે વચ્ચેનો છે. તો, હું અને j વચ્ચે, તમે ફક્ત 1 થી k ની વસ્તુઓ જોઈ શકો છો. તેથી, જો તમે પી 0 પહેલા કરો છો, તો તે કહેશે કે કશું દેખાશે નહીં, કારણ કે તમારી પાસે 1 થી n થી દરેક વસ્તુ હોઈ શકે છે, તેથી, પી 0 માત્ર અડજાનસી (adjacency)ને મેટ્રિક્સ છે.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 16:43)
તેથી, હવે આપણી પાસે ખૂબ જ સમાન અપડેટ નિયમ છે, તેથી જો મને 1 થી 1 ની ઓછા 1 નો ઉપયોગ કરીને શોધી શકાય તેવા પાથને ખબર હોય, તો 1 થી k નો ઉપયોગ કરીને હું કયા પાથ શોધી શકું છું. તેથી, જો k નો ઉપયોગ કર્યા વિના પહેલેથી જ કોઈ પાથ હોય, તો હું તે પાથ રાખી શકું છું. તેથી, અમે કાં તો તે પી કે ધરાવી શક્યા હોત, હવે મને યાદ રાખવું જોઈએ કે આ પાથ કોઈ માર્ગ નથી. તો, આ અડજાનસી (adjacency)ને મેટ્રિક્સ જેવું છે, કોઈ વજન નથી, તે 0, 1 મેટ્રિક્સ અથવા સાચું ખોટું મેટ્રિક્સ હતું. તેથી, શરૂઆતમાં અડજાનસી (adjacency)ને મેટ્રિક્સ 1 અથવા સાચું છે, જ્યારે પણ ધાર હોય ત્યારે ખોટી હોય ત્યારે ખોટી. તેથી, જો મને પહેલેથી જ 1 ઓછા સાથે એન્ટ્રી મળી ગઈ હોય, તો હું તેને રાખી શકું છું. બીજી બાજુ, કદાચ મારી પાસે પાછલા કે પછી જવાની એન્ટ્રી નથી, પરંતુ એક સેકંડ જે ત્યાંની જરૂર છે તે આઈ થી કે એક પાથ છે અને ત્યાં k થી j નો પાથ છે અને અહીં હું ફક્ત કે ઓછા 1 નો ઉપયોગ કરું છું અને અહીં હું ફક્ત કે બાદબાકી 1 નો ઉપયોગ કરું છું, કારણ કે k ને ફક્ત એક જ વાર દેખાય છે. એક ટૂંકી પાથની જેમ, જો હું ફક્ત કેટલાક કનેક્ટિવિટી શોધી રહ્યો છુ, તો મને કે કે થી પાછા જઈને કંઈ પણ ફાયદો થશે નહીં, કારણ કે હું આને દૂર કરી શકું છું, જે હું

અને j જોડાયેલ રહેશે. તેથી, મારે માત્ર એવા પાથ જોઈએ છે જેની પાસે દરેક શિર્ષકની એક નકલ હોય. તો, તેથી, હું ધારી શકું છું કે ત્યાંથી એક થી કે પાથ છે, કે જે k થી k થી, k ની સમાન કંઈપણની બહાર ઉપયોગ કરતું નથી. તેથી, હવે અહીં મિની અને પ્લસ ઓપરેશનોને બદલે, હવે અથવા તે બને છે અને, ક્યાં તો હું આઈ થી કે અને પાથ ટુ થી પાથ પર રસ્તો કરવા માંગું છું અથવા હું i થી j થી અસ્તિત્વમાં રહેલો પાથ જોઈએ છે, જે ક્યારેય કનો ઉપયોગ કરતું નથી. તેથી, મારી પાસે આ બંને અસ્તિત્વમાં રહેલા પાથોને સંયોજિત કરવા માટે અને કાર્ય કરવું જોઈએ, પછી P જોઈએ અને પછી, મારી પાસે NOR ઑપરેશન છે જે કેસ 1 ને જોડે છે. તેથી, પહેલા આપણી પાસે ડબલ્યુ કે કેન 1, આઈજે અને પછી, અમે આ ડબલ્યુ કે કેન્યુ 1 ઉમેરીશું, ik પ્લસ ડબલ્યુ કે માઈનસ 1, કેજે અને અમે આ પર મિની લીધી. તેથી, અહીં આ પ્લસની જગ્યાએ, આપણે AND નો ઉપયોગ કરી રહ્યા છીએ અને min નો ઉપયોગ કરવાને બદલે, આપણે OR નો ઉપયોગ કરી રહ્યા છીએ, તમે જોઈ શકો છો કે એલ્ગોરિધમ બરાબર એ જ નથી. પરંતુ, તે ખૂબ જ સમાન છે, તેથી આ વોરશેલ(Warshall) દ્વારા સૂચિત એલ્ગોરિધમ હતું.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 18:43)
તેથી, તમે બધું ખોટું કરવા માટે પ્રારંભ કરો છો અને પછી, તમારે ફરીથી P માટે i જે 1 થી n બરાબર છે જે j બરાબર 1 થી n માટે છે. તેથી, તમે બધા પાથને ખોટામાં પ્રારંભ કરો છો, જ્યારે તમે સ્પષ્ટ રૂપે સેટ કરો છો તે સાર્થ પર ઝીરોથ લેવલ પાથ છે, જો કોઈ ધાર હોય. અને પછી, તમે કેટી લેવલ પાથને ક્યાં તો એમ કહી રહ્યા છો કે પહેલેથી જ કે માઈનસ 1 લેવલ પાથ છે અથવા મને 2 કે ઓછા માઈન્સ 1 લેવલ પાથો ઈન્ટરમિડિયેટ વર્ટેક્સ કે ઈન્ટરમિડિયેટ લેવલ કે મળી શકે છે. તેથી, તે એકદમ સમાન વસ્તુ છે, અમારી પાસે ફક્ત અને ઓછાના બદલે, ફક્ત અને આ અને આની કામગીરી છે. અને પછી, વૉર્સલ(Warshall)ના એલ્ગોરિધમનો ઉપયોગ સંક્રમિત બંધ અને ફ્લોયડ(Floyd)ને સામાન્ય બનાવવા માટે, તે ટૂંકા માર્ગો અને આ કાર્યોને નકારાત્મક ધારની હાજરીમાં રાખશે, તે કોઈ વાંધો નથી, ફ્લોયડ્સ(Floyd) એલ્ગોરિધમ ધારની કાળજી લેતી નથી, તે નકારાત્મક અથવા સકારાત્મક છે. તેથી, જ્યાં સુધી કોઈ નકારાત્મક ચક્ર હોય ત્યાં સુધી.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 05
લેક્ચર - 29
મિનિમમ કોસ્ટ સ્પૅનિંગ ટ્રીઝ(Minimum Cost Spanning Trees)

વેઈટ્ડ ગ્રાફ્સ(weighted graphs) પર ટૂંકા પાથ માટે વિવિધ અલ્ગોરિધમ્સ જોયા પછી, હવે આપણે કમ્પ્યુટિંગની એક જુદી જુદી સમસ્યા તરફ્ જઈએ છીએ, જેને શું કહેવાય છે મિનિમમ કોસ્ટ સ્પૅનિંગ ટ્રીઝ(Minimum Cost Spanning Trees).

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:12)

તેથી, સમસ્યાને પ્રેરિત કરવા માટે, ચાલો નીચેના ઉદાહરણને ધ્યાનમાં લઈએ. ધારો કે, અમે એક જિલ્લામાં છીએ જેમાં એક માર્ગ નેટવર્ક છે અને ખરાબ ચક્રવાત પછી રસ્તાઓ બધાને નુકસાન પહોંચાડવામાં આવે છે. તેથી, સરકારની પહેલી પ્રાથમિકતા રોડને પુનર્સ્થાપિત કરવી છે, જેથી જિલ્લાના વિવિધ ભાગોમાં રાહત મોકલી શકાય અને લોકો પણ ફ્રી ફ્રતા જઈ શકે. તેથી, પ્રાધાન્ય પર્યાપ્ત રસ્તાઓ પુનઃસ્થાપિત કરવાનું છે, જેથી દરેક જણ આજુબાજુ આગળ વધી શકે. તેથી, માર્ગને પુનર્સ્થાપિત કરવા માટે સરકારનો પહેલો માપદંડ કનેક્ટિવિટીને સુનિશ્ચિત કરવાનો છે. તેથી, કયા રસ્તાઓને સરકારે પ્રથમ પુનર્સ્થાપિત કરવું જોઈએ?

(સ્લાઈડસમયનો સંદર્ભ લો: 00:48)

તેથી, જો મુખ્ય માપદંડ ન્યૂનતમ કનેક્ટિવિટી છે, તો તે સ્પષ્ટ હોવું જોઈએ કે લૂપ શોધવા માટે રસ્તાઓ પુનઃસ્થાપિત કરવામાં કોઈ મુદ્દો નથી. દાખલા તરીકે, જો આપણે આ બધી ચાર રસ્તાઓને પુનર્સ્થાપિત કરીએ છીએ, તો આપણે આમાંથી કોઈપણ એક માર્ગ કાઢી નાખી શકીએ, 3 થી 4 અથવા 2 થી 3 કહીએ અને હજુ પણ આ ચાર નગરોમાંથી કોઈ પણ જિલ્લાના ચાર અન્ય શહેરોમાં જઈ શકે છે. તેથી, લૂપથી ધારને દૂર કરવું ગ્રાફ્ને ડિસ્કનેક્ટ કરી શકતું નથી અને અમારું લક્ષ્ય આ ગ્રાફ્ની અંદરના કેટલાક પેટા સમૂહને શોધવાનું છે જે આ રીતે જોડાયેલ છે કે આ એકદમ ન્યૂનતમ સમૂહનો સેટ છે. તેથી, આપણે જે જોઈએ છીએ તે આ મૂળ ગ્રાફ્ના જોડાયેલા પેટા ગ્રાફ્ છે જેમાં કોઈપણ લૂપ્સ નથી જે વિશ્લેષક છે અને આ બરાબર એક વૃક્ષ કહેવાય છે. તેથી, વ્યાખ્યા દ્વારા વૃક્ષ એ જોડાયેલ વિશ્લેષણાત્મક ગ્રાફ્ છે. અને ખાસ કરીને, અમે મનસ્વી ગ્રાફ્ માટે પ્રારંભ કરીએ છીએ અને અમે આલેખની અંદર બેઠેલા વૃક્ષને શોધી રહ્યા છીએ, જે ધારની સંખ્યાના સંદર્ભમાં પેટા ગ્રાફ્ છે, જે મૂળ ગ્રાફ્માંના બધા શિરોબિંદુને જોડે છે. તેથી, આવા વૃક્ષને સ્પૅનિંગ ટ્રીઝ(Spanning Trees) કહેવામાં આવે છે, તે મૂળ ગ્રાફ્ના શિરોબિંદુઓને ફેલાવે છે, પરંતુ તે ત્રણ કિનારીના પેટા સમૂહને બહારના વૃક્ષની રચના કરે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:00)

તેથી, ઉદાહરણ તરીકે, આ ગ્રાફ્માં, એક ફેલાયેલ વૃક્ષ જે આપણે બનાવી શકીએ, અહીં બતાવેલ અન્ય લાલ ધાર, 1 થી 2, 2 થી 3, 3 થી 4 અને 4 થી 5. અલબત્ત, આપણે દાખલા તરીકે, અન્ય વિસ્તરણ વૃક્ષ બનાવી શકે છે, આ એક લીલો છે, આ 1 થી 3, 2 થી 3, 2 થી 5 અને 4 થી 5 છે. તેથી, આપેલા ગ્રાફ્ પર ઘણા બધા વૃક્ષો બનાવી શકાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:22)

હવે, ધારો કે ગ્રાફ પણ વજન ધરાવે છે. આ ઉદાહરણમાં, ઉદાહરણ તરીકે વજન, રસ્તાના સમારકામની કિંમત હોઈ શકે છે. તેથી, રસ્તાને પુનર્સ્થાપિત કરવાનો ખર્ચ થયો છે અને હવે સરકાર ફક્ત કનેક્ટિવિટીને પુનઃસ્થાપિત કરવા નથી માંગતી, પરંતુ તેનો અર્થ એ છે કે લઘુતમ ખર્ચ પર કરવા માંગે છે. તેથી, જો, ઉદાહરણ તરીકે, સરકારે રસ્તાઓના આ વૃક્ષને સમારકામ કરવાનું પસંદ કર્યું, તો કુલ કિંમત 18 વત્તા 6, 24 વત્તા 70, 94 વત્તા 20, 114 છે. તેથી, આ સંગ્રહ વૃક્ષને 114 ની કિંમત લાગી શકે છે. . બીજી બાજુ, જો સરકાર લીલા વૃક્ષની વૃક્ષ પસંદ કરે છે, તો ખર્ચ 10 વત્તા 6 થી 16 થાય છે 16 વત્તા 20 છે 36 વત્તા 8 એ 44 છે. તેથી, હવે વૃક્ષો અલગ અલગ કરવાથી વિવિધ ખર્ચ થશે અને લક્ષ્ય ઘટાડવું પડશે લઘુત્તમ દીઠ ખર્ચ. આ વિશિષ્ટ ઉદાહરણમાં, તમે આ લીલા વૃક્ષને ચકાસી શકો છો, જેની કિંમત 44 એ આ વિશિષ્ટ ગ્રાફ પર ખરેખર મિનિમમ કોસ્ટ સ્પૅનિંગ (Minimum Cost Spanning) માટે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:22)

તેથી, અમે લઘુતમ ખર્ચાઓના વૃક્ષોની ગણતરી કરવા માટે ઍલ્ગોરિધમ્સ તરફ આગળ વધીએ તે પહેલા, ચાલો આપણે વૃક્ષો વિશેના કેટલાક મૂળભૂત તથ્યો જોઈએ. તેથી, યાદ રાખો કે વ્યાખ્યા દ્વારા, એક વૃક્ષ એ જોડાયેલ વિશ્લેષણાત્મક ગ્રાફ છે. તેથી, સામાન્ય રીતે ગ્રાફમાં n શિરોલંબ હશે, તેથી દાવા એ છે કે કોઈપણ વૃક્ષની બરાબર n ઓછા 1 ધાર હોય છે. તેથી, આ સાબિત કરવું ખૂબ જ સરળ છે, તેને સાબિત કરવાના ઘણા જુદા જુદા રસ્તાઓ છે, અહીં તેના વિશે વિચારવાનો એક રસ્તો છે. તેથી, ધારો કે આપણી પાસે વૃક્ષ છે, તેથી શરૂઆતમાં વૃક્ષ વ્યાખ્યાથી જોડાયેલું છે, તેથી આખું ગ્રાફ એક જોડાયેલ ઘટક બનાવે છે. યાદ રાખો, જ્યારે આપણે પ્રથમ પહોળાઈ અને ઊંડાણને પ્રથમ શોધ વાંચીએ છીએ, ત્યારે અમે કહ્યું હતું કે અમે નોડ લઈ શકીએ છીએ અને તેનાથી કનેક્ટ થયેલ બધું જોઈ શકીએ છીએ અને તે જોડાયેલું છે

(સમયનોસંદર્ભ લો: 04:10).

તેથી, આ વૃક્ષ એકલ જોડાયેલા ઘટકને વ્યાખ્યાયિત કરે છે, જો તમે તેને અલગતામાં ગ્રાફ તરીકે જોશો. હવે, કારણ કે તે એક વૃક્ષ છે, જો મારી પાસે આઇ થી જી સુધીની ધાર હોય, તો બીજા કોઈ પણ ધાર દ્વારા i થી j પર કોઈ અન્ય માર્ગ ચાલી શકતો નથી, કારણ કે જો તે પાથ પ્લસ ઉપરાંત આ ધાર ચક્રીય બનશે. તેથી, જો થી j થી ધાર હોય અને હું તેને દૂર કરીશ, તો વ્યાખ્યા દ્વારા આ ઘટક i અને ઘટક સમાવિષ્ટ j સમાવતું હોવું જોઈએ. તેથી, જો હું એક ઘટક સાથે શરૂ કરું છું, તો હવે મને બે ઘટકો મળે છે. તેથી, હું મારા વૃક્ષમાંથી ગ્રાફમાંથી પ્રથમ ધારને કાઢી નાખું છું અને મારી પાસે એક ઘટક વધુ છે. હવે, હું એજ દલીલ દ્વારા એક વધુ ધાર કાઢી નાખું છું જે કોઈપણ ભાગ જે ધાર ધરાવે છે તે ફરીથી વિભાજિત થશે. તો, દર વખતે જ્યારે હું ધારને કાઢી નાંખું છું, ત્યારે મારી પાસે સંખ્યાબંધ ઘટકો વધારો થાય છે, પરંતુ પછી, મને ખબર છે કે અંતમાં, જો મારી પાસે n એકબીજાવાળા શિરોબિંદુ હોય, તો મારી પાસે પૂર્ણાંક ઘટક હોઈ શકે છે. જો મારી પાસે n શિરોબિંદુ હોય તો, મારી પાસે n ઘટકો કરતા વધુ ન હોઈ શકે. તેથી, હું ફક્ત એક વખત આ n ઓછા કરી શકું છું. તેથી, હું એક વૃક્ષથી પ્રારંભ કરું છું, જ્યાં સુધી બધું ડિસ્કનેક્ટ થઈ જાય ત્યાં સુધી હું ધારને કાઢી નાખું છું, હું આ માત્ર 1 ઓછા કરી શકું છું અને મારે તે કરવું જ જોઈએ, કારણ કે દરેક વસ્તુ ડિસ્કનેક્ટ થઈ જાય છે, તેથી વૃક્ષનું બરાબર n ઓછા 1 ધાર હોવું જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:22)

હવે, જો હું એક વૃક્ષ લઈશ અને પછી હું ધાર ઉમેરીશ, તો તે એક ચક્ર બનાવશે, આપણે આ પહેલાની દલીલમાં પહેલાથી જ જોયું છે, આપણે કહીએ છીએ, તેથી મને લાગે છે કે મારી પાસે એક વૃક્ષ છે, તેથી સામાન્ય રીતે વૃક્ષ એવું કંઈક જુએ છે. તેથી, તે એક ગ્રાફ છે, તેમાં એક પ્રકારનું વધુ ચક્ર છે, પરંતુ ઘણી બધી વસ્તુઓ શાખાઓથી બહાર આવી રહી છે. હવે, જો હું કોઈ વૃક્ષ બનાવીશ તો, હું ધારું છું કે હું તેમાં ઉમેરો કરું છું અને વિચારી રહ્યો છું કે હું અહીં કેટલાક અને કેટલાક

જેનો ઉપયોગ કરું છું, અમે ધાર ઉમેરવાનો નિર્ણય લઈ શકીએ છીએ. તેથી, આપણે જાણીએ છીએ કે તે એક વૃક્ષ છે, ત્યાં પહેલેથી જોડાણ છે. તેથી, ત્યાં કેટલાક પાથ છે જે આ કિસ્સામાં i થી j થી આ શિરોલંબમાં છે. તેથી, તેથી, તે પાથ પૃષ્ઠ વત્તા આ ધાર એસાયકલિક બનાવે છે. તેથી, એક વૃક્ષમાં મારી પાસે બરાબર n માઈનસ 1 કિનારીઓ છે અને જ્યારે હું કોઈ વધારાનો ધાર ઉમેરે છે, હું કઈ ધાર ઉમેરી શકું, તે ચોક્કસપણે એક સાયકલ બનાવશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:06)

તેથી, આ બધી વ્યાખ્યાઓનો બીજો પરિણામ એ છે કે કોઈપણ બે રસ્તાઓ વચ્ચે, વૃક્ષમાં કોઈપણ બે શિરોબિંદુઓ, ફક્ત એક અનન્ય પાથ હોઈ શકે છે. તેથી, વાસ્તવમાં ત્યાં બે રસ્તા છે, તેથી ચાલો આપણે બે શિરોબિંદુઓ જોઈએ, અહીં આપણે તેમને i અને j તરીકે દોર્યા છે અને ધારો કે ત્યાં બે ભાગ છે. તેથી, જો હું બે ભાગોને અનુસરું છું, તો તે ખૂબ જ સ્પષ્ટ છે કે ત્યાં ત્યાં બે જુદા જુદા રસ્તાઓ છે, ત્યાં કેટલાક આંટીઓ વચ્ચે ક્યાંક હશે. તેથી, નોંધ લો કે તેને i અને j સહિત લૂપ ન હોવા જોઈએ, તે i અને j વચ્ચે ક્યાંક હોઈ શકે છે, પરંતુ જો તમે બધા કેસો કાળજીપૂર્વક ધ્યાનમાં લેતા હોવ તો, તમે પોતાને સમજી શકો છો, ત્યાંથી લૂપ બનાવ્યાં વગર i થી j બે અલગ પાથ હોવાનો કોઈ શક્યતા નથી, તો ગ્રાફ હવે એસાયક્લિક(acyclic) નથી, તેથી તે એક વૃક્ષ નથી જે આપણી ધારણા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 06:52)

તેથી, આપણે ખરેખર નીચે આપેલા દાવા પ્રમાણે આપણી પાસે આ ત્રણ ગુણધર્મો છે, G એ જોડાયેલ છે, જી એ જ્ઞાનકોશ છે અને જી પાસે n ઓછા 1 કિનારી છે, તો આમાંથી કોઈપણ બે ત્રીજા સૂચવે છે. તેથી, જી જોડાયેલ છે અને જી વ્યાખ્યાયિત કરીને જ્ઞાનકોશ છે તે એક વૃક્ષ છે, આપણે ફક્ત પ્રથમ દલીલો બતાવી છે કે કોઈપણ વૃક્ષને n ઓછા ખૂણા છે. તેથી, હકીકત એ છે કે પ્રથમ બે એ ત્રીજાને સૂચવે છે તે આપણે જે બતાવ્યું છે તે છે. હવે, તમે ઔપચારિક સાબિત કરવા માટે પોતાને સરળતાથી સમજી શકો છો કે જી જ્ઞાનકોશી છે અને તેની પાસે 1 ની માત્રા છે, પછી હકીકતમાં, તે જોડાયેલું હોવું જોઈએ, બધું જ જોડાયેલું હોવું જોઈએ. અને છેવટે, જો જી કનેક્ટ થયેલ છે અને તેની પાસે n ઓછા 1 ધાર છે, તો તે એસાયક્લિક ગ્રાફિક હોઈ શકે છે, તેમાં કોઈપણ ગ્રાફ્સ હોઈ શકતા નથી. તેથી, આ વૃક્ષો જોવાની રીત અલગ અલગ હોય છે અને કેટલીકવાર અમે એક લાક્ષણિકતા અથવા અન્ય લાક્ષણિકતાનો ઉપયોગ કરી શકીએ છીએ. તેથી, જ્યારે આપણે સામાન્ય રીતે વૃક્ષો વિશે વાત કરીએ ત્યારે આ બાબતોને આપણા ધ્યાનમાં રાખવી ઉપયોગી છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:45)

તેથી, અમારું લક્ષ્ય હમણાં જ મિનિમમ કોસ્ટ સ્પૅનિંગ ટ્રીઝ(Minimum Cost Spanning Trees)ને બનાવવાનું છે. તેથી, ત્યાં કુદરતી રીતે લોભી વ્યૂહરચનાઓ છે જેનો કોઈ વિચાર કરી શકે છે. એક છે, કારણ કે તમે આ લઘુતમ તબક્કાની સાથે પ્રારંભ કરવા માટે અને ઓછામાં ઓછા વૃક્ષનું નિર્માણ કરવા માટે ન્યૂનતમ ખર્ચ વૃક્ષ શોધી રહ્યા છો. તેથી, અમે અસ્તિત્વમાંના વૃક્ષ પર ધાર ઉમેરવાનું ચાલુ રાખીએ છીએ, જેથી નવું ગ્રાફ એક વૃક્ષ રહે અને તે શક્ય તેટલું ઓછું વધશે. આનાથી ઍલ્ગોરિધમનો વધારો થશે જે પ્રિમના અલ્ગોરિધમનો કહેવાય છે. તે એક જ સ્રોત સાથે ડિજ્સ્ટ્રાના ટૂંકા પાથ એલ્ગોરિધમનો સમાન હશે. અમારી પાસે અન્ય વ્યૂહરચના હોઈ શકે છે જે કિનારીઓના ચઢતા ક્રમમાં ગોઠવે છે અને તેમને ઉમેરવાનું ચાલુ રાખે છે, ત્યાં સુધી આપણે વૃક્ષની લાક્ષણિકતાનું ઉલ્લંઘન કરતા નથી. હવે, આ પ્રીમ્સ(Prim's) એલ્ગોરિધમથી જુદું છે, કારણ કે અહીંથી આપણે શરૂ કરવા માટે એક વૃક્ષ બનાવ્યું નથી, અમે ધાર ઉમેરવાનું ચાલુ રાખીએ છીએ, જેથી આપણે ચક્ર બનાવતા નહી, પરંતુ અમે ધારના જૂથોને છુટા કરી શક્યા હોત, પરંતુ આખરે બધા જોડાયેલા હશે એક વૃક્ષ બનાવવા માટે. તો, આપણે આને આગળના 2 પ્રવચનોમાં વધુ વિગતવાર જોઈશું, પરંતુ ચાલો આ બંને વ્યૂહરચનાઓ કેવી રીતે કાર્ય કરે છે તે વિશેનું પહેલું ઉદાહરણ જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:54)

તેથી, ચાલો આ બે એલ્ગોરિધમ્સને ધ્યાનમાં લઈએ પછી તરત જ તેમની પાસે વધુ વિગતો આવશે, તેથી ચાલો આપણે પ્રીમ્સ(Prim's)એલ્ગોરિધમનો પ્રારંભ કરીએ. યાદ રાખો કે પ્રીમ્સ(Prim's)ની એલ્ગોરિધમની વ્યૂહરચના નાના કદના કિનારીથી શરૂ થવી જોઈએ અને ત્યારબાદ વધતી જતી વૃક્ષ ઉગાડવી. તેથી, આપણે અહીં એક નાનો ધાર સાથે પ્રારંભ કરીએ છીએ જે 2 અને 3 ની વચ્ચેના ધાર વજન 6 છે. હવે, આપણે વર્તમાન વૃક્ષને જોવું જોઈએ જેમાં આ ક્રમમાં ગ્રાફનો સમાવેશ થાય છે અને તે નક્કી કરવા માટે આ ચાર ધારમાંથી એક ઉમેરવાનું છે કે નહીં તે નક્કી કરવું. . આપણે અહીં તે ધાર ઉમેરી શકતા નથી, અમે આ ધાર ઉમેરી શકતા નથી, કારણ કે તે વૃક્ષ બનાવશે નહીં, તે આ ધારથી છુટા થશે. તેથી, આપણે આમાંથી કોઈપણ ઉમેરી શકીએ છીએ, પરંતુ આપણે સૌથી નાનું પસંદ કરીએ છીએ. તેથી, આ કિસ્સામાં, આપણે વજન 10 સાથે એજ પસંદ કરીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:37)

તેથી, વૃક્ષનું આગળનું પગલું એજ 1, 2 અને હવે ઉમેરવાનું છે, આપણી પાસે આ વૃક્ષ છે. હવે, જો આપણે શક્ય કિનારીઓ જોઈશું જે આપણે ઉમેરી શકીએ છીએ, તો અમારી પાસે આ ધાર છે, અમારી પાસે આ ધાર છે અને અમારી પાસે આ ધાર છે. હવે, આમાંની સૌથી નાની ધાર રાહ 18 વડે છે, પરંતુ જો મારી પાસે તે હોય તો આપણે એક ચક્ર મેળવીશું. તેથી, ઉમેરવા માટે આ એક સરસ ધાર નથી. તો, તેથી, આપણે બીજા એકને ફરીથી ઉમેરવા જોઈએ, ફરીથી નાનાને પસંદ કરવા માટે, આ કિસ્સામાં વજન 20 સાથે લેબલ ધાર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:06)

તો પછી, આપણને આ વૃક્ષ મળે છે હવે આ આકાર, આ એક વૃક્ષ છે. હવે, આપણે ઉમેરી શકીએ છીએ, આપણે આ ઉમેરી શકતા નથી આપણે જાણીએ છીએ. તો, આપણે કિનારી 70 અથવા કિનારીઓ સાથે વજન 8 ઉમેરી શકીએ છીએ અને દેખીતી રીતે 8 નાની છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:23)

તેથી, છેલ્લે, આપણે ઉમેર્યું કે આમાં એક વૃક્ષ છે જે આપણે મેળવી શકીએ છીએ. તેથી, આ અંતિમ વૃક્ષ છે, અમે પ્રીમ્સ(Prim's)ની એલ્ગોરિધમથી મેળવીએ, ચાલો આપણે સૌથી નાનો ધાર અને વધતી જતી વૃક્ષને વધારીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 10:35)

અમે જે અન્ય વ્યૂહરચના કહી હતી તે ચડતા ક્રમમાં ધાર સાથે શરૂ થવાનું હતું. તો, આપણે 18 થી 8 ની સાથે કિનારીઓથી પ્રારંભ કરીએ, પછી ત્યાં અર્થ છે. તો, આપણી પાસે આ પહેલી છે, આ બીજું છે અને આ ત્રીજુ અને બીજું છે. તેથી, આપણે આ ક્રમમાં, 1, 2, 3, 4, 5 અને 6 માં કિનારીઓને ધ્યાનમાં લઈએ છીએ. તેથી, આપણાં વજન 18, 20, અને પછી કૃપા કરીને 70 છે. તેથી,દરેક 6 કિનારીઓ જ્યારે આપણે આ ક્રમમાં તેમનો વિચાર કરીએ ત્યારે, જો 6 નાના હોય, તો આપણે ઉમેરવું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:07)

તો, આ આપણા વૃક્ષનું સ્ટાર્ટર છે, હવે પછીનું એક 8 છે અને તે ચક્ર બનાવતું નથી, તે વૃક્ષની લાક્ષણિકતાનું ઉલ્લંઘન કરતું નથી. તેથી, આપણે તે ઉમેરીએ, હવે આ મુદ્દે પ્રીમ્સ(Prim's) અને ક્રુસ્કલના(Kruskal's) એલ્ગોરિધમ વચ્ચેના નિર્ણાયક તફાવતને ધ્યાનમાં લો, આપણી પાસે એક વૃક્ષ નથી, આપણી પાસે બે અલગ અલગ વૃક્ષો છે. તેથી, આપણી પાસે આ ગ્રાફમાં બે ભિન્ન વિશ્લેષણાત્મક ઘટક છે જે એકબીજા સાથે જોડાયેલા નથી, પરંતુ અમે ફક્ત કિનારીઓનો ક્રમ લઈ રહ્યા છીએ. તેથી, આગળ આપણે જોશું કે 10 એ આગળની ધાર છે જે આપણે ઉમેરી શકીએ છીએ, આ એક ચક્ર બનાવતું નથી. તેથી, અમે તેને ઉમેરીએ છીએ, તેથી કેટલાક અર્થમાં, આપણે આ ઘટક ઉગાડ્યા છે અને તે ઘટક ઉગાડ્યો છે. હવે, આગળનું એક 18 હશે, પણ જો તમે 18 ઉમેરો, તો તે એક ચક્ર બનાવશે. તેથી, આપણે 18 અવગણો, આપણે આગળના સ્થાને જઈએ, આ 20 છે, 20 બરાબર છે અને 20 વાસ્તવમાં એક વૃક્ષ બનાવવા માટે બે ઘટકને જોડશે. તેથી, આપણે 20 ઉમેરીએ, હવે આપણે પૂર્ણ કર્યું છે, કારણ કે આપણે n ઓછા 1 કિનારીઓ ઉમેરી છે, ત્યાં પાંચ શિરોલંબ છે, આપણને 4 કિનારીઓ મળી છે અને તેથી આપણને ચોક્કસપણે એક વૃક્ષ મળી ગયું છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફ્ેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 04
મોડ્યુલ - 06
લેક્ચર - 30

સ્પૅનિંગ ટ્રીઝ: પ્રીમ્સની અલ્ગોરિધમ (Spanning trees: Prim's algorithm)

તેથી, અમે વેઈટ્ડ ગ્રાફ્સ(weighted graphs)માં મિનિમમ કોસ્ટ સ્પૅનિંગ ટ્રીઝ(Minimum Cost Spanning Trees) બનાવવાની સમસ્યા જોઈ રહ્યા છીએ. અમે કહ્યું કે ત્યાં અમારી પાસે બે મૂળભૂત વ્યૂહરચનાઓ છે જે આ કરવા માટે વિચારી શકે છે. પ્રથમમાં પ્રીમ્સ(Prim's)ની એલ્ગોરિધમ તરીકે ઓળખાતા ઍલ્ગોરિધમ તરફ્ દોરી જાય છે, અને બીજો એક ક્રુસ્કલના(Kruskal's)ના અલ્ગોરિધમ તરીકે ઓળખાતા ઍલ્ગોરિધમ તરફ્ દોરી જાય છે. તેથી, આ ભાષણમાં આપણે પ્રીમ્સ(Prim's)ની એલ્ગોરિધમ જોશું.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:20)

તેથી, સમસ્યા પ્રમાણે છે. અમારી પાસે વજનવાળા અણધાર્યા ગ્રાફ્ છે. તેથી, V શિરોલંબનો સમૂહ છે, E એ ધારનો સમૂહ છે અને w એ વેઈટ ફ્ંક્શન છે. અમે ધારી લીધું કે G જોડાયેલું છે, કારણ કે G જોડાયેલું નથી અને સ્પૅનિંગ ટ્રી બનાવવાની કોઈ રીત નથી. વૃક્ષને ફ્ેલાવવાનું, યાદ રાખવું તે કિનારીઓનું પેટા સમૂહ છે જે g માં બધા શિરોબિંદુઓને જોડે છે. તેથી, જી પહેલેથી જ જોડાયેલ નથી, તો ત્યાં કોઈ રીત નથી કે આપણે ખરેખર કિનારીના સબસેટનો ઉપયોગ કરીને કનેક્ટ કરી શકીએ. તેથી, જી એક જોડાયેલ વજનવાળા અણધાર્યા ગ્રાફ્ છે, અને હવે અમે ન્યૂનતમ વજન સાથે સ્પૅનિંગ ટ્રી ઓળખવા માંગીએ છીએ. તેથી, પ્રીમ્સ(Prim's)ની એલ્ગોરિધમની આ વ્યૂહરચના ન્યૂનતમ ખર્ચના કિનારીથી શરૂ થાય છે, અને વર્તમાન વૃક્ષ સાથે જોડાયેલા નાના ધાર સાથે વૃક્ષને વિસ્તૃત રાખે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:04)

તેથી, અહીં પ્રીમ્સ(Prim's)ના અલ્ગોરિધમનો ઉચ્ચ સ્તરનો સંસ્કરણ છે. તેથી, અમે ન્યૂનતમ ખર્ચ ધાર સાથે પ્રારંભ કરીએ છીએ અને અમે તેને સૂચિમાં ઉમેરીએ છીએ. તેથી, TE એ ધારની સૂચિ છે જે એક વૃક્ષ બનાવે છે. તેથી, આપણે વૃક્ષને ધારના સંગ્રહ તરીકે વર્ણવીશું અને પછી આપણે નોંધ કરીશું કે આપણે વૃક્ષ પર i અને j ઉમેર્યા છે. તેથી, હું અને જે હવે જોડાયેલા છે. તેથી, આ n ઓછા 2 શિરોબિંદુઓને છોડે છે જેને જોડવું પડશે. તેથી, આપણને 2 ગુણ્યા ઓછા 2 વખત કરવું પડશે. તેથી, n ઓછા 2 વખત આપણને ધાર ઉમેરવો પડશે. દર વખતે આપણે ધાર ઉમેરીએ છીએ; એક વધુ શંકુ જોડાયેલું હશે. તેથી, આપણે જાણીએ છીએ કે તે પછી ઘણા ધાર પછી, તમારી પાસે એક વૃક્ષ હશે. યાદ રાખો કે વૃક્ષની તદ્દન n ઓછા 1 ધાર છે. તેથી, અમે ન્યૂનતમ ખર્ચ ધાર સાથે પ્રારંભ કરીને પ્રથમ ધાર ઉમેર્યો છે. તેથી, આપણે n ઓછા 2 ઉમેરી શકીએ છીએ. તેથી, આપણે શું કરીએ છીએ, n ઓછા 2 વખત, આપણે સૌથી નાનો ધાર પસંદ કરીએ છીએ જેની પાસે વૃક્ષમાં એક અંત બિંદુ છે અને વૃક્ષની બહાર એક અંત બિંદુ છે. તેથી, આ એક vertex v હવે છે જે વૃક્ષ

સાથે જોડાયેલ નથી. તેથી, અમે તેને જોડીએ છીએ. તેથી, આપણે આ નવી ધારને વૃક્ષની કિનારીઓની સૂચિમાં જોડીએ છીએ અને આ પટ્ટાને વૃક્ષનાં શિરોબિંદુઓની સૂચિમાં ઉમેરીએ છીએ અને અંતે આ n ઓછા 2 વખત કર્યા પછી, તે દાવો કરે છે કે આપણે બધા ધારને જોડ્યા છે, અમારી પાસે સ્પૅનિંગ ટ્રી છે અને વધુ દાવો કે અમે ધાર પોઈન્ટ ઉમેરવા માટે ન્યૂનતમ ખર્ચ ધાર પસંદ કરી રહ્યા છીએ. એકંદર વસ્તુ એ ન્યૂનતમ કિંમતનો વિસ્તાર છે. અલબત્ત, અમે આ બધાને સાબિત કરીશું નહીં, પરંતુ આ જ છે જેનો મુખ્ય હેતુ એલ્ગોરિધમનો હેતુ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:33)

તેથી, આપણે કંઈક સાબિત કરવાની શા માટે જરૂર છે? ઠીક છે, તમે જોઈ શકો છો કે ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ જેવા,પ્રીમ્સ(Prim’s)નું એલ્ગોરિધમ ખૂબ લોભી એલ્ગોરિધમ છે, સાચું. દરેક બિંદુએ, આપણે વૃક્ષને કેવી રીતે વિસ્તૃત કરવું તે નક્કી કરવું પડશે. તેથી, આપણે આજુબાજુના પડોશમાં જોઈએ છીએ કે વર્તમાન વૃક્ષ, આપણે નજીકના શિરચ્છેદ માટે છીએ જે વૃક્ષ સાથે જોડાયેલ છે, પરંતુ સૌથી નાનો ધાર અને અમે તેને ઉમેરીએ છીએ. તેથી, આ એક સ્થાનિક પસંદગી છે, અને પછી અમે સ્થાનિક પસંદગીઓના આ અનુક્રમોને ચાલુ રાખીએ છીએ અને છેવટે, અમે એક વૃક્ષો પર વૈશ્વિક સ્તરે પહોંચીએ છીએ અને દાવો કે વૈશ્વિક સ્તરે અમે શ્રેષ્ઠ શક્ય વૃક્ષ બનાવી દીધું છે. તેથી, આ હંમેશાં લોભી અલ્ગોરિધમનો દાખલો છે જ્યાં તમે સ્થાનિક પસંદગીઓનો ક્રમ બનાવો છો. ક્યારેય પાછા નહીં જાઓ અને ફરીથી વિચાર કરો અને છેલ્લે, ગ્લોબલ શ્રેષ્ઠ પ્રાપ્ત કરો અને ઘણીવાર આપણે ડિજેસ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ સાથે અગાઉ ઉલ્લેખ કર્યો છે, આવી વ્યૂહરચના તમને યોગ્ય વસ્તુ આપી શકશે નહીં. તેથી, તમારે હંમેશાં સાબિત કરવું પડશે કે આ કાર્ય કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:23)

તેથી, પ્રીમ્સ(Prim’s)ની એલ્ગોરિધમ સાચી અને સાચી સાબિત કરવા માટે, આપણે ક્રુસ્કલના(Kruskal’s)ના અલ્ગોરિધમનો સાબિત કરવા માટે આનો ઉપયોગ કરીશું, પછીથી સુધારીશું. અમે તમામ ન્યુનત્તમ વિભાજક લેમ્માની ખૂબ ઉપયોગી ટિપ્પણી સાબિત કરીએ છીએ. તો, ચાલો ધારીએ કે આપણી પાસે આ વજનવાળા અણધાર્યા ગ્રાફ છે અને આપણે શિરોલંબ વિરુદ્ધના સમૂહ તરફ જોઈએ છીએ, અને આપણે માનીએ છીએ કે તે બે પાથોમાં વહેંચાયેલું છે. તેથી, આને પાર્ટીશન કરવાનું કહેવામાં આવે છે. તેથી, ત્યાં બે અલગ અલગ જોડાણ માર્ગો છે જે હું કહીશ U અને w , અને હું ધારું છું કે આ બંને ખાલી નથી. તો, ઓછામાં ઓછું એક કર્ણ છે અને એક ભાગ છે. હવે, મને આ પાર્ટીશન તરફ જાય છે તે સૌથી નાનો ધાર જોઈએ. યાદ રાખો કે આખું ગ્રાફ જોડાયેલું છે. તેથી, તેઓ તમારી પાસેથી જવાની રીત હોવા જોઈએ. હું તમને જે રીતે લઈ શકું તેમાંથી કેટલાક માર્ગો. ચાલો મને સૌથી નાનો ધાર તપાસો. ચાલો હું તમને નાના u અને નાના w અંત બિંદુ પર લઈ જાવ. તેથી, હવે, દાવો એ છે કે વૃક્ષની દરેક લઘુતમ કિંમત આ ધારને શામેલ કરવી આવશ્યક છે. આ એક ખૂબ જ શક્તિશાળી દાવો છે. અલબત્ત, ત્યાં એક બાજુની સ્થિતિ છે જે આપણે કોઈ ક્ષણ માટે ધારતા નથી કે કોઈ પણ બે ધાર સમાન છે. આપણે પછીથી જોશું કે આપણે કેવી રીતે આ સ્થિતિને આરામ કરીશું. તેથી, શરત હેઠળ કે કોઈ એક જ વજન પર ધાર નથી, લઘુતમ વિભાજક લેમ્મા એ છે કે જ્યારે પણ તમે ખાલી ન હોય તેવા ભાગો કરવા માટે અમે અલગ કરીએ ત્યારે, આ બંને ભાગોને જોડતા સૌથી નાનો ધાર દરેક ફેલાયેલી વૃક્ષમાં રહેવો જોઈએ. લઘુત્તમ ખર્ચ વૃક્ષ ફેલાવો.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:43)

તો હવે કેસ શા માટે છે? તેથી, ચાલો ધારીએ કે આપણી પાસે આ બે ભાગ છે. તેથી, હું આ ભાગને એક પીળી વસ્તુ કહીશ અને ચાલો આપણે આને u કહીયે અને જેના માટે સીમા દોરી નથી, આ v છે અને આ w છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:00)

તેથી, ચાલો હવે યુ અને ડબલ્યુને જોડીને સૌથી નાનો કિનારી જોઈએ. તેથી, હવે દાવો એ છે કે આ દરેક લઘુત્તમ ખર્ચમાં વૃક્ષ હોવા જોઈએ. તેથી, ધારો કે તે નથી. તો, પછી ધારો કે ત્યાં ઓછામાં ઓછા ખર્ચની કિંમત વૃક્ષ હોવા જોઈએ, કારણ કે આપણે જાણીએ છીએ કે ગ્રાફ જોડાયેલ છે. તેથી, જ્યાં સુધી આ ધાર શામેલ ન હોય ત્યાં સુધી વૃક્ષ માં ફેલાતા લઘુતમ ખર્ચને ધ્યાનમાં રાખીને ઘણા વૃક્ષો ફેલાય છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:28)

તેથી, તે વૃક્ષમાં તમારે વૃક્ષ સાથે જોડાયેલું હોવું જોઈએ, કારણ કે કોઈપણ ફેલાયેલ વૃક્ષ બધા શિરોબિંદુઓને જોડે છે. તેથી, મારા અનુમાનિત વૃક્ષ માં યુ માંથી ડબલ્યુ માર્ગ છે જે આ ધારનો સમાવેશ કરતું નથી. તેથી, હવે એવો દાવો છે કે જો હું તે ચોક્કસ વૃક્ષને લઈશ અને પછી હું તમને ધાર પ્રીમ્સ(Prim's) વી પ્રીમ્સ(Prim's)ને દૂર કરીશ અને તેને ધાર દ્વારા બદલી નાંખો, તો આપણને એક નવું વૃક્ષ મળશે. મને વૃક્ષ ટી વડા મળે છે. તેથી, ટી પ્રીમ્સ(Prim's) એ ટી મીનસ ધાર યુ પ્રીમ્સ(Prim's) વી પ્રાઈસ વત્તા ધાર યુવી છે, પરંતુ હવે ધારણા મુજબ યુવી સૌથી નાનો શિખરો હતો, તમારી અંદરની અંદરથી જ નાના વજનની ધાર છે. તેથી, તેથી, uv ને તમે પ્રીમ્સ(Prim's) વી પ્રીમ્સ(Prim's) કરતા કડક રીતે ઓછું વજન આપ્યું છે. તેથી, ટી પ્રીમ્સ(Prim's) ટી કરતાં સખત ઓછું વજન ધરાવે છે અને તમે ચકાસી શકો છો કે બીજું બધું કનેક્ટ થયેલું છે કારણ કે જોડાયેલું છે. પ્રીમ્સ(Prim's) લાંબા સમય માટે છે અને તેથી, અન્ય બધા શિરોબિંદુઓ જે T દ્વારા જોડાય છે તે જોડાયેલ છે ટી પ્રીમ્સ(Prim's)થી. તેથી, ટી પ્રીમ્સ(Prim's) માન્ય ફેલાયેલું વૃક્ષ છે. તે નાની કિંમતના છે અને તેથી ટી લઘુત્તમ ખર્ચવાળું વૃક્ષ ન હોત. તેથી, આ એક પુરાવો છે કે પાર્ટિશનની અંદરથી પાર્ટીશનની બહારની સૌથી નાની કિંમત ધાર દરેક લઘુત્તમ ખર્ચવાળા વૃક્ષમાં રહેલી છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:45)

આગળ વધતા પહેલા, હું ફક્ત એક નાનકડી ટિપ્પણી કરવા માંગું છું. તેથી, જ્યારે આપણે આ પ્રમેયને થોડી સાબિત કરીશું ત્યારે સાવચેત રહેવું પડશે. તેથી, તે સાચું છે કે અંદરથી બહારની બધી ધાર વચ્ચે, યુવી સૌથી નાનું છે. તેથી, આપણે ફક્ત કહેવું લલચાવી શકીએ છીએ, તેથી યુવી સૌથી નાનું છે, મારા વૃક્ષમાં કોઈ પણ ધાર પસંદ કરો ટી મુખ્ય વૃક્ષ જે ટી અંદરથી બહાર જાય છે, પછી બદલો. તેથી, ઉદાહરણ તરીકે, અમે આ ધારને આકસ્મિક રીતે પસંદ કરી શકીએ છીએ અને તેને આ ધારથી બદલી શકીએ છીએ. આ ધારને પસંદ કરીને નોંધો અને આ ધાર સાથે બદલો, પછી કદાચ w પ્રીમ્સ(Prim's) મેળવવાનો બીજો કોઈ રસ્તો નથી. તેથી, તે ખૂબ જ નિર્ણાયક છે કે અમે તેને બદલવાની સાચી ધાર પસંદ કરીએ છીએ. તેથી, અમારી પાસે લક્ષ્ય છે અને અમે તમને v ને રજૂ કરવા માંગીએ છીએ. તેથી, તમારે u થી v માં T માંના પાથને

અનુસરવું આવશ્યક છે અને તે પાથ અંદરથી પ્રારંભ થવા અને બહાર જવું આવશ્યક છે. તેથી, તેને સીમાને ક્યાંક ખર્ચ કરવો જ પડશે, અને આ બદલવા માટેની ધાર છે. તેથી, આપણે કોઈ મનસ્વી ધાર બદલવાની ભૂલ ન કરવી જોઈએ. આપણે તે કિનારીને બદલવી જોઈએ જે તમને યુ ઝેડ થી લઈને હાયપોથેટિકલ વૃક્ષમાં જવા દે છે, નવો વૃક્ષ બનાવવા માટે નહીં.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:50)

તેથી, એક વાર આપણને આ પ્રમેય
થઈ ગયા પછી, પ્રીમ્સ(Prim's)ની એલ્ગોરિધમનો ચોકસાઈ ખૂબ જ સ્પષ્ટ છે. તેથી, દરેક તબક્કે પ્રીમ્સ(Prim's)ના અલ્ગોરિધમમાં યાદ રાખીએ, આપણે આ વૃક્ષ ટીવી બનાવી છે જે થોડા કિનારીઓ ધરાવે છે, અને પછી આપણી પાસે બધું જ બહાર પડેલું છે અને હવે તેમાંના એકને આપણે કનેક્ટ કરવા માંગીએ છીએ. તેથી, જો તમે આનો વિચાર કરો છો કે અંદરનો સેટ મારો છે અને બહારનો સેટ મારું ડબ્લ્યુ છે, અને અમે પ્રીમ્સ(Prim's)ના એલ્ગોરિધમમાં ધારણા દ્વારા ચૂંટતા હોઈએ છીએ, તો તમને સૌથી ઓછો વજન ધાર જે તમને જોડે છે. આ ન્યૂનતમ વિભાજક પ્રમેય દ્વારા, આ ધાર દરેક ફેલાવતા વૃક્ષને રેખામાં લેવી આવશ્યક છે. તેથી, એલ્ગોરિધમ, પ્રીમ્સ(Prim's)નું એલ્ગોરિધમ, એ પ્રીમ્સ(Prim's)ની એલ્ગોરિધમ ચૂંટણીઓ જે ધાર છે તે વાસ્તવમાં ધાર છે કે પ્રમેયમા અમને પસંદ કરે છે. તેથી, તેથી, પ્રાથમિકનું એલ્ગોરિધમ ચોક્કસપણે સાચું છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 08:35)

તેથી, વાસ્તવમાં આપણે પ્રમેયનો ઉપયોગ પ્રીમ્સ(Prim's)ના એલ્ગોરિધમનો થોડો વધુ હળવા બનાવવા માટે કરી શકીએ છીએ. યાદ રાખો કે સાંકડી પ્રાદેશિક ફોર્મ્યુલેશન અમે સૌથી નાનો ધાર સાથે શરૂ કર્યો હતો, પરંતુ હવે જોવું છે કે જો હું કોઈ શિરોબિંદુ લઈશ, તો હું જમણી બાજુ અને તેમાંથી બહાર નીકળતી બધી કિનારીઓને જોઉં છું, તો હું તમને વર્ટીક્સની વિરુદ્ધ લઈ જઈશ અને હું બીજું બધું બનવા માટે લઈ શકું છું. આપણે આ વાક્યરચનાને ઓછા કરીએ છીએ. પછી, હું જાણું છું કે વી માંથી આ ધાર તરફ જાય છે તે સૌથી નાની ધાર દરેક ફેલાયેલી વૃક્ષમાં હોવી આવશ્યક છે. બીજા શબ્દોમાં કહીએ તો, જો હું કોઈપણ શૂન્યાવકાશ પર શરૂ કરું છું અને તેનાથી જોડાયેલા નાના ધાર તરફ જોઉં છું, તો હું તેનાથી પ્રારંભ કરી શકું છું કારણ કે તે લઘુતમ વિભાજક પ્રમેય દ્વારા દરેક ફેલાતા વૃક્ષમાં હોવું આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:18)

તેથી, આની પ્રાથમિક વ્યૂહરચના માટે આ અમને નીચે આપેલી અલ્ગોરિધમનો આપે છે. તેથી, અમે કોઈપણ શૂન્ય સાથે શરૂ કરો. હવે, દરેક શિરોબિંદુ માટે જે આપણા વૃક્ષના શિરોબિંદુના વર્તમાન સમૂહમાં નથી, તે શિરોલંબમાંથી આપણે નાના કદના વજનને જાળવી રાખીએ છીએ અને કેટલાક વૃક્ષની શિખરોને વી ની અંતર કહેવામાં આવે છે અને કારણ કે આપણે વૃક્ષને કિનારીઓનો સમૂહ બનાવવો છે, આપણે યાદ રાખીએ છીએ કે તે ધાર ક્યાં જાય છે. તેથી, આપણે યાદ રાખીએ છીએ કે તે પાડોશી છે. તેથી, જો આપેલ બિંદુ પર મારું ઝાડ હોય અને મને ખબર છે કે આ વી માટે, આ પાડોશી તમે વીમાં જોડતા સૌથી નાનો ધાર છે, તો અહીં હું ધારના વજન જેટલું અંતર રાખું છું, હું તેને રાખીશ તમારા જેવા પાડોશી. આ મને કઈ કિનારીઓ ઉમેરવામાં આવે છે તેનો ટ્રેક રાખવા દેશે. તેથી, હવે, હું દરેક તબક્કે અંતરની દ્રષ્ટિએ બહારના નાના ભાગની શોધ કરું છું. પછી, હું તેને સેટ પર ઉમેરીશ અને હું તેને પાડોશી અંતર અને મૂલ્યો તરીકે અપડેટ કરું છું. તેથી, આ

ડીજેક્સ્ટ્રા(Dijkstra's)ના અલ્ગોરિધમનો સમાન છે. એક માત્ર વસ્તુ એ અંતરનું અપડેટ છે જે મારા અંતર ઉપરાંત વજનને ઉમેરે છે. તે માત્ર વજન ધ્યાનમાં સમાવેશ થાય છે. તેથી, જ્યારે એલ્ગોરિધમ પોતે દેખાય છે, ત્યારે આપણે સમાંતર ડિજેસ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમને પણ વધુ સ્પષ્ટ જોઈશું.

(સ્લીનો સંદર્ભ લો ટાઈમ: 10:36)

તેથી, અહીં પ્રીમ્સ(Prim's)ના લઘુતમ અથવા ન્યૂનતમ ખર્ચાના વૃક્ષ માટે જ અંતિમ અલ્ગોરિધમ છે. તેથી, તમે બધા શિરોબિંદુઓને અવલોકન કરવા માટે પ્રારંભ કરો. આ બોન્ડ વસ્તુ છે. આપણે બધાને કોઈ પડોશીઓ ન હોવાનું યાદ કરીએ છીએ, કારણ કે વૃક્ષમાં કશું જ નથી. તેથી, તેઓ વૃક્ષમાં કોઈ પડોશીઓ નથી અને તેઓ બધા અંતર અનંત છે. આ વૃક્ષને પ્રારંભ કરે છે. હવે, હું 1 ને શૂન્યથી શરૂ કરીને કેટલાક પ્રારંભિક પસંદ કરું છું અને તેને મુલાકાત લેવા માટે ચિલ્નિત કરું છું, પરંતુ મારી પાસે કોઈપણ ધાર નથી. તેથી, હવે દરેક ધાર 1 થી બહાર જવા માટે, હું તેની સ્થિતિ અપડેટ કરું છું. તેથી, હું ફોર્મ 1 ની દરેક ધાર માટે કહું છું, જે, મેં કહ્યું હતું કે જે વૃક્ષમાં જેનો પાડોશી છે, તેથી વૃક્ષમાં હવે આ એક શિરોબિંદુ અને નાનું વૃક્ષ છે, જેનું એક ભાગ છે અને તેથી 1 -1 0 8, બરાબર. તેથી, j નું પાડોશી 1 છે કારણ કે તે તેનું જોડાણ છે અને અંતર એ આ ધારનો વજન છે. તો, આ મારો પ્રથમ પગલું છે. હવે, મારે મારા વૃક્ષ પર બાકીના n ઓછા 1 કિનારીઓ ઉમેરવાની રહેશે. તેથી, અલગ અલગ ફોર્મેટમાં ડીજેક્સ્ટ્રાના એલ્ગોરિધમમાં તમે જે કરો છો તે હું 1 વખત કરું છું. તમે તે યુ પસંદ કરો છો જે મુલાકાત લીધી નથી અને જેની અંતર ન્યૂનતમ છે. મુલાકાત તરીકે ચિલ્નિત કરો. હવે, તમે જાણો છો કે તે વૃક્ષ સાથે કેવી રીતે જોડાયેલ છે. તેથી, તમે ધાર ઉમેરો છો જે તમને જણાવે છે કે તે તમારા અને તમારા પાડોશીને કેવી રીતે જોડે છે. આ ધાર આપણે વૃક્ષની ધારના સમૂહમાં ઉમેરાય છે. હવે, તમારા પડોશીની મુલાકાત લેવામાં આવતી પ્રત્યેક ધાર માટે, જો વૃક્ષની વર્તમાન અંતર આ ધારના વજન કરતાં વધારે છે, તો મૂળભૂત રીતે મેં હવે તમને આ ઉમેર્યું છે અને ત્યાં એક બીજું વર્ટેક્સ વી છે અને તે કનેક્ટ થવાનો દાવો કરે છે. તેથી, કદાચ આ અંતર ડી અને આ અંતર ડી પ્રીમ્સ(Prim's). ધારો કે ડી v કરતાં મોટો છે, તો હવે તે વૃક્ષમાં છે, હવે તે વૃક્ષમાં ઉમેરાઈ ગયું છે, હવે વી નાના કિનારે વૃક્ષ સાથે જોડાયેલ છે, જમણી બાજુ. તેથી, હાલમાં જે અંતર હું b માટે છે તે યુવી ધારના વજન કરતા વધારે છે. પછી, હું તે વજનને યુવ્હના વજનથી બદલી દઈશ અને હું કહું છું કે વીનો પાડોશી હવે તમે છો, જેથી જ્યારે મારી પાસે વૃક્ષ પર વી હોય, હું તમારી ધાર ઉમેરીશ. તેથી, આ સુધારા સિવાય ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ બરાબર છે જે આ અપડેટની જેમ અમે u વત્તા u વજન બરાબર કર્યું હતું. તેથી, આપણે ડિજેસ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમમાં, આપણે એકત્રિત અંતર જોઈએ છે. અહીં આપણે ઝાડના નજીકના નોડમાંથી એક પગથિયું દૂર કરવા માંગીએ છીએ, પરંતુ અન્યથા પ્રીમ્સ(Prim's)નું એલ્ગોરિધમ મૂળભૂત રીતે ફરીથી અપડેટ અથવા ડીજેક્સ્ટ્રા(Dijkstra's)ના અલ્ગોરિધમનો ભિન્ન અપડેટ ફંકશન છે અને વધુમાં અમારી પાસે આ વસ્તુ છે કે આપણે તેને ડીજેક્સ્ટ્રા(Dijkstra's)માં પણ કરી શકીએ. આપણે ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમનો માર્ગ, સાચો રાખ્યો હતો. તેથી, આપણે પાથ જાળવી રાખ્યો હતો, તે આ પાડોશી સંબંધની જેમ અહીં હશે. આપણે જાણવું છે કે આ કિનારીઓ મારા ટૂંકા પાથ સેટમાં શામેલ છે, બર્ન સેટ, કેમ તે ઉમેરવામાં આવ્યું હતું. તેથી, અહીં આપણે કરી રહ્યા છીએ કે આપણે તેને ઉમેરી રહ્યા છીએ અને ધાર ધારને પણ યાદ રાખીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:27)

તેથી, ચાલો આ વસ્તુઓની જટીલતા વિશ્લેષણ દરમિયાન પહેલાં પ્રયાસ કરીએ અને અમલમાં મુકીએ. તેથી, યાદ રાખો કે આપણે ગમે ત્યાંથી શરૂ કરી શકીએ છીએ. તેથી, ચાલો, 1 થી શરૂ કરીએ. અમે 1 થી પ્રારંભ કરીએ છીએ અને અમે અમારા વૃક્ષને સ્વરૂપ સાથે જોડીએ છીએ. હવે, કારણ કે આ 1 થી શરૂ થાય છે, આપણે 1 ની પડોશીઓમાં મૂલ્યોને અપડેટ કરવું પડશે, એટલે કે 2. એટલે, આપણે 3 માટે ચિલ્નિત કરીએ છીએ. આપણે કહીએ છીએ કે તે અંતર છે જે આપણે લીલા રંગમાં ચિલ્નિત કરીએ છીએ, તેથી વૃક્ષ 18 છે કારણ કે વૃક્ષમાં 1 લીટી હોય છે અને તે વૃક્ષમાં પાડોશી છે જે અંતર પર છે. આ જ રીતે, 1 નું બીજું પાડોશી 2 છે. તેથી, આપણે કહીશું કે તેની અંતર 10 છે અને તેના પાડોશી 1 છે. તેથી, બીજાં દરેક જગ્યાએ મેં સ્પષ્ટ રીતે તેનો ઉલ્લેખ કર્યો નથી, પરંતુ બાકીના દરેક જગ્યાએ વેલ્યુ 1 છે અને માફ કરશો, અનંત અને બાદબાકી 1. તેથી, આ અનંત છે અને પાડોશી 1 ઓછા છે. તેથી, આ ડિફૉલ્ટ મૂલ્ય છે. તેથી, જ્યાં પણ ડિફૉલ્ટ મૂલ્ય હાજર હોય, ત્યારે અમે તે સૂચવીશું કે મૂલ્ય અસરકારક રીતે ચક્રીય નથી. આપણે જાણીએ છીએ કે તે એક જોડાયેલ ગ્રાફ છે. તેથી, આપણે આખરે તેને સેટ કરીશું. તેથી, તમે તેના વિશે ચિંતા કરશો નહીં, પરંતુ આ ચર્ચામાં અમે તેને છોડી દઈશું. તેથી, અમારી પાસે હવે આ બે ઉમેદવારો છે જે મુલાકાત લેતા નથી અને જેની પાસે કેટલાક વાજબી અંતર છે. તેથી, આપણે 2 કરતા નાના પસંદ કરીશું. તેથી, આપણે આ એક પસંદ કરીએ જે 10 છે, અને તેથી આગળના પગલા પર આપણે 2 શ્વેત ભાગની મુલાકાત લઈશું અને આ ધાર 1 થી 12 ઉમેરીશું.

(સ્લાઈડસમયનો સંદર્ભ લો: 14:46)

હવે, 2 ઉમેરીને, અમારી પાસે આ અપડેટ છે. તેથી, આપણે પડોશીઓ તરફ નજર કરીએ છીએ. તેથી, 2 ના પડોશીઓ કર્ણ 3 અને કર્ણ 5 છે. તેથી, કર્ણ 2 માટે, આપણી પાસે એક નવી અંતર 6 છે. તેથી, જો તમે 3 ના વૃક્ષની 3 થી અંતર સુધી જાઓ અને ત્યાંથી કનેક્ટ થઈ શકે 2 જે 6 છે તે 18 કરતા નાની છે, બરાબર. તેથી, 18 અગાઉ વૃક્ષના ત્રણ જેટલા હતા તેવો શ્રેષ્ઠ અંદાજ હતો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:16)

તેથી, તમે 18, 1 થી 6, 2 ને બદલી શકો છો જે સૂચવે છે કે હવે કર્ણ 3 વૃક્ષથી 6 અંતર દૂર છે, અને જો તે દૂરથી જોડાયેલું હોત, તો તે 2 સાથે જોડાયેલું હોઈ શકે છે. એ જ રીતે, 5 જે અગાઉ અનલેબલ થયું હતું, હવે 22 તરીકે લેબલ થાય છે જે સૂચવે છે કે તેની અંતર વૃક્ષમાંથી 20 છે અને તેના પડોશી વૃક્ષમાં લેબલ વર્ટેક્સ વી છે. હવે, આપણે ફરીથી બે નાના પસંદ કરીએ છીએ. તો, આપણે વૃક્ષમાં ઉમેરવા માટે આ 3 લીટી 3 પસંદ કરીશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:46)

એકવાર આપણે તેને ઉમેરતા, તમને 4 ની સ્થિતિ મળશે કારણ કે તે એકમાત્ર નવું લેબલ છે જે આપણે 1 ની સ્થિતિને અપડેટ કરતા નથી કારણ કે 1 એ વૃક્ષમાં પહેલેથી ઉમેરાઈ ગયું છે. અમે માત્ર તે વૃક્ષોના પાડોશીઓને જ જોઈએ છીએ જે મુલાકાત લેતા નથી. તેથી, હવે 4 પાડોશી 3 સાથે અંતર 70 મેળવે છે, અને પછી આ બંને વચ્ચે, હવે 20 70 કરતા નાની છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:55)

તેથી, આપણે આપણા વૃક્ષમાં 5 ઉમેરીશું, અને પછી આપણે અપડેટ કરીશું 6 અને 7 ની સ્થિતિ.

(સ્લાઈડસમયનો સંદર્ભ લો: 16:05)

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:10)

તેથી, 6 હવે પાડોશી 5 સાથે અંતર 10 છે, 7 પાડોશી સાથે પણ અંતર 10 છે. હવે, અમારી પાસે અંતર 10 સાથે બે શિરોબિંદુઓ. અમે ક્યાં તો એક પસંદ કરી શકે છે. તેથી, ચાલો આપણે 7 પસંદ કરીએ. જો આપણે 7 પસંદ કરીએ, તો આપણે તેને વૃક્ષમાં ઉમેરીશું અને હવે, આપણે 6 ની સ્થિતિને અપડેટ કરીશું. પહેલા તે પાડોશી 5 સાથે અંતર 10 હતું, પરંતુ હવે તે અંતર 5 છે પાડોશી 6. તેથી, અમે પાડોશી સાથે તેને ઘટાડીએ છીએ 7. અમે તેની અંતર ઘટાડીએ છીએ અને અમે તેના પાડોશીને બદલીએ છીએ. હવે, 5 અને 17 ની વચ્ચે, આપણી પાસે 6 શ્વેત છે, 6 એક નવી છે, અને પછી આપણી પાસે 4 છે અને આ તે વૃક્ષ છે જે આપણને મળે છે. આ રીતે પ્રીમ્સ(Prim's)ન્નું એલ્ગોરિધમ કાર્ય કરે છે. તે ડીજેક્સ્ટ્રા(Dijkstra's)ના સિદ્ધાંતથી ખૂબ જ સમાન છે પરંતુ તે એકદમ અલગ અપડેટનો ઉપયોગ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:57)

તેથી, જટિલતા પણ ડીજેક્સ્ટ્રા(Dijkstra's)ના અલ્ગોરિધમનો સમાન છે. અમારી પાસે એક બાહ્ય લૂપ છે જે n વખત ઓર્ડર n વખત ચાલે છે કારણ કે આપણને તે વૃક્ષ બનાવવા માટે n ઓછા 1 ધાર ઉમેરવું પડશે અને દર વખતે જ્યારે આપણે વૃક્ષ સાથે વર્ટેક્સ ઉમેરીશું. હવે, ન્યૂનતમ કિંમત વર્ટેક્સ ઉમેરવા માટે આ ઑર્ડર n સ્કેન છે. તેથી, આપણે પહેલા જ જોયું છે કે ડિજક્સ્ટ્રાના અલ્ગોરિધમનો સમાવેશ કરવા માટે ન્યૂનતમ અંતરનું વર્ટેક્સ શોધવાનું છે, અને પછી જ્યારે આપણે કર્ણ ઉમેરતા હોઈએ, ત્યારે તમારે બધી એન્ટ્રીઓને અપડેટ કરવા માટે ફરીથી અને ફરીથી સ્કેન કરવું પડશે. તેથી, અમારી પાસે એડજેસંસી મેટ્રિક્સ(adjacency matrix) છે. આ ફરીથી ઓર્ડર એન ટાઈમ લેશે અને તેથી, આખરે તે ઓર્ડર n ચોરસ લેશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:36)

તેથી, જેમ ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ એડજેસંસી મેટ્રિક્સ(adjacency matrix)ને એડજેસંસી (adjacency) લિસ્ટમાં ખસેડવા બરાબર છે, તેથી ધાર્સની રજૂઆત અમને અપડેટ્સની જટિલતાને ઘટાડે છે. તેથી, એન પુનરાવર્તનોમાં, અમે કુલ ઓર્ડર એમ અપડેટ્સ કરીએ છીએ કારણ કે ફક્ત પડોશીઓ, ડિગ્રી, બધા શિરોબિંદુઓની ડિગ્રીની સંખ્યાને આધારે અપડેટ કરો. જો કે, તે ક્રમ n ચોરસ લાવવા માટે, અમને ઓછામાં ઓછા અંતરની કુશળતાપૂર્વક ગણતરી કરવાની પણ જરૂર છે જેના માટે અમને એક ઢગલાની જરૂર છે. તેથી, એકવાર અમારી પાસે એક ઢગલો છે જેને આપણે પછીના ભાષણમાં તપાસ કરીશું, દાવા એ છે કે અમે ન્યુનત્તમ શોધી શકીએ છીએ અને n લૉગ સમયમાં અંતરને અપડેટ કરી શકીએ છીએ. તેથી, આ આપણને એકંદર જટિલતા આપે છે, જે બરાબર x ના m લોગ n વત્તા એમ લોગ n ને દો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:19)

તેથી, આ ન્યૂનતમ શોધવામાં આવે છે કારણ કે n સમય આપણને ન્યૂનતમ શોધવો પડે છે અને તે અપડેટ્સમાંથી આવે છે, કારણ કે અમને એકંદરે એમ અપડેટ્સ કરવું પડશે. દરેક અપડેટ લોગ એન ટાઈમ લે છે, તેથી આપણને n પ્લસ n લોગ n બરાબર મળે છે જેમ આપણે ડિજેસ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ માટે કર્યું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:36)

તો, આપણે પ્રીમ્સ(Prim's)ના અલ્ગોરિધમનો છોડતા પહેલા એક છેલ્લો મુદ્દો. યાદ રાખો કે ચોકસાઈમાં આપણે તે ન્યૂનતમ વિભાજક પ્રમેયનો ઉપયોગ કરવો પડશે જેમાં આપણે ધાર્યું હતું કે ધાર વજન અલગ છે. તેથી, અલબત્ત આપણે ઉદાહરણમાં જોયું છે કે આપણે અમલમાં મુક્યા છે, આપણે ધાર વજન, એક જ વજન સાથે બહુવિધ કિનારીઓ હોઈ શકે છે. તો, આપણે આ પ્રમેય
માં કેવી રીતે કામ કરીશું? ઠીક છે, અમે દલીલ કરી શકીએ કે તમે તે માત્ર એટલું જ વજન નહી, પરંતુ વજન વત્તા બીજા કોઈ પણ શબ્દને પણ બનાવી શકો છો. તેથી, સામાન્ય રીતે આપણે કહી શકીએ છીએ કે આપણે ધારની એકંદરે ઓર્ડરિંગને ઠીક કરીએ છીએ. ત્યાં એમ ધાર છે. તો, આપણે ફક્ત 1 થી n ની ધારણા કરીએ છીએ અને આપણે કહીએ છીએ કે વજન એકદમ નાનો હોય અથવા વજન બરાબર હોય તો એક કિનારી નાની કિનારી કરતા નાની હોય છે, પરંતુ ક્રમમાં અનુક્રમણિકા નાના છે. તેથી, ઈ અને એફ પાસે યુવીનું વજન અને તમારા પ્રીમ્સ(Prim's) અને વી પ્રીમ્સ(Prim's)નું વજન છે, પરંતુ અમારી પાસે ઈન્ડેક્સ i અને j પણ છે. તેથી, આ એક છે હું 1 થી મીટર વચ્ચેનો છું અને આ કેટલાક j વચ્ચે 1 થી મીટર છે. તેથી, ક્યાં તો વજનનું વજન f ના વજન કરતાં ઓછું હોવું જોઈએ અથવા વજન સરખા હોવું જોઈએ, પછી હું જ કરતાં નાના હોવું જોઈએ, બરાબર. તેથી, આ આપણને સમય ભંગ કરવાનો નિયમ આપે છે. તેથી, હવે આ મૂળભૂત રીતે અમને કહેશે કે આપણે હંમેશાં બે ધારની તુલના કરી શકીએ છીએ અને ઘોષિત કરી શકીએ કે 1 બીજા કરતા નાના છે અને પ્રીમ્સ(Prim's) એલ્ગોરિધમ શું કરશે, તે નાના ને પસંદ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:49)

તેથી, અમે ઈચ્છીએ છીએ કે આ એવું કહેવાનું છે કે આપણે વાસ્તવમાં જ્યારે બે સમાન વસ્તુઓ ધરાવીએ છીએ ત્યારે પસંદ કરવાની વ્યૂહરચના આપીએ છીએ. તેથી, એલ્ગોરિધમ કહે છે કે તે અંતર જે લઘુતમ છે અને જો બહુ ઓછા અંતર સાથે બહુવિધ ઉપયોગનો ઉપયોગ કરે છે, તો અમે મનસ્વી બિંદુ પસંદ કરીએ છીએ. તેથી, તેનો અર્થ એ છે કે મનસ્વી બિંદુ પસંદ કરવો કે કેમ તેનો અર્થ એ છે કે તેમાં કોઈ ઓર્ડર પસંદ કરવો અને તે ક્રમમાં જવું. તેથી, જો તમે વિવિધ ઓર્ડર પસંદ કરો છો, તો પછી અમને વિવિધ વૃક્ષો મળે છે. તેથી, તેથી અમારી પાસે એક વૃક્ષમાં બહુવિધ ધાર છે જે સમાન વજન ધરાવે છે. સામાન્ય રીતે, અમે એક અનન્ય સ્પૅનિંગ વૃક્ષ મેળવી શકતા નથી. હકીકતમાં, તમે ચકાસી શકો છો કે તમારી પાસે બધા વજન સમાન છે કે કેમ. મૂળભૂત રીતે તમારે જુદા જુદા કિનારો ઉમેરવા અથવા છોડવાનું ચાલુ રાખવું પડશે અને તમારી પાસે ઝાડની ઘાતાંકીય સંખ્યા હશે કારણ કે એક વૃક્ષમાં કિનારી હોઈ શકે છે અને તે કદાચ બીજા વૃક્ષમાં હોઈ શકે નહીં અને તેથી જ. તેથી, એકંદરે શક્યતમ લઘુતમ ખર્ચે વૃક્ષોની સંખ્યા ખૂબ મોટી હોઈ શકે છે. જ્યારે આપણે આગલા ભાષણમાં જોશું ત્યારે આપણે શું કરી શકીએ અને શું ક્રુસ્કલ(Kruskal) કરશે તે પણ આ શક્ય વસ્તુઓમાંથી એકને

અસરકારક રીતે પસંદ કરવા માટે લોભી વ્યૂહરચનાનો ઉપયોગ કરવો છે. હવે, જો ધાર વજન અનન્ય હોય, તો ત્યાં વધુ પસંદગી નથી. તમારે એક જ વૃક્ષ પસંદ કરવું પડશે. જો ધાર વજનને ડુપ્લિકેટ કરવામાં આવે છે, તો તમે ચોક્કસપણે બહુવિધ વૃક્ષો ધરાવી શકો છો અને દરેક તબક્કે સૌથી નાનો વિકલ્પ પસંદ કરવાની આ વ્યૂહરચના અમને એક નાનો સૌથી નાનો ઓળખવા માટેનો એક ઝડપી રસ્તો આપશે, પરંતુ અનન્ય નહીં.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફ્ેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 07
લેક્ચર - 31
સ્પૅનિંગ ટ્રીઝ: ક્રુસ્કલનું એલ્ગોરિધમ (Spanning trees: Kruskal’s algorithm)

અમે પ્રીમ્સ એલ્ગોરિધમ(Prim’s algorithm)નો લઘુત્તમ ખર્ચ વિસ્તાર વૃક્ષ માટે એક અલ્ગોરિધમનો જોયો છે. હવે, ચાલો આપણે ક્રુસ્કલના અલ્ગોરિધમ (Kruskal’s algorithm) વિશે જે અન્ય વ્યૂહરચના વિશે વાત કરીએ છીએ તેને જોઈએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:10)

તેથી, અમે વેઈટ્ડ અંડિરેક્ટએડ ગ્રાફ્(weighted undirected graph)માં, ન્યૂનતમ કિંમતના વૃક્ષને શોધી રહ્યા છીએ. તેથી, પ્રીમ્સ(Prim’s)નો અલ્ગોરિધમ અમુક ધાર સાથે પ્રારંભ થાય છે અને ધીરે ધીરે એક વૃક્ષને વિસ્તૃત કરે છે, જ્યારે ક્રુસ્કલ(Kruskal)નું એલ્ગોરિધમ બીજી વ્યૂહરચનાને અનુસરે છે, જે વજનના ચડતા ક્રમમાં તમામ કિનારીઓને ઓર્ડર આપે છે. તેથી, તે કાંકરાને નાનાથી સૌથી મોટા અને દરેક ધાર માટે પ્રયાસ કરી રાખે છે જો તે વૃક્ષની મિલકતનું ઉલ્લંઘન કર્યા વિના ધાર ઉમેરી શકે છે, તો તે ઉમેરે છે. હવે, ધાર ઉમેરવાની પ્રક્રિયામાં, તે વાસ્તવમાં એક વૃક્ષનું નિર્માણ કરી શકતું નથી, તે બધું ખાતરી કરે છે, તે વૃક્ષની મિલકતનું ઉલ્લંઘન કરતું નથી, તે પછી તે ચક્ર પેદા કરતું નથી. તેથી, જો આપણે ધાર ઉમેરવાનું ચાલુ રાખીએ, ત્યાં સુધી આપણે ચક્ર બનાવતા નથી અને અંતે દાવા એ છે કે આપણે ઓછામાં ઓછું ખર્ચ કરી વૃક્ષો ફેલાવીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:52)

તેથી, અહીં એલ્ગોરિધમનો એક પ્રકારનો ઉચ્ચ સ્તર દૃશ્ય છે, તેથી કિનારીઓને ક્રમમાં ગોઠવવામાં આવે છે જેથી e 1 થી e. તેથી, આપણે ખાલી વૃક્ષથી પ્રારંભ કરીએ છીએ, તેથી ફરીથી આપણે વૃક્ષોને ધારની સૂચિ તરીકે રાખીશું અને હવે આપણે આને સ્કેન કરવા જઈશું, 1 થી મીટર. તેથી, હું આગલા પ્રયાસ કરવા માટે ધારની સૂચિ બની શકું. તેથી, જ્યાં સુધી આપણે હજુ સુધી n ઓછા 1 ધાર ઉમેર્યા નથી. યાદ રાખો કે જો તમે n ઓછા 1 કિનારીઓ ઉમેરો છો અને અમારી પાસે જોડાયેલ ગ્રાફ્ છે, તો તે એક વૃક્ષ હોવું આવશ્યક છે. વૃક્ષો વિશે આપણે જે કહ્યું તે આ એક જુદી જુદી લાક્ષણિકતા છે. વૃક્ષો n માઈનસ 1 કિનારીઓ છે, પરંતુ n માઈનસ 1 સાથેના કોઈપણ કનેક્ટ કરેલ ગ્રાફ્ તે હોવું આવશ્યક છે. તેથી, એકવાર આપણી પાસે 1 ની બાદબાકી 1 કિનારીઓ બંધ થઈ જાય છે. તેથી, જ્યાં સુધી કિનારીઓની સંખ્યાના સંદર્ભમાં વૃક્ષની લંબાઈ 1 ના નહી હોય, આપણે વધુ ધાર ઉમેરવા પડશે. તેથી, આપણે આગળના ઈ ઈ તરફ્ ધ્યાન આપીએ છીએ, જો આપણે TE માં ઉમેરાય ત્યારે ચક્ર બનાવતા નથી, તો આપણે આગળ વધીએ છીએ જો તે એક ચક્ર બનાવે છે, તો આપણે ફ્કત છોડીને આગળ વધીએ, આ બધા ક્રુસ્કલના અલ્ગોરિધમ (Kruskal’s algorithm) છે

((સમયનોસંદર્ભ લો: 01:49)).

બધા ધાર દ્વારા સ્કેનિંગ અને જ્યાં સુધી આપણે કોઈ વૃક્ષ શોધી ન શકીએ ત્યાં સુધી તે દરેક ધારને ચડતા ક્રમમાં ગોઠવે છે અને ચક્ર બનાવ્યાં વગર ઉમેરવામાં આવે તે દરેક વખતે તે ઉમેરાશે, તે ઉમેરવામાં આવશે નહીં ચક્ર તે ઘટી જશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:03)

તેથી, ચાલો આપણે એ જ ઉદાહરણ જોઈએ જે આપણે પ્રીમ્સ(Prim's)ની એલ્ગોરિધમ્સ વિશે જણાવીએ છીએ. તેથી, અહીં આ સૌથી નાનો ધાર વજન યાદ રાખો કે આપણે તેને સૉર્ટ કરી નથી. તેથી, 5 એ નાનું સૌથી નાનું છે, પછી આપણી પાસે વજન 10 છે, તો આપણી પાસે 18 છે, તો આપણી પાસે 20 છે અને પછી આપણી પાસે છે, હવે જો આપણે 3 નાનામાં એક પસંદ કરીશું, તો આપણે ધાર 5 થી શરૂઆત કરીશું આપણા વૃક્ષ માટે.

(સ્લાઈડસમયનો સંદર્ભ લો: 02:28)

હવે, આગલું એક 6 છે, તેથી આપણે આપણા વૃક્ષમાં 6 ઉમેરીએ છીએ. હવે, આપણે 10 લેબલવાળી ધારમાંથી એક પસંદ કરવું પડશે, આપણે કોઈ પણ એક પસંદ કરી શકીએ, તો ચાલો ધારીએ કે આપણે આને પસંદ કરીએ છીએ. હવે, આપણે બીજું ધાર લેબલ 10 પસંદ કરવા માંગીએ છીએ, કારણ કે આગામી બે ધાર ફરીથી લેબલ 10 છે. તેથી, ધારો કે આપણે આ ધાર લેબલ 10 પસંદ કરીએ, પછી આ ધાર લેબલ 10, આ ચક્ર બનાવશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:47)

તેથી, આપણે તે કરી શકતા નથી, તેથી અમે તેને છોડ્યું, અમે તેને તોડીને આગળનાં પર જઈએ. તેથી, હવે આપણે આ એક પસંદ કરીએ, તેથી આપણે આ દ્વારા દૂર જઈશું અને હવે આપણે આમાં જઈશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:00)

અને હવે આ એક ઓર્ડર બનાવે છે, તેથી હવે આપણે દસ પૂર્ણ કરી દીધા છે, તેથી પછીનું એક 18 છે, પણ ફરીથી 18 ઉમેરી, આપણે એક ચક્ર બનાવવાની તૈયારી કરીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:08)

તો, આપણે તેને રદ્ કરીએ છીએ અને હવે આપણે તે પછી આગળ જઈશું જે આ રીતે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:16)

હવે, 20 ઉમેર્યા પછી, આપણે હજી પણ કર્યું નથી, કારણ કે આપણે ઉમેર્યું છે કે અમારી પાસે 1, 2, 3, 4, 5 ધારની પૂરતી કિનારી ઉમેરવા માટે કૌશલ્ય છે અને આપણને 6 ની જરૂર છે , કારણ કે આપણી પાસે ઝડપથી 7 શિરોબિંદુઓ છે, આપણને n ઓછા 1 ધાર મળે છે. તેથી, એકમાત્ર ધાર બાકી છે આ 70 છે, તેથી તે એક રીતે છે. તેથી, આ

ક્રુસ્કલ(Kruskal)ના એલ્ગોરિધમનું વિસ્તરણ વૃક્ષ સ્વરૂપ છે, અમે વાસ્તવમાં આ વિશિષ્ટ કેસમાંના તમામ ધારમાંથી સ્કેન કરીએ છીએ અને જે રીતે અમે કપ્લરને છોડી દીધું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:43)

તેથી, ક્રુસ્કલ(Kruskal)નું એલ્ગોરિધમ એ પણ લોભી અલ્ગોરિધમ છે, આ કિસ્સામાં આપણે કોઈ નથી બનાવતા . તેથી, પ્રાથમિક એલ્ગોરિધમમાં ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ્સ તરીકે વધુ, અમે વધતી પસંદગી, દરેક જગ્યાએ સ્થાનિક પસંદગીઓ કરીએ છીએ. મુદ્દો, આપણે હવે જે જાણીએ છીએ તેના આધારે, આપણે આગળ શું કરીશું તે જોઈશું. અહીં, અમે અગાઉથી પસંદગી કરીએ છીએ, આપણે શરૂઆતમાં જ કહીએ છીએ કે આપણે બધા ધારને સૉર્ટ કરીએ અને તે ક્રમમાં કરીએ. અને ફરીથી તે આ ચોક્કસ ક્રમમાં કરવાથી સ્પષ્ટ નથી થયું જેણે શરૂઆતમાં જ નક્કી કર્યું છે કે અમને એકંદરે મહત્તમ ઉકેલ મળે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:13)

તો, ફરી એકવાર આપણે તે જ પરિણામનો ઉપયોગ કરીશું જે આપણે પ્રીમ્સ(Prim’s)ની એલ્ગોરિધમનો ઉપયોગ કર્યો હતો, આ ન્યૂનતમ વિભાજક પ્રમેય. તેથી,પ્રમેય એ શું કહ્યું તે યાદ કરો, તે કહે છે કે જો તમે શિરોબિંદુનો સમૂહ લો અને બે બિન-ખાલી જૂથો યુ અને ડબલ્યુમાં વિભાજિત કરો અને જો આપણે સૌથી નાનું ધાર લઈએ, તો હવે ગ્રાફ જોડાયેલ છે. તેથી, આ ટૂલને જોડતી ધાર હોવી આવશ્યક છે, અમે નાના ભાગોને એક સાથે જોડીએ છીએ જે દરેક ભાગને એકસાથે જોડે છે, પછી આ ધાર દરેક ફેલાયેલી વૃક્ષ પર જ હોવો જોઈએ, દરેક લઘુત્તમ ખર્ચે ઝાડવું જોઈએ, આ પ્રમેયએ કહ્યું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:44)

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:29)

હવે, તેથી, જ્યારે પણ આપણે ધાર ઉમેરીએ ત્યારે કોઈ ધાર નથી, તે 2 એન પોઈન્ટ અલગ ઘટકોમાં છે. તેથી, આપણે પ્રમેયનો ઉપયોગ કરી રહ્યા છીએ, ચાલો આપણે કેપિટલ યુને ધાર અને કેપિટલ ડબ્લ્યુ ના ઘટક સાથે રહેવા માટે કહીએ, બાકીના જે પણ બહાર છે, તેથી આ બાહ્ય ઘટક છે. તેથી, આપણી પાસે યુ અને જે પણ તે જોડાયેલું છે. તો, આ મારું સેટ યુ છે અને પછી મારી પાસે બાકીના શિરોબિંદુઓ છે, તેથી યુ આ બિંદુએ જુદા જુદા ઘટકમાં કંઈક જોડાયેલું છે, પરંતુ તે અન્ય બધા ઘટકો છે જેને હું ડબલ્યુ કહું છું. તેથી હવે આપણે ખબર છે કે આ પહેલી વાર છે જ્યારે આપણે યુ થી વી કનેક્ટ કરવાનો પ્રયાસ કરી રહ્યા છીએ, યુ થી વી કનેક્ટ થયેલા નથી, અત્યાર સુધી. પરંતુ, અમે વજનના ચડતા ક્રમમાં ધાર તરફ જોઈ રહ્યા છીએ, તેથી જો તમે બધી કિનારીઓ જુઓ. તેથી, આ વર્તમાન ધારને આપણે ej કહેવામાં આવે છે, તેથી જો તમે બધા કિનારીઓ જુઓ જે તમે ej ઓછા નથી 1 તો તેમાંના કોઈપણ યુને વી જોડાયેલું નથી, તે યુને જોડતા ઘટક છે. કારણ કે, તે ઘટકને યુને સમાવતી ઘટક ધરાવતું હતું. જ્યારે આ ધાર ઉમેરી શકે છે, ત્યારે આ ધાર ચક્ર બનાવે છે. તેથી, આ પહેલો તબક્કો છે જે આપણે બે ચાલમાં કનેક્ટનો ઉપયોગ કરી શકીએ છીએ જે આપણે જોઈ નથી, તેથી આને પણ જોડે

છે. તેથી, આને એકસાથે મૂકવા, આ સૌથી નાની ધાર છે જે મૂડી યુને બહારની મૂડી વી સાથે જોડે છે. અને આ નાના તબક્કાઓ હોવાથી, પ્રમેય અમને કહે છે કે આ વૃક્ષમાં હોવી જોઈએ અને તેથી ક્રુસ્કલ(Kruskal)ના આ વૃક્ષમાં સમાવેશ કરવાનો વિચાર સારો છે, આ એક ધાર છે તે ઈનપુટ વૃક્ષ હોવા જ જોઈએ. તેથી, ક્રુસ્કલ(Kruskal)ના એલ્ગોરિધમ ઉમેરેલી પ્રત્યેક ધાર ખરેખર ન્યૂનતમ વિભાજક પર માન્યતા આપે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:11)

તેથી, હવે આપણે ચક્ર બનાવવાની આ મિલકતને કેવી રીતે ટ્રૅક રાખવું તે નક્કી કરવું પડશે. યાદ રાખો કે જ્યારે પણ આપણે ધાર ઉમેરીશું, તમારે પહેલા તેને ફોર્મ ચક્ર તપાસવું જોઈએ અને જો તે કોઈ ચક્ર બનાવતું નથી, તો તમારે શામેલ કરવું આવશ્યક છે. તેથી, તપાસવાનો સૌથી સરળ રસ્તો, પરંતુ ધાર એ ઘટકને ટ્રૅક રાખવા માટે એક ચક્ર બનાવતું નથી, જો બે અલગ ઘટકો પર ધાર શામેલ હોયતે ધાર અને એક ઘટકોનો એક પોઈન્ટ છે અને અન્ય ભાગમાં ધારની એક પોઈન્ટ છે. પછી, પ્રત્યેક ઘટકો એક વૃક્ષ છે અને તેથી, આ બે વૃક્ષો અલગ છે અને તેથી, નવી ધાર એક ચક્ર બનાવે છે. જો બે પોઈન્ટ સમાન ઘટકમાં હોય તો તે એક ચક્ર બનાવશે. તેથી, નવો ધાર ન હોય કે નહીં તે ટ્રૅક રાખીને ચક્ર ઘટકોને ટ્રૅક રાખવા સમાન છે. તો, આપણે ઘટકોનો ટ્રૅક કેવી રીતે રાખીએ છીએ, સારૂ, આપણે મૂળભૂત રીતે ઘટકોને લેબલ કરી શકીએ છીએ અને તે કારણ કે અમારી પાસે n શિરોબિંદુઓ છે અને શરૂઆતમાં ત્યાં તમામ સ્વતંત્ર સ્વતંત્ર શિરોબિંદુઓ છે જે ક્રુસ્કલના એલ્ગોરિધમ્સના પ્રારંભિક બિંદુમાં કોઈ કિનારીઓ નથી, બધું ડિસ્કનેક્ટ થઈ ગયું છે. તેથી, શરૂઆતમાં આપણે n શિરોલંબ n ઘટકો છીએ, તે એક વસ્તુ જે કહે છે તે ઉમેરે છે, પરંતુ પ્રત્યેક શિરોબિંદુ હું ઘટકો ભાગથી સંબંધિત છું. હવે, હું તમને ધાર ઉમેરી શકું છું, v જો તમારા ભાગનો ભાગ v ના ઘટકથી જુદો હોય અને તમે ઘટકના ઘટક પર v ના ઘટકોથી અલગ હોવ તો આ પછી શું થાય છે તે બે ઘટકો સમાન બને છે. તેથી, કેટલાક ઘટકોને મર્જ કરવા માટે કેટલાકને ઉમેરો, બીજી બાજુ તે સમાન મૂર્તિ પર આધાર રાખે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:37)

તેથી, હવે આ એલ્ગોરિધમનો વિગતવાર વિવરણ છે, તેથી આપણે ઈ 1 ને વજન દ્વારા ગોઠવેલ કિનારીઓ આપીએ તે પહેલાં, શરૂઆતમાં આપણે કહ્યું કે દરેક બિંદુ 1 t તેનાં ઘટકોમાં છે j ની n ઘટકોમાં દરેક j માટે j છે. તો, 1 એ ઘટકો 1 અને 2 ઘટક 2 છે અને તેથી આગળ, હવે આપણે કિનારીઓના ખાલી સમૂહથી પ્રારંભ કરીએ છીએ અને હવે આપણે આ કિનારીઓ આગળ ચઢતા ક્રમમાં ગોઠવીશું. તેથી, અમારી પાસે અનુક્રમિત સૉર્ટમાં પહેલી ધાર તરફ નિર્દેશાંક ઈન્ડેક્સ છે, તેથી આ મારી અનુક્રમણિકા છે. તેથી, જ્યાં સુધી આપણે તેને n ઓછા ખૂણામાં ઉમેરતા નથી, આપણે આ ક્રમમાં ગોઠવેલ ક્રમમાં હું ધારની ધાર તરફ જોયેલો છે, તે તમારી વિરુદ્ધ છે. જો તમારું વર્તમાન ઘટક v ની વર્તમાન ઘટકમાં જુદું હોય, તો પછી અમને ખાતરી છે કે ધાર ઉમેરવું ચક્ર બનાવશે નહીં. તેથી, આપણે એક વૃક્ષમાં ઉમેર્યા છે અને હવે આપણે ઘટકોના આ જોડાણને કરવું પડશે, હવે કર્ણનો ઘટક માત્ર એક સંખ્યા છે. તેથી, અમારા સમૂહમાં દરેક j માટે હવે સમસ્યા એ છે કે આ ઘટકો પહેલાં મર્જ થઈ શકે છે. તેથી, તમારા ઘટકોમાં v ને અન્ય શિરોબિંદુ ઘટક શામેલ હોઈ શકે છે જે બીજા શિરોલંબને કરી શકે છે અને તે બધાને હવે આપણે સમાન ઘટક સંખ્યા આપવી જોઈએ. તેથી, તે બધા નવા સમાન સંખ્યા છે, તેથી મારી પાસે કોઈ વિકલ્પ નથી, પરંતુ મારા j શિર્ષકોમાં દરેક j સ્કેન કરવા માટે અને જ્યારે પણ ધારે પાસે સમાન ઘટકો છે, ત્યારે હું તમને કયા ઘટકને ફરીથી સેટ કરું છું. તેથી, આ બધું પછી જે સમાન ઘટકોની સંખ્યા છે v તરીકે v એ

તમને સમાન ઘટક નંબરો મળ્યા છે. તેથી, તેથી, અસરકારક રીતે બે ઘટકોને એકના ઘટકોમાં મર્જ કરવામાં આવ્યા છે. તેથી, આ ક્રુસ્કલ(Kruskal)ના એલ્ગોરિધમ્સ છે, ફક્ત કહે છે કે ધારને ધ્યાનમાં રાખવું અને જો તેઓ જુદા જુદા ઘટકોને જોડે છે, તો તેમાં ઉમેરો અને ઘટકોને મર્જ કરો, જો તેઓ વિવિધ ઘટકોને કનેક્ટ ન કરે તો તે એક દેખાવ બનશે, તેથી માત્ર તેમાંથી જ દૂર થઈ જશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:26)

તેથી, ક્રુસ્કલ(Kruskal)ના એલ્ગોરિધમ્સમાં પ્રથમ જટિલતા શું છે તે તેને ધારને સૉર્ટ કરવાની જરૂર છે. તો, આપણે જાણીએ છીએ કે આપણે n ધાર અને m લોગ m ટાઈમને સૉર્ટ કરી શકીએ છીએ, પરંતુ n એ મોટાભાગના n ચોરસમાં છે, તેથી લોગ ઈન 2 વખત લોગ n હશે. તો લોગ n નો ક્રમ લોગ એમના ક્રમમાં સમાન છે. તેથી, હું એમ પણ લખી શકું છું કે m લોગ n છે, તેથી આ લખવાનું ફક્ત એક બીજું રસ્તો છે. તેથી, આ સૉર્ટિંગ એમ લોગ એમ સમય લેશે. હવે, અમારી પાસે બાહ્ય લૂપ છે

((સમયનોસંદર્ભ: 11:00)),

તેથી આ લૂપ જે અહીં ચાલે છે, સામાન્ય રીતે આ સૂચિમાંના તમામ કિનારીઓ દ્વારા ચાલે છે. કારણ કે, તે ખૂબ જ સારી રીતે થઈ શકે છે કે કોઈ પણ કારણસર પાછલા ભાગે વૃક્ષને મળ્યું. તેના બદલે, ઉદાહરણમાં આપણે સૌથી મોટા કિનારે જોયું તે વૃક્ષમાં શામેલ છે, કારણ કે તે એકમાત્ર એક છે જે આપણને ચોક્કસ શિરોલંબ સાથે જોડે છે. તો, આપણી પાસે બાહ્ય લૂપ છે, તે એમ લખાણ છે, હવે આ એમ સમયમાં આપણે ઘટકોને સુધારવાના દરેક ભાગ માટે અપડેટ કરીશું જે ફક્ત n ઓછા 1 વખત થાય છે. પરંતુ, જ્યારે આપણે ઘટકને અપડેટ કરીએ છીએ, ત્યારે જ આપણને સમસ્યા હોય છે, કારણ કે ઘટકોની સંખ્યા બદલવા માટે આપણે બધા શિરોબિંદુઓ દ્વારા સ્કેન કરવું પડશે. તેથી, ઘટકના પ્રત્યેક અપડેટ, દરેક મર્જિંગ ઘટકો કે જે તે ઓર્ડર એન ટાઈમ છે અને આ પોતે n ઓછા 1 વખત થાય છે. કારણ કે, આપણે દર વખતે જ્યારે વૃક્ષ પર ધાર ઉમેરીએ છીએ, ત્યારે તે બરાબર n ઓછા 1 થાય છે. તેથી, અમારે 1 ની બાદબાકી 1 અપડેટ્સ અને દરેક અપડેટ ઑર્ડર n વખત હોવું જોઈએ. તેથી, આપણી પાસે n ચોરસ ક્રમમાં એકંદર જટિલતા છે. હવે, યાદ રાખો કે પ્રીમ્સ(Prim's) એલ્ગોરિધમ એન સ્ક્વેર સૌથી ખરાબ કેસ હતો અને પછી અમે વધુ સારી માહિતી માળખું સાથે કહ્યું હતું કે અમે તેને નીચે વત્તા એમ પ્લસ n ને લાવી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:12)

તો, આપણે અહીં શું કરી શકીએ? તેથી, ચાલો આપણે સમસ્યાની તપાસ કરીએ, સમસ્યા મૂળભૂત રીતે આ લેબલિંગ વ્યૂહરચના છે. હવે, જ્યારે આપણે ઘટકોને લેબલ કરીએ છીએ અને આ રેખીય સ્કેન કરીએ છીએ ત્યારે ઘટકોને મર્જ કરવા માટે અમે ઓર્ડર n સમય પસાર કરી રહ્યા છીએ. તેથી, આપણે ડિજક્સ્ટ્રાના એલ્ગોરિધમ્સ અને પ્રીમ્સ(Prim's) એલ્ગોરિધમ્સ માટેના ઢગલા જેવા જ શોધી શકીએ છીએ જેથી તેમને લઘુતમ કણો ઉમેરવામાં અને અંતરને અપડેટ કરવા માટે, અહીં આપણે શોધીશું કે આપણે આ ઘટક માળખું જાળવી રાખવાની જરૂર છે. તેથી, ધારો કે આપણે સેટ વી સાથે કામ કરવા માંગીએ છીએ જે વિવિધ ઘટકોમાં ભાંગી છે. તેથી, તેમાં કોઈ પણ સમયે ઘટકો છે જેને તેઓ A, B, C, D, E કહે છે અને પછી મારી પાસે આ ઘટકોથી સંબંધિત વ્યક્તિગત ઘટકો છે. તેથી, તેમાંની સંખ્યા ઘણી છે, એકનો સરવાળો

અને તેથી વધુ. તેથી, તમે જે વસ્તુઓ કરવા માગો છો તેમાંથી એકનું નામ વર્ટેક્સ વી છે, તમારે પૂછવું છે કે કયા ઘટકો નથી. તેથી, આ શોધી કાઢવામાં આવે છે, ઘટક શોધી કાઢેલ છે, પરંતુ n બીજી વસ્તુ વર્ટેક્સ ને આપવામાં આવે છે. તો, માની લો કે મને au અને v આપવામાં આવે છે તે આ બંનેને મર્જ કરશે, તેથી આ બે ઘટકો લો અને પછી સમાન નામ મેળવો. તેથી, તેમને બધા સી કૉલ કરો અથવા તેમને બધા કૉલ કરો, તેથી તેને યુનિયન કહેવામાં આવે છે, યુનિયન તે ઘટકો લેતું નથી જે તે બે પ્રતિનિધિઓ લે છે, એક પ્રતિનિધિ એક ઘટકો અને અન્ય ઘટકોમાંથી એક પ્રતિનિધિ અને પોતે ઘટકોમાં રહેલી દરેક વસ્તુને મર્જ કરે છે. તમે ઘટક વી માં બધું સાથે. અને આ બંને કામગીરી છે; દેખીતી રીતે, આપણે ક્રુસ્કલ(Kruskal)ના એલ્ગોરિધમ્સ કરીએ છીએ, v એ શોધી કાઢવું જોઈએ કે જો તમારી પાસે ધાર હોય તો, તે જ ઘટક પર તપાસ કરવા માટે v એ તમને ઘટકની ઘટક મળી છે, તે સમાન છે. તેનો અર્થ એ છે કે, શામેલ ઘટક તેઓ અલગ અને અલગ ઘટકો છે અને અમે આ ધાર ઉમેરી શકીએ છીએ. હવે, જો તમે ધાર ઉમેરો છો, તો અમારે યુનિયન વિરુદ્ધ કરવું પડશે, તેથી આ ચોક્કસપણે બે ઓપરેશન્સ છે અને અમે શોધીશું કે આ ડેટા માળખું છે જેને યુનિયન તરીકે ઓળખવામાં આવે છે, આ બે ઓપરેશન્સને કારણે, જે વાસ્તવમાં આ કાર્યક્ષમ રીતે કરી શકે છે. . અને અસરકારક રીતે અર્થ થાય છે કે, તેનો અર્થ એ છે કે તે આ ચોરસ વસ્તુને દૂર કરી શકે છે અને m લોગ n ની કિંમત પર બધું લાવી શકે છે અથવા યાદ રાખવું જોઈએ કે અમને આ પ્રકારની જરૂર છે. અમારું પ્રથમ પગલું એ કિનારીઓમાં સૉર્ટ કરીને ક્રુસ્કલના એલ્ગોરિધમ્સ પ્રકારના કિનારીઓ જેવું હતું. તો, m લોગ n એ કંઈક છે જે આપણને ધારને સૉર્ટ કરવા માટે સમાન છે. આ ઉપરાંત, આવશ્યકપણે આ ઢગલામાં આ અપડેટ્સ, યુનિયન અપડેટ્સ વાસ્તવમાં લોગ n ને પાછું શોધે છે. અને યાદ રાખો કે આ અપડેટ્સ આ પુનરાવર્તિત લૂપમાં થઈ રહ્યું છે. તેથી, આપણે વાસ્તવમાં પ્રીમ્સની એલ્ગોરિધમની જટીલતાને આના જેવી જ કંઈક મેળવીએ છીએ.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક 5
મોડ્યુલ - 01
લેક્ચર - 32
યુનિયન - ડેટા સ્ટ્રક્ચર શોધો

તેથી, જ્યારે આપણે લઘુતમ પાથો માટે અને લઘુતમ પાથ માટે વજનવાળા આલેખ જોવા અને વૃક્ષોના લઘુત્તમ ખર્ચ માટે, કેટલાક અપડેટ્સને કાર્યક્ષમ બનાવવા માટે ડેટા સ્ટ્રક્ચર નો ઉપયોગ કર્યો. તેથી, તે સમયે અમે ધારીએ છીએ કે આ ડેટા સ્ટ્રક્ચર ઉપલબ્ધ હતા અને અમે આ અલ્ગોરિધમ્સના વિશ્લેષણને નિર્ધારિત કરવા આગળ વધ્યા. હવે, ચાલો પાછા જઈએ અને આ ડેટા સ્ટ્રક્ચરને જોઈએ. તેથી, અમે યુનિયનને ડેટા સ્ટ્રક્ચર સાથે પ્રારંભ કરીએ છીએ જેનો ઉપયોગ ક્રુસ્કલના અલ્ગોરિધમમાં (Kruskal's algorithm) ઓછામાં ઓછા ખર્ચવાળા વૃક્ષ માટે થાય છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:28)

તો, યાદ કરો કે ક્રુસ્કલનું એલ્ગોરિધમ (Kruskal's algorithm) કેવી રીતે કાર્ય કરે છે. અમે કિનારીઓના ચઢતા ક્રમમાં ગોઠવણી કરીએ છીએ અને અમે આ ક્રમમાં કિનારીઓની પ્રક્રિયા કરીએ છીએ. તેથી, અમે દરેક ધાર પસંદ કરીએ છીએ. જો તે કોઈ ચક્ર બનાવતું નથી, તો આપણે તેને વૃક્ષમાં ઉમેરીએ છીએ અને આપણે અવલોકન કરીએ છીએ કે ચક્ર બનાવવું એ અત્યાર સુધીના ઘટકોને ટ્રૅક રાખવા જેવું જ છે, અને એ તપાસવું છે કે અંતિમ બિંદુઓ વિવિધ ઘટકોને રેખા બનાવે છે. તેથી, જો u અને v હાલમાં જોડાયેલ ન હોય તો ધાર યુવી ઉમેરી શકાય છે. તેઓ એક જ ઘટકમાં નથી. હવે, ધાર ઉમેરવાનું પરિણામ તરીકે, બે ઘટકો જોડાયા છે. તેથી, આપણે તે બે ઘટકોને મર્જ કરવું પડશે. તેથી, ક્રુસ્કલના એલ્ગોરિધમનો (Kruskal's algorithm) અમલ કરવા માટે મુશ્કેલી એ કાર્યક્ષમતાના આ સંગ્રહને ટ્રૅક રાખવા માટે છે જેમાં ખૂણો કયા ઘટકને અનુસરે છે અને બે ઘટકોને મર્જ કરવા માટે, જ્યારે પણ અમે તેમને જોડતા ધાર ઉમેરીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:22)

તેથી, ઔપચારિક રીતે આપણે જે સમસ્યાનો ઉકેલ લાવવાનો દાવો કર્યો છે તે સેટનો પાર્ટીશન જાળવવા અને આ પાર્ટીશનને અપડેટ કરવા માટે છે. તેથી, પાર્ટીશન દ્વારા અમારું મતલબ એ છે કે આપણી પાસે એક સેટ છે અને તે કેટલાક અલગ અલગ ભાગોમાં તૂટી ગયું છે. તેથી, આ સબસેટ ઓવરલેપ થતા નથી અને દરેક તત્વ આમાંના કોઈ પણ વસ્તુથી સંબંધિત નથી. તેથી, ત્યાં કેટલાક હોઈ શકે છે કે જેમાં એકથી વધુ તત્વ હોય અને દરેક તત્વ એક ભાગમાં બરાબર સોંપાયેલ હોય, અને આપણે આ પાર્ટીશનોને ઘટકો પણ કહીએ. તેથી, ક્રુસ્કલના એલ્ગોરિધમ (Kruskal's algorithm) અને અન્ય એપ્લિકેશન્સમાં, ઘણીવાર આપણે એક પાર્ટીશન શરૂ કરીએ છીએ જેમાં દરેક તત્વ ચાલુ છે, તે સમાપ્ત થાય છે. તો, આપણી પાસે પાર્ટીશનમાં કદાચ બે ઘટકો હોતા નથી. તેથી, અમે આ સેટિંગને ડેટા સ્ટ્રક્ચર તરીકે કૉલ કરીશું. તેથી, અમે તેને યુનિયનને બોલાવીશું કારણ કે આ ડેટા સ્ટ્રક્ચર પર ખરેખર જે બે ઓપરેશન્સ અમે સપોર્ટ કરીએ છીએ તે શોધવામાં આવે છે. તેથી, આ એક ક્વેરી ઓપરેશન છે. તે એક તત્વ છે. મને જણાવો કે હાલમાં તે કયા ઘટકથી સંબંધિત છે. તેથી, આ

એક અપડેટ છે જે તે ડેટા સ્ટ્રક્ચર અપડેટ કરતું નથી. તે માત્ર ડેટા સ્ટ્રક્ચર પૂછે છે અને અમને જણાવે છે કે આમાંના કયા પાર્ટીશનો હાલમાં અસ્તિત્વમાં છે, અને પછી અમારી પાસે એક અપડેટ છે જે આપણને બે ભાગો લેવાની પરવાનગી આપે છે, . તેથી, આપણે આ બે પાર્ટીશનો લઈ શકીએ છીએ અને હવે તેમને એક પાર્ટીશનમાં ભેગા કરીએ. તેથી, આપણે આ યુનિયનને, સાચું કહીએ છીએ. તેથી, એક યુનિયન ઓપરેશન છે જે એક સાથે પાર્ટીશનોને મર્જ કરે છે અને ત્યાં એક શોધ ઓપરેશન છે કે જે કયા પાર્ટિશનને કહેવામાં આવે છે તેનો ટ્રેક રાખવો પડે છે જે તત્વ સાથે સંકળાયેલી હોય છે. આ બે ઓપરેશન્સ યુનિયનને લીધે તે મૂળભૂત રીતે અન્ય પાર્ટીશનો સાથે મર્જ થઈ જશે અને શોધશે. અમે આ યુનિયનને ડેટા સ્ટ્રક્ચર શોધીએ છીએ જે આ બે કામગીરીને અસરકારક રીતે ટેકો આપે છે, અને આ યુનિયનનો પ્રારંભ એ એક ઑપરેશન છે જે એક સેટ લે છે અને તેને તોડે છે. તેમાં n એ ઘટક છે જે દરેક ઘટકો ધરાવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:09)

તેથી, પ્રથમ સમસ્યા જેનો આપણે સામનો કરવો પડશે તે ઘટકોના નામો વિશે છે. આપણે આ ઘટકોને શું કહીએ? તે એક સરળ ઉકેલ છે. આપણે ફક્ત સમૂહના ઘટકોને નામો તરીકે જ ઉપયોગમાં લઈશું. તેથી, આપણે કયા નામ આપીએ તે ખરેખર કોઈ વાંધો નથી. આપણે માત્ર સમયાંતરે તપાસ કરવા માટે સમર્થ હોવા જોઈએ કે શું S અને T અથવા u અને v એ સમાન ઘટક છે. તેથી, આપણે જાણવાની જરૂર છે કે તમે શોધવા માટે v ની શોધ બરાબર છે. તેથી, અમે તમને શોધી કાઢીએ છીએ અને વી શોધવા માટે કેવી પસંદગી કરીએ તે ચોક્કસ પસંદગી નથી. તેથી, જ્યાં સુધી આપણે ચકાસી શકીએ કે બે લેબલ્સ સમાન અથવા અલગ છે, પરંતુ તેનામાંથી લેબલ્સના નિર્માતા સમૂહને બદલે, આપણે વાસ્તવમાં તત્વોને સેટ કરવા માટે લેબલ્સ પસંદ કરીશું. તેથી, શરૂઆતમાં આપણે કહ્યું હતું કે દરેક તત્વ એક ભાગમાં છે. તેથી, માની લો કે મારી પાસે STU શામેલ છે, પછી શરૂઆતમાં ત્રણ ભાગો હશે. એકમાં એસ હોય છે, જેમાં એક સમાવતું હોય છે અને તે એક છે. તમારી પાસે પ્રશ્ન છે કે આપણે આ પાર્ટીશનને શું કહીએ છીએ. ઠીક છે, આપણે તેમને એક જ વસ્તુ કહીએ છીએ. અમે આ પાર્ટીશનને બોલાવીએ છીએ, આ પાર્ટીશન ટીને બોલાવો અને આને પાર્ટીશન કહીએ. તેથી, કેટલીકવાર તત્વના નામો પાર્ટિશનોના નામનો સંદર્ભ લેશે. કેટલીકવાર તેઓ પોતાને તત્વોના નામનો ઉલ્લેખ કરશે. હવે, જ્યારે આપણે મર્જ કરીએ ત્યારે શું થાય છે? ઉદાહરણ તરીકે, ધારો કે આપણે આ બે પાર્ટીશનો મર્જ કરીએ, . પછી, લેબલ સમાન હોવું જોઈએ. તેથી, તત્વ બદલાતું નથી, પણ કદાચ આપણે સેટ લઈ શકીએ, લેબલ આપીએ અને તેને બનાવી શકીએ. તેથી, હવે, બંને તત્વ u અને તત્વ ટી પાર્ટીશન લેબલ ટી, જમાણી છે. તેથી, આપણે ફક્ત લેબલ્સના સમૂહ તરીકે, તત્વોના નામનો ઉપયોગ કરીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:44)

તેથી, ખાસ કરીને જો તમે ગ્રાફ સાથે કામ કરી રહ્યા છો, તો તત્વો માટે શિરોબિંદુઓ છે અને અમારી પાસે પહેલાથી જ સંમેલન છે કે આપણી પાસે સેટમાં n શિરોબિંદુ છે અને આપણે તેને 1 ને n ને કહીએ છીએ. તેથી, તત્વોનો સમૂહ n થી n છે અને સ્રોત ઘટકોનો સમૂહ છે. તો, અરે, આપણે હવે શું કરીશું તે ટ્રેક રાખવાનો સૌથી સરળ રસ્તો એરે સુયોજિત કરવા માટે છે. તો, આપણી પાસે એરે છે જે આપણે ઘટ્ટ કરીશું અને આ એરે શું કહેશે. ઠીક છે, આ કહેશે કે દરેક શિરોલંબ માટે ગાંઠો સમાપ્ત થાય છે, તે કયા ઘટક સાથે છે, . તેથી, શરૂઆતમાં આપણે કહ્યું હતું કે પ્રત્યેક ઘટકમાં બરાબર એક ભાગ હશે. તેથી, અમે થોડા સમય પછી દરેક માટે સામાન્ય રીતે શિરોબિંદુ n, શિરોબિંદુ I અને ઘટક I પાસે હોઈ શકે છે, આ બદલાશે.

તેથી, આ ઘટક 3 પર ગયો હોઈ શકે છે કદાચ તે ઘટક 7 પર ગયું હશે. તેથી, સમય જતાં ઘટક જે યુનિયન ઓપરેશન્સને લીધે બદલાવ સાથે સંકળાયેલો છે, તેથી જ્યારે આપણે તમને શોધીશું, ત્યારે માત્ર ઘટકનું વર્તમાન મૂલ્ય પરત કરીશું અને યુનિયન માટે આપણે ફક્ત તે જ કરવું પડશે, તમારે બધા ઘટકો તપાસવા અને બનાવવું પડશે. , બંને ઘટકો k અને k પ્રાઈમ સમાન લેબલ ધરાવે છે. તો, નવું લેબલ શોધવાની જગ્યાએ, આપણે ક્યાં તો કે અથવા કે પ્રાઈમ પસંદ કરીશું. આ કિસ્સામાં આપણે કી વડા પસંદ કરીએ છીએ. તેથી, આપણે શું કરીશું, આપણે પસાર કરીશું અને જ્યાં પણ આપણે AK જોશું, આપણે એ.કે. પ્રાઈમ દ્વારા બદલીશું. તેથી, આપણે આ એરેમાં ફોર્મ કે ની પ્રત્યેક એન્ટ્રીને કે કીમાં ગોઠવીશું. તો, આ બધા ઘટક મૂલ્યો પછી આપણે k અથવા હવે k પ્રાઈમ, બધા વર્ચ્યુઅલ કે વડા હોવાનું પસંદ કરીએ છીએ અને કી મુખ્ય રહે છે. તેથી, અસરકારક રીતે બે ઘટકો મર્જ કરવામાં આવ્યા છે. તેથી, આ યુનિયન શોધ એક ખૂબ સરળ અમલીકરણ છે. તેથી, ચાલો આપણે એક જટીલતા દૃષ્ટિકોણથી આ એક ખૂબ જ સારી અમલીકરણ કેમ નથી તે સમજવા અને સમજવા દો.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 06:35)

તેથી, પ્રારંભિક વસ્તુઓ બનાવવા માટે સ્પષ્ટ રીતે, આપણે એકવાર એરેને સ્કેન કરવું પડશે, અને પછી મારે ફક્ત મૂલ્ય I. તરીકે ઘટકને પ્રારંભ કરવું પડશે. તેથી, આ ક્રમમાં n સમય. તેથી, આ શોધી કાઢ્યું છે. એ જ રીતે, તત્વ શોધી કાઢવું એ કાર્યક્ષમ છે. આપણે ફક્ત એરેમાં ith ઘટકને જોવું જોઈએ અને યાદ રાખવું જોઈએ કે કોઈપણ તત્વને ઍક્સેસ કરતી એરેમાં સતત સમય લાગે છે. તેથી, આ એક કાર્યક્ષમ ઑપરેશન છે. તે સતત સમય લે છે. બીજી બાજુ, યુનિયન એ એક સમસ્યા છે કારણ કે આપણે યુનિયનને વર્ણવ્યું છે તે રીતે, આપણે દરેક નોડ દ્વારા પસાર થવું પડશે, તેનું ઘટક છે કે નહીં તે તપાસવું જોઈએ અને જો તેનું ઘટક એ છે, તો ઉપકરણ કે ઘટક જો ઘટક છે, તો તેને અપડેટ કરવું પડશે. તેથી, કે જે ઘટકો કે કે કે જે હાલમાં ઘટકો જેવા દેખાય છે તેના પર ધ્યાન આપ્યા વિના, આપણે બધા ઘટકોને સ્કેન કરવું પડશે અને કે જે કી કે કી છે તે અપડેટ કરવું પડશે. તેથી, આ માત્ર એક યુનિયન ઓપરેશન માટે ઓર્ડર એન સમય લેશે જે વર્તમાન પાર્ટીશન સેટ્સ તરીકે કે કે કે કદના કદથી સ્વતંત્ર છે. તેથી, જો આપણે એમ જેવી કામગીરીનું અનુક્રમણિકા કરીએ, તો આ ઓર્ડર એમ ટાઈમ્સ n હશે, અને જો તે એન જેવી ક્રિયાઓ હશે, તો તે n ચોરસ હશે, . તેથી, એક અનુક્રમણિકા એમ ઓપરેશન, તેમાંના દરેક ઑર્ડર એન ટાઇમ લેશે અને અમે તેના પર વધુ સુધારો કરવા માંગીએ છીએ. તેથી, મૂળભૂત રીતે આપણે જોવા માંગીએ છીએ કે આપણે યુનિયન ઓપરેશનની ગતિમાં સુધારો કરી શકીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:54)

તેથી, ચાલો આપણે થોડો વધારે વિસ્તૃત પ્રતિનિધિત્વ કરીએ. તેથી, આપણે આ એરે ઘટકને પહેલાની જેમ રાખીએ છીએ જે આપણને દરેક ભાગ માટે કહે છે જે તે જે ઘટક છે તે, અને પહેલાનાં ઘટકોને શરૂઆતમાં 1 થી n તરીકે લેબલ થયેલ છે. નામો એક જ સેટમાંથી દોરવામાં આવે છે. તેથી, મૂળરૂપે 1 થી n એ શિરોલંબ 1 થી n છે અથવા નામો ઘટકો પણ છે, અને શરૂઆતમાં હું ભાગનો ભાગ છે. હું એક ઘટક સમય છે. હવે, આપણી પાસે અલગ સૂચિ સૂચિ છે, . તેથી, દરેક ઘટક માટે, અમે હાલમાં સૂચિ રાખીએ છીએ. તેથી, શરૂઆતમાં સૂચિ એ છે કે ઘટક 1 માં વર્ટીક્સ 1 ઘટક, 2 શામેલ છે અને 2 થી વધુ છે, પરંતુ સમયાંતરે આપણે પરિસ્થિતિ ધરાવી શકીએ છીએ કે ઘટક 4, 1, 2, 4 અને શામેલ છે. 7, બરાબર. તો, દરેક ઘટક માટે આપણી પાસે શિર્ષકોની સૂચિ છે જે તે સંબંધિત છે અને અમે પણ કદને અલગ રીતે રાખીશું. તેથી, આપણે કહીશું કે આ ઘટકનું કદ પણ 4 છે અને આ ઘટકનું કદ 1 છે. તેથી, અમારી પાસે બે સહાયક વસ્તુઓ છે. અમે તેના ઘટકોની યાદી

સ્પષ્ટપણે દરેક ઘટક માટે રાખીએ છીએ અને અમે આ સૂચિનું કદ પણ રાખીએ છીએ. તેથી, આપણે જાણીએ છીએ કે શરૂઆતમાં કોઈ પણ સમયે દરેક ઘટક કેટલો મોટો છે, અલબત્ત, કદ 1 છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 09:10)

તેથી, જ્યારે આપણને જરૂર હોય ત્યારે યુનિયનને શોધી કાઢો, અમે કહ્યું કે ઘટક I સમાન છે પહેલાંના I ની જેમ, પછી અમે પ્રારંભ કરીએ છીએ કે I સૂચિના સભ્યો જે ઘટક છું તે I દરેક ઘટકના કદમાં રહેલી સૂચિને સમાયોજિત કરું છું. હવે, પહેલા બરાબર તે જ છે તે શોધો. આપણે ફક્ત ઘટકને જોવું જોઈએ અને મૂલ્ય પરત કરીશું. I જે ઘટકને બે યુનિયનો નિર્દેશ કરું છું તે પહેલાં પણ સમાન છે. તેથી, આપણને જે કરવાની જરૂર છે તે છે, આપણે k ને નિર્દેશ કરતી બધી વસ્તુને k ને અંદરની બાજુએ મુકવાની જરૂર છે. તેથી, ઘટકની પ્રત્યેક એન્ટ્રી કે જે કે, કી હોવા જોઈએ, પરંતુ હવે આપણે સભ્યોને જોઈને આ કરી શકીએ છીએ. તેથી, આપણે બધા તત્વો 1 ને n ને સ્કેન કરવાની જરૂર નથી. આપણે k ના સભ્યોમાં દેખાય છે તે દરેક તત્વને જોઈ શકીએ છીએ અને તેના મૂલ્યને કી પ્રાઈમ, અપડેટ કરી શકીએ છીએ. તેથી, આ એક બચત છે. હવે આપણે 1 થી એન સુધી જવાની જરૂર નથી. આપણે ફક્ત તે જ સભ્ય પર જ જોયું છે જે સેટ કે ની છે. પછી, અલબત્ત, આપણે આ નવી વસ્તુઓને અપડેટ કરવાની જરૂર છે. તેથી, કે ના સભ્યો હવે કે પ્રાઈમ્સના સભ્ય બન્યા છે. તેથી, આપણે આ બે યાદીઓને મર્જ કરીશું. હવે, યાદ રાખો કે આ બે સૂચિ સૉર્ટ કરેલ ક્રમમાં છે. આપણે ધારી લઈએ છીએ કે તેઓ હંમેશાં તત્વોના નામ ઉપર ચઢતા રહે છે. તેથી, બે સૉર્ટ કરેલ સૂચિને મર્જ કર્યા પછી અમે સૉર્ટ મર્જ કરવામાં કર્યું, અંતિમ સૂચિની લંબાઈ માટે પ્રમાણસર સમય લે છે. તેથી, આ ઘટક કદ અને કે વડાના કદના પ્રમાણમાં રેખાકીય સમય વસ્તુ હશે. છેવટે, હવે આપણી પાસે એક નવું ઘટક છે જે પહેલાની તરફ વહે છે. તેથી, તેનો કદ બરાબર બે પાછલા બેનો સરવાળો છે. આ પાર્ટીશનો છે. કોઈ હા માર્ગ પુનરાવર્તન કરવામાં આવ્યું હતું. તેથી, દરેક ઘટક જે પાર્ટીશનમાં જોડાય છે તે એક નવું છે. તેથી, k પ્રાઈમનું માપ બરાબર માપનું કે વત્તા જૂના કદના કી વડા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:47)

તેથી, અમને આનાથી કેટલાક ફાયદા કેમ મળે છે? તો, પ્રથમ વસ્તુ જે આપણે કહ્યું છે તે ઘટકને અપડેટ કરવાનું હવે તેના કદ માટે પ્રમાણસર છે. અપડેટ ઘટકને ચાલુ રાખવાથી k નો ઓર્ડર કદ રાખવામાં આવે છે. તે ક્રમમાં પગલાં લેવા નથી. આપણે દરેક વર્ટીક્સ 1 થી એન સુધી જવાની જરૂર નથી. આપણે કે સભ્યોના સભ્યોમાં ઉલ્લેખિત તે તત્વોને સ્પષ્ટ રૂપે જોઈ શકીએ છીએ અને તે મૂલ્યોને ફક્ત અપડેટ કરી શકીએ છીએ, પરંતુ કે ના કદ ખરેખર વધુ મહત્વપૂર્ણ ભૂમિકામાં સ્થાન લે છે. આપણે હવે શું કરી શકીએ તે આપણે નક્કી કરી શકીએ કે કે ફરીથી કી લેવું કે કે કી કે કે પ્રાઈમ અથવા કે પ્રાઈમ છે કે. યાદ રાખો કે જ્યારે આપણે k અને k પ્રાઈમ મર્જ કરીએ ત્યારે આપણી પાસે પસંદગી છે. બધા તત્વો એક જ ઘટકમાં ભાગ બનશે. તેથી, નવો ઘટક કાં તો કૉલ કે હશે અથવા તે કૉલ કે પ્રાઈમ હશે. તેથી, આપણે કોની પસંદગી કરવી જોઈએ? તો, આ વ્યૂહરચના જે આપણે કરવા જઈ રહ્યા છીએ તે મોટાનું નામ રાખવા માટે છે. તેથી, k ના કદ એ કી વડાના કદ કરતાં નાનું છે. તેનો અર્થ એ કે, k વડામાં હાલમાં k માં વધુ સભ્યો છે. પછી, આપણે અંતિમ સમૂહના નામ તરીકે કી વડા રાખશું. તેથી, આપણે બધા કેસ કે કીને બદલીશું અને કીનું સપ્રમાણ કદ માપશે કી કદના કદ કરતા મોટું. આપણે બધા k primes ને k તરીકે, બદલીશું. તેથી, નાનો સેટ તેનું નામ બદલી દે છે અને મોટો સેટ તેનું નામ રાખે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:58)

તેથી, આ કોઈ વ્યક્તિગત સંઘ કામગીરીના ખરાબ કેસમાં અમને કોઈ લાભ આપતું નથી. ધારો કે આપણી પાસે કદ અને કદના કદનો કદ છે જે લગભગ અડધા કદના છે. તેથી, જો આપણે બે ઘટકો બનાવ્યાં છે જે કુલ સેટના આશરે અડધા કદના છે, તો પછી આપણે કે કે બીજાના સમૂહમાં મર્જ કે કે, આપણે અડધા મૂલ્યોને અપડેટ કરવું પડશે. તેથી, આ આર ઓર્ડર ઓપરેશન છે, . તેથી, અગાઉ આપણે કહ્યું છે કે કંઈપણ ફેન્સી વગર, અમે બધા શિરોબિંદુઓને સ્કેન કરીશું અને વાસ્તવમાં સૌથી ખરાબ કેસ એ છે કે અપડેટના દરેક કેસ ઑર્ડર એન ટાઈમ તરીકે લેશે. હવે, તે કહે છે કે ત્યાં એક ખરાબ કેસ છે જ્યાં અમે ઓર્ડર એન ટાઈમ ટાળવાનું ટાળી શકતા નથી. તેથી, n દ્વારા 2 એ ક્રમ n છે. તેથી, તેથી આપણે શું પ્રાપ્ત કરીએ? તેથી, આપણે જે મેળવી છે તે વ્યક્તિગત મર્જ ઑપરેશનની શરતો માટે ખરેખર જવાબદાર નથી. આપણે મર્જર્સના પંચની નમ્રતા અસર જોવાની છે. તેથી, આપણે વધુ સાવચેત એકાઉન્ટિંગ કરવાની જરૂર છે. હવે, આપણે ઓપરેશન્સ માટે જવાબદાર છે. તેથી, યાદ રાખો કે અમે કેટલીક સાવચેતીપૂર્ણ એકાઉન્ટિંગ કરી હતી જ્યારે અમે તમારા જેવા વસ્તુઓ કરી હતી જ્યારે અમે પહોળાઈની પ્રથમ શોધમાં નજીકની સૂચિનો ઉપયોગ કર્યો હતો. અમે બધા આંટીઓ પર કહ્યું, અમે દરેક ધાર બરાબર બે વખત જોશું. તેથી, બધા આંટીઓ પર અમે ઓર્ડર એન સમય લે છે. તેથી, આપણે એક સમાન પ્રકારનું સાવચેતીપૂર્ણ એકાઉન્ટિંગ કરવાની જરૂર છે, એક મર્જ થઈ શકશે નહીં, પરંતુ તે તમામ મર્જ કરે છે કે તેઓ x લે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:17)

તેથી, નાના સેટને મોટા સમૂહમાં મર્જ કરવાની અસર એ છે કે જો હું વ્યક્તિગત તત્વમાં જોઉં, તો ઘટક બદલાય તો તે લેબલ થયેલ છે. જો ઘટક વર્તમાનમાં k છે અને તે કી વડા બને છે, તો મતના કારણે નવા સમૂહ ઓછામાં ઓછા બે વખત છે. તો, માની લો કે મેં કેલ અને એક તત્વ સુયોજિત કર્યું છે અને મારી પાસે બીજું સેટ કે પ્રાઈમ છે, તો પછી હું તેને બે સિંગલ સેટમાં મર્જ કરવાનો નિર્ણય કરું છું. હવે, ધારણા દ્વારા જો નવા સેટને કે વડા કહેવામાં આવે છે, તો તેનો અર્થ એ કે કે k એ કી કરતાં ઓછું ખરાબ છે અથવા તમે તેના કરતા ઓછું કહી શકો છો, તે ખરેખર વાંધો નથી. તેથી, જ્યારે કે કી કરતાં નાના કિસ્સામાં, તેનો અર્થ એ છે કે જો હું કે પ્લસ કે જે જુનું સેટનું કદ બમાણું હશે, તો તે સ્થાનાંતરણ દ્વારા આ રીતે કે પ્લસ કે વડા કરતા ઓછું અથવા સમાન હશે. તેથી, આ નવા સેટનો આકાર છે. કે પ્લસ કે પ્રાઈમ એ નવા સેટનું કદ છે જે મેં બનાવેલા છે અને તે કદના ઓછામાં ઓછા બમણા કદનું હશે. તેથી, જ્યારે પણ કોઈ તત્વના સેટ લેબલિંગ ઘટકનું નામ બદલાતું હોય, ત્યારે નવું સેટ કદને ઓછામાં ઓછું બમણો છે. તો, હવે ચાલો પ્રારંભિક સ્થિતિથી શરૂ કરીને એમ જોડાણ પ્રક્રિયાના કેટલાક અનુક્રમને જોઈએ, જ્યારે મારી પાસે અલગ પાર્ટીશનોમાં બધા ઘટકો હોય. તો, દરેક ઓપરેશનમાં શું થઈ શકે છે, કદાચ હું બે ઘટકો ભેગા કરીશ. હવે, જો આગલી વખતે જો હું આ બંનેને જોડું, તો પછી સંપૂર્ણપણે બે ઓપરેશનોમાં હું ફક્ત ત્રણ ઘટકોને અસર કરીશ. તેથી, સૌથી ખરાબ કિસ્સામાં જો હું આ અલગથી કરવાનું શરૂ કરું, તો દરેક વખતે એક ઓપરેશનમાં હું સ્વીકારું છું કે તે અસર કરે છે અને અન્ય કામગીરીમાં તે વધુને વધુ અસર કરે છે. તેથી, મોટાભાગના 2 મી તત્વો પર એમ ઓપરેશન્સના અનુક્રમ પછી પ્રારંભિક શરતમાંથી સ્થિતિ બદલાઈ ગઈ છે, જ્યારે તેમની પાસે પોઈન્ટિંગ બે અથવા ઘટક છે જે ફક્ત પોતાને જ સમાવે છે. તેથી, તેનો મતલબ એ છે કે, મારા યુનિયન ઓપરેશન્સ દ્વારા ફક્ત 2 મી તત્વોને અસર થઈ છે. તેનો અર્થ એ છે કે ઘટક 2m કરતા મોટો ન હોઈ શકે, કારણ કે કોઈ ઘટકમાં પ્રવેશ કરવા માટે, કંઈક બદલવું આવશ્યક છે. ફક્ત 2 મી તત્વોને જ તેમને કોઈપણ ફેરફારો લાગુ કરવાની મંજૂરી

છે. તેથી, મોટા ભાગનાં 2 મીટરના જોડાણ પછી, કોઈપણ ઘટકનું કદ, પરંતુ કદ કેવી રીતે વધે છે? તે 1, 2, 4 જાય છે કારણ કે તે એમ સુધી બમાણું રાખે છે, સાચું. તેથી, જો લોગ એમ પગલાંઓ પછી હું 1 લોગ એમ વખત બમાણી કરીશ, તો હું એમ મેળવીશ. તેથી, તેથી, કોઈ ઘટકને નિશ્ચિત રીતે ફરીથી લેબલ કરી શકાય છે, માફ કરશો, નિશ્ચિત તત્વને મોટાભાગના લૉગ એમ સમયે ફરીથી લેબલ કરી શકાય છે. તેથી, કારણ કે આપણી પાસે આ બમાણું છે કે જે સેટ કરવામાં આવે છે તે સંખ્યા એક તત્વ હોઈ શકે છે તેને નવા સમૂહમાં ખસેડી શકે છે. તે પ્રતિબંધિત છે કારણ કે દર વખતે તે તેના ઘટક ડબલ્સના સાઈટ કદને ખસેડે છે, અને તે મોટા ભાગનાં ઘટકની મર્યાદા છે જે તે સંબંધિત રહેશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:22)

તેથી, જો આપણે કેટલાક કુલ સંધીય કામગીરીને જોયા, તો આપણે જાણીએ છીએ કે 2 મીટર. તેથી, ઓર્ડર એમ ઘટકો પાસે ઘટકને અપડેટ કરાયું છે અને મોટાભાગના લોગ એમ ટાઈમ્સ પર દરેકને અદ્યતન કરવામાં આવ્યું છે. યાદ રાખો કે જ્યારે આપણે ઘટકોને અપડેટ કરી રહ્યા છીએ, ત્યારે અમે કોઈપણ તત્વને સ્પર્શતા નથી જે અદ્યતન નથી. તે જૂનું સેટઅપ નથી, જ્યાં આપણે નોડ 1 ને બધાને સ્કેન કરવું પડશે, તે નક્કી કરવા માટે કે જે કોઈ નથી, કારણ કે અમારી પાસે સૂચિ સભ્યો છે. જ્યારે આપણે ઘટક k ને અપડેટ કરવા માંગીએ છીએ, ત્યારે આપણે તે ઘટકોને બરાબર અપડેટ કરીએ છીએ. તેથી, અમે ફક્ત ઘટકોને જ સ્પર્શ કરીએ છીએ. તેથી, એમ તત્વો લોગ એમ ટાઈમ્સમાં ફેરફાર કરે છે. તેથી, તદ્દન અમારી પાસે m લોગ m પરિવર્તનના પગલાં છે, . તેથી, જો આપણે યુનિયન ઓપરેશન્સ કરીએ, તો અમે એમ લઈશું. તે એમ નથી કે પ્રત્યેક વ્યક્તિ એમ લોગ એમ ટાઈમ સીધી લે છે, પરંતુ એમ યુનિયન ઓપરેશન્સનું સંયુક્ત કુલ માત્ર એમ લોગ એમ છે, . તેથી, આ રીતે તમે કહી શકો તે રીતે અમે સરેરાશ કહી શકીએ છીએ, કારણ કે જ્યાં એમમાં કુલ ક્રિયાઓ એમ લોગ એમ હોય છે, તેમછતાં પણ સરેરાશમાં થોડી વધારે મોટી હોય છે, તો તેઓ લોગ એમ ઓપરેશન્સ લે છે. તેથી, આ એક અલગ પ્રકારની વિશ્લેષણ છે. તે માટેનું વિશ્લેષણ અમે કર્યું નથી. દા.ત. મુદ્દતની મુદ્રા એસ ની છે, જ્યાં અમે હમણાં જ સમગ્ર વસ્તુમાં બધું ઉમેર્યું છે. અહીં પણ આપણે ઉમેરતા હોઈએ છીએ, પણ આપણે પાછળ તરફ તરફ ધ્યાન આપીએ છીએ અને કહીએ છીએ કે આટલા બધા ઓપરેશન્સ છે અને આ બધા ઓપરેશનોમાં કુલ સમય લેવામાં આવ્યો છે. અમે વિભાજીત કરીએ છીએ અને કુલ ખર્ચના દરેક ભાગને આપીએ છીએ. તેથી, જો તે દરેક કિસ્સામાં લોગ એમ સમય લેતો નથી, તો પણ આપણે એવું માનતા હોઈ શકીએ કે આ એક આપે છે. તેથી, યુનિયન ઓપરેશનનું અમલીકરણ લો. તેથી, આ પ્રકારની વિશ્લેષણને એમોર્ટાઈઝ્ડ જટિલતા કહેવામાં આવે છે. તેથી, આ તે શબ્દ છે જે વાસ્તવમાં નાણાંકીય બાબતોમાંથી એકાઉન્ટિંગ કરવાથી આવે છે, જ્યાં તમારી પાસે ચોક્કસ ખર્ચ છે જે ઉદાહરણ તરીકે તમારી પાસે વ્યવસાય ચલાવવા માટે હોઈ શકે છે. તમારે કંઈક સેટ કરવું પડશે, . તમે એક ઑફિસ સેટ કરી શકો છો, અને તે પછી હવે જો તમે એમ નહીં કહી શકો કે એક આપવા માટેનો મારો ખર્ચ ઘણો છે, કારણ કે ઑફિસને સુયોજિત કરવા માટે મને ઘણું કામ કરવું પડશે. તેથી, તમારે જે કરવાનું છે તે વિસ્તૃત સાથેનો દિવસ એ તમારા ઑપરેશનના કુલ જીવનકાળને જુએ છે અને કહે છે કે આ કોર્સ સમગ્ર વસ્તુઓમાં વહેંચાયેલું છે. તેથી, તે કામ કરે છે, કે જેને ઋણમુક્તિ કહેવામાં આવે છે, જ્યાં તમે એફની ચોક્કસ કિંમત લે છે, સાધનસામગ્રીના નિયત ભાગ કહેવામાં આવે છે અથવા કાર્ય કરે છે અને તેનો ઉપયોગ કરવામાં આવશે ત્યારે સમગ્ર જીવનકાળમાં વિભાજિત થાય છે. તેથી, તે જ રીતે અમે અમૂર્તિત વિશ્લેષણ કરી રહ્યા છીએ. અમે તમામ એમ યુનિયન ઓપરેશન્સમાં ગણતરી કરીએ છીએ કે અમે કેટલો સમય લઈએ છીએ અને પછી કામ કરીએ છીએ કે દરેક એક લગભગ લોગ સમય લે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:49)

તો, આપણે આ યુનિયનનો ઉપયોગ ડેટા સ્ટ્રક્ચર અને ક્રુસ્કલના અલ્ગોરિધમનો (Kruskal's algorithm) કેવી રીતે શોધી શકીએ? યાદ રાખો કે ક્રુસ્કલના એલ્ગોરિધમ (Kruskal's algorithm) શરૂઆતમાં કિનારીઓને સૉર્ટ કરશે. તેથી, અમારી પાસે તમારી પાસે છે અને ખર્ચની કિંમત અને ચઢવાની માંગ છે. હવે, આપણે નાના ભાગો સાથે પ્રારંભ કરીએ છીએ. તેથી, અમે યુનિટ્સને અમારા શિરોબિંદુઓના સમૂહને શોધી કાઢીએ છીએ. તેથી, દરેક શિરોબિંદુ j એ લેબલ j છે. તેથી, અમારી પાસે બરાબર n પાર્ટીશન છે. એક દરેક શિરોબિંદુ ધરાવે છે અને હવે આપણે જોઈ રહ્યા છીએ તે વર્તમાન ધારને ઉમેરવાની જરૂર છે, જો કે તે કોઈ ચક્ર બનાવતું નથી. આ કહેવાનું સમાન છે કે મોકલો પોઈન્ટ અલગ ઘટકો છે. આ એમ કહેવા જેવું જ છે કે જો હું તમારી પાસે શોધી કાઢું છું અને ધારની લંબાઈ માટે વી શોધવાનું છું, તો સાચું છે કે તમારી શોધ V ના શોધવા માટે સમાન નથી અને જો તમારી શોધ V ની શોધ સમાન નથી, તો મને જરૂર છે મર્જ કરવા માટે. તેથી, મારે આ બે ઘટકોનું જોડાણ કરવાની જરૂર છે. હું તમને શોધી કાઢીને અને v શોધીને તેમના ઘટક નામો મેળવી શકું છું. તેથી, આ સામાન્ય રીતે કેટલાક થે ઘટક છે જે uth ઘટક ધરાવે છે. આ એ મુખ્ય ઘટક છે જે વી છે અને હું એક યુનિયન કરું છું અને મને નવા ઘટક મળ્યા છે જેમાં u અને v બંનેનો સમાવેશ થાય છે.

(સ્લાઈડટાઈમ: 19:57 નો સંદર્ભ લો)

તેથી, કારણ કે તે વૃક્ષ ફક્ત n ઓછા 1 એજ છે , અમે સ્પેનિંગ ટ્રી શોધવા માટે ફક્ત કુલ સેટમાંથી n માઈનસ 1 ધાર ઉમેરીશું. તેથી, તેથી, અમારી પાસે ક્રમમાં ઓ યુનિયન ઓપરેશન્સ બધા અને વી કે છે. હવે, તેથી, કોઈપણ એમ ઓપરેશન્સ માટે જે અમે કહ્યું છે તે એમ લોગ એમ કરશે. તેથી, એમ ઓપરેશન્સ માટે, જો આપણે n લોગ n, તો પાર્ટીશનોને જાળવી રાખીએ અને એમ.એન. લોગ n લેતા હોય તે બધામાં ઉમેરીએ તો અમારું એકંદર અમરકરણ છે. હવે, શરૂઆતમાં આપણે ધારને સૉર્ટ કરવું પડશે. તેથી, તે એમ લોગ એમ લે છે, પરંતુ જ્યારે આપણે ખરેખર ક્રુસ્કલને વિગતવાર વિગતવાર જોતા હતા, કારણ કે એમ મોટાભાગના એન ચોરસમાં છે, આપણે જાણીએ છીએ કે લોગ એમ એ 2 લોગ n છે. તેથી, તીવ્રતા લોગ એમ અને તેના પર લોગ n ના ઓર્ડરના સ્તરે, તેથી આપણે એમ લોગ nm ને એમ લોગ n જેવું જ જોઈ શકીએ છીએ. તેથી, કુલ ખર્ચ હવે એમ લોગ એમ થાય છે, માફ કરશો એમએમ લોગ એન વત્તા એન લોગ n. તેથી, આપણને એમ પ્લસ એન લોગ n મળે છે. જો તમે પ્રિમની એલ્ગોરિધમ માટે દાવો કરેલું જટિલતા અને તે પણ છે, વાસ્તવમાં આ એલ્ગોરિધમ્સ માટે, જે ઢગલાઓનો ઉપયોગ કરીને પ્રાથમિકમાં સમાન હોય છે, તો અમે દાવો કરીએ છીએ કે જે લોકો તે પ્રકારના એમ પ્લસ એન છે, જો આપણે ઢગલો કરવા માટે ઢગલોનો ઉપયોગ કરીએ ન્યૂનતમ અંતર ગણતરી. તેથી, તેથી, ક્રુસ્કલના એલ્ગોરિધમને(Kruskal's algorithm) યુનિયન સાથે ડેટા સ્ટ્રક્ચર મળી આવે છે, જે પ્રાથમિક ઢગલાની જેમ જ એલિગોરિધમની સમાન જટિલતા ધરાવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 21:28)

તેથી, આપણે જે જોયું છે તે સારાંશ આપવા માટે આપણે એરેના ઘટકો અને એરે ના લિસ્ટની મદદથી દરેક ઘટકોમાં શિર્ષકોને નામ આપવા માટે સૂચિનો એરે અમલમાં મૂકી શકીએ છીએ, અને અન્ય એરે કદના ટ્રૅક રાખવા માટે દરેક ઘટક. આ સાથે યુનિયનને નિષ્ક્રીય વ્યક્તિગત તત્વ પાર્ટિશન્સના ડેટા માળખાને શોધવાનું પ્રારંભિક પગલું છે ઓર્ડર n શોધવા

સતત સમય લે છે. એમ ઓપરેશન્સની શ્રેણી માટે જટિલતાને અમલીકરણ કરો એમ લોગ n છે. તેથી, અમે દરેક યુનિયન ઓપરેશન પ્રારંભિકથી શરૂ થયેલા સર્ચ એમ કામગીરીના અનુક્રમ માટે લોગ એમ સમય લઈ રહ્યા છીએ તે વિચારી શકીએ છીએ.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 02
લેક્ચર - 33
યુનિયન - પોઈંટર્સનો ઉપયોગ કરીને ડેટા સ્ટ્રક્ચર શોધો.

છેલ્લા લેક્ચરમાં, અમે સંગઠનને ડેટા સ્ટ્રક્ચર શોધવાનું એરે આધારિત અમલીકરણ જોયું, જેમાં શોધ કામગીરી સતત સમય અને અમે એવો દાવો કરવા માટે અમલીકૃત વિશ્લેષણનો ઉપયોગ કરીએ છીએ કે એમ યુનિયનોના ક્રમમાં યુનિયન ઓપરેશન એમ લોગ એમ સમય લેશે. તેથી, અસરકારક રીતે દરેક યુનિયન એક લઘુગણક સમય કામગીરી હતો, સમગ્ર ક્રમ પર અમૃતિત. હવે, આપણે વધુ વ્યવહારિક અમલીકરણ જોશો જે વધુ સારી જટિલતા ધરાવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:29)

તેથી, યાદ રાખો કે યુનિયન ડેટા સ્ટ્રક્ચરને સેટ સેટના ભાગનું ટ્રૅક રાખે છે અને ત્રણ ઓપરેશન્સને સપોર્ટ કરે છે. એક મેક યુનિયન છે જે એક નાનું પાર્ટીશન બનાવે છે, જ્યાં દરેક તત્વો તેનાથી સંબંધિત હોય છે, દરેક તત્વ એક જ શબ્દના પાર્ટિશનમાં હોય છે. તેથી, આપણે પાર્ટીશનો અને ઘટકો માટે સમાન નામોનો ઉપયોગ કરીએ છીએ, તેથી દરેક S એ એક નાના નાના એસ તરીકે ઓળખાતા પાર્ટીશનમાં છે. શોધો અમને કયું પાર્ટીશન આપેલ ઘટક સાથે સંકળાયેલ છે અને યુનિયન એ બે ઘટકો અથવા બે પાર્ટીશનોને એકમાં જોડે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:01)

તેથી, એરે આધારિત અમલીકરણ એ ઘટક તરીકે ઓળખાતી એક એરેનો ઉપયોગ કરે છે જે અમને જણાવવા માટે છે કે દરેક ઘટક કયા ઘટક સાથે સંકળાયેલ છે, તે અમને સૂચવવા માટે સૂચિની સૂચિ રાખે છે કે કયા ઘટકો દરેક ઘટકોથી સંબંધિત છે. તેથી, તે એક પ્રકારનું રિવર્સ મૅપિંગ છે, દરેક ઘટક માટે કે તે આપણને જણાવે છે કે સમૂહના કયા સભ્યો તે છે. અને આખરે, તે દરેક ઘટકના કદને ટ્રૅક રાખે છે, કારણ કે અમે ઘટકોને એકવાર નાના લેતા અને એકવાર મોટા નામે ફરીથી લેબલ કરીને મર્જ કરીએ છીએ. તેથી, યુનિયનને શોધી કાઢો ઓર્ડર એન ઓપરેશન, ઓર્ડર એક ઑપરેશન હતું અને એમ ઓપરેશન્સના અનુક્રમમાં યુનિયન ઓપરેશનની એમ્ટોર્ટાઈઝ્ડ જટિલતા (amortized complexity) લોગ એમ હતી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:37)

તો, હવે આપણી પાસે અલગ પ્રતિનિધિત્વ હશે, આપણે પોઈન્ટર સાથે નોડ નો ઉપયોગ કરીશું. તેથી, સમૂહના દરેક તત્વને હવે નોડ તરીકે રજૂ કરવામાં આવશે, નોડમાં બે ભાગ હશે, તે નામ હશે જે તે તત્વ છે જેનો તમે ટ્રૅક રાખી રહ્યાં છો અને લેબલ ભાગ જે તેના ઘટકને ટ્રૅક રાખે છે, તેથી તે ઘટક માટે એક નિર્દેશક છે. હવે, યાદ રાખો કે ઘટકોના નામો એ તત્વોના નામો છે. તો, જો આપણી પાસે અહીં નામ છે, તો આપણી પાસે નામ j છે, તે કહે છે કે આ નોડ તત્વ j અને તેનું લેબલ પોઈન્ટ્સને રજૂ કરે છે, તેથી તે કહે છે કે j એ ઘટક j થી સંબંધિત છે. અહીં બીજી તરફ, આપણી પાસે તત્વ છે અને આ

દ્વારા આપણે પ્રથમ નોડ પર જઈએ છીએ જે પોતાને નિર્દેશ કરે છે અને આ સેટ એ છે કે Ai ઘટક જે છે. તેથી, હું ઘટક જેનો છું, તેથી આપણે એક પ્રકારનું વાંચ્યું છે કે આપણે આ હકીકતનો ભારે ઉપયોગ કરી રહ્યા છીએ કે તત્વોના નામ અને ઘટકોના નામો સમાન છે. તો, આપણે કેટલાક સંદર્ભમાં છીએ, આપણે નામનું નામ તત્વના નામ તરીકે અર્થઘટન કરી રહ્યા છીએ અને જ્યારે તમે પોઈન્ટરને અનુસરી રહ્યા છો, ત્યારે આપણે નામનું નામ ઘટકના નામ તરીકે અર્થઘટન કરી રહ્યા છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:55)

તેથી, યુનિયનને શોધવાનું પ્રારંભિક છે, તેથી તે શું કરે છે, તે દરેક ઘટકને એક જ શબ્દ તરીકે સેટ કરશે જે સમાન નામના ઘટક ધરાવે છે. તેથી, હું ઘટકોથી સંબંધિત છું, તેથી એક નોડ છે જે પોતે પોઈન્ટ કરે છે, પછી એક નોડ કે જે પોતે પોઈન્ટ કરે છે તે સૂચવે છે કે આ K ને સમાવે છે જે ઘટક k છે, આ ઘટકો j જે સમાવિષ્ટ j છે. તેથી, આપણી પાસે આવા ઘટકો n થી n છે, તેથી આપણી પાસે આવાં નોડ્સ છે જેમાં પ્રત્યેક ઘટકનું નામ પ્રથમ ઘટકમાં છે અને પોઈન્ટર પોતાને બીજા ઘટકમાં શામેલ કરે છે જે પોતે એક શબ્દ ઘટક છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 03:31)

તેથી, થોડા સમય પછી આપણે જોશું કે જ્યારે આપણે નોડને મર્જ કરીએ છીએ, ત્યારે આ પોઈન્ટર પોતાને બદલે પોઈન્ટ કરશે, તે અન્ય વસ્તુઓ તરફ નિર્દેશ કરશે. તેથી, અહીં ઉદાહરણ તરીકે, આઈ, જે, કે અને એલ સમાવિષ્ટ હોઈ શકે છે અને માળખાં કહે છે કે ઘટકનું નામ વાસ્તવમાં જે છે. તેથી, આ ઘટક j છે અને તેમાં i, j, k, l શામેલ છે, તેથી તમે જોઈ શકો છો કે તત્વ મારી પાસે પોઈન્ટર નથી, પરંતુ અન્ય ગાંઠો છે. તેથી, જો તમે આને અનુસરો છો તો આપણે શોધી કાઢીએ છીએ કે તે તત્વ તરફ ધ્યાન કેન્દ્રિત કરે છે જે પોતે પોઈન્ટ્સ કરે છે, તેથી તે છે જેનો ભાગ છે જે I છે. એ જ રીતે, જો આપણે l થી શરૂ કરીએ, l થી i તરફ, i થી j અને so on, તો આપણે જોશું કે આનો આપણે આ કેવી રીતે ઉપયોગ કરીશું તે એક વ્યૂહરચના છે, આપણે તેનો અમલ કરવા માટે ઉપયોગ કરીએ છીએ. આપણે નોડ સાથે પ્રારંભ કરવો પડશે અને પાથ ઉપર જવું પડશે અને આ એક વૃક્ષ હશે, તેથી દરેક પાથ રૂટને સમાપ્ત કરશે અને રુટ ઘટકનું નામ હશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:25)

તો, આ વૃક્ષને નેવિગેટ કરવામાં થોડી મદદ કરવા માટે, ચાલો ધારીએ કે આપણી પાસે નીચે આપેલા વધારાના એરે પોઇંટર્સ અને નંબર્સ છે. તેથી, આપણે દરેક નોડ જ્યાં છે તે સૌ પ્રથમ ટ્રેક રાખશું. તો, આપણી પાસે અહીં નોડ તરીકે ઓળખાતી એક એરે છે અને એરેના દરેક તત્વ તે તત્વ ધરાવતા નોડ પર નિર્દેશક છે. તો, મે નોંધ્યું છે કે નોડ i અહીં છું, એન્ટ્રી જે કહે છે કે નોડ જે અહીં છે અને આગળ. તેથી, અનુરૂપ ત્યાં એન્ટ્રી હશે જે કહે છે કે નોડ કે અહીં છે અને નોડ l અહીં છે. તેથી, આપણા વાસ્તવિક સમૂહમાં દરેક ઘટક પાસે આ નોડ એરેથી વૃક્ષના વાસ્તવિક નોડમાં પોઈન્ટર હશે, જ્યારે વૃક્ષનો સમૂહ ઘટકોનો સમૂહ છે, તે વાસ્તવિક નોડ જ્યાં તે મૂલ્ય રહે છે, પછી અમને તે જાણવાની જરૂર છે કે ઘટકો જે હાલમાં સક્રિય છે. તેથી, અહીં આ ચોક્કસ સેગમેન્ટ i, j, k, l જે j માટે ઘટક છે, ત્યાં હું ઘટક બનવા માટે ઉપયોગ કરું છું, જ્યારે હું ઘટક k તરીકે ઉપયોગ શરૂ કરતો હતો, ત્યારે ઘટક l નો ઉપયોગ થતો હતો. પરંતુ, આ નામો હવે ઉપયોગમાં નથી આવ્યાં, કારણ કે આ ઘટકોને સીધા જ જી અથવા સીધું જ મર્જ કરવામાં આવ્યું છે. તેથી, હું, જે, કે, એલ વચ્ચે સાચું જાણું છું કે ફક્ત જ

રહે છે, તેથી આપણે કહીએ છીએ કે j નું મૂળ છે, ઘટક j એ j નો સમાવેશ કરેલો નોડ છે અને અન્ય બધા લેબલો i, j, k અને l માટે, આપણે કહીએ છીએ કે હું, કે, એન, એલ ત્યાં કોઈ ઘટક નથી જેનો અર્થ છે કે ત્યાં કોઈ ઘટક નથી, તેથી આપણું ખાલી ખાલી છે, તે જ રીતે આ વસ્તુઓ માટે કોઈ રુટ નથી. તેથી, જો તમે આ ઘટકના રુટને શોધવા માંગો છો, તો નોડ I પર જાઓ અને પછી તમારી રીત ઉપર કાર્ય કરો. છેવટે, આપણે કદ દ્વારા મર્જ કરવાના સંદર્ભમાં પહેલાની જેમ સમાન વ્યૂહરચનાનો ઉપયોગ કરીશું, તેથી અમે ફક્ત દરેક ઘટકના કદને સ્પષ્ટપણે ટ્રૅક કરીશું. તેથી, અહીં એવું કહેવામાં આવે છે કે આ ઘટકોનું કદ i, j, k, l એ 4 છે કારણ કે હાલમાં 4 તત્વો છે. તો, આપણી પાસે આ નોડ પોઈન્ટર છે જે સેટમાં દરેક તત્વને નિર્દેશ કરે છે જ્યાં તે હાલમાં આવેલું છે, જેમાં નોડ તે તત્વ ધરાવે છે. અમારી પાસે રુટ પોઈન્ટર છે જે આપણને દરેક ભાગ માટે કહે છે જે હાલમાં આપણા પાર્ટીશનમાં સક્રિય છે, જે તે પાર્ટીશનનો રુટ છે અને પછી તે કદ છે જે આપણને દરેક ઘટકનું માપ કહે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:29)

તેથી, જ્યારે આપણે યુનિયન કરીએ છીએ ત્યારે મૂળભૂત રીતે આપણે નાનામાંથી એકને મોટામાં મર્જ કરીએ છીએ. તેથી, અહીં આપણી પાસે બે ઘટકો છે, આપણી પાસે ઘટક j છે જે j, k અને m છે અને આપણી પાસે ઘટક છે જે i અને l છે. તેથી, હવે આપણે i અને j નું જોડાણ કરવું છે, તેથી હવે આપણે પહેલા ક્યાંક નોંધ કરીશું કે હું 2 નું કદ છે અને જે 3 નું માપ 3 છે. તેથી, આને રેકોર્ડ કર્યા પછી, આપણે હવે હું જે માં મર્જ. તેથી, આનો અર્થ શું છે કે આ ધાર જે કહે છે કે હું સ્ટ્રિંગનો રુટ નાશ કરું છું અને તેના બદલે આપણે આને એક લિંક મુકવી પડશે અને આ બે વૃક્ષો અને એક મોટા વૃક્ષને મર્જ કરીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:26)

તેથી, જો આપણે તે કરીએ તો આ પરિણામી વૃક્ષ છે, હવે આપણી પાસે j એક જ ઘટક છે જે આપણે j ના કદને, i કદ અને પ્લસના કદના કદ માટે અપડેટ કરી હોત. જે. તેથી, હવે તે 5 કહેશે3 ની જગ્યાએ અને નોંધ લો કે આ સંપૂર્ણ ઑપરેશન ઑર્ડર 1 ઑપરેશન છે, કારણ કે આપણે બે ઘટકોની મૂળ શોધવા માટે બે ઘટકોના કદ શોધવા માટે એક નિશ્ચિત સંખ્યામાં પગલા લઈએ છીએ અને નક્કી કરીએ છીએ કે કયા રુટને મસાજ કરવા માટે અન્ય રુટ . તેથી, હું j પર નિર્દેશ કરું છું કે હું જે તરફ ધ્યાન આપું છું, તેથી આ એક ઑર્ડર 1 ઑપરેશન છે અને તે એટલા માટે છે કારણ કે કદ કદની માહિતી અમને કદ જણાવે છે, તે કદ શોધવા માટે ઘટકમાંથી પસાર થતી નથી. અમારી પાસે આ રુટ માહિતી છે જે સીધી અમને જણાવે છે કે બે પાર્ટીશનની મૂળ ક્યાં છે. તેથી, આપણે તે સ્થળ પર સીધા જ જવું પડશે અને એક પોઈન્ટ બીજા અથવા તેનાથી વિરુદ્ધ બનાવવું પડશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:09)

હવે, બીજી તરફ જોશો જેમ આપણે જોયું છે કે આપણે મૂલ્યમાંથી પ્રારંભ કરવાની જરૂર છે જે આપણે તપાસવા માંગીએ છીએ અને તેમને દૂર કરવા માંગીએ છીએ. તેથી, અમને લાગે છે કે અમે તમને શોધી કાઢીએ, પછી યાદ રાખો કે આ નોડ એરે હશે અને તેથી તમારો નોડ કહેશે કે તમારો નોડ અહીં છે. તેથી, આપણે ત્યાં શરૂ કરીશું અને પછી આપણે આ પાથને અનુસરીશું, તેથી આપણે આ મુદ્દાઓ અહીં કહીશું અને અહીં આ મુદ્દાઓ અને આ મુદ્દાઓ અહીં છે અને છેવટે, જ્યારે

આપણે કોઈ નોડ સુધી પહોંચીએ છીએ જે નામ બતાવે છે. તેથી, પછી આપણે કહીશું કે, તમને તે સમાન છે, તેથી આ માર્ગને અનુરૂપ સમય લાગે છે જે આપણે નોડથી રુટ સુધી પસાર કરીએ છીએ. હવે, યાદ રાખો કે શરૂઆતમાં અમારી પાસે તમારા માટે એક રસ્તો હતો જે આના જેવો દેખાય છે, તેથી પ્રારંભમાં દરેક નોડ પાસે લંબાઈ 1 નો માર્ગ મોકલે છે જે તે જ રુટ છે. હવે, પાથ પરિવર્તન કેમ કરે છે, તે યુનિયન ઓપરેશનને કારણે, દરેક વખતે જ્યારે આપણે એક યુનિયન 1 નોડ જે પોઈન્ટ કરવા માટે ઉપયોગ કરે છે, તે હવે નિર્દેશ કરે છે, તે હવે બીજા નોડ તરફ નિર્દેશ કરે છે અને તેના નીચે બધું જ છે કારણ કે તે પાથ વધે છે. તેથી, તેથી, દર વખતે જ્યારે j j ના લેબલમાંથી j ની શોધમાંથી જવાનું હોય ત્યારે દર વખતે પાથ વધે છે, જ્યારે દરેક વખતે જે બદલાશે તે લેબલમાં કેટલો સમય લાગે છે તે લેબલ છે. પરંતુ, પહેલાં આપણે ઘટક કદને બમણું કરી રહ્યા છીએ, આપણે કયા સમયે મર્જ કરીએ છીએ. તેથી, આપણે કયા સમયે યુનિયન કરીએ છીએ, પછી j સમાવતી લેબલમાં પહેલાની જેમ ઘણા ઘટકો ક્વિઝ શામેલ છે. હવે છેલ્લે, તમે તમારા ગ્રાફમાંના તમામ ગાંઠો એક ઘટકમાં એકીકૃત કરવા માટે કરી શકો છો, મહત્તમ ઘટક કદ n છે. તો, તેથી, આપણે કદ 1 ના ઘટકથી પ્રારંભ કરીએ છીએ, પછી આપણે તેને 2 થી બમણું કરીએ છીએ અને તે 4 થી બમણું કરીએ છીએ અને છેવટે આપણે આગળ વધતા નથી. તેથી, આ સંપૂર્ણ વસ્તુ મોટાભાગના લોગ n પગલાંઓ પર હોઈ શકે છે, તેથી, j ના લેબલને બદલી શકો છો, જ્યારે લેબલ બદલાય ત્યારે દર વખતે, કારણ કે આપણે બીજા ઘટકની રુટને અહીં જોડીએ છીએ, પાથ લિંક 1 થી વધી રહી છે. તેથી, પાથની લંબાઈ 1 વત્તા 1 વત્તા 1 લોગ n ગુણ વધે છે, તેથી પાથ મોટા ભાગના લોગ n છે. તેથી, આ વિશ્લેષણ સાથે આપણે તે જોઈ શકીએ છીએ, કારણ કે આપણે નાના ઘટકોને મર્જ કરી રહ્યા છીએ અને મોટા ઘટકો શોધવા માટે લોગ એન ટાઈમ લેશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:29)

તેથી, આપણે એવા સ્થાને છીએ જ્યાં પહેલાનાં ડેટા માળખામાં આપણે ડ્યુઅલ છે, પહેલાનાં ડેટા માળખામાં એરે સાથે. તેથી, અમને સતત સમયનું ઓપરેશન મળ્યું હતું અને અમારે એલોર્ટાઈઝ્ડ વિશ્લેષણ કર્યું હતું કે જે યુનિયન લોગ એન હતું, હવે આપણી પાસે આ પોઈન્ટર બેઝ વિચારી છે કે યુનિયન ઓર્ડર 1 છે, પરંતુ ઓર્ડર લોગ n છે. તેથી, તેનો ઉપયોગ એ છે કે આપણે આવશ્યક રીતે સમાન પ્રકારની માળખું મેળવ્યું છે, તે નિર્ધારિત કરે છે કે અમે બે ઓપરેશન્સની ફરિયાદને ઉલટાવીએ છીએ. હવે, તે તારણ આપે છે કે આપણે વધુ ક્લિવર કંઈક કરી શકીએ છીએ અને વાસ્તવમાં તે શોધવા માટે વધુ કાર્યક્ષમ બનાવે છે. તેથી, અમે એક ક્રમમાં યુનિયન માટે, ક્રમમાં રાખી શકીએ છીએ, પરંતુ ઓર્ડર લોગ ખરેખર ઘટાડી શકાય છે. તેથી, ચાવી એ છે કે એક વાર આપણે આ પાથને u થી j માં ફેરવી દઈએ, આપણે જાણીએ છીએ કે તે એક એવી સમજણ છે કે ફક્ત તમે જ નથી જાણતા કે જે તમને સમાવતા ઘટકવાળા પાથ જે છે, આપણે જાણીએ છીએ કે l સમાવિષ્ટ છે જે તમે જાણો છો હું સમાવિષ્ટ ઘટક જે છે. તેથી, અમે આ માહિતીનો ઉપયોગ કેમ કરતા નથી અને જ્યારે પણ અમે પાછા આવો ત્યારે ફરીથી વિવાદિત થતા નથી. તેથી, આપણે શું કરીશું, આપણે સામાન્ય શોધ જ કરીશું, તેથી આપણે પાથને આગળ વધારીશું અને પછી આપણે j થી root તરફ પાછા જઈશું અને હવે કહીશું કે આ j છે. તેથી, મને બાયપાસ કરવું સંપૂર્ણ છે જે સીધા જ j તરફ નિર્દેશ કરે છે, તેથી મેં પહેલાની વસ્તુઓને દૂર કરી દીધી છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:02)

તેથી, હું આ ધારને દૂર કરીશ અને સીધા જ જઈશ, ત્યાં જતાં પહેલાં હું ત્યાં જવાનું દૂર કરીશ. તેથી, હું કહું છું કે હું આ ધારને સીધી દૂર કરી શકું છું, તેથી હું દરેક પગલું માટે, જ્યાં સુધી હું છેલ્લે સુધી રુટની નીચે સુધી પહોંચું ત્યાં સુધી દૂર કરી શકું છું. તેથી, મૂળરૂપે હવે હું આ પાથ સાથે એક સપાટ વૃક્ષ ધરાવીશ, તેનો ઉપયોગ આ માર્ગ જેવો છે તે આના જેવું દેખાય છે. તેથી, દરેક વિચારની પાસે હવે પાથની નિર્દેશિત ઍક્સેસ છે, તેથી અમે વૃક્ષને સપાટ બનાવી દીધી છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:26)

તેથી, જો તમે ચિત્રશૈલીને જોવા માંગો છો, તો હું આનાથી પ્રારંભ કરું છું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:36)

અને પછી જો હું હાલની પરિસ્થિતિને જોઉં, તો હું આ ધારને તમારી સામે l થી સીધા જ એક તરફ થી સીધા બદલીશ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:42)

ત્યારબાદ, હું l થી i થી સીધા જ l થી j સુધી ધારને બદલીશ અને તેથી હવે હું આ ફ્લેટ ટ્રીને સમાપ્ત કરીશ. તેથી, હવે પછીથી જુઓ, તેથી હું તમને એક અને એક પગલું લઈશ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:52)

તેથી, પ્રથમ શોધ લોગ એન ટાઈમ લે છે, પછી તે સતત સમય લે છે, તેથી હવે તમે ખરેખર બતાવી શકો છો કે મારો અર્થ એ છે કે આ એક સરળ ગણતરી નથી. તેથી, આપણે હમણાં પ્રયાસ કરવા અને હાજરી આપવા જઈ રહ્યા નથી, પરંતુ આ પછી તમે બતાવી શકો છો કે જેમ આપણે અગાઉના લોગ n, m log n, cost m log m મેળવવા માટે પહેલાનાં એરે આધારિત અમલીકરણ માટે અમૂર્તિત વિશ્લેષણ કરી શકીએ છીએ. એમ યુનિયન કામગીરી માટે ખર્ચ. અહીં, તમે વાસ્તવમાં બતાવી શકો છો કે કુલ n શોધ ખરેખર n માં લગભગ રેખીય કંઈક લેશે. N માં લગભગ રેખીય ખરેખર n ના આલ્ફા તરીકે ઓળખાતા ફંક્શનનો n વખત છે અને n નો આ આલ્ફા એ કંઈક છે જે એર્કમેન (Ackermann) ફંક્શનની વિરુદ્ધ છે, એર્કમેન (Ackermann) ફંક્શન કંઈક છે જે ખૂબ ઝડપથી વધે છે. તેથી, તે એક ફંકશન છે જે કોઈપણ સમજૂતી કરતા ઝડપથી વધે છે, હવે આ વ્યસ્ત એર્કમેન (Ackermann)ના લોગ જેવું છે, મારો અર્થ એર્કમેન (Ackermann)નો લોગ છે. તેથી, તમે જાણો છો કે ઘાતાંકીય કાર્ય માટે, વ્યસ્ત એ લોગ છે. તેથી, લોગ n એ આપણા માટે સરસ છે, તે 1000 નું ધીમી લોગ 10 છે, 1 અબજનું લોગ 20 છે, 1 મિલિયન 20 છે, 1 અબજનું લોગ 30 છે, તે ખૂબ ધીમું હતું, કારણ કે અન્ય દિશામાં ઝડપથી વધતા 2 ની શક્તિ 10, 2 ની શક્તિ 20, 2 ની 30 ઘાત ખૂબ ઝડપથી જાય છે. તેથી, સમાન રીતે એર્કમેન (Ackermann) ખૂબ ઝડપથી જાય છે, તેથી જો તમે અનુરૂપ લોગ ફંક્શન શું લો છો, તો એર્કમેન (Ackermann) ઈનપુટ્સ તે કેટલી ધીરે વધે છે તે માટે. હકીકતમાં, એવો દાવો છે કે n ના કોઈપણ મૂલ્ય માટે n નો આલ્ફા એ 4 છે જે તમને વ્યવહારુ કમ્પ્યુટિંગ સમસ્યામાં આવી શકે છે જેને તમે હલ કરી શકો છો. તેથી, તમે વિચારી શકો છો કે આ કંઈક ઓર્ડર જેવી છે, કારણ કે આપણે જાણીએ છીએ કે n નો આલ્ફા 4 થી ઓછા છે. તેથી, આ 4 n ની જેમ છે, તેથી પાથ કમ્પ્રેશન એ n લોગ n થી નબળી ઘટાડો તરીકે આપે છે લીનિયર કંઈક જે પાથ કમ્પ્રેશન વગર મળશે. તેથી, હવે, આ

નિર્દેશક આધાર પર વિચાર કરીને, આપણે વાસ્તવમાં વધુ સીધા સીરિ આધારિત અમલીકરણ સાથે નોંધપાત્ર બચત તુલના પ્રાપ્ત કરી છે જે છેલ્લા ભાષણમાં વર્તે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:48)

તેથી, જો આપણે પોઈન્ટર સાથે ગાંઠોનો ઉપયોગ કરીને યુનિયનને અમલમાં મૂકવા માટે અમલમાં મૂકતા હોય તો સારાંશ આપવા માટે, આપણે અલબત્ત, મેક યુનિયનની શોધ અને રેખીય સમયનો પ્રારંભિક સેટ કરી શકીએ છીએ, હવે યુનિયન ઑર્ડર 1 ઑપરેશન બની જાય છે. કારણ કે, આપણે ફક્ત બે મૂળને સીધા જ ઍક્સેસ કરીએ છીએ અને આ નાના રુટને મોટા રુટ સાથે મર્જ કરીએ છીએ અને હવે જો તમે આ પાથ કમ્પ્રેશન યુક્તિનો ઉપયોગ કરો છો, તો પણ સિદ્ધાંત તે એન ઓપરેશન્સ માટે n લોગ n છે, તે n ઓપરેશનલ માટે n આલ્ફા n છે. અમે પાથ કમ્પ્રેશનનો ઉપયોગ કરીએ છીએ.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 05
મોડ્યુલ - 03
લેક્ચર - 34
પ્રાયોરિટી ક્યુ(Priority queues)

તેથી, આપણે જોયું છે કે હોંશિયાર ડેટા માળખુંનો ઉપયોગ કેવી રીતે કરે છે, યુનિયન ડેટા માળખું શોધે છે તે ક્રુસ્કલના અલ્ગોરિધમ(Kruskal’s algorithm)ને વધુ કાર્યક્ષમ બનાવે છે. અન્ય બે એલ્ગોરિધમ્સ કે જેને આપણે સિંગલ સ્ત્રોત ટૂંકા પાથ માટે કાર્યક્ષમ અથવા ડિજેસ્ટ્રાના(Dijkstra's) એલ્ગોરિધમ બનાવવાની જરૂર છે, અને ન્યૂનતમ પાર્સ સ્પેનિંગ ટ્રી માટે પ્રાઈમના અલ્ગોરિધમનો સમાવેશ કરીએ છીએ. તે બંનેમાંથી તે એક ડેટા માળખું જરૂરી બને છે જેને સામાન્ય રીતે પ્રાયોરિટી ક્યુ(Priority queues) કહેવામાં આવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:23)

તેથી, પ્રાયોરિટી ક્યુ(Priority queues) સમજવા માટે, ચાલો નીચે આપેલ દૃશ્યને જોઈએ. તેથી, અમને લાગે છે કે અમારી પાસે ઑપરેટિંગ સિસ્ટમ તરીકે ચાલતી નોકરી શેડ્યૂલર હશે. તેથી, જ્યારે આપણે ઑપરેટિંગ સિસ્ટમ પર બહુવિધ કાર્યો ચલાવીએ છીએ ત્યારે તેમાંના દરેક થોડો સમય માટે ચાલે છે અને પછી તે નાપસંદ થાય છે. તેથી, જોબ શેડ્યૂલર તેમની બધી પ્રાથમિકતાઓ સાથે બાકી રહેલી બધી નોકરીઓની સૂચિને જાળવી રાખે છે. જ્યારે પણ પ્રોસેસર મુક્ત હોય, ત્યારે સુનિશ્ચિતકર્તા નોકરીમાં મહત્તમ પસંદગી સાથેની નોકરી પસંદ કરે છે અને તેને સુનિશ્ચિત કરે છે. હવે શું થાય છે? આ ગતિશીલ છે. તેથી, જ્યારે કેટલીક નોકરીઓ ચાલી રહી છે, અન્ય નવી નોકરીઓ આવી શકે છે. તેથી, દર વખતે જ્યારે નવી નોકરી આવે ત્યારે તે વિવિધ પ્રાથમિકતાઓ સાથે આવે છે, અને તે સુનિશ્ચિત કરવા માટે શેડ્યૂલરનું કામ છે કે ઉચ્ચ પ્રાધાન્યતા સાથેની નવી નોકરીઓ ઓછી નોકરી સાથે જૂની નોકરીથી આગળ આવે. તેથી, ડેટા માળખું દ્રષ્ટિકોણથી, અમારું પ્રશ્ન એ છે કે સુનિશ્ચિતકર્તા માટે બાકીની નોકરીઓની સૂચિને તેમની પ્રાથમિકતાઓ સાથે જાળવી રાખવા માટેનો શ્રેષ્ઠ રસ્તો શું છે? તેથી, આ સૂચિ ગતિશીલ રીતે અપડેટ કરી શકાય છે કારણ કે આ સૂચિ અપડેટ થઈ જાય છે

((સમયનોસંદર્ભ લો: 01:22))

નવી નોકરીઓ આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:26)

તો, આ એ છે જે આપણે અગ્રતા કતાર કહીએ છીએ? તેથી, અમારી પ્રાથમિકતા કતારમાં બે મૂળભૂત કામગીરી છે; પહેલું કામ એ આગળની નોકરીને કાઢવાનો છે જે આ કિસ્સામાં નોકરીને ઉચ્ચતમ પ્રાધાન્યતા સાથે લે છે અને તેને કતારમાંથી દૂર કરે છે. હવે અલબત્ત, અમારી પાસે એવી સ્થિતિ છે જ્યાં તમારી સમાન પ્રાધાન્યતામાં બહુવિધ નોકરીઓ છે. તેથી,

અમે તમારી પ્રાથમિકતાઓને ધ્યાનમાં રાખતા નથી, પરંતુ જો તમને જરૂર નથી, તો અમે ક્યાં તો લખી શકીએ છીએ કે ત્યાં કોઈ પ્રકારનું લેખન નિયમ છે અથવા અમે કાળજી રાખીએ છીએ કે કોઈ ઉચ્ચ પ્રાધાન્યતા નોકરીમાંથી કોઈ એક યોગ્ય છે. તેથી, મુખ્ય કામગીરી મહત્તમ કાઢી નાંખે છે; કતારમાંથી મહત્તમ પ્રાધાન્યતા આઈટમ કાઢી નાખો. અને પછી એક શામેલ છે. તેથી, નોકરી આવશે તે દાખલ કરવામાં આવશે, તે અગ્રતા સાથે આવશે. તેથી, આપણે સૂચિમાં શામેલ કરવાની જરૂર છે. તેથી, તે પછીથી કાઢી નાખો મહત્તમ આને એક ગણાશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:09)

તેથી, સૌ પ્રથમ ઉકેલ જે કોઈ ડેટા રાખવા માટે વિચારી શકે છે તે સૂચિને જાળવી રાખવા, કોઈ રેખાકીય માળખું અને એરે અથવા સૂચિ જાળવવાનું છે, કારણ કે તે ગતિશીલ રીતે સૂચિબદ્ધ છે. તમારી સ્પષ્ટ પસંદગી. હવે જો આપણે કોઈ ક્રમાંકિત સૂચિ રાખીશું, તો નોકરી ઉમેરવાનું સરળ છે, જે સૂચિમાં ઉમેરાય છે. તેથી, નિવેશ એ સતત સમય ઓપરેશન સમય (O)લે છે, પરંતુ જો આપણે કોઈ ક્રમમાં ગોઠવેલ સૂચિ હોય તો ઘણીવાર જોયેલી હોય, તો અમને ન્યૂનતમ અથવા મહત્તમ શોધવા માટે સમગ્ર સૂચિને સ્કેન કરવી પડશે. તેથી, આ કિસ્સામાં જો અમે કોઈ નૉર્ટસ્ટેડ સૂચિમાંથી કાઢી નાંખવાનું મહત્તમ કરવા માંગીએ છીએ, તો આ ઑપરેશન સૂચિના કદના પ્રમાણમાં રેખાંશ સમય અથવા આ ક્ષણે બાકીની નોકરીઓની સંખ્યા જેટલું લેશે. તેથી, સૂચિ સાથે કોઈ એક જ ઉપયોગી વસ્તુ કરી શકે છે, નહીં તો તેને સૉર્ટ કરવું જોઈએ. તેથી, જો આપણે સૉર્ટ કરીએ તો અમે પ્રાધાન્યતાના ઘટાડવાની હુકમમાં નોકરીઓનો પ્રયાસ અને જાળવણી કરી શકીએ છીએ, જો આપણે તેને પ્રાધાન્યતાના હુકમ ઘટાડે છે, તો મહત્તમ પ્રાધાન્યતા કાર્ય હંમેશા સૂચિની શરૂઆતમાં છે. તેથી, આપણે તેને સતત સમયસર તરત શોધી શકીએ છીએ. તેથી, મહત્તમ કાઢી નાખો હવે એક સતત સમય ઓપરેશન બને છે, પછી દાખલ કરવા શું થાય છે? આપણે સૂચિમાં મૂલ્ય શામેલ કરવાની જરૂર છે, અને જ્યારે અમે સૉર્ટ કરેલ સૂચિમાં મૂલ્ય શામેલ કરીએ છીએ ત્યારે અમે તેને દાખલ કરવાની સૉર્ટને સમાધાન કરવાની જરૂર છે, તેને સૂચિબદ્ધ કરવા માટે સૂચિ પર ચાલવું પડશે. તેથી, તે હવે લેશે ઓ સમય એન ઓર્ડર. તેથી, કોઈ નૉર્ટસ્ટેડ સૂચિ અથવા સૉર્ટ કરેલી સૂચિમાં, અમારી પાસે એક અનિર્ણીકૃત સૂચિમાં એક ટ્રેડ બંધ છે, મહત્તમ રેખાંશ સમય કાઢી નાખે છે, સૉર્ટ કરેલી સૂચિ શામેલમાં લીનિયર સમય લે છે. તેથી, એક બીજું સમસ્યા છે. તેથી, જો આપણે એન નોકરીઓના અનુક્રમમાં કોઈ ક્રમમાં ગોઠવેલ સૂચિ અથવા સૉર્ટ કરેલી સૂચિનો ઉપયોગ કરીએ છીએ, તો એમ લાગે છે કે સિસ્ટમમાં N નોકરીઓ આવે છે તે તેમને પ્રક્રિયા કરશે. પછી આપણે તેમને N વખત દાખલ કરીશું અને મહત્તમ N વખત કાઢી નાખીશું. તેથી, આ બંને વચ્ચે આપણે કયા સોલ્યુશનનો ઉપયોગ કરીએ છીએ, આપણે એન સ્ક્વેર ટાઈમનો ખર્ચ ક્રમમાં સમાપ્ત કરીશું. તેથી, આ અમને સૂચવે છે કે એક રેખીય માળખું. પ્રાધાન્યપૂર્ણ કતારમાં કાર્યક્ષમ રીતે રજૂઆતને ઉકેલવા માટે એક પરિમાણીય માળખામાં ગંભીર મર્યાદા છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:13)

તેથી, આપણે એક પરિમાણીય માળખુંથી બે પરિમાણીય માળખું પર જવાનું છે. તેથી, ચાલો આપણે ખૂબ જ નિષ્કપટ બે પરિમાણીય માળખાથી પ્રારંભ કરીએ, ફક્ત બતાવવા માટે કે આપણે કેવી રીતે સખત સુધારા મેળવી શકીએ, માત્ર એક પરિમાણ, બે પરિમાણોથી જ આગળ વધી રહ્યા છીએ. તેથી, અહીં અમે માનીએ છીએ કે આપણે કોઈ પણ પ્રકારની કુલ નોકરીમાં વધારો કરવો એ અગાઉથી જાણી શકીએ છીએ. તેથી, આપણે આ કેસમાં ધારી લીધું છે કે એન 25 છે. તેથી, આપણે જે કરીએ છીએ તે 25 લંબાઈની એક પરિમાણીય સૂચિને જાળવવાની જગ્યાએ છે - મહત્તમ લંબાઈ 25, આપણે

આ સૂચિને 5 થી 5 ની સ્ક્વેર્ડ એરે તરીકે ફરીથી ગોઠવીએ છીએ. , સ્ક્વેર રુટ n દ્વારા ચોરસ રુટ n. તેથી, હવે તમે અહીં આવી વસ્તુનો દાખલો જોઈ શકો છો. તેથી, અમારી પાસે હાલમાં 25 અલગ-અલગ 6 19 નોકરીઓ છે જેમાં વિવિધ પ્રાથમિકતાઓ છે. કેટલીકવાર બે નોકરીમાં સમાન પ્રાધાન્યતા હોઈ શકે છે જે કોઈ વાંધો નથી. પરંતુ મહત્વની વાત એ છે કે આપણે આ બે ડાયમેન્શનલ એરે દ્વારા ક્રમિક રીતે એક પ્રકારની અનુરૂપ રીતે જાળવી રાખવાની જરૂર નથી, અમે બાંહેધરી આપીએ છીએ કે દરેક પંક્તિમાં જે પંક્તિથી ડાબેથી જમણી તરફની તરફ જુએ છે તે એક ચડતા ક્રમ છે. તેથી, અમારી પાસે પહેલી હરોળ ચઢતી હુકમ છે, બીજી પંક્તિ છે, પરંતુ પંક્તિઓ પોતાનામાં કશું જ નથી

((સમયનોસંદર્ભ: 05:22)).

તેથી, તમે નાની હરોળમાં બીજી પંક્તિ અને તેનાથી વિપરિત મોટા મૂલ્યો ધરાવી શકો છો. તેથી, ઉદાહરણ તરીકે, 19 12 કરતા મોટો છે, પછી 17 એ 8 કરતા મોટો છે અને તેથી આગળ. તેથી, આ પ્રકારની નોકરીઓની પાંચ સ્વતંત્ર સૂચિ છે, જે દરેક સૂચિમાં એકસાથે કામની કુલ સેટને સૉર્ટ કરવામાં આવે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:40)

તો, હવે ધારો કે તમે આ સૂચિમાં નવી નોકરી દાખલ કરવા માંગો છો. તેથી, વ્યૂહરચના એ ખૂબ જ સરળ છે કે અમે તેને પ્રથમ સ્થાને ખાલી સ્થાન ધરાવતી સાચી સ્થાને શામેલ કરવા માંગીએ છીએ. હવે જો કોઈ પંક્તિઓમાં ખાલી જગ્યા છે કે નહીં તે શોધવા માટે, ત્યાં કોઈ ઘટકો કેવી રીતે છે તે શોધવા માટે અમને આ વસ્તુને નીચે ચાલવાની જરૂર છે. તેથી, તમે આ વધારાની માહિતીને અહીં રાખીને તે બચાવી શકો છો, જે દરેક પંક્તિઓનું કદ છે. તેથી, ચાલો ધારીએ કે આપણી પાસે પંક્તિઓ શામેલ કરવા માટે કદ ઉપલબ્ધ છે, હવે આપણે આ વિશિષ્ટ કેસમાં પંક્તિ કરીએ છીએ કે દરેક પંક્તિ મોટાભાગના કદ પાંચ છે. તેથી, જો કદ 5 છે, જ્યારે આપણે જાણીએ છીએ કે જો તે આ પહેલી પંક્તિમાં 11 શામેલ કરવાનો પ્રયાસ કરે છે, તો 11 ત્યાં ફિટ થઈ શકશે નહીં, કારણ કે મૂળરૂપે કોઈ સ્થાન નથી. તેથી, પછી આપણે આગળની હરોળમાં જઈએ, ફરી એક વાર આપણે જોશું કે કદ 5 છે. તો, પછી આપણે ત્રીજા ભાગ ઉપર જઈશું. હવે આપણે જોઈ શકીએ છીએ કે ત્યાં એક જગ્યા છે, કારણ કે આ પંક્તિમાં ફક્ત ત્રણ ઘટકો છે અને તેથી તે લઈ શકે છે, હવે આપણે આ સૂચિને કાર્ય કરીએ છીએ અને તેને શામેલ કરવા માટે યોગ્ય સ્થાન શોધીએ છીએ અને દાખલ કરીએ છીએ તે રીતે દાખલ કરીએ છીએ. તેથી, બીજો પ્રશ્ન એ છે કે આ કેટલો સમય લે છે. સારુ, આપણે જાણીએ છીએ કે તેની પાસે એન પંક્તિઓનો વર્ગમૂળ છે. તેથી, તે ટોચથી તળિયે જવા માટે યોગ્ય પંક્તિ શોધવા માટે N પગલાંઓનો સ્ક્વેર રુટ લે છે, અને પછી અમારી પાસે દરેક એન્ટ્રીઝ એન એન્ટ્રીઝના સ્ક્વેર રૂટ છે. તેથી, જો આપણે સામાન્ય સંમિશ્રણ કરીએ તો તે પંક્તિની લંબાઈ માટે પ્રમાણસર સમય લે છે, અમે તેમાં શામેલ છીએ. તેથી, અમારી પાસે યોગ્ય પંક્તિ શોધવા માટે N સ્કેનનું સ્ક્વેર રુટ છે, તે પંક્તિની અંદર યોગ્ય સ્થિતિ શોધવા માટે N સ્કેનનું વર્ગમૂળ. તેથી, એકંદરે તે સમયનો વર્ગમૂળનો ઓ ઓર્ડર છે. તો, કાઢી નાખવા વિશે શું?

(સ્લાઈડસમયનો સંદર્ભ લો: 07:15)

તેથી, આપણે આ એરેમાં મહત્તમ તત્વ કાઢી નાખવા માંગીએ છીએ. હવે કારણ કે દરેક પંક્તિ બીજી પંક્તિથી સ્વતંત્ર છે, આપણે અગાઉથી જાણતા નથી કે આ મહત્તમ તત્વ ક્યાં છે. જો કે, આપણે જાણીએ છીએ કે દરેક હરોળમાં મહત્તમ તત્વ

અંતમાં છે, કારણ કે દરેક પંક્તિ સૉર્ટ કરવામાં આવે છે. તો, પ્રથમ વસ્તુ જે આપણે જાણીએ છીએ તે છે કે દરેક સંભવિત મહત્તમ તત્વ વાસ્તવમાં તેની પંક્તિના અંતમાં છે. તો, આપણી પાસે આ પાંચ પંક્તિઓ છે, આપણી પાસે પાંચ મહત્તમ તત્વો છે. અને હવે એકંદરે મહત્તમ આ પાંચ તત્વોમાં હોવું આવશ્યક છે, તે સૌથી મોટું હોવું જોઈએ. તો, આપણે ફક્ત કદની માહિતીનો ઉપયોગ કરી શકીએ છીએ. તેથી, આપણે જાણીએ છીએ કે તે પ્રથમ પંક્તિમાં પાંચમું તત્વ છે, બીજી પંક્તિમાં પાંચમું તત્વ, ત્રીજી અને ચોથી પંક્તિમાં ચોથું તત્વ, અને બીજું તત્વ છેલ્લી પંક્તિ છે. તેથી, કદની માહિતીનો ઉપયોગ કરીને, આપણે આ બધામાંથી સ્કેન કરી શકીએ છીએ અને 43 ની જરૂરિયાતોને કાઢી નાખવાની જરૂર છે તે ઓળખવા માટે આ કેસમાં સૌથી મોટો કોણ છે તે ઓળખી શકે છે. અને પછી 43 ને કાઢી નાખ્યા પછી આપણે તેને દૂર કરીએ, અને પછી આપણે માપ 4 થી 3 સુધી પણ ઘટાડીએ. તેથી, આપણે જ્યારે માપમાં ગયા ત્યારે આપણે અગાઉના ઉદાહરણમાં ચેપ લગાવી શકીએ તે પ્રમાણે આ માપનો ટ્રેક રાખી શકીએ છીએ. 11, આપણે આ 3 ને 4 માં બનાવીએ. તેથી, ફરીથી તે ક્રમમાં ક્રમિક ક્રિયા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:34)

તેથી, આપણે હવે ડેટા માળખું પ્રાપ્ત કર્યું છે, જે પ્રાયોરિટી ક્યુ(Priority queues)માં ઘટકોને ટ્રૅક રાખે છે જ્યાં દાખલ થવા માટે ક્રમમાં N સમય લે છે, મહત્તમ ક્રમમાં ઓર્ડર રુટ N ને કાઢી નાખો, અને તેથી, હવે પ્રક્રિયા કરી રહ્યું છે એન નોકરીઓનો ક્રમ એન રૂટ એન સમય લે છે. યાદ રાખો કે અગાઉ તે એન ચોરસ ઓર્ડર હતો. તેથી, જો તમે ઈચ્છો છો

(અમેસમય: 08:54)),

તો આપણે એન સ્ક્વેરમાંથી ક્રમમાં N 3 ને 2 દ્વારા ક્રમમાં ઘટાડીએ છીએ. તેથી, આ સમજૂતી કરવા માટે ફક્ત એક નમૂના છે કે બે પરિમાણીય માળખું તમને રેખીય શોધ પર નોંધપાત્ર બચત આપી શકે છે. તેથી, અલબત્ત, આપણે આનાથી ખુશ થવાનો નથી, બીજાઓ તમે ફક્ત આનાથી રોકશો. તો, આપણે વાસ્તવમાં એન થી 3 ની 2 ઘાત કરતા વધુ સારું કરી શકીએ છીએ, અને આ પછી આપણે લેક્ચરમાં ચર્ચા કરવા જઈ રહ્યા છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:14)

તમને એક પૂર્વાવલોકન આપવા માટે, આપણે શું કરવા જઈ રહ્યા છીએ તે તેને સરળ એરે અથવા સ્ક્વેર મેટ્રિક્સમાં જાળવવાનું છે, પરંતુ એક વિશિષ્ટ પ્રકારના બાયનરી ટ્રીમાં તેને લેપ કહેવામાં આવે છે. તેથી, આ એક દ્વિસંગી વૃક્ષ હશે. તેથી, તેના જેવી રચના હશે, અને તે સંતુલિત થશે. તેથી, મૂળભૂત રીતે તમામ પાસની લગભગ સમાન લંબાઈ હશે, અને આનો અર્થ શું થશે તે છે કે વૃક્ષની ઊંચાઈ વૃક્ષના કદમાં લઘુગણક હશે. અને આ મેક્સ લે લોગ એન ઓપરેશન્સને શામેલ કરશે અને કાઢી નાખશે, અને આ એન લોગ એન ના એન ઓપરેશન્સ માટે એકંદર નીચે આપવામાં આવશે. અન્ય વસ્તુ એ છે કે આપણે ખરેખર તેને ગતિશીલ વૃક્ષ તરીકે જાળવી રાખીએ છીએ, અમારે તેને બનાવવાની જરૂર નથી. એક ધારણા જેમ કે અમે સરળ અથવા સોલ્યુશન સોલ્યુશન કર્યું છે, જે અમે હમણાં જ પ્રસ્તાવ મૂક્યો છે, જ્યાં અમે ઉપલા બાહ્ય એન. અમે એક ઉકેલ મેળવી શકીએ છીએ જ્યાં કી આપણે જેટલી મોટી બની શકીએ. તેથી, જ્યાં સુધી આપણે ખાતરી કરીએ કે આપણે તેને સંતુલિત રાખવા માટે વ્યવસ્થિત રીતે વૃદ્ધિ કરીએ છીએ. તો, આ આપણે આગળ જોશું, આપણે આ ડેટા સ્ટ્રક્ચરને એક ઢગલો કહીશું જે સેટ ટાઈપ બાયનરી ટ્રીને જાળવી રાખીને પ્રાથમિકતા કતારને લાગુ કરે છે.

ડિઝાઈન અને એનાલિસિસ ઓફ્ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 05
લેક્ચર - 35
હિપ્સ

હવે ચાલો હિપ્સ જુઓ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:03)

તેથી, યાદ રાખો કે અમારું લક્ષ્ય પ્રાયોરિટી ક્યુ(Priority queues)ને અમલમાં મૂકવું છે. પ્રાયોરિટી ક્યુ(Priority queues)માં, અમારી પાસે કામની શ્રેણી છે જે સિસ્ટમમાં પ્રવેશ કરે છે, દરેક કામને અગ્રતા છે. જ્યારે પણ, અમે અમલ કરવા માટે કામને સુનિશ્ચિત કરવા માટે તૈયાર છીએ, ત્યારે અમને નવીનતમ જોબ અથવા પ્રારંભિક કામ ન મળવી જોઈએ, પરંતુ જે કામ હાલમાં રાહ જોતી કામઓમાં સૌથી વધુ પ્રાધાન્ય છે. તેથી, અમને ડિલીટ મેક્સ કહેવાતા ઑપરેશનની જરૂર છે જે બાકી રહેલા અને તેમાં શેડ્યૂલ કરેલા લોકોમાં ઉચ્ચ પ્રાધાન્યતા જોબ શોધશે. અને આપણે દેખીતી રીતે, એક શામેલ ઓપરેશન કર્યું છે જે આ કામને ગતિશીલ રીતે તેઓ પહોંચે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:37)

તેથી, આપણે છેલ્લે જોયું કે રેખીય માળખું આપણને આ બંનેને એકસાથે ઑપ્ટિમાઈઝ કરવા દેશે નહીં. મહત્તમ અથવા કાઢી નાખવા માટે ઑર્ડર N ઑપરેશન માટે અમે ઑર્ડર N ઑપરેશન સાથે અંત કરીએ છીએ. પછી, આપણે ટૂંકા બે પરિમાણીય એરે જોયા જે આપણને એન રૂટ એન સોલ્યુશન આપે છે જે આમાંના દરેક માટે રુટ એન ઓપરેશન છે, તેથી અમારા એન ઓપરેશન્સ એન થી એન ઓર્ડર છે. પરંતુ, અમે કહ્યું કે અમે વધુ સારી માહિતી માળખું શોધીશું એક ખાસ પ્રકારની વૃક્ષનો ઉપયોગ કરીને હિપ્ કહેવાય છે. તેથી, હિપ્ એક સંતુલન વૃક્ષ બનશે, જેની ઊંચાઈ કદમાં લઘુગણક છે, જો વૃક્ષમાં ધ્વજ એન ગાંઠો હોય, તો ઊંચાઈ કે જે રુટથી કોઈપણ પાંદડા સુધી ધારની સંખ્યા છે તે લૉગ એન હશે અને આ સાથે , તે બંનેને શામેલ કરશે અને મહત્તમ એનને લોગ કરવા માટે પ્રીપોઝિશનલ કાઢી નાખશે અને તેથી, N નો કાર્યવાહી કરવામાં આવશે સમયનો સમય લેશે N એ એન.આર. રૂટ એન માટે માનવામાં આવે છે અથવા રેખીય રજૂઆત માટે N ચોરસ માટે. અમે એમ પણ કહ્યું કે સિદ્ધાંતમાં આ હિપ્ લવચીક છે અને આપણે જે જોઈએ તેટલું વધારી શકીએ છીએ. તેથી, આપણે અગાઉથી નક્કી થતાં ઢગલાના માપને સુધારવાની જરૂર નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:39)

તેથી, ચાલો આપણે એક હિપ ટેસ્ટ શું છે તે જોવાનું શરૂ કરીએ. તેથી, દ્વિસંગી વૃક્ષ એ એક વૃક્ષ છે જ્યાં આપણી પાસે રુટ છે અને દરેક નોડમાં 0, 1 અથવા 2 બાળકો છે. તેથી, સામાન્ય રીતે દ્વિસંગી વૃક્ષો મનસ્વી આકાર હોઈ શકે છે. તેથી, અમારી પાસે દ્વિસંગી વૃક્ષો હોઈ શકે છે જે આ જેવા દેખાય છે, જ્યાં રુટમાં 1 બાળક હોય છે, તેમાં 2 બાળકો હોય છે અથવા તે એક દિશામાં વધુ અવ્યવસ્થિત દેખાય છે. તેથી, દ્વિસંગી વૃક્ષો ખૂબ જ વિચિત્ર આકાર હોઈ શકે છે, પરંતુ એક

હિપ્ ટ્રિવસંગી વૃક્ષ છે જે ખૂબ જ ચોક્કસ આકાર ધરાવે છે, જ્યાં આપણે વૃક્ષ ગાંઠો ભરીએ છીએ અથવા આપણે ચોક્કસ ક્રમમાં વૃક્ષના ગાંઠોને ઉમેરીએ છીએ. તેથી, પ્રથમ આપણે રુટ પર પ્રારંભ કરીએ, પછી આપણે રુટના ડાબા બાળકને ઉમેરવું જોઈએ, પછી યોગ્ય બાળક અને આ રીતે સ્તરથી ડાબેથી જમણે સ્તર ચાલુ રાખવું જોઈએ. તેથી, આપણે આ નોડ ઉમેરીએ, પછી આપણે આ નોડ ઉમેરીશું, પછી આપણે આ નોડ ઉમેરીશું. તેથી, એકવાર મને ખબર છે કે વૃક્ષમાં કેટલા ગાંઠો છે, મને ખબર છે કે આકાર શુ બરાબર છે, તેથી આકાર સુધારાઈ ગયો છે. તેથી, તે ઢગલાની પહેલી લાક્ષણિકતા છે કે જો મારી પાસે n ગાંઠો સાથે હિપ્ હોય, તો વૃક્ષના આકારને આ નિયમ દ્વારા નિશ્ચિતપણે નિશ્ચિત કરવામાં આવે છે કે n ગાંઠો નીચેથી નીચે, ડાબેથી જમણે શામેલ હોવા જોઈએ, પછી અમારી પાસે મૂલ્ય મિલકત. તેથી, મૂલ્ય ગુણધર્મ કહે છે કે ... તેથી, વૃક્ષમાં શું થઈ રહ્યું છે તે છે કે આપણી પાસે ગાંઠો છે અને દરેક ગાંઠો મૂલ્ય છે, તેથી જ્યારે પણ હું મૂલ્ય વિરુદ્ધ 1 નો નોડ જોઉં છું જેમાં બાળકો v2 અને v3 હોય, તો પછી આપણે જે જોઈએ તે છે, તે વી 2 કરતા મોટું અથવા તેના બરાબર છે અને મોટા 3 ની બરાબર અથવા બરાબર છે. તેથી, આ ત્રણ નોડો વચ્ચે સૌથી મોટું એક v1 હોવું જોઈએ, તેથી આને મહત્તમ હિપ્ ગુણધર્મ કહેવામાં આવે છે. તેથી, આ સ્થાનિક મિલકત છે, તે ફક્ત તે નોડ પર દરેક નોડ પર અમને કહે છે, 2 બાળકોને જુઓ, નોડ તેના કરતાં 2 બાળકો વધારે મોટું હોવું જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:23)

તેથી, અહીં 4 નોડ્સ સાથે હિપનું ઉદાહરણ છે, તેથી પ્રથમ કારણ કે તે 4 નોડ્સ છે, દરેક 4 નોડ હિપમાં આકાર હશે. કારણ કે, પ્રથમ નોડ રુટ હશે, બીજો મૂળ મૂળ છોડશે, ત્રીજો નોડ યોગ્ય બાળક હશે અને ચોથા નોડ નવી લાઈન શરૂ કરશે, પછી વધુ આપણે હિપ પ્રોપર્ટીને ચકાસી શકીએ છીએ. તેથી, આપણે જોયું કે 24 એ 11 કરતા મોટું છે, 24 7 કરતા મોટું છે. તેથી, આ હિપ પ્રોપર્ટી માટે માન્ય નોડ છે, 11 એ 10 કરતા મોટો છે, કોઈ યોગ્ય બાળક નથી, તેથી આ માન્ય હીપ છે, ત્યાં કોઈ નથી 7 નું બાળક તેથી, ટૂંકાગરીથી આ માન્ય હિપ નોડ છે અને 10 એ જ કારણોસર માન્ય હિપ નોડ છે. તેથી, દરેક પાંદડા નોડ જેનો કોઈ બાળકો હોતો નથી તે હંમેશાં ઢગલાબંધ સંપત્તિને સંતોષે છે. તેથી, એકવાર તમારી પાસે પર્ણ નોડ હોય, તો તપાસ કરવા માટે કંઈ નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:16)

તેથી, અહીં બીજી હિપ્ છે, આમાં 7 નોડ છે, તેથી ફરીથી આકાર સુધારાઈ ગયો છે અને ફરીથી તમે ચકાસી શકો છો કે આ 11 અને 7 કરતા વધારે છે, 11 10 અને 5 કરતા વધારે છે અને 7 6 અને 5 કરતા મોટો છે અને બાકીના બધા પર્ણ નોડ છે, તેથી કોઈ સમસ્યા નથી. તેથી, આ ઢગલાનાં બે ઉદાહરણો છે. તેથી, કંઈક કે જે હિપ્ નથી તેનું ઉદાહરણ શું છે?

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:35)

તેથી, અહીં આપણી પાસે કંઈક છે જે હિપ્ નથી, કારણ કે માળખું ખોટું છે. તેથી, અમે કહ્યું કે તમે પાંદડા છિદ્રો નાખી શકતા નથી, તમારે ઉપરથી નીચે ડાબેથી જમણે જવું આવશ્યક છે, તેથી જમણી બાજુ નોડ ઉમેરતા પહેલા, અહીં કેટલાક નોડ હોવા જોઈએ. તેથી, આ નોડ ક્યાં છે? આ નોડ ખૂટે છે, તેથી આ માળખું યોગ્ય નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:51)

આ જ કારણસર, આ માળખું પણ સાચું નથી, કારણ કે અહીં આપણી પાસે કંઈક છે જે ખૂટે છે, આ સ્તરે નોડ છે અને આપણે નવી લાઈન શરૂ કરી છે. તેથી, આ બંને માળખાકીય કારણોસર પણ નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:04)

બીજી બાજુ, આપણે કંઈક એવું જોયું જે માન્ય માળખું છે, હકીકતમાં આપણે ઢગલાને જોયું તે પહેલાં માળખું ધરાવે છે, સમસ્યા આ નોડ સાથે છે. તેથી, આપણે 7 ને 8 અને 5 કરતા મોટા હોવા જોઈએ, પણ આ કોર્સ નથી. 7 8 કરતા મોટું નથી, 7 એ 8 કરતા નાની છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:20)

તેથી, આ નોડ 8 વાસ્તવમાં હિપ પ્રોપર્ટીનું ઉલ્લંઘન કરે છે, તેથી કંઈક ઢગલામાં નિષ્ફળ થઈ શકે છે, કાં તો વૃક્ષનું માળખું ખોટું છે અથવા કારણ કે કેટલાક નોડ પર હિપ્ મિલકતનો ભંગ થાય છે. આ કિસ્સામાં, નોડ જે હીપ પ્રોપર્ટીનું ઉલ્લંઘન કરે છે તે 7 છે, કારણ કે તેમાંનું એક તે છે જે ખરેખર નોડ કરતા બાળકો મોટા હોય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:37)

તેથી, હવે આપણે આ બે ઓપરેશન્સને હિપ્સ પર અમલમાં મૂકવું પડશે, મહત્તમ દાખલ કરવું અને કાઢી નાખવું પડશે. તેથી, ચાલો જોઈએ કે તે કેવી રીતે કાર્ય કરે છે? તેથી, ચાલો પહેલા 12 દાખલ કરીએ, તેથી 12 દાખલ કરો, એટલે કે મને હીપમાં વેલ્યુ ઉમેરવું પડશે. તો, પ્રથમ વસ્તુ જ્યારે હું હીપમાં મૂલ્ય ઉમેરું છું ત્યારે તે એક વૃક્ષને વિસ્તૃત કરવું આવશ્યક છે. તેથી, હું આ નોડ ક્યાં મૂકું? તેથી, આ હવે સુધારેલ છે કારણ કે આપણે જાણીએ છીએ કે ઢગલાઓ માત્ર એક જ રીતે વધવા અને સંકોચાઈ શકે છે, તેથી મારે જમણે જમણે, ઉપર થી નીચે નવો નોડ ઉમેરવો જોઈએ. તેથી, આ કિસ્સામાં જો હું ડાબેથી જમણે જઈશ, ઉપર થી નીચે હું આ બિંદુ પર આવીશ, હવે હું વધુ કંઈ ઉમેરી શકતો નથી, તેથી તે આગલા સ્તર પર આવવું આવશ્યક છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 06:16)

તેથી, પ્રથમ વસ્તુ એ છે કે આ નવો નોડ 12 માં આવવો આવશ્યક છે. હવે, જો હું આ સ્થિતિમાં 12 મુકું, તો સમસ્યા એ છે કે મારી પાસે હિપ ગુણધર્મો સંતોષી શકશે નહીં . આ કિસ્સામાં, તમે જોઈ શકો છો કે હિપ પ્રોપર્ટી વાસ્તવમાં અહીં જ આવે છે, કારણ કે તે 12 કરતા વધારે મોટું છે. તેથી, 10 હીપ પ્રોપર્ટીનું ઉલ્લંઘન કરે છે, પરંતુ નોંધ લો કે આ ફક્ત ઉપર થઈ શકે છે, તેથી મારો અર્થ એ છે કે જ્યારે તમે કંઈક શામેલ કરો છો તે એક પર્ણ છે. તેથી, જ્યારે તમે 12 થી પર્ણ છો, તે 12 ની નીચેની હીપ મિલકતને નિષ્ફળ કરી શકતું નથી, કારણ કે ત્યાં 12 વર્ષથી નીચેના કોઈ બાળક નથી. જો બધી સંપત્તિ નિષ્ફળ જાય, તો તે છે કારણ કે નવા નોડના માતાપિતાનું મૂલ્ય ઘણું ઓછું હોય છે . તેથી, ઓછામાં ઓછા આને ઠીક કરવા માટે એક સરળ રીત છે અને તે આ બંનેને વિનિમય કરવો છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:57)

તેથી, હું આ 12 અને 10 નું વિનિમય કરું છું અને હવે હું અહીં સમસ્યાને ઠીક કરું છું, પરંતુ હવે હું આ બિંદુએ વેલ્યુ બદલીશ. તેથી, હું આ ગોઠવણીને જોવા માટે જોઉં છું કે હું 12 વડે માલ મિલકતનું ઉલ્લંઘન કરું છું કે નહિ અને અહીં ફરીથી તમે જોઈ શકો છો કે ત્યાં કોઈ સમસ્યા છે કારણ કે તે 12 કરતા પણ મોટું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:20)

તેથી, પછી હું તે વિનિમય કરું છું, તેથી હવે હું આનું વિનિમય કરું છું, હવે પ્રક્રિયામાં શું થયું છે તે અહીંથી કૂવામાં આવ્યું હતું. અને હવે, સૌ પ્રથમ આપણે પોતાને સમજાવવાની જરૂર છે કે આપણે જે હિપ્ સ્થાનિક હિપ્ બનાવ્યો છે તે કોઈ પણ વધુ ફિક્સિંગની જરૂર છે. બીજા શબ્દોમાં, આપણે ખાતરી આપવી જોઈએ કે ત્યાં કોઈ સમસ્યા નથી અને બીજી બાજુ 12 અને 5 ની વચ્ચે છે અને આ થઈ શકતું નથી, કારણ કે મૂળરૂપે આપણી પાસે 11, 5 અને અહીં કંઈક હતું, આપણે જાણીએ છીએ કે 11 એ 5 કરતા મોટું હતું અને તે કરતા મોટું કંઈક, પછી અમે કર્યું અને વિનિમય અને અહીં 12 ખાતે ખરીદી શકાય છે. તેથી, જો બધી સમસ્યાઓ ફક્ત 11 અને 12 હશે અને ઉલ્લંઘન સારું હોય તો 11 કરતા મોટી ઈચ્છા હોય તો ઉલ્લંઘન થાય છે, પરંતુ 11 એ બીજી બાજુ કરતાં મોટી હોય છે. તેથી, જો હું આ ત્રાણ નોડ માળખાની ટોચ પર 12 ખસેડે, તો તે બીજી બાજુ કરતા નાની હોઈ શકતી નથી. કારણ કે, રુટ હાલમાં બીજી બાજુ કરતાં પહેલાથી મોટો છે, એકમાત્ર કારણ એ છે કે હું આમાંના 12 ને ખસેડી શકું છું કારણ કે તે હજી કરતા વધારે છે

((સંદર્ભઆપો: 08:18)).

તો, પછી જ્યારે હું આ પ્રમાણે કરીશ અને ઉપર સ્વેપ કરીશ ત્યારે હું ખાતરી કરી શકું છું કે બીજી તરફ નજર રાખવાની જરૂર નથી, હું ફક્ત આગળ જતો રહ્યો છું. તેથી, હવે મને આ બિંદુ સુધી 12 મળ્યું છે, તેથી હવે હું તપાસ કરું છું કે ત્યાં કોઈ વધુ ઉલ્લંઘન છે કે નહીં અને તે 12 ની સંખ્યા 24 કરતા નાની છે. તેથી, હું રોકી શકું છું, તેથી આમાં 12 શામેલ કરવાનું પરિણામ હતું. હિપ્.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:41)

હવે, હું નવા નોડ 33 દાખલ કરી શકું છું કે આપણે કેવી રીતે કરવું તે જાણે છે, તેથી પહેલા જો હું 33 શામેલ કરું. તેથી, 12 પછી અને ત્યારબાદ આ ઢગલાનું પરિણામ હતું. અહીં 33 શામેલ કર્યું છે, તેથી તે સાચું હોવું જોઈએ કારણ કે તે માળખું માં આગલું નોડ છે અને મેં મૂલ્ય મૂક્યું છે, પરંતુ તે હિપ પ્રોપર્ટીનું ઉલ્લંઘન કરશે 11 અને 33, તેથી હું તેને સ્વેપ કરીશ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:00)

અને પછી ફરીથી હું અહીં ઉલ્લંઘન કરું છું, તેથી હું તેને ફરીથી સ્વેપ કરું છું.

(સ્લાઈડસમયનો સંદર્ભ લો: 09:06)

ફરીથી મારી પાસે ઉલ્લંઘન છે, તેથી હું તેને ફરીથી સ્વેપ કરું છું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:07)

અને હવે હું રુટ સુધી પહોંચું છું, તેથી વાસ્તવમાં તે સૌથી મોટો નોડ બની ગયો છે અને હવે હું બંધ કરી શકું છું.

(સ્લાઈડસમયનો સંદર્ભ લો: 09:15)

તો, આ કેટલો સમય લે છે? તેથી, દર વખતે આપણે જોયું કે આપણે દરેક વખતે દાખલ કરીએ છીએ, આપણે લીફ નોડ પર એક નવું લીફ નોડ શરૂ કરીએ છીએ જે આપણે બનાવે છે અને રુટ સુધી ચાલીએ છીએ. તેથી, આ વસ્તુનો સૌથી ખરાબ કેસ તે વૃક્ષની સૌથી ખરાબ કેસની ઊંચાઈ પર આધાર રાખે છે, જો મારી પાસે આ પ્રકારનું વૃક્ષ હોય તો વ્યાખ્યા દ્વારા વૃક્ષની ઊંચાઈ, વૃક્ષની ઊંચાઈને બંધ કરવું પડશે. તેથી, વૃક્ષની ઊંચાઈ એ સૌથી લાંબી શોધ માર્ગ છે, જે રુટમાંથી તેમને સૌથી લાંબી માર્ગની લંબાઈ છે. તો, આપણે ક્યાં તો ધારની સંખ્યાઓની સંખ્યા અથવા શિરોબિંદુઓની સંખ્યા ગણી શકીએ છીએ

((સમયનોસંદર્ભ: 09:47))

તે શિરોલંબ કરશે કે તે ટૂંક સમયમાં 4 હશે, જો તે ધાર હશે તો તે 3 થશે, પરંતુ ખરેખર મુદ્દો એ છે કે આનો સૌથી લાંબો માર્ગ જટિલતા નક્કી કરશે, કારણ કે જે માર્ગે લાંબા સમય સુધી મને માર્ગ પર સ્વેપ કરવાની જરૂર છે. તેથી, ઢગલાની ઊંચાઈ વિશે શું કહી શકાય, તેથી પ્રથમ વસ્તુ ધ્યાનમાં લેવી તે છે કે આપણે ઢાંકનારી રીતે તે કરી શકીએ છીએ. તો, રૂટ નોડ પર આપણી પાસે લેવલ 0 છે, આપણે આ સ્તર 0 ને બોલાવીએ છીએ, આપણી પાસે સ્તર 1 પર બરાબર એક નોડ છે અને મોટા ભાગના પાસે 2 નોડ્સ છે. તેથી, આપણે લખી શકીએ કે 2 ની શક્તિ માટે 2 છે, આ 2 ની શક્તિ 2 છે, આમાંના પ્રત્યેક પાસે 1 હશે. તેથી, આપણે બમણું કરીશું, તેથી દરેક સ્તર પર નોડ્સની સંખ્યા ડબલ્સ થશે, કારણ કે દરેક અગાઉના સ્તરમાં બે બાળકો હોય છે. તેથી, આપણે સ્વેપ કરવું પડશે, તેથી આ રીતે આપણે સ્તર 0 પર 2 ની સંખ્યાઓ 0 ની શક્તિથી સ્તર 1 પર 2 ની સંખ્યા છે, પાવર 1 પર, કોઈપણ સ્તર પર હું 2 છે. તેથી, જો તમારી પાસે કે સ્તરો છે, તો સ્તરો 0 છે, 1 થી 1 ની બાદબાકી 1 થી કામ કરે છે, આપણે હમણાં જ કહ્યું છે કે 2 થી 0 વત્તા 2 થી 1 વત્તા 2 સુધી, કે બાદબાકી 1

((સમયનોસંદર્ભ લો)

: 10:55)).

તેથી, તે 2 થી 0 વત્તા 2 થી 1 વત્તા 2 ની બાદબાકી 1 ની થશે, હવે આ અને તેના વિશે વિચારવાનો એક રસ્તો એ કે 1 ની સાથે બાયનરી નંબર છે. તેથી, બાયનરી નંબર કે 1 એસ એ બીજા કેવળમાં પાવર કે બાદ 1, માત્ર 2 છે, જો હું કે સ્તરો માટે દ્વિસંગી ઝાડ ભરીશ તો મારી પાસે મોટાભાગના 2 કે ઓછા 1 નોડ્સ હશે. તેથી, વીઅગાઉ, ગાંઠોની સંખ્યા ઘટક છે તે

સંખ્યાઓની સંખ્યા છે. તેથી, જો મારી પાસે સંખ્યાઓની સંખ્યામાં નોડોની સંખ્યા હોવી જ જોઈએ, તો લઘુગણક હોવા જોઈએ

((સમયનોસંદર્ભ લો: 11:31)).

અને સ્તરોની સંખ્યા એ નક્કી કરે છે કે, સૌથી લાંબી પાથની લોગરિધર્મિક લંબાઈ અને તેથી, કોઈપણ હિપ્ ઇાખલ કરો એનનો સમય લોગ લેશે, કારણ કે લૉંગ એન હોવાના દરેક માર્ગને ગેરંટી આપવાની છે.

(સ્લાઈડજુઓ સમય: 11:46)

તેથી, અગ્રતા કતાર માટે અમલીકરણ કરવાની જરૂર રહેતી અન્ય કામગીરી મહત્તમ કાઢી નાખવી છે. તેથી, પ્રથમ પ્રશ્ન એ છે કે ઢગલામાં મહત્તમ ક્યાં છે? તેથી, દાવો એ છે કે રુટમાં હંમેશાં મોટાભાગના છે, તે શા માટે છે કારણ કે જો હું ગમે ત્યાંથી શરૂ કરું તો મને ખબર છે કે કોઈપણ 3 નોડ્સમાં મહત્તમ તે ટોચ છે. તેથી, જો હું 33 ઉદાહરણ તરીકે જોઉં છું, 33 એ 24 અને 7 કરતા મોટો છે, પરંતુ ઈન્ટરેક્ટીવ રીતે હું જાણું છું કે 24 આ પેટા વૃક્ષમાં સૌથી મોટો નોડ હોવો જોઈએ અને 7 ઉપ ટ્રીમાં સૌથી મોટો નોડ હોવો જોઈએ. તેથી, તેથી, 34 એ 33 કરતા વધારે મોટું છે તે સીધું મોકલવામાં આવેલું સૌથી મોટું નોડ હોવું આવશ્યક છે. તેથી, મૉડ્યૂલ રૂટ પર પહેલાથી જ મોડેલ 3 મહત્તમ મૂલ્યો છે. તેથી, હવે પ્રશ્ન એ છે કે રુટ પર મહત્તમ મૂલ્યો, આપણે તેને વૃક્ષથી અસરકારક રીતે કેવી રીતે દૂર કરીશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:39)

તેથી, કહો કે તમે મહત્તમ મૂલ્ય દૂર કરો, તેથી આ પર્ણ છિદ્ર પાસે છે, આપણી પાસે રુટ પર કોઈ મૂલ્ય નથી, તે જ સમયે, કારણ કે અમે મૂલ્યને દૂર કર્યું છે એક દ્વારા વૃક્ષમાં મૂલ્યોની સંખ્યા ઘટાડે છે. પરંતુ, અમે કહ્યું હતું કે વૃક્ષનું માળખું નિશ્ચિત છે, જો તમે એક દ્વારા ઘટાડો કરો છો, તો આપણે રુટને દૂર કરી શકતા નથી, આપણે આ ડાબેથી જમણે ઉપરથી નીચેની તરફ જતાં છેલ્લા નોડને દૂર કરવું જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:05)

તેથી, આપણે આ નોડને અહીં જવા માટે અહીં નોડને દૂર કરવા જ જોઈએ, તેથી હવે આપણી પાસે મૂલ્ય છે જે બેઘર છે, તેના માટે મૂળ રજકણો નથી અને અમારી પાસે ઘર ખાલી છે. તો, આપણે શું કરીશું તે આપણે 11 ને રુટ પર ખસેડીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:25)

હવે, દુર્ભાગ્યવશ, કારણ કે આપણે પાંદડામાંથી કેટલાક મનુષ્ય નોડને રુટમાં ખસેડતા આમ કરીને ઢગલાબંધ હુકમને વિક્ષેપિત કરી રહ્યા છીએ, આપણે જાણીએ છીએ કે અમારી પાસે હિપ પ્રોપર્ટી સંતુષ્ટ છે કે નહીં. હવે, એકમાત્ર જગ્યા જ્યાં રત્ન પર ઢગલાબંધ મિલકતનો ભંગ કરી શકાય છે, કારણ કે દરેક જગ્યાએ સ્થાનિક પડોશી તે ઓપરેશનને સ્પર્શતું નથી, તમે માત્ર તેમના સ્થાને તેમનો પડોશી સ્પર્શ કર્યો હતો, પરંતુ હંમેશાં નોડને દૂર કર્યો હતો. જો તમે પાંદડાને દૂર કરો છો તો

તે હિપ પ્રોપર્ટીનું ઉલ્લંઘન કરી શકતું નથી, કારણ કે ઉપલા નોડ માટે તે આ વૃક્ષ બંને કરતા પહેલાથી મોટું છે. તેથી, આ ... તેથી, એકમાત્ર સ્થાન જેનું ઉલ્લંઘન થઈ શકે તે અહીં છે અને ખરેખર આપણે કરીએ છીએ, કારણ કે 11 અને 24 એ ખોટા પર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:01)

તેથી, આપણે હવે હીપ પ્રોપર્ટીને નીચે સંગ્રહિત કરીશું, જ્યારે આપણે દાખલ કર્યું હતું અમે ઉપર કર્યું હતું, તે અહીંથી શરૂ થાય છે અને આ તરફ ધ્યાન દોરે છે અને પછી આપણે બંને દિશાઓ તરફ જોશું અને આપણે મોટી સંખ્યામાં લઈશું બેમાંથી એક અને તેને ખસેડો. તેથી, અમે સૌથી મોટા બાળક સાથે સ્વેપ કરીએ, ધારો કે આ સંચાલક 7 નથી, પરંતુ 17 તો હું શું કરી શકું, જો તમે 17 ની 11 અને 24 ની જોયેલી 17 ની સંખ્યામાં વધારો કર્યો હોય અને આ ઢગલાબંધ મિલકતને ઠીક કરશે નહીં, કારણ કે આ જોઈશે કે તેઓ જોઈએ છે. તેથી, આપણે બંનેને મોટામાં વધારે લેવું જોઈએ અને તેને ખસેડવું જોઈએ, કારણ કે આ ત્રણમાંથી સૌથી મોટું મૂલ્ય એ ટોચ પર હોવું જોઈએ કે જે મહત્તમ હિપ ગુણધર્મની વ્યાખ્યા છે, તેથી અમે 11 અને 24 નું વિનિમય કરીએ.

(સ્લાઈડસમયનો સંદર્ભ લો : 14:44)

તેથી, 24 રુટ સુધી જાય છે અને પછી 11 આ દિશામાં આવ્યા છે, તેથી અમારે બીજું તપાસ કરવું જોઈએ કે આ ભાગ જે હવે વિક્ષેપિત છે તે ભૌતિક મિલકતને સંતુષ્ટ કરે છે કે નહીં. અને અલબત્ત, આ કિસ્સામાં તે 11 અને 12 યોગ્ય વસ્તુમાં નથી કારણ કે તે નથી. તેથી, આ 3 ની વચ્ચે ફરીથી મને મહત્તમ વેલ્યુ અપ લેવા પડશે, તેથી હું 12 ઉપર લઈ જાઉં છું અને 11 ને નીચે ખસેડીશ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:04)

અને હવે મને આ વિભાગ તપાસવાની જરૂર છે કે કેમ આ હિપ પ્રોપર્ટી સંતુષ્ટ છે, અહીં તે સંતુષ્ટ છે, હવે આપણે બંધ કરવા માંગીએ છીએ. તેથી, મહત્તમ કાઢી નાખવા માટે હું રુટથી શરુ કરું છું અને હું નીચે જવામાં છું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 15:16)

તેથી, ધારો કે આપણે આ ફરીથી કરીએ, તો હું 24 ને દૂર કરીશ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 15:25)

અને પછી હું 10 ને છેલ્લા પર્ણથી ટોચ પર ખસેડીશ.

(સ્લાઈડસમયનો સંદર્ભ લો: 15:28)

પછી, મારે ફરીથી આ સમસ્યાને ઠીક કરવી પડશે, તેથી હું 10 અને 12 નું વિનિમય કરું છું.

(સ્લાઈડસમયનો સંદર્ભ લો: 15:34)

પછી, મારે આને જોવું પડશે અને આ સમસ્યાઓને ઠીક કરવી પડશે આ વૃક્ષોમાંથી સૌથી મોટું 11 છે હું 10 અને 11 ઠીક કરું છું અને મને તે મળે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:39)

તેથી, હવે ખાતરી કરીને કે હું ફરીથી સૌથી મોટો કરું છું, હું અન્ય દિશામાં ન જતો. તેથી, હું હંમેશાં એક પાથ તરીકે ચાલતો રહ્યો છું, તેથી ફરી એક વખત ફરીથી શામેલ કરવા જેવી કિંમત ઊંચાઈને પ્રમાણસર છે. અને કારણ કે આપણે જાણીએ છીએ કે એક ઢગલામાં ઊંચાઈ લઘુગણક છે, મહત્તમ કાઢી નાખો ઓર્ડર લોગ ઑ ઓપરેશન પણ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:00)

તેથી, આપણે જે કર્યું છે તે છે, આપણે બતાવ્યું છે કે એક હિપ્ વાસ્તવમાં મહત્તમ કાઢી નાખે છે અને લૉગ એન ટાઈમમાં દાખલ કરે છે. હવે, અને હિપ્ વિશેની બીજી ઘણી સરસ સંપત્તિ એ છે કે આપણે ખરેખર ખૂબ જ જટિલ વૃક્ષની લાક્ષણિકતાને જાળવવાની જરૂર નથી, અમે આ કરવા માટે અરેમાં ખરેખર હિપ્ કરી શકીએ છીએ, અમે નિરીક્ષણ કરીએ છીએ કે આપણે એક ઢગલામાં બધા ગાંઠોને સિદ્ધ કરી શકીએ છીએ, અમે રુટમાં 0 થી શરુ કરો, પ્રથમ નોડ જે રુટ નીચે 1, બીજા બાળક 2 અને બીજું ભરો. તેથી, મારી પાસે સંખ્યા 0, 1, 2 છે, તેથી હું ખરેખર આ ઢગલાને એરેઝ હીપ તરીકે રજૂ કરી શકું છું જે આ 0, 1, 2 અને બીજું છે. હવે, આમાં દાવો એ છે કે હું પોઝિશન પર વ્યાખ્યાયિત છું, તેથી જો મારી પાસે કોઈ સ્થાન હોય, તો આનાં બાળકો 2 i પ્લસ 1 અને 2 i પ્લસ 2 તમે આ દરેક જગ્યાએ ચકાસી શકો છો. તેથી, જો મારે વાસ્તવમાં એક ઢગલા પર જવાનું છે અને હિપ પ્રોપર્ટી વિશે કંઈક પૂછવું છે, તો હું ફક્ત પોઝિશન પર ધ્યાન આપીશ, પછી i પોઝિશન 2 અને પ્લસ 1 અને 2 ની વત્તા 2 થી આગળ વધું છું. તેથી, સંપૂર્ણપણે એરેમાં એકલા એરેનો ઉપયોગ કરીને હું નોડનાં બાળકોને જોઈ શકું છું અને આ ઓપરેશનને રદ કરીને, જો i જે -1 ઓછા 2 ને જોઉં છું અને તેનો ફ્લોર લઈશ તો હું આ પાછો આવીશ. તો, બાળક સાથે 2 વત્તા 1 2 i પ્લસ 2 પછી માતાપિતા જે માઈનસ 1 થી 2 છે અને પછી તે ભિન્ન હોઈ શકે છે, તેથી પૂર્ણાંક ભાગો લો. તેથી, ફ્લોર એટલે કે જે માઈનસ 1 નો પૂર્ણાંક ભાગ 2 થી લઈ જાય છે. તેથી, આ j ઓછા અડધા નથી, તેથી તે જે માઈનસ 1 ને આખા 2 વડે જુએ છે. તેથી, ઉદાહરણ તરીકે, 12 જે માઈનસ 1 એ 11 છે, 11 દ્વારા 2 એ 5 છે અને તે અડધા માળ 5 છે, તેથી માતાપિતા 12 છે. તેથી, તમે ચકાસી શકો છો કે આ ફોર્મ છે, તેથી હું હવે એરેમાં મારા બધા મેપ મેનિપ્યુલેશન કરી શકું છું, જે ખૂબ અનુકૂળ છે. મારે ફક્ત એક એરે લખવાનું છે અને પછી જ્યારે પણ હું આ ક્રિયાઓ કરીશ, જેમાં ઢગલાને ઉપર અને નીચે જવાનો સમાવેશ થાય છે ત્યારે હું ફક્ત 2 આઇ પ્લસ 1 2 આઇ પ્લસ 2 સૂત્રનો ઉપયોગ કરીશ જે જે માઈનસ 1 થી 2 સૂત્રના આ ફ્લોરનો ઉપયોગ કરે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 18:08)

તો, આપણે આ સંપૂર્ણ પ્રક્રિયા કેવી રીતે શરૂ કરીશું, આપણે મૂલ્યોના સમૂહમાંથી એક હિપ કેવી રીતે બનાવવી જોઈએ. તેથી, ખૂબ જ નિષ્ક્રીય વ્યૂહરચના નીચે પ્રમાણે કાર્ય કરશે અને મૂલ્યો n ના મૂલ્યો x 1 થી x n ને આપવામાં આવશે. તેથી, હું ખાલી ઢગલાથી પ્રારંભ કરું છું અને પછી હું x 1 શામેલ કરું છું, તેથી મારી પાસે કદ 1 નું ઢગલું છે, પછી હું x 2 શામેલ કરું છું, તેથી હવે હું કદ 2 ની હિપ અને તેથી આગળ. તેથી, હું એન દાખલ કરું છું અને પ્રત્યેક શામેલ મોટાભાગે લોગ એન ટાઈમ લે છે, તેથી અમે ઓછા સમય લેશે, જો હું તેને દાખલ કરું તો હું લોગનો સમય લેશે

((સમયનોસંદર્ભ: 18:39)),

તેથી માટે પણ ચાલો આપણે ઉચ્ચ બાઉન્ડ્રી તરીકે લોગ એન. તેથી, જો હું આ એન તત્વોને દાખલ કરું તો હું પણ હિપ અને એન લોગ એન સમય બનાવશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:49)

હવે, હકીકતમાં તે બહાર આવ્યું છે કે આ કરવા માટે વધુ સારો માર્ગ છે, તેથી જો હું કોઈપણ એરે પર જોઉં છું તો હું આને એક ઢગલા તરીકે વિચારી શકું છું, હું કલ્પના કરી શકું છું કે આ આદેશ આપ્યો છે. કારણ કે, દરેક એરે એક હીપ અર્થઘટન તરીકે, હું કલ્પના કરી શકું છું કે આ રીતે એરે દેખાય છે, જો હું તેના વિશે અચાનક વિચારીશ, તો તે હિપ પ્રોપર્ટીને સંતુષ્ટ કરતું નથી. પરંતુ, જો હું હિપ ગોઠવીશ તો તે જોશે. હવે, પાંદડાના સ્તરે જે કંઈપણ છે તે તપાસ કરવાની જરૂર નથી, કારણ કે તેમાં કોઈ સંતાન નથી, બધા પાંદડાઓ તૃતીયાંશથી ભૌતિક સંપત્તિને સંતુષ્ટ કરે છે. તેથી, મારે પહેલાનાં સ્તરે ફક્ત ફિક્સિંગ કરવાનું શરૂ કરવું પડશે, તેથી હું આ માટે પાછું કામ કરું છું. તેથી, હું અહીં આવ્યો છું અને x 3 હું હીપ પ્રોપર્ટીને સંબંધિત રીતે ઠીક કરું છું, તે બાળકો છે, જ્યારે x 2 માં પ્રત્યેક બાળકો સંબંધિત મિલકતને ઠીક કરવા, પ્રક્રિયામાં કંઈક માત્ર એક સ્તર સુધી ઉપર અને નીચે, પછી હું આવીશ એક્સ 1 અને હું તેને ઠીક કરીશ, તે સમસ્યા છે, હવે આ 2 સ્તરો શામેલ હોઈ શકે છે. તેથી, દરેક સ્તર કે માઈનસ 1 કે માઈનસ 2 માટે તે. તેથી, પાંદડા સ્તરે સ્તર કે કે ઓછા સ્તર 1 કે માઈનસ 2 પર હોય છે, અમે હીપ પ્રોપર્ટીને ઠીક કરીએ છીએ. તેથી, જ્યારે આપણે હિપ પ્રોપર્ટીને ફિક્સ કરવાનું નક્કી કરીએ છીએ, એટલે કે આપણે નીચે જઈને પર્ણ સુધી ચાલીને મહત્તમ કાઢી નાખવા માટે કર્યું છે. તેથી, આપણે જે સ્તર પર જઈએ છીએ તે આ પાથની લંબાઈ 1 થી વધે છે, પરંતુ કારણ કે આપણે જેટલા સ્તર નીચે જઈએ તે બમણું છે તેમ તેમ આપણે વધીએ છીએ. તેથી, નોડ્સની સંખ્યા જેના માટે આપણે આ વધારાની લંબાઈની પાથ તપાસવી પડશે તે હકીકત પંક્તિ દ્વારા નીચે જાય છે. તેથી, હવે જો તમે વિશ્લેષણ કરો છો, તો આપણે બરાબર કરવા જઈશું નહીં, આ પ્રક્રિયામાં તે બહાર આવે છે કે તમારે હીપ બનાવવા માટેના અપડેટ્સની સંખ્યા વાસ્તવમાં માત્ર ઓર્ડર N છે. તેથી, જો તમે આ બોટમઅપ હિપીફિકેશન નો ઉપયોગ કરો છો, પછી તે ઓર્ડર એન પ્રક્રિયા હશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 20:34)

તો, ચાલો ગાઈડર ચિત્ર મેળવવા માટે શું ચાલી રહ્યું છે, તો ચાલો ધારીએ કે મારી પાસે સૂચિ ધરાવતી 15 તત્વ છે અને આપણે ખરેખર તેના દ્વારા આમ કરી શકીએ છીએ અને ક્લેમ એ છે કે આ n માઈનસ 2 નોડ n દ્વારા 2 નોડ્સ, જે આશરે છે તે વાસ્તવમાં 15 માંથી 8 છે, તે પહેલાથી જ હિપ પ્રોપર્ટીને સંતુષ્ટ કરે છે, તેથી આ કરવાનું કંઈ નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 20:50)

તેથી, હું એક સ્તર ઉપર જાઉં છું અને હું આને ઠીક કરું છું, જ્યારે હું આને ઠીક કરું છું ત્યારે મને તે 4 નોડ્સ માટે કરવું પડશે અને તેમાંના દરેકને સમારકામમાં મોટાભાગના અથવા કોઈ સ્વેપ્સમાં એક સ્વેપ શામેલ હશે, સૌથી ખરાબ કેસ

((સમયનોસંદર્ભ લો: 21:02))

તેથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 21:05)

પછી, હું એક સ્તર ઉપર જઈશ અને હવે આ દરેક ગાંઠો માટે મને સંભવતઃ લંબાઈ 2 ના 2 પાથ નીચે જવા પડશે. તેથી, તેમાંના દરેકમાં ઊંચાઈ 2 સમારકામ શામેલ હશે જે 2 છે 4 નોડ્સથી બધા કરતાં વધુ સારી વસ્તુનું પગલું હું 2 નોડ્સ પર ગયો છું. તેથી, ફક્ત અર્ધ મિનિ ગાંઠો છે જેને ફક્ત એક પગથિયું નોડની જરૂર છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 21:25)

અને પછી છેલ્લે, જ્યારે હું રુટ પર જાઉં ત્યારે મારે એક પ્રકારનું ઠીક કરવું પડશે જેમાં છેલ્લા પાન પર સ્વેપ કરવું આવશ્યક છે, તેથી 3 સ્વેપ્સ. પરંતુ, આ એક માત્ર નોડ છે જે આ કરે છે, તેથી ત્યાંથી વેપારનો નોડોનો જથ્થો છૂટી રહ્યો છે અને લંબાઈ ફક્ત એક જ વધી રહી છે તે બતાવે છે કે આ સમગ્ર ઑપરેંડને ફક્ત એન સમયની જરૂર છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 21:45)

તેથી, સારાંશ માટે આપણે જોયું છે કે વિશિષ્ટ સંતુલન વૃક્ષોનો ઉપયોગ કરીને પ્રાથમિકતા કતારને અમલમાં મૂકવા માટે, આ વૃક્ષમાં મહત્તમ અથવા લઘુગણક શામેલ કરો અને કાઢી નાખો, અમે આ તળિયેનો ઉપયોગ વાસ્તવમાં બિલ્ડ કરવા માટે યોગ્ય બનાવી શકીએ છીએ. હિપ્ અને ઓર્ડર એન સમય. અને સૌથી વધુ ઉપયોગી તે છે કે આ હિપ્ ખરેખર એરે તરીકે સરળતાથી ઉપયોગમાં લઈ શકાય છે, હવે એક વસ્તુ જે આપણે કરી શકીએ છીએ તે ઢગલાને અવગણવાની છે. તેથી, આપણે કહી શકીએ કે જ્યારે પણ v1, v2 અને v3 જુઓ ત્યારે આપણે v1 ને v2 અને v બંને કરતા નાના હોવા જોઈએ. તેથી, આને મિન હીપ કહેવામાં આવે છે, આપણે શું કર્યું છે, તેથી માટે મહત્તમ હિપ્ છે. તેથી, કેટલીકવાર તમે સૌથી નાની પ્રાધાન્યતાને ટ્રૅક રાખવા અને નાની પ્રાધાન્યતા આઈટમને દૂર કરવા માંગો છો, ફક્ત ઉદાહરણ તરીકે, તમે પરીક્ષામાં લોકોને ક્રમ આપો છો. તેથી, જો તમે કોઈ સ્પર્ધાત્મક પરીક્ષામાં હોવ તો, કોઈ પણ ક્રમાંક ઓછી હોવ તો તેને પ્રાધાન્ય આપો. તેથી, જો તમારી પાસે ક્રમ 1 હોય તો તમારી પાસે ઉચ્ચતમ પ્રાધાન્યતા છે. તેથી, નાની સંખ્યાઓ વિશે વિચારવું એ સ્વાભાવિક છે, તે ઉચ્ચ પ્રાથમિકતા છે. તેથી, તમારે ઘણું બધું કરવું પડશે, આપણે માત્ર લઘુતમ હોવા માટે હીપ રુટને બદલવું પડશે. તેથી, દરેક નોડ બે બાળકો કરતા નાના છે અને પછી આપણે જે કર્યું છે તે બધું બરાબર કાર્ય કરશે. તેથી, અમારી પાસે બે પ્રકારનાં ઢગલા છે, તમારી પાસે મહત્તમ હિપ્ છે અને તમારી પાસે મહત્તમ હિપ્ છે અને મહત્તમ

ઢગલામાં તે બધા જુદા છે અને લઘુ ઢગલા નોડ્સ પર લિપ્ છે અને તમે અમલમાં આવતાં ઑપરેશનને કાઢી નાખો તે મહત્તમ છે અને મહત્તમ કાઢી નાખો .

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 05
મૉડ્યૂલ - 04
લેક્ચર – 36

હિપ્સની અમારી ચર્ચાને પૂર્ણ કરવા માટે આપણે હિપ્સ અને ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમનો ઉપયોગ કેવી રીતે કરવો તે અને વિવિધ પ્રકારની સૉર્ટિંગ માટે હિપ્સનો ઉપયોગ કેવી રીતે કરવો તે પણ જોશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:10)

તેથી, યાદ રાખો કે હિપ્સ પ્રાયોરિટી ક્યુ (priority queues)નો વૃક્ષ અમલીકરણ છે જેમાં દાખલ કરો અને કાઢી નાખો અથવા જટિલતા લોગ એન બંને. તમે એન ટાઈમમાં હીપ તળિયે બનાવી શકો છો અને તમે તેને રજૂ કરી શકો છો એક વૃક્ષ અને તેથી તમે તેને બદલી શકો છો, માફ કરશો, તમે એરેમાં એક વૃક્ષનું પ્રતિનિધિત્વ કરી શકો છો, જેથી તમે તેને ખૂબ સરળતાથી ઉપયોગમાં લઈ શકો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:30)

તેથી, ચાલો આપણે ડિજસ્કસ્ટ્રાના એલ્ગોરિધમ પર પાછા જઈએ. તેથી, ડિજસ્ક્રાસ્ટના અલ્ગોરિધમમાં, આપણે પ્રારંભિક શિરચ્છેદથી પ્રારંભ કરીએ છીએ અને અમે આ શિરોબિંદુને દઝાડતું રાખીએ છીએ, તેથી, આપણે તેમની અંતર મુજબ વિનિમય શિરોબિંદુ રાખીએ છીએ. તેથી, શરૂઆતમાં, આપણે અંતરને દરેક વસ્તુ માટે અનંત હોવાનું સેટ કર્યું, . અને આપણે શરૂઆતના બિંદુનો ઉપયોગ, શરૂઆતના અંતરની અંતરને 0 તરીકે સેટ કરીને, અને પછી આપણે સૌથી નાનું અવલોકન કરેલું, ઉભા, વર્ટેક્સ શોધીએ છીએ, તેને મુલાકાત લેવા માટે સેટ કરીએ છીએ અને તેના પ્રત્યેક પાડોશીઓની અંતરને ફરીથી ઠપકો આપીએ છીએ. તેથી, લૂપના આ ભાગમાં ખરેખર મુશ્કેલીઓ હતી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:03)

તેથી, ઓછામાં ઓછા અંતર સાથે j શોધવા માટે, અંતરાયો સૌથી પહેલા છે. તેથી, નિષ્ક્રીય અમલીકરણ આપણને ઓર્ડર એન ટાઈમ લેશે કારણ કે આપણે બધા અવ્યવસ્થિત શિરોબિંદુઓને સ્કેન કરવું પડશે, જે કોઈ પણ અંતરના કોઈ પણ ક્રમમાં નથી અને તેમાંના ઓછામાં ઓછા, છે. તેથી, અત્યાર સુધી આપણે જે ચર્ચા કરી છે તેમાંથી સ્પષ્ટ દેખાય છે તે છે કે આપણે એક હિપ્સ તરીકે અંતર જાળવી રાખવું જોઈએ અને મહત્તમ કાઢી નાખો અથવા આ કિસ્સામાં વાસ્તવમાં મિનિ કાઢી નાખવું જોઈએ, કારણ કે આપણે એક મિનિટ હિપ્સ જોઈએ છે, ઠીક છે. તેથી, કાઢી નાંખો મિનિટે ઘટાડો, પણ પછી આપણે અંતરને પુનર્વિચાર કરવા પણ પડશે. તેથી, અંતરને પુન: કમ્પ્યૂટર કરવાનો અર્થ છે જે જેનો દરેક પાડોશી k નું પરીક્ષણ કરવું. તેથી, ખાતરી કરવા માટે, આપણે પાડોશીઓને અસરકારક રીતે શોધી શકીએ છીએ, અમે અડજાસનસી (adjacency) સૂચિનો ઉપયોગ કરી શકીએ છીએ જેથી અમે અડજાસનસી (adjacency)ને મેટ્રિક્સ પરની સંપૂર્ણ

પંક્તિની યોજના ઘડીએ સમય બગાડીશું નહીં. પરંતુ અહીં અવરોધ એ છે કે, આપણે અંતરને અપડેટ કરવાની જરૂર છે, એટલે કે, આપણે તેને લિપ્સ અને મૂલ્યમાં ફેરફાર કરવાની જરૂર છે, તેથી આપણે હીપ મૂલ્યોને અપડેટ કરવાની જરૂર છે. હવે, આપણે ખરેખર હિપ મૂલ્યોને કેવી રીતે અપડેટ કરવું તે જોયું નથી, અમે ફક્ત શામેલ કર્યું છે અને મિની કાઢી નાખો અને મહત્તમ કાઢી નાખો. તો, કામ કેવી રીતે અપડેટ કરે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:12)

તેથી, ધારો કે આપણે આ વેલ્યુ 12 થી 44 માં બદલવું છે, બરાબર. જો આપણે, આ મૂલ્ય વધારીએ, તો તેના બાળકોના સંદર્ભમાં તે કોઈ પણ નાનું, બરાબર નહીં મેળવી શકે. તેથી, જો 12 10 અને 11 કરતા મોટો હોય, તો કોઈપણ મોટું મૂલ્ય 10 અને 11 કરતા પણ વધારે હશે. તેથી, અમે તેને 44 બનાવીએ છીએ. આપણે નજર રાખવાની જરૂર નથી, પરંતુ આપણે તેને વધુ મોટી બનાવીએ છીએ, તે કરતાં મોટી થઈ શકે છે તેના પેરેન્ટ અને હકીકતમાં, તે થાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:39)

તેથી, જો આપણે 24, 12 દ્વારા 44 ને બદલીએ, તો પછી અમને લાગે છે કે ઉપરના લિપ્સબંધ ઉલ્લંઘન છે. તેથી, હવે, આપણે આનો બરાબર ઉપચાર કરીએ છીએ જેમ આપણે દાખલ કર્યો છે. અમે પિતૃ તરફ જુએ છે. તેથી, અમે આ ઉલ્લંઘન ઉપરોક્ત ઠીક કરીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:51)

તેથી, હવે આપણે જોઈએ છીએ અને અમને અહીં 44 મળ્યા છે અને હવે આપણે તપાસવું પડશે કે તે તેના પેરેન્ટ સાથે કંઈપણનું ઉલ્લંઘન કરે છે, અને તે કરે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 03:00)

તેથી, તેનું વિનિમય કરો અને છેલ્લે, જ્યારે તે બંધ થાય, ત્યારે ક્યાં તો વધુ ઉલ્લંઘન ન થાય અથવા જ્યારે આપણે રુટ સુધી પહોંચીએ ત્યારે આપણે જે આગળ વધી શકીશું નહીં. તેથી, મૂલ્યમાં વધારો કરવાથી તમે ઉલ્લંઘનને ઠીક કરો છો, કારણ કે મૂલ્યમાં વધારો કરવાથી તમે તમારા બાળકો કરતા નાના થઈ શકતા નથી, પરંતુ તમે તમારા પેરેન્ટ કરતા મોટા થઈ શકો છો.

(સ્લાઈડસમયનો સંદર્ભ લો: 03:17)

અન્ય પ્રકારનું પરિવર્તન મૂલ્ય ઘટાડવાનું છે. તેથી, ધારો કે હું આ 33 લઈશ અને હું તેને 9 બનાવીશ. હવે, ફરીથી તે જ તર્ક દ્વારા તે 33 હતું, 44 કરતા નાની હતી, 9 પણ 44 કરતા નાની હશે. 33 કરતા નાના કોઈપણ મૂલ્ય ઉલ્લંઘન કરી શકશે નહીં. તેથી, હું જમણી તરફ જોવું જોઈએ. તો, જો હું તેને 9 સુધી નીચે લાવીશ, તો નંબર 33 અને 24 ની વચ્ચેની સમસ્યા છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 03:36)

તો, જો મારું નવું મૂલ્ય છે, તો મારે તેના બે બાળકો સાથે તપાસ કરવી પડશે અને સૌથી મોટી એક અપ આ કિસ્સામાં તે 24 છે, બરાબર.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:43)

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:48)

તેથી, હવે અહીં આવીને, હવે હું તેના બે બાળકોને તપાસવાની જરૂર છે. અને હવે, સૌથી મોટી એક અપ લો કે જેમાં તે 11 છે, અને તેથી તે કેવી રીતે છે, બરાબર. તેથી, જ્યારે હું મૂલ્યોને અપડેટ કરું છું અને મૂલ્ય ઘટાડે છે, ત્યારે મને ઉલ્લંઘનોને નીચે ઠીક કરવું પડશે કારણ કે મૂલ્ય ઘટાડવા તેના પેરેન્ટ કરતાં તેને વધુ મોટું બનાવી શકતું નથી, પરંતુ તે તેના બે બાળકોમાંના એક કરતાં નાના બનાવી શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:04)

તેથી, જો તમે ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ જુઓ છો, તો તે જે રીતે કાર્ય કરે છે તે છે, આપણે શિરોબિંદુ j લઈએ અને કહીએ, આ અંતરને અપડેટ કરો. તેથી, આપણે કેટલાક વર્ટેક્સ જેની અંતર અપડેટ કરવી પડશે, જે ક્યાંક હિપ્સમાં છે. તેથી, આપણું અગાઉનું ઉદાહરણ જે દર્શાવે છે તે છે, જો આપણે, આંગળીમાં નળી પર અમારી આંગળી મૂકીએ અને તેનું મૂલ્ય બદલીએ, તો આપણે જાણીએ છીએ કે હિપ્સને કેવી રીતે ગોઠવવું. પરંતુ જીપમાં ક્યાં છે તે આપણે ક્યાં શોધી શકીએ? તેથી, જ્યાં j સ્થિત થયેલ છે? તેથી, અમને આ વધારાની માહિતીની અલગ રાખવાની જરૂર છે. તેથી, આપણે નોડ્સથી પોઈન્ટ કરતા બે નવા એરે રાખીએ છીએ, જે એક હિપ્સને સમાપ્ત કરવા માટે છે, જે 0 ની લઘુતમ 1 છે અને ઊલટું 0 થી N નો નોડ પર છે. તેથી, અહીં આ ચિત્રમાં, જમણી બાજુ, આ બે વસ્તુઓ વિવિધ રંગોમાં દોરવામાં આવે છે. તેથી, નોડ સામેના લાલ લેબલ્સ શિરચ્છેદ સૂચવે છે. તેથી, 44 વર્ટીક્સ 8 ની અંતરનું પ્રતિનિધિત્વ કરે છે અને તે સૂચવવા માટે કે આપણી પાસે એરે છે, તે કહે છે કે નોડ

((સમયનોસંદર્ભ: 05:06)),

આલેખમાં નોડ 8 એ હિપ્સમાં 0 વક્ક છે. એ જ રીતે, જો હું આને જોઉં તો તે કહે છે, આ કર્ણ 3 છે. તો, મારા ગ્રાફમાં કર્ણ 3 એ મારા હિપ્સમાં 6 નોડ છે. તેનાથી વિપરીત, જો હું હિપ્સમાં છું અને હું છું કે હું નોડ 4 પર છું, જે લૂપનું પ્રતિનિધિત્વ કરે છે, તો પછી કયું વાક્ય આને અનુરૂપ છે? તેથી, તે કહે છે કે, હિપ્સમાં નોડ 4 કર્ણ 5 થી સંબંધિત છે. તેથી, હિપ્સમાંથી નોડના ઈન્ડેક્સને હિપ્સમાં આપવામાં આવે છે, જે ગ્રાફમાં નોડ તે અનુરૂપ છે? તેથી, અમારી પાસે વધારાની વસ્તુઓ છે જે અમે સેટ કરી છે અને જ્યારે આપણે સ્વૅપ કરીએ છીએ અથવા

((સમયનોસંદર્ભ લો: 05:49)),

ત્યારે જ આને અપડેટ કરવું પડશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:50)

તેથી, હવે માટેદાખલા તરીકે, અમને લાગે છે કે અમારી પાસે આ પાછલી વસ્તુ છે અને આપણે આ 33 ને 9 માં બનાવવું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:57)

તો, અમારું લક્ષ્ય 33 થી 9 ઘટાડવાનું છે, જે આપણે પહેલાનાં સેટમાં કર્યું હતું. હવે, ચાલો આપણે આમ કરીએ. તેથી, હવે, જ્યારે હું આ કરું છું, ત્યારે મારે તેના બે બાળકોને હિપ્સમાં નીચે જવાની જરૂર છે અને તે સ્વીકારવું જોઈએ કે આ 24 બદલવું આવશ્યક છે. હવે, 24 થી બદલવું જ પડશે, મારે તેને અપડેટ કરવું તે પણ જાણવું જોઈએ. તો, 24 એ હિપ્સમાં નોડ 3 છે. તેથી, નોડ 3 એ વર્ટેક્સ 2 છે. તો, મને અહીં શિરોબિંદુ 2 પર જવું પડશે. તેથી, મને આ એન્ટ્રીઓને અપડેટ કરવાની જરૂર છે. તેથી, મને 6 અને 1, અને 3 અને 2 અપડેટ કરવાની જરૂર છે. તેથી, જ્યારે હું 9 અને 24 નું વિનિમય કરું છું, ત્યારે મને 6 અને 2 નું પણ વિનિમય કરવું આવશ્યક છે. મારે કહેવું જોઈએ કે, હવે મારા ગ્રાફમાં વર્ટેક્સ નોડ, વર્ટેક્સ 2 છે. મારા ગ્રાફમાં વૃક્ષ અને ગાંઠ 6 માં નોડ 1 પર હવે વૃક્ષમાં નોડ 3 પર છે. તેનાથી વિપરીત, વૃક્ષમાં નોડ 1, પોઈન્ટમાં પોઈન્ટ, નોડ 3 ટ્રી પોઈન્ટ્સમાં વેરટેક્સ, વર્ટેક્સ 2 તરફ નિર્દેશ કરે છે, નોડ 3 પોઈંટથી વેરટેક્સ

((સમયનોસંદર્ભ: 06:55)),

(સ્લાઈડસમયનો સંદર્ભ લો: 06:57)

તેથી, આ સુધારા પછી, મૂળભૂત રીતે, આ બંને મૂલ્યો પહેલાં જે છે તેમાંથી વિનિમય પામ્યા છે, બરાબર. તેથી, આ સ્તરે સ્વેપ કરવા ઉપરાંત, મને પણ સ્વીકૃત કરવું પડશે કે આ સ્વેપ છે. હવે, હું એક વધુ સ્વેપ કરું છું. કારણ કે 9 અને 11 નું સ્વૅપ્સ હતું, હવે હું 3, 6 અને 8, 9 માટે પ્રવેશો સ્વેપ કરીશ. અને તે જ રીતે, 6, 3 અને 9, 8, . તેથી, એન્ટ્રીઝ વર્ટીક્સ 6 અને વર્ટેક્સ 9 અને નોડ 3 અને નોડ 8 સાથે જમણી બાજુએ છે. તેથી, વર્ટેક્સ 6, વર્ટેક્સ 9, નોડ 3 અને નોડ 8, આને સ્વેપ કરવું પડશે, .

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:26)

તેથી, તેને વધારાની એરેમાં રાખીને, , હું આ અપડેટ્સને ખૂબ જ સરળતાથી કરી શકું છું, કારણ કે મારી પાસે હેપ ઈન્ડેક્સ અને નોડ ઈન્ડેક્સ વચ્ચે પાછળ અને પાછળ જવાનો માર્ગ છે. અને જ્યાં સુધી હું આ કરું નહીં, હું ખરેખર ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમનો ઉપયોગ કરી શકતો નથી કારણ કે ડીજેક્સ્ટ્રા(Dijkstra's)ના એલ્ગોરિધમ

અમને હીપમાં મૂલ્ય અપડેટ કરવા માટે કહેશે, પરંતુ મને જાણવાની જરૂર છે કે હેપમાં મને કઈ મૂલ્ય અપડેટ કરવાની જરૂર છે. સુધારો, આ ઉપર અથવા નીચેના મેનીપ્યુલેશનનો ઉપયોગ કરીને બરાબર દાખલ અથવા કાઢી નાખવા જેવા કરી શકાય છે, પરંતુ વાસ્તવિક સમસ્યા એ છે કે જ્યાં અપડેટ પ્રારંભ થાય છે તે ઓળખી શકાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:55)

તેથી, જેમ આપણે પહેલા જોયું તેમ, હવે આપણે આ સુધારા ઓપરેશન સાથે આ લિપ્સનો ઉપયોગ કરી શકીએ છીએ અને ન્યૂનતમ ટાઈમ વર્ટેક્સ અને લોગ એન ટાઈમ શોધી શકીએ છીએ. અને કારણ કે અમારી પાસે અડજાસનસી (adjacency)ને સૂચિ છે, બરાબર, બર્ન ટાઈમ્સને અપડેટ કરતી લૂપ્સ એકંદરે લોગ એન ટાઈમ લે છે, અને ત્યાં સંપૂર્ણપણે ઓર્ડર એમ ધાર છે. તેથી, એકંદરે, લઘુત્તમ એમ ટાઈમ્સને અપડેટ કરવા માટે ન્યૂનતમ એન ટાઈમ્સ અને એમ લોગ n શોધવા માટે, આપણે અપડેટ કરવા માટે n લોગ n મેળવીએ છીએ. તેથી, એન પ્લસ એન લોગ n. તમે પ્રાઈમના અલ્ગોરિધમનો સમાન વ્યૂહરચનાનો ઉપયોગ કરી શકો છો. પ્રાઈમના અલ્ગોરિધમમાં માત્ર એક જ અંતર છે, અંતરની કલ્પના સમાન નથી. અમે અંતરને સંગ્રહિત કરતા નથી, અમે વાસ્તવિક ધારની કિંમત પર ધ્યાન આપીએ છીએ. પરંતુ હજી પણ, આપણે વર્તમાન વૃક્ષ તરફ નોડને જોડીને લઘુતમ ખર્ચ ધારને આકર્ષવાની જરૂર છે અને પછી આ ધારને વૃક્ષ પર ઉમેરવા પછી, આપણે આ વસ્તુઓને અપડેટ કરવાની જરૂર છે, . તેથી, અપડેટ્સ સાથે એક મિનિટ લિપ્સનો ઉપયોગ કરવો તે જ વિચાર અમે તેનો ઉપયોગ કરીશું. તેથી, પ્રાઈઝની એલ્ગોરિધમ ડિજસ્કસ્ટ્રાના એલ્ગોરિધમ જેવી જ જટિલતાને પણ ઘટાડે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 08:55)

તેથી, આપણે લિપ્સ છોડતા પહેલા, ચાલો જોઈએ કે સૉર્ટ કરવા માટે લિપ્સનો ઉપયોગ કેવી રીતે કરવો. તેથી, આપણે મૂલ્યોની સૂચિને સૉર્ટ કરવા માંગીએ છીએ. તો, આપણે શું કરી શકીએ, આપણે પહેલા એક લિપ્સ બનાવી શકીએ, . તેથી, હવે આપણે મૂલ્યોના કેટલાક અનિશ્ચિત અનુક્રમથી શરૂ કરીએ છીએ, x 1 થી xn, પછી આપણે એક લિપ્સ બનાવીએ છીએ અને આપણે સંભવતઃ xi 1, xi 2, xin ની કોઈ રીતમાં પુનઃક્રમાંકિત થઈએ છીએ, પરંતુ આ સૉર્ટ કરેલ રીતે નથી, આ તે છે

((સમયનોસંદર્ભ લો: 09:21)),

તે છે, આપણે જાણીએ છીએ કે મહત્તમ ડાબી બાજુએ છે અને તેથી આગળ. હવે, હું તેને મહત્તમ કાઢી નાખીશ અને હું આ વ્યક્તિને બહાર કાઢીશ, . તેથી, હું જાણું છું કે આ મહત્તમ છે. પછી, આ પછી, આ કંઈક આગળ આવશે. પછીના બિંદુએ, તે આવશે, અને પછી હું બીજા મહત્તમ અને તેથી જ મળશે, . તેથી, જો હું મહત્તમ n વખત કાઢી નાંખો, સ્પષ્ટ રીતે, દરેક બિંદુએ મને પછીની મહત્તમ મળશે. તેથી, હું ઉતરતા ક્રમમાં વૃક્ષમાંથી ઘટકો કાઢું છું, અને તેથી મને સૉર્ટ કરેલ આઉટપુટ મળે છે. હું સેટને રિવર્સ કરી શકું છું, હું ચઢતા ક્રમમાં રહી શકું છું, તે વાંધો નથી, પણ હું ફક્ત ચોક્કસ ક્રમમાં તત્વોને બહાર કાઢું છું. અને તેથી આ એક નાના સોર્ટિંગ અલ્ગોરિધમનો છે. હવે, દરેક નિષ્કર્ષણ લોગ એન ટાઈમ લે છે કારણ કે તે કાઢી નાખવાનો સમય મહત્તમ, , અથવા મહત્તમ કાઢી નાખો અથવા કાઢી નાખવા માટેનો પ્રકાર છે જેનો ઉપયોગ કરવા માટે કયા પ્રકારનાં લિપ્સને આધારે. તેથી, તમે એન આવા નિષ્કર્ષણ કરો. તેથી, લોગ એન ટાઈમમાં તમે

તત્વોને ક્રમમાં ગોઠવી શકો છો અને તેમાં મૂકવા માટે, તમે ફક્ત ઓર્ડર n સમય લીધો હતો. તેથી, એકંદરે અમે એન લોગ એન સમય સૉર્ટ.

(સ્લાઈડસમયનો સંદર્ભ લો: 10:19)

ત્યાં એક નાની ગૂંચવણ છે જે તમે આ વિશે પૂછી શકો છો. પ્રશ્ન એ છે કે, આ કિંમતો ક્યાં જાય છે, સાચું. તેથી, શરૂઆતમાં, પ્રથમ પુનરાવર્તનમાં, જો મારી લિપ્સ મારી જેમ દેખાય, તો તે મારી મહત્તમ છે. તેથી, એવું લાગે છે કે મારે નવી સૂચિમાં મૂકવું પડશે, પરંતુ આ પગલાં પછી શું થાય છે, તે આ મૂલ્ય અહીં શામેલ થઈ ગયું છે અને તે પછી હું તેને ઠીક કરું તો, લિપ્સ દ્વારા તે નીચે ઉતરે છે. તો, જ્યારે હું રુટ પર વેલ્યુ રદ્દ કરું છું, ત્યારે હું છેલ્લો પર્ણ લઈશ, તેનું મૂલ્ય રુટમાં મુકું અને પછી હું તેને નીચે મુકું જેથી બધી હેપ પ્રોપર્ટી

((સમય:10:54)).

અને આ પછી, હવે મારો લિપ્સ આ જ દેખાય છે કારણ કે આ મૂલ્ય જતું રહ્યું છે. તેથી, મારું હીપ x માઈનસ n માઈનસ 1 થી સમાપ્ત થાય છે. તેથી, મારી પાસે મારા એરેમાં એક સ્થાન છે, જે હવે પછીના ડિલીટ મેક્સ માટે વધુ ઉપયોગમાં નથી રહ્યું, કારણ કે પછીના ડિલીટ મેક્સમાં ફક્ત n ઓછા 1 ઑપરેશન શામેલ છે. તો, તેથી, હું, વાસ્તવમાં, આ બિંદુએ જઈ શકું છું અને આ મૂલ્યને આ સ્થાને લિપ્સમાં મૂકી શકું છું અને ખાતરી આપી શકું છું કે તે વધારે નથી. તો, હવે શું થશે, મારી પાસે અહીં મહત્તમ આવવાનું રહેશે, પછી મારી પાસે આગામી બીજા મેક્સ અહીં આવી જશે. તેથી, જો હું મહત્તમ કાઢી નાંખું ચાલુ રાખું છું, તો હું ઉતરતા ક્રમમાં મૂલ્યની જમાણી બાજુથી પ્રચાર કરી રહ્યો છું અને આખરે, તે વાસ્તવમાં ચઢતા ક્રમમાં ગોઠવાયેલા ક્રમને સમાપ્ત કરી શકે છે. તેથી, આ મને એક સ્થાને ઓર્ડર n લોગ n સ્વેપ આપે છે, સાચું, ખાતરી કરીને, કે જ્યારે હું તેને ફેંકી દેવા અથવા નવી સૂચિમાં મૂકવાને બદલે મૂલ્ય કાઢી નાખું છું, તો હું તેને ખાલી જગ્યામાં મુકું છું હવે એક મૂલ્ય દ્વારા લિપ્સ શોધીને, અને પછી હું આપોઆપ આઉટપુટને સૉર્ટ કરેલા માર્ગમાં મેળવી શકું છું, . તેથી, લિપ્સનો ઉપયોગ ખૂબ અલગ પ્રકારની ઓર્ડર n લોગ n ટૂંકા સ્થાને કરવા માટે થઈ શકે છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 6
લેક્ચર - 37
ડિવાઈડ અને કોનકોર(Divide and Conquer): ઊલટસૂલટ ગણતરીઓ

ચાલો આપણે પાછા જઈએ અને ડિવાઈડ અને કોનકોર(Divide and Conquer) ફરી જોઈએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:06)

તેથી, ડિવાઈડ અને કોનકોર(Divide and Conquer)યાદ કરો જેમાં સમસ્યાને તોડવાથી વિવાદાસ્પદ ઉપપ્રોબ્લેમ્સમાં સમાવેશ થાય છે. પછી, અમે આ દરેક પેટાપ્રવાહને અલગથી હલ કરીએ છીએ અને પછી મૂળ સ્વરૂપના ઉકેલ માટે અમે તેમને કાર્યક્ષમ રીતે જોડીએ છીએ. તેથી, આપણે ડિવાઈડ અને કોનકોર(Divide and Conquer) બે ઉદાહરણો જોઈ છે, સૉર્ટ કરો વિલીનીકરણ એ ડિવાઈડ અને કોનકોર(Divide and Conquer)નો ક્લાસિક ઉદાહરણ છે, જ્યાં અમે સૂચિને સૉર્ટ કરવા માટે સૂચિને વિભાજિત કરવા માટે સમાન ભાગોમાં સૉર્ટ કરીએ છીએ. અમે આ બે ભાગોને અલગથી સોર્ટ કરીએ છીએ અને પછી, આપણે તેમને કાર્યક્ષમ રીતે મર્જ કરીએ છીએ, તે સૂચિબદ્ધ છે. ઝડપી સૉર્ટમાં જુદી જુદી વ્યૂહરચના છે, તે જે કરવાનો પ્રયાસ કરે છે તે મર્જિંગ પગલાને ટાળે છે. તેથી, તમે મૂળ સૂચિને ફરીથી ગોઠવો છો, જેથી તમારી પાસે પીવૉટના સંદર્ભમાં નીચલું અને ઉપલા ભાગ હોય. તેમને ફરીથી ગોઠવવું, તમે નીચેની અડધી અને ઉપલા અડધાને સ્વતંત્ર રીતે અને હવે સૉર્ટ કરી શકો છો, કારણ કે તેઓ પહેલેથી જ ફરીથી ગોઠવાયેલા છે, તમારે તેમને મર્જ કરવાની જરૂર નથી. તેથી, મૂળભૂત રીતે આ પેટા સમસ્યાઓ અને સબપ્રોબ્લેમ્સને સંયોજિત કરવા સાથે સંકળાયેલા ખર્ચને સેટ કરવા માટે ખર્ચ સામેલ છે. અને જો આ સેટ અપ ખર્ચ અને સંયોજન ખર્ચ કાર્યક્ષમ હોય, તો સંપૂર્ણ ઉકેલ તમને પ્રત્યક્ષ અભિગમ કરતાં વધુ સારી કંઈક આપે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:15)

તેથી, ચાલો નીચેની પરિસ્થિતિ તરફ ધ્યાન આપીએ, ઘણી વખત જ્યારે તમે કોઈ ઑનલાઈન સ્ટોર પર જાઓ છો, ત્યારે તમને ભલામણ મળે છે. ઉદાહરણ તરીકે, તે કહેશે, તમારા જેવા ગ્રાહકો કે જેમણે પુસ્તકોમાં રસ લીધો હતો અથવા આ ફોન ખરીદનારા ગ્રાહકો પણ હેડ ફોન્સની આ જોડી શોધી રહ્યાં છે. તેથી, આ સેવાઓ તમને તમારી પ્રોફાઈલના આધારે ભલામણ કરવામાં આવે છે, તમારી ઑનલાઈન સેવા તમને જે ગમે છે અને તમને ગમતી નથી તે વિશે કેટલીક પ્રોફાઈલ માહિતી જાળવે છે. તે તમને જે ગમે છે તેની તુલના કરે છે અને અન્ય લોકો સાથે ગમતું નથી અને તે સમાન શ્રેણીના લોકોને ઓળખે છે. અને પછી, તે કેટેગરીઝ દ્વારા પસંદ કરાયેલ ઉત્પાદનો અથવા સેવાઓની શોધ કરે છે, જેને તમે ત્રાણ અને પછી ભલામણ કરી નથી. તેથી, આવી ભલામણ પ્રણાલીમાં મૂળભૂત પગલું એ પ્રોફાઈલ્સની સરખામણી, એક વ્યક્તિ કેવી રીતે પસંદ કરે છે, અન્ય લોકોની સરખામણીમાં એક વ્યક્તિને કેવી રીતે પસંદ અને નાપસંદ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:13)

તેથી, આનો એક દાખલો જ્યારે તમારી પાસે મૂવીઝ અથવા પુસ્તકો જેવી વસ્તુઓ પર પસંદગી હોય છે, તો માની લો કે સમયની ઉપર, તમે કેટલીક વેબસાઈટ પર ગયા છો અને તમે જોયેલી મૂવીઝ વિશે તમારી પસંદગીઓ દાખલ કરો . તેથી, કહો કે ત્યાં પાંચ મૂવીઝ છે, ચાલો આપણે તેમને એ, બી, સી, ડી અને ઈ કહીએ જે તમે અને બીજા કોઈએ આ વેબસાઈટ પર ક્રમાંકિત કર્યા છે. તેથી, તમે સામાન્ય રીતે બે વ્યક્તિઓ કે જે વિવિધ રેન્કિંગ સાથે આવે છે. તેથી, કદાચ તમે પહેલા D અને E ને ક્રમ આપ્યો છે અને તમારા મિત્રે બી પ્રથમ અને ઈ છેલ્લે ક્રમાંકિત કર્યો છે. તેથી, તમે બન્ને સંમત થયા હતા કે ઈ સૌથી ખરાબ છે, પરંતુ તમે બી કે ડી વધુ સારા હતા કે નહીં તે અંગે અસંમત છો. હકીકતમાં, તમારા મિત્ર વિચારે છે કે ડી ખૂબ જ ખરાબ હતો, તે વાસ્તવમાં ફક્ત પછીની ઈ અને તળિયેથી છે. તેથી, ડી જે તમારો પ્રથમ હતો તે તમારા મિત્રો બની ગયા છે. તો, હવે આપણે શું કરી શકીએ તે છે કે આપણે રેન્કિંગના આવા બે સેટ્સ લઈ શકીએ છીએ અને પૂછીએ કે તે સમાન અથવા અસમર્થ છે. તેથી, આને માપવાનો એક રસ્તો એ છે કે તમે મૂવીઝનાં જોડીઓને કેવી રીતે ક્રમ આપો છો. તેથી, મૂવીઝના દરેક જોડી માટે, તમે તુલના કરી શકો છો કે તમે બીજા કરતાં વધુ સારા છો અને તમારા મિત્ર પણ કરે છે અથવા તમે કર્યું છે. તેથી, અહીં ઈન્સ્ટન્ટ્સ માટે, જો તમે બી અને સી જુઓ છો, તો તે તમને અને તમારા મિત્ર દ્વારા સમાન રીતે ક્રમાંકિત કરવામાં આવે છે. બીજી બાજુ, જો તમે ડી અને બી જોતા હોવ, તો એક કિસ્સામાં, તમે ઉપરના D અને D ઉપર તમારા મિત્ર ક્રમ બીને ક્રમ આપશો. તેથી, આપણે ખાસ કાળજી રાખતા નથી કે ત્યાં કેટલો ભાગ છે, ત્યાં કેટલી અન્ય વસ્તુઓ છે, અમે માત્ર બે વસ્તુઓની પસંદગી આપી રહ્યા છીએ, જે તમે પસંદ કરો છો, જે તમારા મિત્રને પસંદ કરે છે અને આપેલી સૂચિમાં ઉપલબ્ધ બધી પસંદગીઓની પસંદગીને પસંદ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:54)

તેથી, આપણે જે કરવાનો પ્રયાસ કરી રહ્યા છીએ તે અમે વિપરીતતાના સંદર્ભમાં વિપુલતાને માપીએ છીએ. તમે અને તમારા મિત્ર વચ્ચેની વિપરીત રીતમાં મૂવીઝ અથવા ક્રમ કેટલી જોડીઓ છે? તેથી, જો તમે અને તમારા મિત્ર એક જ ક્રમમાં મૂવીઝના દરેક જોડીને ક્રમ આપે છે, તો તમારા પ્રદર્શનના કુલ ક્રમ સમાન હોવું આવશ્યક છે. તેથી, જો શૂન્ય વ્યુત્ક્રમો હોય, તો તમારા મિત્રમાં તમારા સ્વાદમાં બરાબર સમાન છે અને રેન્કિંગ સમાન છે. બીજી તરફ, જો તમારી પાસે એન મૂવીઝ હોય, તો તમે કોઈ પસંદગી કરી શકો છો, n જોડી બનાવવા પસંદ કરો. તેથી, ચલચિત્રોના જુદા જુદા જોડીઓની સંખ્યા 2 ને પસંદ કરી છે જે n ની 1 ઘાત વડે 1 છે. તેથી, જો શક્ય હોય તો તમે તમારા મિત્ર સાથે અસંમત હોવ તો, ઈનવર્ઝનની સંખ્યા n થી minus n માં 2 થશે, જે ક્રમાંક n સ્ક્વેર્ડ છે. તેથી, તમે હવે આ માપદંડ તરીકે ઉપયોગ કરી શકો છો, ક્રમાંકની સંખ્યા કેટલી છે, તે કેવી રીતે સમાન અથવા અસમર્થ છે, રેંકિંગના બે સેટ છે અને આ ભલામણમાં ઉદાહરણ તરીકે ઉપયોગ કરી શકાય છે કે કેમ તે નક્કી કરવા માટે ગ્રાહકો કયા તુલના કરે છે ભલામણ કરવામાં. તેથી, તમે માત્ર એવા ગ્રાહકોને પસંદ કરવા માંગો છો કે જેઓ નજીકના છે, જેની ભલામણ કરવામાં આવી રહી છે. કોઈ વસ્તુ એવી ભલામણ કરતું નથી કે જે કંઈક વ્યક્તિગત ગમશે નહીં કારણ કે તમે તેનાથી સરખામણી કરો છો કે જેની પાસે એકદમ અલગ સ્વાદ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:10)

તો, આપણે તેને બીજી રીતે બનાવી શકીએ છીએ. તેથી, હવે, આપણે આપણી રેન્કિંગ લઈએ છીએ અને આપણે ધારીએ છીએ કે આ આદેશ આપ્યો છે. તેથી, આપણે મૂવીઝ માટે ચોક્કસ ઓર્ડર પસંદ કરીએ છીએ અને અમે તેને મૂળ ક્રમાંક 1, 2, 3, 4 સુધી n કહીએ છીએ. હવે, આપણી મિત્રોની રેંકિંગ એ રેંક કરશે જે આપણે 1 તરીકે કહી શકીએ તે 1 હોઈ શકે છે, જેને આપણે 2 કહીએ છીએ તે 3 હોઈ શકે છે અને બીજું. તેથી, જે ક્રમાંક આપણે ક્રમાંક સાથે ક્રમાંકિત કરીએ છીએ તે ક્રમાંકિત કરવામાં આવશે, જ્યાં આપણા મિત્ર દ્વારા જુદા જુદા ક્રમાંકો અને અલબત્ત, દરેક ક્રમાંક ત્યાં દેખાશે. તેથી, મિત્રો રેન્કિંગ 1 થી n ની ક્રમચય હશે અને તમે જે માગી રહ્યા છો તે છે કે જો હું j પહેલા રેંક કરું, તે પહેલા હું j કરતા નાની સંખ્યા છે, તો શું મિત્ર પહેલા મને j ક્રમ આપે છે, આવી કોઈ વિચાર હશે એક બદલાવ. તેથી, મારી યાદીમાં વ્યુત્ક્રમ એક જોડી છે, હું કોમા જે હશે, જ્યાં હું j થી નાનું છું. તેથી, મારી સૂચિ હું j નું અપડેટ કરું છું, પરંતુ મારા મિત્રોની સૂચિમાં, j એ એડ ક્રમચયમાં પહેલા દેખાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:07)

ચાલો આ થોડું વધારે કોંક્રિટલી જોઈએ. તેથી, ધારો કે આ અમારું મૂળ ઉદાહરણ હતું. તેથી, ત્યાં 5 મૂવીઝ એ, બી, સી, ડી, ઈ અને હું તેમને ક્રમ આપું છું અને તમે તેમને ક્રમ આપો છો ડી, બી, સી, એ, ઈ. પછી હું કહીશ કે ડી 1, બી 2, સી 3 છે એ 4 અને ઈ 5 છે. તેથી, આ મારી મૂળ સૂચિ 1 થી 5 છે. હવે, કારણ કે મારી પાસે તેમની મૂવીઝ અને તે વસ્તુઓ અને રેન્કિંગ્સ વચ્ચેનો પત્રવ્યવહાર છે, તો પછી મને ખબર છે કે ત્યારથી D માટે 2 છે. તેથી, બી એ 2 છે, એ 4 છે, સી 3 છે, ડી 1 છે અને ઈ 5 છે. તેથી, આ પસંદગીઓની સૂચિમાંથી, હું તેને વાંચી શકું છું, તે મારા ક્રમાંકનની રેકોર્ડીંગ તરીકે લખી શકું છું અને હવે, આપણે પૂછીએ છીએ કે જ્યારે જોડીઓ છે કે નહીં આ 2 અને 1 ની જેમ. તેથી, 2 એ 1 અને મારા મિત્રોની સૂચિ પહેલા દેખાય છે, તે દેખીતી રીતે મૂળ સૂચિમાં 1 પછી દેખાય છે, તેથી આ એક બદલાવ છે. તેથી, 2, 1 એ વ્યુત્ક્રમ છે, તેવી જ રીતે 4, 1 અવલંબન છે, તેથી તે 3, 1 છે, તેથી આપણી પાસે આ ત્રણેય અવ્યવસ્થા છે, પછી ભલે આપણે તેને 1 થી 2, 1 તરીકે લખીએ; 3 ના 3, 1, કારણ કે આ જોડી છે, તેથી ક્રમમાં તે મહત્વનું નથી, મૂવીઝનાં આ જોડી તમારા અને તમારા મિત્ર દ્વારા વિરોધાભાસી રૂપે છે. અને આ વિશિષ્ટ ઉદાહરણમાં અંતિમ વ્યુત્ક્રમ 3 અને 4 છે, તેથી 3 અને 4 વિપરીત ક્રમમાં દેખાય છે, તેથી તમે ઉદાહરણ તરીકે, દરેક અન્ય જોડી, 4, 5 અથવા 2, 3 અથવા 2, 4 ઑર્ડર હાજર છે તે જોઈ શકો છો. તેથી, તમારી રેંકિંગ અને તમારા મિત્રો વચ્ચે ક્રમાંકિત આ વિશિષ્ટ સૂચિમાં ચાર ઈનવર્ઝન છે અને અમારું ધ્યેય ક્રમચયને આપવામાં આવતી આ સંખ્યાઓની સંખ્યાને ગણવું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:43)

તેથી, આના વિશે વિચારવાનો બીજો રસ્તો, જો કે આપણે તેની ગણતરી કેવી રીતે કરી શકીએ તે અસરકારક રીતે અસર કરશે નહીં, આ પ્રકારના ગ્રાફને દોરવાનું છે. તેથી, તમે ટોચ પર યોગ્ય ક્રમમાં સાથે શરૂ કરવા માટે રેન્કિંગ લે છે. તેથી, તમારી પાસે 1 થી 5 શિર્ષકોનો એક સમૂહ છે અને પછી, તમારે તમારા મિત્રોની ક્રમાંકનના ક્રમમાં સૂચિબદ્ધ 1 થી 5 શિર્ષકોનું ક્રમચય કરવું પડશે. અને પછી, તમે 1 થી 1, 2 થી 2 ને જોડો છો અને તેથી, તમે ગ્રાફમાં બિલ્ડ કરો છો, જેમ કે તમારી પાસે 5 ની ઉપર અને 5 ની નીચે છે, તો તમારી પાસે 5 ધાર છે, જો તમારી પાસે n અને n છે એન ધાર છે. હવે, આ રેખામાં દરેક વખતે જ્યારે કોઈ રેખા પાર કરે છે ત્યારે તે મેળ ખાય છે તે સૂચવે છે, તેથી 2 એ 1 થી આગળ નીકળી ગયું છે, તેવી જ રીતે 4 4 આગળ આગળ વધ્યું છે અને તેથી આગળ. તેથી, આ રેખાઓ વચ્ચે 4 ક્રોસિંગ છે અને તે ખરેખર

આ ઉદાહરણમાં ચાર ઈનવર્ઝન સાથે સંબંધિત છે. તેથી, હવે, તપાસ કરવા માટે એક ખૂબ જ સરળ બ્રુટ ફોર્સ રસ્તો છે, કારણ કે આપણે જાણીએ છીએ કે દરેક ઈનવર્ઝન એક જોડી ij છે, જેમ કે હું મારા મિત્રોની સૂચિ પહેલા જ દેખાય છે. તેથી, આપણે તે ચકાસી શકીએ છીએ, અમે માત્ર દરેક i અને દરેક j ને ચકાસી શકીએ છીએ જે i થી જુદું છે, પછી ભલે હું અને j એક બદલાવ છે અને આ આપણને બ્રુટ ફોર્સ ઓર્ડર n સ્ક્વેર્ડ અલ્ગોરિધમનો આપશે. તેથી, આ ખરેખર બધી વ્યુત્ક્રમોને સમજાવે છે, તે દરેક સંભવિત જોડીઓની તપાસ કરે છે અને જો તે વ્યુત્ક્રમ છે તો તે હા કહે છે, જો તે કોઈ વ્યુત્ક્રમ નથી તો તે પછી અને પછી કહે છે, તમે કેટલી સંખ્યામાં વિચલનો હલ કરો છો તે તમે જાણો છો. અને અમે જોયું અને તે છે કે આપણે ખરેખર ખરાબ કેસમાં સંપૂર્ણ રીતે n ને 1 ઓછા 2 વટાવી લઈએ છીએ. તેથી, આ નિરાશાજનક રીતે દરેક ઉલ્લંઘનને તેને હા અથવા ના સાથે તપાસવામાં આવશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:15)

તો, આપણું આખું વધુ કાર્યક્ષમ અલ્ગોરિધમ આપવાનું છે, તેથી આપણે આ વિભાજન તરફ આગળ વધીશું અને પ્રતિબિંબ જીતીશું. તો, માની લો કે તમારા મિત્રનું ક્રમચય, આપણું ક્રમચય હંમેશાં 1 થી n છે, તેથી અમે એમ માનીએ છીએ કે તે આપવામાં આવ્યું છે. તેથી, શરૂઆતમાં રસપ્રદ શું છે આપણા મિત્રો ક્રમચય છે, તેથી મિત્રો ક્રમચય એ 1 થી n ના કેટલાક ક્રમમાં છે, ચાલો આપણે તેને 1 થી i ને કહીએ. તેથી, હવે, આપણે મર્જ સૉર્ટ જેવું કંઈક કરીશું, તેથી તમે આ સૂચિને 1 માં લો અને બે ભાગોમાં વહેંચી લો. તેથી, આપણી પાસે 1 થી 2 ની સંખ્યા છે જે ડાબી બાજુ છે અને 2 વત્તા 1 થી n સુધી જે જમણી છે. તેથી, વિભાજન અને જીતવું એ ખૂબ જ સરળ વિચારસરણીની વ્યૂહરચના છે, તમે ફક્ત એક જ વસ્તુ કરી શકો છો, તમે આને હલ કરી શકો છો અને તે પછી, તમે જોડાઈ શકો છો. તેથી, આ મૂળભૂત અનુરૂપ છે, તેથી તમારે વિભાજિત ભાગોને ઉકેલવા અને જોડવાનું છે. તેથી, આપણે વારંવાર ધારીશું કે આપણે ઈનવર્ઝનને ડાબી અને જમણી બાજુએ ગણી શકીએ છીએ. હવે, ગણતરી કરવા માટે બાકી શું છે તે અવરોધો જે સીમાને પાર કરે છે. તેથી, ત્યાં જે છે, હું જોડી જે આ જેવો દેખાય છે. જ્યારે હું માત્ર ડાબી જ ગણું ત્યારે આ ગણતરી કરવામાં આવશે નહીં, કારણ કે હું ડાબે નથી, તે માત્ર જમણી બાજુએ જ ગણવામાં આવશે નહીં, કારણ કે j એ જમણી બાજુ નથી. તેથી, એક બદલાવ થશે જે હું નંબરો જેટલા જ ઓછા છે, પરંતુ ડાબી બાજુ જ j દેખાય છે, હું જમણી તરફ દેખાય છે. તેથી, આપણે વારંવાર ઉકેલવા પછી આ કરવું જોઈએ, તેથી આ મૂળભૂત રીતે સંયોજનો પગલું છે, જમણી બાજુના કેટલા તત્વો ડાબી બાજુના ઘટકો કરતા વધારે છે. ડાબી બાજુની કોઈપણ વસ્તુ, જો તે જમણી બાજુની કોઈ વસ્તુ કરતાં નાની હોય, તો તે બદલાવ નથી, કારણ કે તે પહેલેથી જ સંપૂર્ણ સૂચિમાં યોગ્ય ક્રમમાં છે. પરંતુ, ડાબે કંઈક કરતાં જમણી બાજુએ કંઈક વધારે છે, તે એક બદલાવ છે, આપણે આવા બધા જોડીઓને ગણવું પડશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:55)

તેથી, આ ઉકેલવા માટે, આપણે માત્ર ગણતરી કરતાં થોડી વધારે મજબુત પ્રક્રિયા કરીશું અથવા રીકર્સીવ કરીશું. તેથી, આપણે ધારીશું કે માત્ર બે ભાગમાં કાર્યોની ગણતરી નથી; અમે તેમને સારી રીતે સૉર્ટ કરીએ છીએ અમે ગણતરી કરી રહ્યા છીએ. તો, હવે શું થાય છે, હવે આપણે બે ભાગમાં આપણી સમસ્યા વહેંચી લીધી છે અને પછી, આપણે પાછા આવીશું, તો આ મારું L છે અને આ મારું આર છે, મેં હવે R ને સૉર્ટ કર્યું છે અને મારી પાસે કાઉન્ટરો છે, તેથી મારી પાસે ગણતરી છે એલ અને એક ગણતરી આર. હવે, તે જે પરિબળો સૉર્ટ કરેલા છે, તેનો અર્થ એ છે કે હું કોઈ પ્રકારનું મર્જિંગ કરી શકું

છું. તેથી, હું મર્જની આવૃત્તિનો ઉપયોગ કરી શકું છું. તેથી, અમે મર્જિંગનાં સંસ્કરણનું વર્ણન કરીશું જે અમને ગણતરી કરવાની મંજૂરી આપે છે. તેથી, આ અન્ય ગણતરીઓ આપે છે અને તે પછી, આપણી પાસે ત્રણ ગણતરીઓ છે જે બાકીની ગણતરીને યોગ્ય ગણાય છે અને મર્જ દ્વારા ગણતરી પરત આવે છે અને તેથી તમે જે મર્જ કરવા માંગો છો તે છે કે વાસ્તવમાં કેટલા તત્વો છે અથવા કેટલા તત્વો કરતાં મોટા ડાબી બાજુ પર.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:57)

તો, આપણે આ કેવી રીતે કરી શકીએ? તેથી, મર્જ અને ગણતરી સિદ્ધાંત શું છે? તેથી, યાદ રાખો કે એલ અને આર તરફનો ઉદ્ભવ આરમાં એક તત્વ ધરાવે છે. તેથી, આપણી પાસે R માં એક તત્વ છે અને એલમાં એક તત્વ છે, જેમ કે આ સંખ્યા આ સંખ્યા કરતા નાની છે. તેથી, અમારી પાસે અહીં કેટલાક અને અહીં કેટલાક છે, જેમ કે હું j થી નાના છે. તેથી, મર્જ પ્રક્રિયામાં શું થાય છે તે કોઈક સમયે છે, તેથી અમે મર્જ કરી રહ્યા છીએ, તેથી અમે આ બંનેમાંથી નાનાને પસંદ કરીએ છીએ અને ખેંચીએ છીએ. તેથી, હવે જો હું અહીંથી એક તત્વ ખેંચ્યો હતો; એટલે કે, આ વર્તમાન નિર્દેશક પર મેં આ સુધી એક સાથે મર્જ કરી છે. તેથી, મારી સૂચિમાં બે પોઈન્ટર છે ડાબે અને જમણે, સૂચિબદ્ધ સૂચિ કે જેના પર મર્જ થઈ ગઈ છે. હવે, આ બિંદુએ હું જમણી બાજુની તત્વ પસંદ કરું છું, કારણ કે તે નાનું છે. તેથી, જો તે નાનું હોય, તો તે બધું કરતાં નાના છે જે મેં જોયું ન હતું, તેથી જ, કારણ કે તે પ્રથમ તત્વ કરતાં નાનું છે કારણ કે હું એલ માં જોઈ રહ્યો છું. તેથી, એલમાં બીજું બધું, કારણ કે બીજું બધું એલમાં સૉર્ટ

((સમયનોસંદર્ભ લો: 13:04)).

તેથી, તેથી, આ સમગ્ર સેગમેન્ટ જે એલમાં રહે છે તે તત્વોને અનુરૂપ છે જે વર્તમાન તત્વ કરતાં નાના છે તેનાથી બહાર ખેંચી રહ્યો છું. બીજા શબ્દોમાં કહીએ તો, આરમાં આ તત્વ ઘણાં અવ્યવસ્થાને ફાળો આપે છે કારણ કે મર્જ પ્રક્રિયામાં જ્યારે તે કાઢવામાં આવે છે ત્યારે એલમાં તત્વો હોય છે. તેથી, જ્યારે પણ હું આર માંથી આઉટપુટમાં એક તત્વ ઉમેરું છું, ત્યારે તે આમાં લીધેલા બધા ઘટકોને પાછું વાળવામાં આવે છે. તેથી, મને પ્રવર્તમાન કદને L ને વર્તમાન સંખ્યામાં ઉમેરવું જોઈએ. તેથી, આ મર્જ થાય છે અને ગણતરી થાય છે, જ્યારે આપણે મર્જ કરી રહ્યા છીએ, દરેક વખતે જ્યારે આપણે ડાબેથી ખેંચીએ છીએ, ત્યાં કોઈ બદલાવ નથી, દરેક વખતે જમણી તરફ ખેંચાય છે, આપણે જોઈ શકીએ છીએ કે ડાબે અને કેટલી બધી વસ્તુઓ બાકી છે અમારા ઈનવર્ઝન ભાગમાં ઉમેરવાની જરૂર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:50)

તેથી, અહીં મર્જ અને ગણતરી માટે મર્જ પ્રક્રિયા છે જે મૂળભૂત મર્જ સેટમાં મર્જ પ્રક્રિયા જેવી જ છે. તેથી, અમારી પાસે મર્જ કરવા માટે A અને B ની સૂચિ બે હતી, બંનેને સૉર્ટ કરવામાં આવી છે અને A પાસે m તત્વો છે અને B પાસે n એલિમેન્ટ્સ છે અને આપણે આઉટપુટ સૂચિ સી બનાવવું છે, જેમાં m plus plus તત્વો છે. તો, આપણે પોઈન્ટર પર હસ્તાક્ષર કરવાથી પ્રારંભ કરીએ છીએ કે અમને દરેક સૂચિ પર કેવી રીતે જન્મ આવે છે, આપણે ફક્ત 0 તરીકે સેટ કરીએ છીએ. અને હવે, આપણે ઈનવર્ઝનની સંખ્યાને ટ્રૅક રાખવી પડશે, તેથી આપણે ચલ તરીકે ઓળખાય છે જેને ગણતરી કહેવામાં આવે છે. 0, અત્યાર સુધીમાં જોવાયેલા સંખ્યાબંધ અવરોધો. તેથી, સીમાં જવા માટે કંઈક છે, આપણે કંઈક ખસેડીએ છીએ, તેથી ત્યાં બે કિસ્સાઓ છે, એમાંથી ખસેડવાના પ્રથમ કેસો છે, તેથી મારી પાસે બી ખાલી છે, જે ઈ

બરાબર n એ માથાના ઘટક છે. એ, હું B ની સમાન કરતાં નાના છે જે કિસ્સામાં આપણે સામાન્ય વિચાર કરીએ છીએ, અમે A ની એથ તત્વને સીમાં કે કેમાં નકલ કરીએ છીએ અને પછી, આપણે ફરીથી એન કે બંનેને વધારીએ છીએ. અન્ય કિસ્સાઓમાં જ્યારે ખાલી A એ ખાલી હોય છે, હું નાના મૂલ્ય તરીકે એમ અથવા બી જે સમાન હોય છે, આ કિસ્સામાં સંભવતઃ એક વ્યુત્ક્રમ હોઈ શકે છે. તેથી, આપણી પાસે કેટલી સંખ્યા છે, અમારી પાસે બરાબર ઓછા ઓછા છે, અમારી પાસે અસંખ્ય અવ્યવહાર છે તેથી આ હું છે, આપણી પાસે અસંખ્ય અવ્યવહાર છે જેમ કે એ હાલમાં તત્વો છે, જે n ઓછા છે 1. હવે, નોટિસો કે ચોક્કસ કિસ્સામાં જ્યાં આપણે બીમાંથી કૉપિ કરી રહ્યા છીએ, કારણ કે A ખાલી છે, આપણે મારી પાસે સમાન છે. તેથી, આ ખરેખર બી 0 હશે. તેથી, તેઓ એક બદલાવ ન હોવું જોઈએ, જો તેઓ માત્ર બી અને સીને ડુપ્લિકેટ કરે છે, કારણ કે એ થાકી ગયું છે. તેથી, તે 0 ની માઈનસ ની પણ સંભાળ લે છે, તેથી જો આપણે ગણતરીને અપડેટ કરી રહ્યા છીએ, તો અમે તેમાં ઉમેરી રહ્યા નથી એમ ધારી રહ્યા છીએ. તો, આપણે ફક્ત એક નવિનય મર્જ કરી રહ્યા છીએ, જ્યારે આપણે બીમાંથી કંઈક ખસેડી રહ્યા છીએ, જ્યારે A હજુ પણ ગણતરીમાં વધારો કરવા માટે ખાલી નથી, તો આપણે આગળ વધી રહ્યા છીએ કારણ કે આપણે આ ઘટકોની ગણતરી કરવા માટે ગણતરીમાં છીએ. તેથી, આ વિલીનીકરણની પ્રક્રિયાનો અંત, અમે આ સંખ્યાઓની સંખ્યાને પાછો જોયો છે જે આપણે જોઈ છે મર્જ સૂચિ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:48)

તેથી, હવે, આપણે મર્જ સૉર્ટમાં પ્રક્રિયાને સમાવિષ્ટ અથવા મર્જ કરવા નીચે પ્રમાણે ગણાય છે. હંમેશની જેમ, અમે કેટલાક ડાબે અનુક્રમણિકાથી કેટલાક અધિકાર અનુક્રમણિકામાં સામાન્ય રીતે એક એરેને સૉર્ટ કરીશું, કારણ કે અમે વિવિધ સેગમેન્ટ્સ અને વિવિધ સમયે સૉર્ટ કરીશું. તેથી, જો વર્તમાન સેગમેન્ટને લંબાઈ 1 તરીકે ગોઠવવાનું છે, તો ત્યાં કંઈ કરવાનું બાકી નથી, આપણે ફક્ત એક નવું સેટ કર્યું છે, સૉર્ટ કરેલ સેગમેન્ટ એ ફક્ત તે મૂલ્ય છે જે આપણે જોઈએ છીએ અને તેમાં કોઈ બદલાવ નથી, કારણ કે તે ફક્ત એક જ છે મૂલ્ય બીજી તરફ, જો તે 1 કરતા મોટો હોય, તો આપણે મિડપોઈન્ટની ગણતરી કરીએ છીએ અને આપણે આ બંને પુનરાવર્તિત કૉલ્સ કરીએ છીએ, તેમાંથી દરેક ક્રમે અનુક્રમે ડાબી અને જમણી બાજુની ગણતરી આપશે, પછી આપણે પ્રક્રિયાને કૉલ કરીશું અથવા મર્જ કરીશું અને આની ગણતરી કરીશું મર્જ વિભાગ. તો, પછી આપણી કુલ સંખ્યાઓની સંખ્યા ડાબી બાજુની ગણતરી છે અને મર્જથી સાચી ગણતરી થાય છે અને નવી એરે મર્જ દ્વારા પરત આવે છે. તો, આ મર્જ સૉર્ટનું એક ખૂબ જ સરળ એક્સ્ટેંશન છે જે અમને ભલામણોની સિસ્ટમ માટે ઉપયોગી છે તેવા ઉલ્લંઘનોની ગણતરી કરવાની મંજૂરી આપે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 16:47)

તેથી, અલબત્ત વિશ્લેષણ, સંસાધનોના સંસાધનો મર્જ સૉર્ટ જેવું જ છે, વિશ્લેષણ સમાન છે, અમે એન પગલાંઓ પર લીધેલા સમય માટે આ રિકર્ઝન છે. તો, 1 નું ટી 1 છે અને n ના T એ 2 ગણી ટી n બરાબર 2 વત્તા n બરાબર છે. તેથી, સૉર્ટ કરો મર્જ કરો કારણ કે તે ગણતરી સાથે સૉર્ટ મર્જ કરવા માટે ફક્ત રેખીય સમય તરીકે લે છે. તેથી, તમે ગણતરી સાથે મર્જ કરો તે ગણતરી વિના મર્જ થવા જેટલી જ સમય લે છે, તેથી અમે આને હલ કરીએ છીએ અને આપણને n લોગ n મળે છે. હવે, નોંધ લેવા માટેની એક મહત્વપૂર્ણ વિચાર એ છે કે જ્યારે આપણે બ્રુટ ફોર્સની ગણતરી કરી હતી, ત્યારે અમે દરેક સંભવિત જોડીને જોયા અને નિર્ણય કર્યો કે શું કોઈ વ્યુત્ક્રમ નથી, તેથી વાસ્તવમાં તે સ્પષ્ટતાપૂર્વક જવાબ મેળવવા માટેના ઉલ્લંઘનની ગણતરી કરે છે. હવે, અહીં આપણે એન લોગ n કરી રહ્યા છીએ જે તમને જોઈએ છે તે ઈનવર્ઝનની

સંખ્યા કરતાં સંભવિત રૂપે ઘણું ઓછું છે. તેથી, આપણે વાસ્તવમાં અસમાન સંખ્યાઓની ગણતરી કરી રહ્યા છીએ અથવા અંદાજ આપીએ છીએ કે વાસ્તવમાં તે કેવી રીતે સંખ્યાબંધ વ્યુત્ક્રમો છે તે વાસ્તવમાં જાતે એક ગણવામાં આવે છે. કારણ કે, તેઓ સ્ક્વેર્ડ ઈનવર્ઝન હોઈ શકે છે, પરંતુ ત્યાં એન સ્ક્વેર્ડ ઈન્વર્ઝનનો ઉપયોગ થાય છે, અમે તેને n log n ટાઈમમાં શોધી શકીએ છીએ. તેથી, અમે દરેક પગલાને મેન્યુઅલી ગણતા નથી, તેના બદલે આપણે તેને પુનરાવર્તિત ગણતરી દ્વારા મેળવી રહ્યા છીએ.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 07
લેક્ચર - 38
ડિવાઈડ અને કોનકોર(Divide and Conquer): પોઈંટ્સની સૌથી નજીકનું જોડણી

હવે આપણે બીજા ડિવાઈડ અને કોનકોર(Divide and Conquer) તરફ ધ્યાન આપીએ છીએ, આ ભૌમિતિક સમસ્યા છે, જે પોઈન્ટનો સેટ આપવામાં આવે છે તેમની વચ્ચેની સૌથી નજીકના પોઈન્ટની ગણતરી કરવા.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:11)

તેથી, યાદ રાખો કે લેક્ચર્સના આ સમૂહની શરૂઆતમાં વધુ કાર્યક્ષમ અલ્ગોરિધમ્સની જરૂરિયાતને પ્રોત્સાહિત કરવા માટે, જો તમે સ્ક્રીન પર કેટલીક ઑબ્જેક્ટ્સ હોય અને તમે કદાચ વિડિઓ ગેમનો દાખલો લઈ શકો તેને કોઈ પણ બિંદુએ શોધી કાઢવું છે, તેમની વચ્ચેની સૌથી નજીકની વસ્તુઓ. તેથી, આ માટે ફરી એક નિષ્કપટ અલ્ગોરિધમ આ n વસ્તુઓના દરેક જોડી વચ્ચેની અંતરની સ્પષ્ટ રીતે ગણતરી કરી શકે છે, જે ઓર્ડર n સ્ક્વેર્ડ અલ્ગોરિધમનો હોવો જોઈએ. તેથી, આપણે જે જોવા જઈ રહ્યા છીએ તે છે કે આપણે વાસ્તવમાં આ સમસ્યા માટે ડિવાઈડ અને કોનકોર(Divide and Conquer) મેળવી શકીએ છીએ અને ઓર્ડર n લોગ એન અલ્ગોરિધમનો ઉપયોગ કરી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:45)

તેથી, ઔપચારિક રીતે આપણે બે પરિમાણોમાં પોઈન્ટ જોઈ રહ્યા છીએ, તેથી દરેક બિંદુ xy coordinate xp, yp દ્વારા આપવામાં આવે છે અને આપણે સામાન્ય ઉપયોગિતા અને અંતરની ગતિનો ઉપયોગ કરી રહ્યા છીએ જે અંતર દ્વારા આપવામાં આવેલ અંતર છે પાયથાગોરસ સૂત્રનો ઉપયોગ કરે છે, જે છે કે પી 1 અને પી 2 ની અંતર એ x 2 ઓછા x 1 સંપૂર્ણ ચોરસ વત્તા વાય 2 ઓછા વાય 1 સંપૂર્ણ ચોરસનું વર્ગમૂળ છે. તેથી, તમે માત્ર ધારે છે કે ત્યાં એક અંતર સૂત્ર છે જેનો ઉપયોગ આપણે જ્યારે પણ પોઈન્ટ્સની જોડી વચ્ચે અંતરની ગણતરી કરવા માંગતા હોઈએ ત્યારે કરી શકીએ છીએ. તેથી, અમારા લક્ષ્યને n પોઈન્ટ્સ પી 1 નો સમૂહ તેમના વચ્ચેના સૌથી નજીકના જોડીને શોધવા માટે પીન પર સેટ કરવામાં આવે છે અને તે એલ્ગોરિધમનો વિશ્લેષણ માટે અનુકૂળ રહેશે કે અમે સૂચવવા જઈ રહ્યા છીએ કે આ સેટમાં કોઈ પણ બે પોઈન્ટ સમાન એક્સ નથી અથવા વાય સંકલન. તેથી, આ એનએક્સ કોઓર્ડિનેટ્સ વચ્ચેના દરેક x સંકલન અલગ છે, આ n કોઓર્ડિનેટ્સ વચ્ચેના દરેક વાય સંકલન અલગ છે. હવે, એલ્ગોરિધમ દ્વારા વિસ્તૃત કરી શકાય છે જે આપણે બતાવવા જઈ રહ્યા છીએ, તે ધારણા સાથે વ્યવહાર કરવા માટે વિસ્તૃત કરી શકાય છે જ્યાં આ ધારણા સાચી નથી, પરંતુ તે પછી અલ્ગોરિધમનો સમજાણ બિનજરૂરી રીતે જટિલ કરશે. તેથી, ચાલો ધારીએ કે આપણે સમસ્યાના આ વિશિષ્ટ કેસને હલ કરી રહ્યા છીએ, જ્યાં દરેક બિંદુ એક અલગ x સંકલન પર છે અને દરેક અન્ય બિંદુથી જુદા જુદા વાય સંકલનમાં છે. તેથી, જેમ આપણે જોયું છે કે બ્રુટ ફોર્સ સોલ્યુશન દરેક જોડીને અજમાવવા, p i p j ની ડીની ગણતરી કરવા અને પછી ઓછામાં ઓછા અંતરની જાણ કરશે. તેથી, આ ઓર્ડર n સ્ક્વેર્ડ અલ્ગોરિધમનો હશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:08)

તેથી, ચાલો પહેલા એક જ સમસ્યા જોઈએ જો આપણી પાસે માત્ર એક પરિમાણીય બિંદુ હોય. જો આપણી પાસે એક પરિમાણીય બિંદુઓ હોય તો આ બધા બિંદુઓ રેખા સાથે રહે છે, જેને આપણે એક્સ અક્ષ ગણી શકીએ છીએ. તેથી, અમારી પાસે પોઈન્ટનો સમૂહ છે અને આપણે નજીકના બિંદુને શોધવા માંગીએ છીએ. તેથી, આપણે શું કરી શકીએ તે છે કે આપણે તેમને પ્રથમ સૉર્ટ કરી શકીએ, જેથી આપણી પાસે x સંકલનના ક્રમમાં વધતા પોઈન્ટ હોય અને પછી તે જોવાનું સરળ છે કે આપણે જે અંતરની જરૂર છે તે બે નજીકના બિંદુઓ વચ્ચેનો અંતર છે. કારણ કે, જો હું કોઈ બિંદુ તરફ જોઉં છું, તો નજીકનો પોઈન્ટ જો તેના પરનો કોઈ એક બાકી હોય અને તેના પરનો એક જ યોગ્ય હોય. તેથી, મારે આ x 2 ઓછા x 1 અંતર, પછી x 3 ઓછા x 2 ને સ્કેન કરવાની જરૂર છે, તેથી મારે આ n માઈનસ 1 અંતરને સ્કેન કરવાની જરૂર છે અને પછી આ બે બિંદુઓ વચ્ચેનો સૌથી નાનો તફાવત ટ્રૅક રાખવો પડશે અને તે મને આપશે એકંદરે n પોઈન્ટ્સની એકંદર જોડી વચ્ચેનો સૌથી નાનો અંતર. તેથી, અહીં એલ્ગોરિધમ એ n લોગ n છે, કારણ કે તે x સંકલનમાં પોઈન્ટને સૉર્ટ કરવા માટે n લોગ n વખત લે છે, તે પછી ન્યૂનતમ શોધવું ખરેખર ખૂબ જ સરળ છે. તેથી, એક પરિમાણમાં આ સમસ્યા હલ કરવી ખૂબ જ સરળ છે, પડકાર તે બે પરિમાણોમાં હલ કરવાનો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:12)

તેથી, જો આપણે ડિવાઈડ અને કોનકોર(Divide and Conquer)નો ઉપયોગ કરવા જઈ રહ્યા છીએ તો તે બે પરિમાણો છે, આપણે પોઈન્ટને બે જૂથોમાં વિભાજીત કરવાની રીત, લગભગ સમાન કદ અથવા આશરે બરાબર સમાન કદની જરૂર છે. તેથી, એક કુદરતી રીત એ છે કે ભૌમિતિક સમસ્યા તેમને તેમના સ્થાનો પર આધારિત છે. તો, આ મારા પોઈન્ટ્સનો એકંદર સેટ છે, પછી આ કુદરતી છે અને કોઈ પણ પ્રકારની રેખા દોરવા અને કહે છે કે અડધા પોઈન્ટ અહીં છે અને અડધા પોઈન્ટ છે. તો, આપણે તેને વર્ટીકલ લાઈનનો ઉપયોગ કરીશું, તેથી આપણે આ સમૂહને આના જેવી અનિશ્ચિત રેખાથી વિભાજીત કરીશું, પરંતુ તેના બદલે આપણે તેને ઊભી રેખા દ્વારા પ્રયાસ કરીશું અને વિભાજીત કરીશું. તેથી, ક્યાંક કે જે અડધા માર્ગે ન હોઈ શકે, કારણ કે પોઈન્ટ અસમાન રીતે ફેલાયેલા હોઈ શકે છે, આપણે તેને વિભાજીત કરીશું, જેથી ત્યાં જમણી બાજુની લીટીઓની સમાન સંખ્યા અને લીટીની જમણી બાજુ હોય. તેથી, હવે આપણે જે વસ્તુને વિભાજીત કરીશું અને જીવીશું તે આપણે કરીશું, આપણે ડાબી બાજુના બિંદુઓ વચ્ચેના નાના અંતરની ગણતરી કરીશું, અલગથી આપણે જમણી બાજુના નાના અંતરની બિંદુઓની ગણતરી કરીશું. પરંતુ, તે અમને ડાબી બાજુના બિંદુઓ અને જમણી બાજુના બિંદુઓ વચ્ચેની અંતર વિશે કંઈ પણ જણાતું નથી અને તે સીમામાં એકબીજા સાથે ખૂબ જ નજીકના બિંદુઓ હોઈ શકે છે. તો, અહીં ચાર પોઈન્ટ ઉમેરી શક્યા હોત, ચાર પોઈન્ટ્સ ઉમેરી શક્યા હોત અને હું આ પોઈન્ટ જોડી ઉમેરી શકું જે આ કાળો રેખા ફેલાવે છે, જે વાસ્તવમાં એકબીજા સાથે નજીક છે, પછી કોઈપણ બે બિંદુઓ ડાબે અથવા બે પોઈન્ટ જમણે છે. તેથી, આપણે જુદા જુદા લીટીમાં નજીકના જોડીની ગણતરી કરવાની જરૂર છે અને આ એક પડકાર છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:44)

તેથી, ચાલો આપણે આ કેવી રીતે કરીએ તેના પર થોડી નજીક જોઈએ. તેથી, અમે આ વસ્તુને વારંવાર કરવા પહેલાં આગળ પગલું બનાવીશું. પોઈન્ટ આપ્યા પછી આપણે પોઈન્ટની ગણતરી કરીશું, પોઈન્ટ્સનો સેટ આપણે બે સોર્ટ

ઓર્ડર્સની ગણતરી કરીશું. તેથી, આપણે પહેલા આ મુદ્દાઓને ડાબેથી જમણે x સંકલન દ્વારા સ્કેન કરીશું અને અમે આ ક્રમમાં સૂચિબદ્ધ કરીશું અને તેને પી એક્સ કહીશું, પછી આપણે આને સ્કેન કરીશું, આ સૂચિ P માંથી ... તેથી, અમે y coordinate પર સૉર્ટ કરીશું અને આ પી વાય કૉલ કરો, તેથી પીથી આપણે બે સૂચિ બનાવીશું, એક x દ્વારા સૉર્ટ થશે અને વાય દ્વારા સૉર્ટ કરવામાં આવશે. પછી, આપણે ધારીશું કે આપણે આ કોર્સ કર્યું છે, આપણે તે જ કરી શકીએ છીએ જે આપણે જાણીએ છીએ કે શરૂઆતમાં જ n લોગ n સમય છે. હવે, આગલું પગલું આ પુનરાવર્તિત કૉલ કરવું છે અને તેથી જ્યારે આપણે પુનરાવર્તિત કૉલ કરીએ છીએ, કારણ કે આપણી પાસે પી x એ x coordinate દ્વારા તેને સૉર્ટ કરે છે, આપણે જાણીએ છીએ કે જે રેખાને આપણે દોરવા માટે જરૂર છે તે તે છે જે પી x ને બેમાં વિભાજીત કરે છે સમાન ભાગો. તેથી, આપણે પી એક્સના મધ્યબિંદુ પર જવાની જરૂર છે અને પછીના બિંદુથી મિડપોઈન્ટને અલગ કરીને x સંકલન પર એક રેખા દોરીએ છીએ. તેથી, જ્યારે આપણે પી એક્સ જાણીએ છીએ ત્યારે આ વાક્યની સ્થિતિ નિશ્ચિત કરવામાં આવે છે, જેનો અર્થ એ છે કે આપણે બે બિંદુઓ વચ્ચે જાણીએ છીએ. યાદ રાખો, આપણે ધારીએ છીએ કે સમાન એક્સ કોઓર્ડિનેટ પર બે બિંદુઓ, સમાન વાય સંકલનમાં કોઈ બે બિંદુઓ નથી. તો, જો હું ફક્ત મધ્યમ મૂલ્ય અથવા x ની મધ્યમાં મૂલ્ય માટે જોઉં છું, તો ત્યાં એક રેખા દોરે છે. હવે, હું ધારણા કરું છું કે તેઓ બે સમાન ભાગોમાં ધારણા કરે છે, કારણ કે મેં પી એક્સની આ સક્ષમ મિડપોઈન્ટ કરી છે. તેથી, મારી પાસે બે સેટ Q અને R છે, પરંતુ આ રિકર્ઝન ચાલુ રાખવા માટે મારે એમ ધારવાની જરૂર છે કે ક્યૂ x અને y બંનેમાં સૉર્ટ થાય છે અને તેથી આર છે. તેથી, મારે એ ધારવાની જરૂર છે કે મારી પાસે એક્સ્ટ્રેક્ટ કરવા માટેનો કેટલાક કાર્યક્ષમ રીત છે. આમાંની કેટલીક સમસ્યાઓ, એક્સ દ્વારા સૉર્ટ કરેલા પોઈન્ટનું સમાન ક્રમ, y દ્વારા સૉર્ટ કરેલા. તેથી, મને કુશળતાપૂર્વક ક્યૂ એક્સ અને ક્યૂ વાય, આર એક્સ અને આર વાય ગણતરી કરવાની જરૂર છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:39)

ક્યૂ એક્સ અને ક્યૂ વાય સરળ છે, કારણ કે આ બધી વસ્તુ પહેલા પી એક્સ હતી અને પછી આપણે આ વસ્તુ મિડપોઈન્ટને વિભાજીત કરી હતી. તેથી, મધ્યબિંદુની ડાબી બાજુની દરેક વસ્તુ Q x છે અને મધ્યબિંદુની જમણી બાજુએ દરેક વસ્તુ આર x છે. તેથી, પી એક્સના એક સ્કેનમાં હું અડધી રીતે જાઉં છું અને મેં ક્યૂ એક્સ માં બધું મૂકી દીધું છે અને પછી અડધા માર્ગે હું બધું આર એક્સ લમ્બાડામાં મુકું છું. હવે, વાય વિશે શું? હવે, સમસ્યા એ છે કે હું આગળ વધી રહ્યો છું આ બે બિંદુઓ Q વાયમાં છે, પછી આ બિંદુ આર વાયમાં જાય છે કારણ કે પી વાયમાં એકંદર યોજનામાં, આ બધા વૈશ્વિક સ્તરે y સંકલન દ્વારા સૂચિબદ્ધ છે. તેથી, મારે આને આર તરફ ખસેડવા પડશે, પછી હું આને પાછું મુકવા માંગું છું ... તેથી, ફરી એક પછી ફરી સંભવિત રૂ. તમે આનો ઉપયોગ કેવી રીતે કરો છો, પછીનો એક ક્યુ છે અને બીજું. તેથી, જેમ હું ઉપર જાઉં છું, કેટલાક મુદ્દાઓ ક્યૂ વાય અને કેટલાક જઈ રહ્યા છે. હવે, હું કેવી રીતે નક્કી કરી શકું કે આ ઘણી વાર ન જાય, તેથી કી એ છે કે x નું આ વિભાજન કર્યા પછી, આપણે આ વિભાજન રેખા નોંધીએ છીએ. તેથી, આપણે જાણીએ છીએ કે ક્યો x કોઓર્ડિનેશન Q થી R ને અલગ કરે છે. તેથી, આપણે પી વાય મારફતે જઈએ ત્યારે, આપણે દરેક poi ને જોઈશુંnt અને જો x સંકલન આ મિડપોઈન્ટ કરતાં ઓછું હોય, તો ચાલો આ એક્સ ક્યૂ કહીએ, જો તે x ક્યુ કરતાં ઓછું હોય, તો આપણે તેને ક્યૂ વાયમાં દબાણ કરીએ, જો તે x એ x ક્યુ કરતા મોટો હોય તો આપણે તેને આર વાય અને કારણ કે તે મૂળભૂત રીતે સૉર્ટ થાય છે, તો અમે ક્યૂ વાય અને આર વાય પણ સૉર્ટ કરેલ ક્રમમાં બનાવી રહ્યા છીએ. તેથી, પી વાયના સ્કેન તમને ક્યૂ વાય અને આર વાય મળે છે. તેથી, રેખાંકિત સમયમાં આપણે આપેલ સૉર્ટ કરેલી સૂચિ પી એક્સ અને પી

વાય લઈ શકીએ છીએ અને સૉર્ટ કરેલી સૂચિમાં અલગ શંકા ક્યૂ એક્સ, ક્યૂ વાય, ડાબી અર્ધ આર x, આર વાય માટે જમણી અર્ધ માટે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:29)

તેથી, હવે જ્યારે આપણે નજીકની જોડી પી એક્સ, પી વાય સાથે અમારા એલ્ગોરિધમને કૉલ કરીએ છીએ ત્યારે પી વાય તે નજીકના જોડીમાં ક્યૂ એક્સ, ક્યૂ વાય અને આર એક્સ, આર વાય ડાબી અને જમણી અર્ધમાં વિભાજિત કરશે. તેથી, અમે તમને હંમેશાં ધારી લઈએ છીએ કે અમે આ વારંવાર હલ કરીશું, ત્યાં અમને સૌથી નજીકની અંતર મળશે. હવે, આપણે તેમને કેવી રીતે જોડવું તે વિશે ચિંતા કરવાની જરૂર છે, આ બિંદુઓની કાળજી કેવી રીતે રાખવી, જેની અંતર અંતરાલને પેન કરે છે, જે બે ભાગને અલગ કરે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:58)

તેથી, જો આપણે આ સમસ્યાને ડાબી બાજુના બધા મુદ્દાઓમાં ડાબી બાજુએ હલ કરીએ, તો હું અંતર સાથે નજીકના જોડી તરીકે નજીકના જોડી તરીકે ઓળખી શકું છું. અને હવે, જો હું સમસ્યાને હલ કરીશ, હું અંતર ડી આર સાથે જમણે જમણી બાજુના જમણા જમણા ખૂણા તરીકે ઓળખીશ. તેથી, હવે આપણે આ નાના અથવા આ બંનેમાં રસ ધરાવો છો, કારણ કે આ બન્ને નાના નાના લઘુત્તમ માટે ઉમેદવાર છે. અંતર તેથી, હવે આપણે પોઈન્ટ જોઈ રહ્યા છીએ જે ડી ક્યુમાં હોઈ શકે છે જે આ સીમા પર ડી ક્યુ કરતાં નાના છે. તેથી, d ને d અને d R ની ન્યૂનતમ હોવી જોઈએ, તેથી આ બંનેની નાનું છે, તેથી આ વિશિષ્ટ ઉદાહરણમાં ડી છે ડી ક્યુ. હવે, દાવો છે કે જો તમે આ ઝોનમાં જુઓ છો, જે વત્તા ઓછા છે ડી અંતરથી દૂર છે, તો જો મારી પાસે આની બહાર કોઈ બિંદુ હોય અને જો હું બીજી તરફ કોઈ પણ બિંદુ તરફ જોઉં, તો અહીંથી અહીંની અંતર કી છે અને તેથી બંને બાજુએ થોડી અંતર છે, તે વધુ છે ડી કરતાં. તેથી, તે ડી ક્યુ અને ડી આર ની તુલનામાં વધુ છે, તેથી તે એકદમ નાનો સૌથી નાનો અંતર માટે ઉમેદવાર હોઈ શકતો નથી. તેથી, આ પ્રકારની વસ્તુઓ નકામી છે, કોઈ નિર્દેશક કોઈ પણ લીટીની શોધમાં નથી જે કોઈપણ જોડી છે જેની એક અંત બિંદુ આ ઝોનની બહાર છે. કારણ કે, જો તે ઝોનની બહાર હોય, તો તે ઓછામાં ઓછા બી પ્લસ હોઈ શકે છે જે બીજી બાજુથી બીજી બાજુથી કંઈક દૂર છે. તેથી, ઝોન અને બંને બાજુના બિંદુઓને જોવા માટે તે પૂરતું છે. તેથી, આપણે ફક્ત વિભિન્ન બિંદુઓને ધ્યાનમાં રાખવાની જરૂર છે જે આ વત્તા ઓછા વત્તા અંદરની રેખા છે, આની બહારની કોઈપણ જોડ લાઈનની સૌથી નજીક જોડી હોઈ શકતી નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:44)

તેથી, ચાલો નજીકનો દેખાવ કરીએ, તો આ મારો પ્લસ ઓછા છે, ચાલો આ ઓછા છે, આ વત્તા ડી છે. હવે, આપણે શું કરવા જઈ રહ્યા છીએ, આપણે તેને 2 વડે કદના આ વર્ગમાં ભંગ કરવા જઈ રહ્યા છીએ, તેથી મારી પાસે 2 વત્તા d થી 2 હતી. તેથી, આ સંપૂર્ણ વસ્તુ ડી સાથે, હું જઈશ આ બધા પગલાંને ધ્યાનમાં લો. તેથી, દાવો એ છે કે આવા બૉક્સની અંદર મારી પાસે મોટાભાગે એક બિંદુ હોઈ શકે છે, મારી પાસે બે પોઈન્ટ હોઈ શકતા નથી. શા માટે, કારણ કે બૉક્સની અંદરનો સૌથી દૂરનો ભાગ ત્રિકોણમાં હોય છે, તેથી આગળનું પગલું બૉક્સમાં હોઈ શકે છે, જે ત્રિકોણ તરફના 2 પોઈન્ટ છે. પરંતુ, કદના આ વર્ગના ત્રિજ્યા દ્વિ 2 ની વર્ગમૂળ છે, તેથી આ કેટલાક બિંદુઓ 0.047 ડી છે. તેથી, આ ડી કરતાં સખત ઓછું

ડી છે, પરંતુ ધ્યાન રાખો કે આ ચોરસ એક જ સમયે યોગ્ય સમયે, ડાબે અથવા જમણે અને ડાબે અને જમણે બંને છે, આપણે જાણીએ છીએ કે લઘુત્તમ વિભાજન એ છે, ત્યાં ત્યાં વાક્યની એક બાજુ પર કોઈ બિંદુ નથી, ડી કરતાં બે બિંદુઓ નજીક નથી, કારણ કે ડી એ ન્યૂનતમ ડી ક્યુ અને ડી આર છે તેથી, આમાંના દરેક બૉક્સમાં આપણી પાસે એકથી વધારે બિંદુ હોઈ શકે છે અને હવે આપણે શોધી રહ્યા છીએ કોઈપણ બિંદુએ. તેથી, આ બૉક્સમાં અહીં એક બિંદુ જુઓ, હવે એવો દાવો છે કે જો મને અંતરની અંદર હોવું હોય તો, હું તેના વિશે વિચારી શકું તો તમે ક્યાંથી દૂર રહેશો અથવા દૂરના બૉક્સમાં જઈ શકો છો તે બોક્સ પર જાઓ. તેથી, તમે થોડું વધારે સાવચેત થઈ શકો છો, પરંતુ તે ચોક્કસપણે તે બોક્સ કરતાં વધુ હોઈ શકતું નથી. તેથી, જો તમારે આ બૉક્સમાંથી જવું પડે તો મૂળભૂત રીતે જો આપણે આ અંતરથી વધુ દૂર જવાનું હોય તો, તે દૂરથી વધુ હશે. તેથી, દાવો એ છે કે આ અંતરની અંદરનો કોઈ પણ બિંદુ આગામી 4 થી 4 સેગમેન્ટ્સમાં રહેલો હોવો આવશ્યક છે. તેથી, આપણે દરેક બિંદુની માત્ર 15 વિરુદ્ધ તુલના કરવી પડશે. હવે અલબત્ત, આપણે તેની તુલના કરીશું, તે એક સ્કેનથી નીચેથી ઉપરની તરફ જશે. તેથી, આ નીચેનાં પોઈન્ટ હોઈ શકે છે જ્યારે આપણે તે અને તેના પહેલા ધ્યાનમાં લઈએ છીએ. તેથી, આપણે ધ્યાનમાં લઈશું, આપણે આ સ્કેનમાં આ ક્રમમાં કરીશું કારણ કે આપણે નેટમાં જોશું. તેથી, તેથી, આપણે આ બિંદુને આ 16 ચોરસ વર્ગની આસપાસના પોઈન્ટ્સ સામે ધ્યાનમાં લેવાની જરૂર છે. તેથી, તે વાસ્તવમાં કેટલા બિંદુઓ છે તેનાથી ફિક્સ નંબર સ્વતંત્ર છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 13:09)

તેથી, ઔપચારિક રીતે આપણે શું કરીશું તે આપણી પાસે છે અને આ બાજુ પાસે આપણી પાસે Q વાય છે જે સૉર્ટ કરેલું છે અને આ બાજુમાં આપણી પાસે RY છે જે સૉર્ટ કરેલ છે, પરંતુ યાદ રાખો કે તે બંને સૉર્ટ કરેલા છે. તેથી, આપણે એક પ્રકારનું મર્જ કરીશું, તેથી આપણે ક્યૂ વાય અને આર વાય મારફતે જઈશું, આપણે ક્યૂ વાય અને આર વાય વચ્ચે સૉર્ટ કરેલ વાય ઓર્ડરમાં આગલું એક પસંદ કરીશું. જો x સંકલન યાદ રાખતું હોય તો આ x ક્યૂ મૂળરૂપે પોઈન્ટ વત્તા ડી અને બાદબાકી ડીને અલગ કરે છે. જો x પો પ્લસ ડી અને x ક્યુ બાદબાકી ડી વચ્ચેના બિંદુનું એક્સ સંકલન હોય, તો પછી આપણે સૂચિ Y માં સૂચિ ઉમેરશું. તેથી, એસ વાય હવે રેખા રેખામાં ક્યૂ વાય અને આર વાયમાંથી કાઢેલા તળિયેથી ઉપરના આ પેનની અંદર પોઈન્ટની સૉર્ટ કરેલી સૂચિ હશે. હવે, આ સૂચિની અંદર હવે આ બૉક્સીસ ઑર્ડર કરવામાં આવશે. તેથી, જો હું તેમને ફક્ત સ્કેન કરું તો મને તે મળશે કે હું આ બધા ... તેથી, તેઓ સંભવતઃ 4 પોઈન્ટ હશે જે આ 4 બોક્સમાંથી આવે છે, 4 આ 4 બોક્સમાંથી અને બીજું. તેથી, હું આ સૂચિમાં આગામી 15 મુદ્દાઓને ચોક્કસપણે શોધી શકું છું અને આની સાથે તુલના કરી શકું છું. તેથી, આ રેખીય સ્કેન છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:15)

તેથી, બીજા શબ્દોમાં કહીએ તો આપણી પાસે આ નીચેની એલ્ગોરિધમ છે, તેથી અમે ધારીએ છીએ કે અમે કોઈ સમસ્યા સેટ કરી છે. તેથી, તે પીને બે નકલોમાં વિભાજિત કરવામાં આવ્યો છે પી એક્સ અને પી વાય તેને x દ્વારા સૉર્ટ કરો, તેને y દ્વારા સૉર્ટ કરો. હવે, જો આપણી પાસે 3 થી ઓછા બિંદુઓ હોય, તો 3 પોઈન્ટ ઓછાં હોય, તો આપણે ફક્ત બ્રુટ ફોર્સ દ્વારા તેને નજીકના જોડીની ગણતરી કરીએ અને જવાબ આપીએ. તેથી, જવાબમાં જોડી પરની અંતર, અંતર પરના બિંદુઓનો સમાવેશ થાય છે, જો તે 3 કરતા વધારે હોય, તો પછી આપણે આ પુનરાવર્તિત વસ્તુ કરીએ છીએ. તેથી, આપણે પી એક્સ અને પી વાય થી કહ્યું છે કે આપણે ક્યુ એક્સ, ક્યૂ વાય અને આર એક્સ, આર વાય બનાવ્યું છે અને આ

આપણે કહીએ છીએ કે આપણે રેખીય સમય કરી શકીએ છીએ. પછી, આપણે આ ઉકેલ, ક્યૂ અને આર ની હલ સમસ્યાને વારંવાર ઉકેલીએ છીએ અને આમાંથી બે અંતર ડીએક્સ અને ડી આર મેળવશું, આપણે d ને ડી અને ડી આર ની લઘુતમ ગણીશું. તેથી, આનો ઉપયોગ કરીને આપણે આ ઝોનને તે ઝોનની અંદરના બધા બિંદુઓની સૂચિ સેટ કરો અને પછી તે સૂચિને સ્કેન કરશે જે ઝોનમાં છે જે પોઈન્ટ છે, જે નજીકના અંતરે એસ. એસ છે. હવે, જો ડી એસ ક્યુ કરતાં વધુ હોય તો લઘુતમ ડી ક્યુ અને ડી આર, તો પછી કંઈપણ યોગદાન આપતું નથી, પરંતુ જો તે ઓછું હોય તો સાચા જવાબ તરીકે આપે છે. તેથી, આપણે કાં તો ક્યુ ક્યૂને ડાબી બાજુથી ઉત્પન્ન કરેલ ઉત્પાદન અથવા જમણી બાજુના ઉત્તરાર્ધ ઉત્પાદન અથવા ઝોન વચ્ચેના અમારા દ્વારા ઉત્પન્ન કરેલ જવાબને આધારે પાછા ફરવું પડશે. તો, આ મૂળભૂત એલ્ગોરિધમ છે જે તમારે થોડું વધુ કામ કરવું પડશે વાસ્તવમાં તે યોગ્ય રીતે કોડ કરો, પરંતુ આ એલ્ગોરિધમનો એકંદર માળખું આપે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 15:43)

તેથી, આપણી પાસે બિન રિકર્સિવ ભાગ છે જે પ્રથમ પી અને પી વાયની પ્રારંભિક સૉર્ટની ગણના કરે છે, આ એન લોગ એન ટાઈમ ફોર્સ પર ઓર્ડર લે છે. કારણ કે, ટેક્સ્ટ n લોગ n સમય સૉર્ટ કરીને પછી આ પુનરાવર્તિત વસ્તુને સેટ કર્યા પછી સમગ્ર રિકર્સિવ સમસ્યા n પોઈન્ટને n દ્વારા 2 ના બે ભાગમાં વિભાજિત કરે છે. તેથી, આપણી પાસે પરિચિત વસ્તુ છે જે T ની n એ 2 times T ના n દ્વારા 2 છે. અને પછી કારણ કે તમામ સમય સેટિંગ અને આ બધા સંયોજન સમય રેખીય છે, અમારી પાસે મર્જ સૉર્ટ માટે બરાબર પુનરાવર્તન છે. તેથી, તેથી, પુનરાવર્તિત ભાગ પણ ઓર્ડર n લોગ n તરીકે આપે છે, તેથી આપણી પાસે પ્રારંભિક સૉર્ટિંગ તબક્કો છે જે n લોગ n છે, આપણી પાસે એક પુનરાવૃત્ત ભાગ છે જે n લોગ n છે અને તેથી, આ અલ્ગોરિધમ એ n લોગ n છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 01
લેક્ચર - 39
સર્ચ ટ્રી(Search Trees)

હવે આપણે સર્ચ ટ્રી(Search Trees) નામની અન્ય માહિતી માળખું પર ધ્યાન આપીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:05)

તેથી, શોધ વૃક્ષોને પ્રોત્સાહિત કરવા દો આપણે નીચેના ઉદાહરણને ધ્યાનમાં લઈએ. તેથી, ધારો કે તમે એર ટ્રાફિક કંટ્રોલર છો અને તમે એક રન રસ્તાની ઍક્સેસને નિયંત્રિત કરો છો, ફ્લાઈટ્સને જમીન અને ટેકઓફ કરવાની જરૂર છે. તેથી, એર ટ્રાફિક કંટ્રોલ ટાવર અપેક્ષિત આગમન અને પ્રસ્થાન સમય વિશે એર ક્રાફ્ટથી વિનંતી પ્રાપ્ત કરે છે અને આ વિનંતી રેન્ડમ સમયે આવે છે. તેથી, એવું નથી કે તેઓ વાસ્તવિક ઘટનાના સમય માટે આવે છે. તેથી, અમને પહેલીવાર 16:23 વાગ્યે ચેન્નાઈમાં ઉતરાણ માટે વિનંતી મળી શકે છે અને પછીથી અમને 16:12 વાગ્યે ટેકઓફ માટે વિનંતી મળી શકે છે જે વાસ્તવમાં અગાઉની છે. અને પછી, અમને 16:55 વાગ્યે આગમનની બીજી માહિતી મળી શકે છે, પછી 16:18 વાગ્યે અન્ય આગમનની માહિતી અને નિયંત્રણ ટાવર સાથેની વ્યૂહરચનાનું પાલન કરવાનું માનવામાં આવે છે, જેનો પ્રારંભિક ઉપયોગ સમય સાથે ફ્લાઈટને પ્રાથમિકતા આપવાનો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:59)

તેથી, આ અગ્રતા કતાર છે, દરેક વિનંતી કતારમાં જાય છે, પરંતુ તેની આવશ્યકતાના આધારે તેની પ્રાધાન્યતા હોય છે. તેથી, અમે એક મિનિટ લિપ રજૂઆત આપી શકીએ છીએ અને જો તમે 6 વિનંતીને ક્રમમાં ગોઠવો છો કે તે આવે છે, તો અમે તેને લઘુ લિપમાં શામેલ કરીએ છીએ, પછી અમે જમણી બાજુએ જોયેલી મિની લિપ મેળવીશું. અને તેથી આમાં, કારણ કે 16:12 આ 6 ની વચ્ચેની સૌથી પહેલી ઘટના બનશે, તે આપમેળે રૂટ પર જશે અને તે આપણા માટે ઉપલબ્ધ થશે, પછી પ્રક્રિયા માટે આગળ વધશે અને પછી તે ઘટનાથી 2 મિનિટ પહેલા. થઈ શકે છે, અમે ઉડાનને મંજૂરી આપતા ફ્લાઈટને સિગ્નલ મોકલીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:42)

તો, માની લો કે અમારી પાસે આ વધારાની જરૂરિયાત છે, ન ફક્ત પ્રારંભિક ઈવેન્ટને પ્રાથમિકતા આપવાનું છે, પણ આપણે કેટલાક નાના વિમાનોને ટાળવા માંગીએ છીએ રન માર્ગ પર collusions કોઈપણ શક્યતા. તેથી, અમે એમ કહીએ છીએ કે રન કરવા માટેના 2 વિમાનોનો ઉપયોગ ઓછામાં ઓછો 3 મિનિટનો હોવો આવશ્યક છે. હવે, આ ઉદાહરણમાં આપણે જોઈ શકીએ છીએ કે 16:53 અને 16:55 વાગ્યે 2 ઈવેન્ટ્સ દ્વારા નિયમનું ઉલ્લંઘન થાય છે. તેથી, શું થાય છે તે છે કે આપણે 16:55 પછી વિનંતી કરી કે અમે 16:55 પછી વિનંતી સ્વીકારીએ અથવા વિનંતી સ્વીકારીએ.

આ બિંદુએ, આપણે તપાસ કરીશું કે તે હાલની બધી નજીવી વિનંતી કરતા ઓછામાં ઓછા 3 મિનિટ છે, જો નહીં તો અમે પાઈલટને સંદેશ મોકલવો જોઈએ 16:55 કહેવું શક્ય નથી, તમારે તમારા ટેકઓફને ખસેડવા અથવા ઉતરાણ 16 સુધી કરવું જોઈએ : 56 અથવા પછીથી. હવે, કમનસીબે લિપ ડેટા સ્ટ્રક્ચરમાં આ કરવા માટેનો કોઈ સરળ રસ્તો નથી. તેથી, જ્યારે 16:55 લિપમાં ઉમેરવામાં આવે છે, ત્યારે તેને હીપમાંના બધા ઘટકો સામે સ્કેન કરવું પડશે કે જેથી તે 3 મિનિટની અંદર અથવા તેમાંથી કોઈ પણ અંદર હોય. તેથી, આ રેખીય સ્કેન હશે, તેથી અન્ય તમામ ઑપરેશન લોગરિધમિક માટે લિપ પર શામેલ કરો અને કાઢી નાખો. પરંતુ, જો તમે આ અવરોધ લાદવા માંગતા હોવ કે જે એકબીજાના 3 મિનિટની અંદર લિપમાં તત્વો તરફ જાય, તો અમે દરેક દાખલ કરવા માટે વધારાની રેખીય સમયનો ખર્ચ ઉમેરીશું અને આ પછી આ પ્રકારના ડેટા માટે એક ઢગલો અવિશ્વસનીય બનશે. .

(સ્લાઈડસમયનો સંદર્ભ લો: 03:10)

તેથી, આ પ્રકારની જરૂરિયાતની ગણતરી કરવાની એક રીત એ પુરોગામી અને અનુગામીની તપાસ કરવાનો છે. બીજા શબ્દોમાં કહીએ તો, મૂલ્ય સમૂહની સૂચિ આપેલ કોઈપણ મૂલ્ય માટે આપણે નજીકના નાના મૂલ્યના સંદર્ભમાં અગાઉની ગણતરી કરી શકીએ છીએ અને પછીના અનુગામી નજીકના મોટા મૂલ્યની શું છે. તેથી, જો તમે ઉદાહરણ તરીકે 16:18 જુઓ, તો પછીનું નાનું મૂલ્ય અહીં છે, તેથી આ પુરોગામી 16:12 છે અને પછીનું મોટું મૂલ્ય અહીં છે, તેથી આ અનુગામી છે. હવે, એવું લાગે છે કે પૂર્વગામી અને અનુગામી વચ્ચેની વૃક્ષની રચનાના સંદર્ભમાં લિપમાં કેટલાક સારા સંબંધો છે. પરંતુ, જો તમે ઉદાહરણ માટે અલગ મૂલ્ય જુઓ છો, 16:53 તો આપણે જોશું કે પુરોગામી વૃક્ષની જુદી જુદી શાખામાં રહે છે અને અનુગામી તે જ શાખામાં રહે છે. તેથી, લિપ અને પૂર્વગામીના વૃક્ષની માળખા અને ઉત્તરાધિકારી સંબંધ વચ્ચેની કોઈ સીધી સંબંધ નથી, જેની અમને જરૂર છે. જો કે, જો આપણે પુરોગામી અને ઉત્તરાધિકારીની ગણતરી કરી શકીએ, તો ધ્યાન આપવાની મહત્વપૂર્ણ બાબત, પછી આપણે આ સમસ્યાને હલ કરી શકીએ છીએ. કારણ કે, એક વાર આપણે અનુગામી અથવા પુરોગામી જોવું જોઈએ અને અમને કંઈક એવું લાગે છે જે 3 મિનિટની અંદર છે, તો અમે તેને ઠીક કરી શકીએ છીએ, ચેતવણીને ભૂલ કરી શકીએ છીએ અને તે એરક્રાફ્ટને સંકેત આપી શકીએ છીએ કે વિનંતી બંધ થઈ નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:30)

તેથી, જો તમે આ માહિતીને જાળવી રાખવા માંગતા હો, તો આપણે અત્યાર સુધી જોયેલી વિવિધ ડેટા સ્ટ્રક્ચરોને અજમાવી જુઓ અને જુઓ. તો, આપણી પાસે અચોક્કસ એરે, સૉર્ટ કરેલ એરે અને મિન હીપ્સ છે, હવે આ ચોક્કસ માળખાંમાં આપણે સીધી કાઢી નાખો મિની અથવા ડીલીટ મેક્સ પર નજર રાખતા નથી. પરંતુ, બે જુદા જુદા ઓપરેશન્સ જે ઓછામાં ઓછા અને મહત્તમ અને એક જે તર્કને તદ્દન અલગ રીતે કાઢી નાખે છે તેની તપાસ કરે છે. તેથી, મિનિ અને મેક્સ સુધી જેટલા મહત્તમ અને મહત્તમ જવા માટે, તે સ્પષ્ટ છે કે શ્રેષ્ઠ ડેટા માળખું એરેને સૉર્ટ કરેલું છે, કારણ કે આ મૂલ્યો એરેના અત્યંત અંતર પર છે. સૌથી ખરાબ ખરાબ ક્રમાંકિત એરે છે, કારણ કે તે ગમે ત્યાં હશે અને બંને રેખીય સમય તરફ જોશે અને પછી અમારી પાસે લઘુ ઢગલો અથવા મહત્તમ ઢગલો છે તેના પર આધાર રાખીને, તેમાંથી એક સારો છે અને બીજો ખરાબ છે. તેથી, એક મિનિટ લિપમાં આપણે લઘુતમ રુટ સાથે શોધી શકીએ છીએ, મહત્તમ ગમે ત્યાં હોઈ શકે છે અને તે જ રીતે મહત્તમ લિપમાં, મહત્તમ રૂટ પર હશે, લઘુત્તમ ગમે ત્યાં હોઈ શકે છે. તેથી, જો તમે આગળના

ઓપરેશનો સેટ જુઓ છો, તો આ શામેલ છે અને કાઢી નાખો, હવે અહીં તે તારણ આપે છે કે કોઈ અનૉર્ટ કરેલ એરે શામેલ કરવા માટે સારું છે, કારણ કે, આપણે તેને અંતમાં ટિક કરી શકીએ છીએ. કાઢી નાખવા માટે, આપણે એરેથી પસાર થવું પડશે અને મૂલ્યને જોઈએ છીએ અને તેને દૂર કરવું પડશે, તેથી તે લીનિયર સમય લેશે. સૉર્ટ કરેલા એરેમાં બંને ધીમું રહેશે, કારણ કે આપણે ક્યાં તો યોગ્ય સ્થાને શામેલ કરવું પડશે, જેમ કે શામેલ સૉર્ટમાં અથવા જ્યારે અમે કાઢી નાખીએ છીએ ત્યારે અમને એરેને સંકોચવું પડશે જે રેખીય સમય લેશે. હવે, આપણે જાણીએ છીએ કે એક મિનિટ લિપમાં શામેલ લઘુગણક છે, કાઢી નાખો પણ લઘુગણક છે જો કે તે મિનિટે કાઢી નાંખે છે. તેથી, આને બે પગલાં ભાંગી શકાય છે, તેથી વર્તમાન ન્યૂનત્તમ એકંદર માઈનસ 1 પર અપડેટ કરો અને પછી કાઢી નાખો. તેથી, તમે વર્તમાન કિંમતને કાઢી શકો છો જેને તમે કાઢી નાખવા માંગો છો, તેને ફરીથી સેટ કરો મૂલ્ય છે, તેથી તે ન્યૂનતમ બને છે, તે કિસ્સામાં જ્યારે તમે નાના મૂલ્ય પર અપડેટ કરો ત્યારે જોયું, તે રૂટ સુધી ફેલાશે. અને પછી, જો તમે મિનિટને કાઢી નાખો તો તે મૂલ્ય અદૃશ્ય થઈ જાય છે અને તેથી અમે લોગ n ના બે પગલાઓમાં અસરકારક રીતે અસર કરીશું, તેથી આપણે તેને કાઢી નાખીશું, તેથી એકંદરે તે લોગ n છે. અને હવે આપણે સૉર્ટ કરેલી સૂચિમાં પુરોગામી અને ઉત્તરાધિકારીને કહ્યું છે કે તેઓ ગોઠવણ કરે છે, તેથી આ બંને સતત કામગીરી છે. બીજી બાજુ, કોઈ નામ વગરની સૂચિમાં આપણે સંપૂર્ણ એરેને જોવું જોઈએ અને અમે એક લિપમાં પણ કહ્યું હતું કે, આપણે સ્કેન કરવું પડશે. કારણ કે, કોઈ ચોક્કસ ઓર્ડર નથી કે જેમાં લિપબંધ તત્વોને લિપમાં સંગ્રહિત કરવામાં આવે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:59)

તેથી, આપણે આજે અને બાયનરી શોધ વૃક્ષો અને દ્વિસંગી શોધ વૃક્ષમાં જોવાનું ચાલુ રાખીએ છીએ, આપણે દલીલ કરવા જઈ રહ્યા છીએ કે આ તમામ કામગીરી વાસ્તવમાં લઘુગણક છે, જો કે આપણે દ્વિસંગી શોધ વૃક્ષને જાળવી રાખીએ સારી રચના સાથે. અને આપણે શોધ નામના ઑપરેશનને પણ જોશું, જે મૂલ્ય માટે શોધે છે અને આ પણ લઘુગણક હશે. તેથી, અમે બાયનરી શોધ વૃક્ષમાં આ બધા 7 ઑપરેશંસને ઑપ્ટિમાઈઝ કરી શકશો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:27)

તેથી, દ્વિસંગી વૃક્ષ ફક્ત એક વૃક્ષ છે જે રુટ સાથે છે જેમાં દરેક નોડમાં 1, 2 અથવા 3 બાળકો હોય છે, ઢગલો એક ખૂબ જ ખાસ પ્રકારનો દ્વિસંગી વૃક્ષ છે, જેને આપણે ઉપર ભરો જમાણે ડાબેથી જમાણે, પરંતુ સામાન્ય રીતે દ્વિસંગી વૃક્ષની કોઈ મર્યાદા હોતી નથી. તેથી, આપણી પાસે દરેક નોડ પર મૂલ્યો છે અને અમે માનીશું કે ફક્ત પેરેન્ટ પાસેથી જતા બાળક પાસે જવાનો માર્ગ નથી, પણ બાળક પાસે પેરેન્ટ પાસે જવાનો પણ અમારો માર્ગ છે. તેથી, દરેક નોડ જે આપણે ધારણ કરવા જઈ રહ્યા છીએ તેના માટે એક લિંક હશે પેરેન્ટ, ડાબું બાળક અને યોગ્ય બાળક, આ લિંક્સ કદાચ ગુમ થઈ શકે છે, પરંતુ જો તે ત્યાં છે તો તે ત્રણેય છે. તેથી, આપણે વૃક્ષને નીચે અથવા નીચે નીચે બે નોડો તરફ નિર્દેશ કરી શકીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:10)

તેથી, ફક્ત પરિભાષાના સંદર્ભમાં રુટ સૌથી વધુ નોડ છે, તેનામાં કોઈ પિતૃ નથી, પર્ણ એ કોઈ નોડ છે જે કોઈ બાળકો નથી અને કોઈ પણ બિંદુએ, જો તમે નોડ જુઓ , તો તે તેના પેરેન્ટ છે અને તે બાળકને છોડી દે છે અને તે યોગ્ય બાળક છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 08:28)

તેથી, હવે આપણે દ્વિસંગી વૃક્ષ લઈએ છીએ અને આપણે મૂલ્યોની અવરોધ લાવીએ છીએ. તેથી, એક હિપમાં યાદ રાખો કે આપણે સૌપ્રથમ માળખાગત સ્થિતિ મુજબ એક વૃક્ષ પર ઉપરથી નીચે ડાબેથી જમણે ભરી દીધું છે અને પછી અમે કહ્યું કે, ત્યાં એક ઢગલો મિલકત છે જે મહત્તમ ઢગલો અથવા લઘુ ઢગલો કહે છે, ક્યાં તો નોડ તેના કરતાં મોટો છે તે 2 બાળકો છે અથવા તે 2 બાળકો કરતા નાની છે. તેથી, તેમની પાસે માળખા પર મિલકત છે અને તે મૂલ્ય પરની સંપત્તિ હતી, તેથી તેઓ એકબીજા પ્રત્યે આદર સાથે કેવી રીતે ગોઠવાય છે. તેથી, દ્વિસંગી શોધ વૃક્ષમાં, દ્વિસંગી વૃક્ષની મનસ્વી રચના હોઈ શકે છે કે મૂલ્યો ચોક્કસ રીતે ગોઠવવામાં આવે છે. તેથી, દરેક નોડ માટે મૂલ્ય વી સાથે, v ના બધા નોડ્સ, v કરતાં નાના, ડાબી ઉપ ટ્રીમાં છે અને વી કરતાં મોટા બધા મૂલ્યો અધિક ઉપ ટ્રીમાં છે. અને આપણે સામાન્ય રીતે આપણે ધારીશું કે કોઈ ડુપ્લિકેટ મૂલ્યો નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:20)

તેથી, ઉદાહરણ તરીકે, જો તમે રુટ નોડ 5 જુઓ છો, તો 5 ના ડાબા ઉપ ટ્રીમાં 1 થી 4 મૂલ્યો શામેલ છે જે આ વૃક્ષના બધા ગાંઠો 5 કરતાં નાના છે.

(સંદર્ભઆપો સ્લાઈડ ટાઈમ: 09:26)

અને જમણા પેટા વૃક્ષમાં 8 અને 9 શામેલ છે, હવે આ તે મિલકત છે જે સમગ્ર વૃક્ષમાં વારંવાર સંતોષાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:33)

તેથી, જો તમે નોડ 2 ને ઉદાહરણ તરીકે જુઓ છો, તો તે બાળક છોડી દીધું છે, બાકી પેટા વૃક્ષો ફક્ત એક જ નોડ 1 છે જે 2 ની રુટવાળા વૃક્ષ કરતાં નાના છે અને તે યોગ્ય છે વૃક્ષો મૂલ્ય 4 જે યોગ્ય છે તે 2 કરતાં મોટી છે, આ ઉપ ટ્રી કરતાં 2 ની માત્રા એકમાત્ર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:49)

એ જ રીતે, જો તમે 8 માં 8 ની સાબર વૃક્ષમાં 8 કરતા વધુ માત્ર એક જ મૂલ્ય જુઓ તો તેનું મૂલ્ય 9 છે અને તે પછીથી આપણે જમણી ઉપ ટ્રીમાં જઈએ છીએ અને આપણી પાસે ડાબી ઉપ ટ્રી ખાલી. તેથી, આપણે આ પ્રકારના અંતરને હિપમાં વિપરીત કરી શકીએ છીએ, આપણી પાસે એક તફાવત હોઈ શકે છે, પરંતુ આપણી પાસે સમાન સમાન મિલકત છે જે પેટા વૃક્ષમાંના કોઈપણ નોડ પર છે, નોડ કરતા નાના બધા મૂલ્ય ડાબી બાજુ છે, અને બધી કિંમતો વધુ મોટી છે. જમણી બાજુએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:13)

તેથી, સામાન્ય રીતે આપણે તેને એક લિંક માળખું તરીકે અમલમાં મૂકીશું, જ્યાં આપણી પાસે દરેક નોડ મૂલ્ય કરતાં અન્ય 3 ક્ષેત્રો સાથે હશે. તેથી, અમે પેરેન્ટ તરફ ધ્યાન દોર્યું છે, ડાબી તરફ નિર્દેશ કરેલા અને યોગ્ય બાળક તરફ ધ્યાન દોર્યું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:31)

તેથી, પ્રથમ વસ્તુ જે આપણે દ્વિસંગી શોધ વૃક્ષ સાથે કરી શકીએ તે છે તે સૂચિબદ્ધ ક્રમમાં મૂલ્યો છે તે સૂચિબદ્ધ કરવાનું છે, આને ઑર્ડર ટ્રાવર્સલ કહેવામાં આવે છે. તેથી, ઓર્ડર ટ્રાવર્સલ રિકર્સિવ ટ્રાવર્સલ છે તે વૃક્ષ દ્વારા ચાલવાનો માર્ગ છે. તેથી, આપણે પ્રથમ ડાબા ઉપ ટ્રી દ્વારા, પછી વર્તમાન નોડ અને પછી જમણે સબ વૃક્ષથી પસાર થાય છે. ઉદાહરણ તરીકે, જો તમે ઓર્ડર ટ્રાવર્સલ અહીં કરો છો, તો અમે રુટ પર પ્રારંભ કરીએ છીએ અને પછી આપણે પહેલા લીફ્સ ઉપર ઝાડ પર ચાલવું પડશે, તેથી અમે ડાબી બાજુએ ચાલીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:03)

અને પછી, ફરી એકવાર આપણે વૃક્ષની ડાબી બાજુએ જવું જોઈએ, જેથી તમે ડાબી બાજુએ જશો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:07)

અને હવે, ત્યાં કોઈ ડાબે બાળક નથી, તેથી હવે અમે છાપશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:10)

અને બેકઅપ ખસેડો, કારણ કે કોઈ યોગ્ય બાળક નથી, હવે આપણે 2 પ્રિન્ટ કરીએ છીએ

(સ્લાઈડસમયનો સંદર્ભ લો: 11:15)

અને જમણી ઉપ ટ્રી પર જાઓ અને હવે ફરીથી કારણ કે 4 પાસે ડાબે અથવા જમણે સબ ટ્રી નથી, તેથી છાપશે અને બૅકઅપ લેશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:20)

અને કારણ કે આપણે પહેલાથી જ સમાપ્ત કરીએ છીએ તે જ રુટનો અંત આવશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:26)

શું આપણે 5 પેટા વૃક્ષ તરફ જવાનું છીનવીશું, ત્યારબાદ ત્યાં ડાબે ઉપ ટ્રી 8 છાપશે અને જમણી ઉપ ટ્રી પર જશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:30)

અને અંતે, છાપશે 9.

(સ્લાઈડસમયનો સંદર્ભ લો: 11:33)

તેથી, આ ગુણધર્મને કારણે તેને જોવાનું સરળ છે, આપણે જાણીએ છીએ કે ડાબી બાજુની દરેક વસ્તુ તે મૂલ્ય કરતા ઓછી છે અને જમણી બાજુની દરેક વસ્તુ મૂલ્ય કરતાં મોટી છે. તેથી, આ એક પુનરાવર્તન જેવું છે, આ એક અર્થમાં ઝડપી સૉર્ટનું વિભાજન સમાન છે. તેથી, આપણે પહેલાથી જ નાના વેલ્યુનાં વેલ્યુઓ ડાબે કરી દીધી છે. તેથી, જમણી બાજુએ મોટા મૂલ્યો છે અને જો તમે આ પાર્ટીશનિંગને વારંવાર સૂચિબદ્ધ કરવા માટે અનુસરો છો તો અમે કરીશું; દેખીતી રીતે, સૉર્ટ કરેલા એરેમાં મૂલ્યો મેળવો.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:59)

તેથી, હવે એક વૃક્ષની શોધ કરી રહ્યા છે, બાયનરી શોધ વૃક્ષમાં મૂલ્ય બાયનરી શોધ જેવું છે. યાદ રાખો, દ્વિસંગી શોધમાં તમારી પાસે એક એરે છે અને પછી અમે મિડપોઈન્ટ પર પ્રારંભ કરીએ છીએ અને પછી જો તમને તે મળે તો તમે હા કહી શકો છો; નહિંતર, જો તે નાનું હોય તો તમે જાઓ અને ડાબી બાજુ જાઓ; અન્યથા, એક મોટો આપણે જમણી તરફ જુએ છે. તેથી, અમારી પાસે ખૂબ જ સમાન વસ્તુ છે, તેથી આપણે ઝાડમાં મૂલ્ય વી શોધવા માંગીએ છીએ, જો વૃક્ષ અસ્પષ્ટ છે, તો આપણે કહીએ છીએ કે તે ત્યાં નથી, તેથી અમે ખોટા પાછા ફર્યા છે. જો વર્તમાન નોડ મૂલ્ય ધરાવે છે, તો બીજી તરફ આપણે તેને સ્થાનાંતરિત કર્યા છે અને 2 પરત કરીએ છીએ, જો તમે તેની સ્થાપના ન કરી હોય તો, ત્યારબાદ આપણી પાસે એક વૃક્ષ છે, આપણે જોવું જોઈએ કે મૂલ્ય વર્તમાન મૂલ્ય કરતાં નાના છે, તે નાના છે આપણે વારંવાર ડાબે શોધીએ અને તેની સાથે જે પણ શોધીશું તે પાછું મેળવીએ ત્યાં આપણે સાચું કહીએ, તે ત્યાં નથી મળ્યું જે આપણે વૃક્ષને કહીએ છીએ; અન્યથા, અમે વારંવાર જમણી શોધ કરીએ છીએ, તેથી આ ખૂબ સમાન દ્વિસંગી શોધ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:59)

હવે, બાઈનરી શોધની જેમ તમે આનું પુનરાવર્તિત સંસ્કરણ કરી શકો છો, તેથી તમે રૂટ પર પ્રારંભ કરો અને તેટલો લાંબો સમય સુધી તમે પાથ પર વિશ્વાસ કરશો નહીં. તેથી, વર્તમાન મૂલ્યો આપણે સાચા પાછા આપીએ છીએ; નહિંતર, તમે ડાબી તરફ જતા હતા, તમે વાસ્તવમાં રૂટ પર પ્રારંભ કરો છો અને તમે જે પાથને મૂલ્ય આપી શકો તે શોધી કાઢો તે છે. અને પછી જ્યારે તમે પહોંચશો ત્યારે નોડ હોવ, બીજી તરફ જો તમે કોઈ બિંદુ સુધી પહોંચો છો જ્યાં તમે આગળ વધી શકતા નથી, જો તમે નોડ્સથી બહાર નીકળ્યા, જો તમે બધી રીતે નીચે આવો અને પછી કહો કે ત્યાં કોઈ એક્સ્ટેંશન નથી, જ્યાં હું તે શોધી શકું પછી અમે પણ સુરક્ષિત થઈશું. તેથી, આ શોધનું સરળ પુનરાવર્તિત અને પુનરાવર્તિત સંસ્કરણ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:34)

તેથી, આગામી બે ઑપરેશન શામેલ છે અમે તમારી શોધ લઘુતમ અને મહત્તમ કરવા માંગીએ છીએ. તેથી, ટ્રીમાં ન્યૂનતમ નોડ ડાબી બાજુનો નોડ છે, અન્ય વૉર્ડ્સમાં તે નોડ છે જે તમે જાઓ છો, અને જો તમે ફક્ત ડાબી બાજુએ જ અનુસરી શકો છો. તેથી, આ કિસ્સામાં ડાબી બાજુ 5 જવા માટે હું 3 થી જાઉં છું, 3 થી 3 જાઉં તો હું 1 જાઉં છું અને પછી ડાબી બાજુ જઈ શકતો નથી, તે એક જ નીચે છે, તે જમણી તરફ છે, પણ ઓછામાં ઓછું નથી. ડાબું એક વૃક્ષ પર ન્યૂનતમ તત્વ હોવા જ જોઈએ, કારણ કે જે કંઈક કરતાં નાના છે તે 2 રહેશે, તેથી જો હું ડાબી ન જઈ શકું તો તે 1 કરતા ઓછું નથી. તેથી, વૃક્ષનું ડાબે સૌથી મૂલ્ય શા માટે છે લઘુત્તમ હશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:14)

તેથી, આપણે તેને સરળતાથી શોધી શકીએ છીએ, હવે આપણે આનો ઉપયોગ સામાન્ય રીતે વૃક્ષો ખાલી ન હોવાનું માનતા, તે થોડું કોડિંગ સરળ બનાવે છે. તેથી, વૃક્ષો ખાલી ન હોવાનું માનતા આપણે કોઈ ખાસ સ્થિતિ તપાસવાની જરૂર નથી અને જ્યારે તે ખાલી નથી, ત્યારે તે ખાલી છે. તેથી, આપણે માનીએ છીએ કે તે ખાલી નથી, તેથી જો આપણે ડાબી બાજુ જઈ શકીએ તો આપણે કરી શકીએ છીએ, તેથી જો ટી ટૉટ બાકી હોય તો નલ હોય તો આ લઘુત્તમ મૂલ્ય છે અને આપણે અન્યથા પાછા ફરો, આપણે વારંવાર ડાબી બાજુના ન્યૂનતમ મૂલ્ય માટે શોધ કરીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:47)

અને ફરી એકદમ સરળ પુનરાવર્તિત સંસ્કરણ છે, આપણે વર્તમાન વસ્તુથી પ્રારંભ કરીએ છીએ અને આપણે જેટલું કરી શકીએ ત્યાં સુધી ડાબી બાજુએ જતા રહીશું. તેથી, જ્યાં સુધી આપણે તે નહીં કરીએ અને જ્યારે આ બિંદુ સુધી પહોંચીએ ત્યારે ટી ટૉટ બાકી કેમ નથી, આપણે આ લૂપથી બહાર આવીએ અને તમે આ બિંદુએ વેલ્યુ પરત કરો, તેથી અહીંથી આપણે 5 ટી ડોટ ડાબેથી શરૂ કરીશું 3, ટી ડોટ ડાબે 1 છે, ટી ડોટ ડાબે છે. તેથી, આ બિંદુએ આપણે બંધ કરીએ છીએ, તેથી ટી 1 તરફ પોઈન્ટ કરે છે, તેથી ટી ડોટ વેલ્યુ 1 છે અને આ વળતર દ્વારા મૂલ્ય છે, તેથી આપણે લઘુત્તમ શોધી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:14)

તેથી, સમપ્રમાણતા મુજબ મહત્તમ, વૃક્ષ પર સૌથી વધુ યોગ્ય મૂલ્ય છે, કારણ કે તમે જમણી બાજુએ જાઓ છો, તો તમે વધુ મોટા થઈ શકો છો, જો તમે વધુ યોગ્ય રીતે જઈ શકતા નથી, તો તે નોડ કરતાં મોટું નથી. તો, અહીં 5 થી આપણે 7 થી 9 અને 9 પર જઈ શકીએ છીએ, તે સૌથી મોટું મૂલ્ય છે, કારણ કે આગામી નોડ જે વૃક્ષની નીચે 9 કરતા નીચે હોવા છતાં તે ડાબી બાજુથી છે અને તેથી 9 કરતા ઓછું છે. તેથી, આપણી પાસે સપ્રમાણ રિકર્સિવ સોલ્યુશન મેક્સવલ, જો જમણી બાજુ ન હોય તો આપણે જમણી બાજુએ તપાસ કરીશું તો જમણી બાજુ ન હોય તો આપણે મહત્તમ મૂલ્ય પરત કરીશું, પછી આપણે વારંવાર મહત્તમ માટે અધિકાર શોધ કરીશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:46)

અને ફરીથી તેનું પુનરાવર્તિત સંસ્કરણ, આપણે ફક્ત જમણી બાજુના ચેઝ પોઈન્ટને અનુસરીએ છીએ. તેથી, જ્યાં સુધી જમણી ટી ડોટ જમણી નહી હોય ત્યાં સુધી આપણે ટી થી ટી ડોટ જમણી તરફ જઈએ છીએ અને જયારે આપણે મહત્તમ વેલ્યુ ઉપર આગળ વધી શકતા નથી, તેથી આપણે તે વેલ્યુ પરત કરીએ છીએ. તેથી, આ બિંદુએ આપણે શોધ વૃક્ષમાં ન્યૂનતમ અને મહત્તમ શોધી કાઢ્યા છે. તેથી, આ ત્રણ કાર્યવાહી અમે કરી છે, હવે જો તમને યાદ છે કે અમે પૂર્વગામી અને અનુગામી છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 16:20)

તેથી, ચાલો પહેલા અનુગામીને જોઈએ, તેથી યાદ રાખો કે x માં અનુગામીસૂચિ, આગામી મૂલ્ય હોવાનું માનવામાં આવે છે, તે સૂચિ પછી x પછીનું સૌથી નાનું મૂલ્ય છે. તેથી, જો આપણે સૉર્ટ કરેલ ક્રમમાં તે છાપીશું, તો તે તે મૂલ્ય હશે જે x ની જમણી બાજુએ દેખાશે અને ક્રમમાં વૃક્ષની ઓર્ડર ટ્રાવર્સલ સૉર્ટ કરેલ ક્રમમાં મૂલ્યોને છાપે છે. તેથી, તે અસરકારક છે જે x પછી તરત જ છાપશે. તેથી, આપણે જાણીએ છીએ કે જે ક્રમમાં કામ કરે છે, તે ડાબી પેટા વૃક્ષને છાપે છે પછી તે x ને છાપે છે, પછી તે યોગ્ય પેટા વૃક્ષને છાપે છે. તેથી, જો તમે x પછી તરત જ આવવા માંગતા હો, તો તે અહીં પ્રથમ મૂલ્ય હશે. તેથી, જમણા પેટા વૃક્ષમાં સૌથી નાનું મૂલ્ય, બીજા શબ્દોમાં, સાચા ઉપ ટ્રીની લઘુત્તમ અને આપણે પહેલાથી જ એક વૃક્ષની ન્યુનતમ ગણતરી કેવી રીતે કરવી તે જાણીએ છીએ, પરંતુ તે શક્ય છે કે આપણે મૂલ્યના અનુગામીની સાથે અધિકાર સબ વૃક્ષ નથી, તો પછી આપણે શું કરીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 17:16)

તેથી, પ્રથમ નિરીક્ષણ એ છે કે જો મૂલ્યનો કોઈ ઉપ ઉપાડ નથી, તો તે ઉપ તે વૃક્ષનો મહત્તમ હોવો આવશ્યક છે. તેથી, આ કિસ્સામાં આપણે મૂલ્ય x ને જોઈએ છીએ, તેમાં કોઈ ઉપ ઉપાડ નથી, તેથી તે મહત્તમ ઉપ ટ્રી છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:29)

તેથી, આપણે આ સબ ટ્રીને પ્રથમ ઓળખવું પડશે, તેથી આપણે પેટા વૃક્ષને કેવી રીતે ઓળખીએ, જાગશે અને શોધીશું કે આ સબ વૃક્ષ કેવી રીતે જોડાયેલ છે. તેથી, આપણે જમણી તરફના પોઈન્ટર્સના અનુક્રમે x ને ત્યાં મેળવીએ છીએ. તેથી, અમે તે જમણી તરફના પોઈન્ટરને પાછળથી અનુસરે છે અને જ્યારે આપણે ડાબે નિર્દેશકનું અનુસરણ કરીએ છીએ, ત્યારે આપણે જાણીએ છીએ કે આપણે આ ઉપ ટ્રીમાંથી બહાર આવ્યા છીએ. તેથી, તેથી તે નોડ નીચે ઉપ ઉપ ટ્રીનો રુટ હોવો જોઈએ, તે ડાબી ઉપ ટ્રી છે, તેથી તે આગલું મૂલ્ય છે. તેથી, આ સંપૂર્ણ લાલ, બ્લોક સબ ટ્રી છાપવામાં આવશે અને x સાથે સમાપ્ત થયેલ ટૂંકા ક્રમમાં, પછી રુટ છાપવામાં આવશે આથી આ નોડ છે જે આપણને અનુગામી તરીકે લેબલ કરે છે. તો, તેથી, એ તે નોડ છે જેને આપણે x માટે સફળતામાં નિયુક્ત કર્યું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:10)

તેથી, અહીં આ માટેનો એક સરળ સ્યુડો કોડ છે, તેથી આપણે નોડ ટીના અનુગામીને શોધવા માંગીએ છીએ. હવે, જો આ નોડનો યોગ્ય પોઈન્ટર હોય, તો બીજી તરફ જમણી બાજુના વૃક્ષનું જમણા મૂલ્ય પાછું મળે છે, જો તેની પાસે યોગ્ય ન

હોય તો, પછી આપણે આગળ વધવું જોઈએ અને ભાગ જ્યાં બદલાવો તે પ્રથમ સ્થાને જોઈએ. તેથી, આપણે ટી સાથે પ્રારંભ કરીએ છીએ અને પછી આપણે ગણતરી કરીએ છીએ કે તે પેરેન્ટ છે જેને આપણે હવે y કહીએ છીએ. તેથી, જ્યાં સુધી આ દિશા સાચું છે, આપણે આગળ વધી રહ્યા છીએ અને આપણે એક વધુ પિતૃ પર જઈએ છીએ અને આપણે ટી ઉપર જઈએ છીએ, તેથી આ લૂપમાં અહીં જે ચાલી રહ્યું છે. તેથી, આપણે ટી અને વાય ખસેડીએ છીએ અને પછી કોઈક સમયે આપણે move તરફ જઈએ છીએ અને વાય આ રીતે જશે. તેથી, તે વાય ડોટ બાકી રહેશે અને તેથી, આ વાય હવે આ મૂળ નોડ ટીનું અનુગામી છે અને પછી એક વિશિષ્ટ કેસ છે જ્યાં આપણે ટોચ પર પહોંચી ગયા છીએ અને આપણે ક્યારેય જમણી તરફ વળ્યાં નથી અને તે જ સ્થિતિમાં આપણે તમે વૃક્ષમાં એકંદર મહત્તમ મૂલ્યથી શરૂ કર્યું છે, તે કિસ્સામાં આપણે ત્યાં સુધી પહોંચીએ છીએ તે પછી તેમાં કોઈ સફળ નથી, આપણે ફક્ત પાછા જઈશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:19)

તેથી, ચાલો આપણે આ વિશિષ્ટ વસ્તુને જોઈએ, તેથી 3 ઘટનાઓ માટે યોગ્ય સબ વૃક્ષ અસ્તિત્વમાં છે અને આ નાનું મૂલ્ય 4 છે. તેથી, 1 ની સૌથી નાની સફળતા માટે મૂલ્યો 2, તેથી તે 7 માટે અનુગામી છે, આ ઉપ ટ્રીમાં લઘુતમ મૂલ્ય 8 છે. તેથી, 8 એ અનુગામી અથવા 7 છે, તેથી તમારા મૂળ ઉપાયમાંથી આ બધા મૂળ કેસમાંથી બહાર આવે છે, કેમ કે તમારી પાસે યોગ્ય પેટા વૃક્ષ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:42)

બીજી બાજુ, જો તમારી પાસે યોગ્ય સબ ટ્રી ન હોય તો ઉદાહરણ તરીકે તમે 2 શરૂ કર્યું, પછી તમે જાગી અને તે જમણી બાજુએ તમે જ્યાં ફેરવો છો, ત્યાં તમને લાગે છે કે 3 સતત છે અથવા 4 થી તમે જાગૃત થાઓ છો અને તમે ખૂબ જ સમય પસાર કરો છો તે તમને જણાય છે કે 5 અનુગામી છે. એ જ રીતે, હું 8 માટે તમે તરત જ જમણે ચાલુ કરો, તેથી 9 સતત અને છેલ્લે છે, 9 માટે તમે આવશે અને તમે

((સમયનોસંદર્ભ લો: 20:02))

આ નાનો કેસ. તેથી, 9 કહેશે કે ત્યાં કોઈ અનુગામી નથી, કારણ કે તે શક્ય એટલું મોટું મૂલ્ય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 20:09)

તેથી, પુરોગામી માટે પરિસ્થિતિ સમપ્રમાણ છે, તેથી જો આપણી પાસે ડાબો ઉપ ટ્રી હોય, તો પુરોગામી આમાં મહત્તમ મૂલ્ય છે. તેથી, આપણે ડાબે નીચે જઈએ અને પછી આપણે બધા જ રીતે જમણી બાજુએ જઈએ અને જો આપણી પાસે ડાબું ઉપ ટ્રી ન હોય તો તેનો અર્થ એ છે કે તે પહેલાથી જ લઘુતમ મૂલ્ય પેટા વૃક્ષો છે. તેથી, આપણે ડાબે પોઇંટર્સના આ અનુક્રમને અનુસરીને પાછળ ચાલીએ છીએ, જ્યાં સુધી આપણે બીજી રીત ચાલુ ન કરીએ ત્યાં સુધી આ નોડ આપણે પુરોગામી બનીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 20:41)

તેથી, ઉદાહરણ તરીકે જો તમે આ મૂલ્યો જુઓ જેણે ઉપ વૃક્ષો છોડી દીધા છે, તો 5 માટે આપણે ડાબી ઉપ ટ્રી નીચે જઈએ છીએ અને આપણને યોગ્ય મૂલ્ય 4 છે જે 4 છે. તેથી, 4 છે પુરોગામી 5 છે, તેવી જ રીતે 2 એ 3 અને 8 નું પુરોગામી 9 નું પુરોગામી છે, હવે આપણી પાસે આ બીજું મૂલ્ય છે. તેથી, જે આપણે ઉદાહરણ તરીકે છોડી શકતા નથી, આપણે તેને છોડી શકતા નથી 2, આપણે તેને છોડી શકતા નથી 4, આપણે તેને છોડી શકતા નથી 7, તેને છોડી શકતા નથી 8.

(સ્લાઈડસમયનો સંદર્ભ લો: 21:09)

તો, શું કરવું અમે આ કિસ્સાઓમાં કરીએ છીએ ઉદાહરણ તરીકે આપણે 2 ને શરૂ કર્યા, અને પછી આપણે ઉપર જવાનો પ્રયાસ કરીએ છીએ અને જ્યાં આપણે જમણી તરફ વળીએ છીએ ત્યાં અમને મળે છે. તેથી, અમે અહીં આવીએ છીએ અને જો તમે જાઓ છો, તો અમે શોધી કાઢીએ છીએ કે કોઈ એક પુરોગામી છે જે 4 થી આવે છે અને તેથી અમારું લક્ષ્ય જમણી બાજુએ જવાનું છે અને પછી જ્યાં આપણે પુરોગામી શોધીએ તો ડાબી તરફ વળીએ. તેથી, આ ત્રણ નોડ્સની ડાબી બાજુએ તરત જ ચાલુ કરો અને પછી 8 આપણે જમણી બાજુએ જઈએ અને જ્યાં આપણે ડાબી તરફ વળીએ છીએ ત્યાં આપણે પુરોગામી શોધીશું. તો, આ રીતે, પુરોગામી કામ કરે છે, તે સતત કાર્ય માટે બરાબર સપ્રમાણ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 21:46)

તો, હવે ચાલો જોઈએ કે વેલ્યુ કેવી રીતે દાખલ કરવી, સર્ચ ટ્રીમાં વેલ્યુ સીધું વેલ્યુ શામેલ કરવું જોઈએ, યાદ રાખવું એ માત્ર એક જ જગ્યા છે કારણ કે વૃક્ષ ક્રમમાં સૂચિબદ્ધ છે. ટૂંકા યાદી. તેથી, હવે આ મૂલ્યને ઉમેરતા પછી, તેને સેકન્ડમાં ટૂંકા સૂચિ આપવી આવશ્યક છે, તેથી તે એવું કહેવાનું છે કે ટૂંકા સૂચિમાં મૂલ્ય શામેલ કરવા માટે તે ફક્ત એક જ સ્થાન છે. તેથી, ટૂંકા સૂચિમાં દાખલ થવા જેવું છે, સિવાય કે તેને વૃક્ષને લટકવા માટે તે યોગ્ય સ્થાન શોધવું પડે. તેથી, જ્યારે આપણે આ ટ્રાવર્સલ ક્રમમાં કરીએ ત્યારે, જ્યારે અમે તેને સૂચિબદ્ધ કરીશું ત્યારે તે યોગ્ય ક્રમમાં હશે. અને તે તારણ આપે છે કે મૂળભૂત રીતે આપણે શોધવું જોઈએ કે તે ક્યાં શોધવું જોઈએ અને જો તે હાજર ન હોય, તો અમે આ શોધ ક્ષેત્રો દ્વારા ઉમેરીશું. તેથી, ઈન્સ્ટન્ટ્સ માટે જો તમે 21 શામેલ કરવા માંગો છો, તો આપણે વૃક્ષ નીચે ચાલતા જઈશું અને આપણે તે સ્થળ શોધીશું જ્યાં આપણે 21 મેળવવું જોઈએ. તેથી, તે 52 કરતા નાની છે, આપણે ડાબી બાજુએ જઈએ, 37 કરતા નાની અને ડાબી બાજુએ જઈએ. , 16 થી મોટા આપણે ઠીક જઈએ છીએ અને પછી આપણે શોધી કાઢીએ કે 28 ના તે 21 ની કોઈ કિંમત નથી, કારણ કે ત્યાં બાકી નથી, તેથી આ ડાબી બાજુએ હોવું જોઈએ, તેથી તે 28 ઉમેરીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 22:45)

હવે, ઈન્સ્ટન્ટ્સ માટે જો આપણે 65 શામેલ કરવા ઈચ્છીએ, તો પછી આપણે 52 થી જઈશું અને હવે તે 74 છે, આપણે ડાબી તરફ જોશું, જ્યાં તે ડાબી બાજુ નથી. તેથી, અમે દાખલ કરીશું

((સમય:22:59 નો સંદર્ભ લો)),

તેથી તે વૃક્ષમાંની કોઈપણ સ્થિતિમાં થાય છે, જે 28 મી, 28 જેવા પાંદડા નોડ પર હોવું જરૂરી નથી અને તે નીચે 21 શામેલ છે. . પરંતુ, 74 ને યોગ્ય બાળક હતો, પરંતુ અમે તેમાં કંઈક શામેલ કર્યું, કારણ કે ડાબું બાળક

((સમયનોસંદર્ભ: 23:11)).

હવે, આપણે કહી શકીએ કે પહેલેથી જ ત્યાં છે તે મૂલ્ય શામેલ કરવાનો પ્રયાસ કરો, મારા પ્રોમ્પ્ટમાં ઈન્સ્ટન્ટો માટે 91 શામેલ કરો. તેથી, આપણે 91 માટે શોધી રહ્યા છીએ અને તેને શોધી શકીએ, તેથી આપણે 91 ની દાખલ કરવાની અર્થઘટન કરીશું જે કંઈક કરે છે તેને ખલેલ પહોંચાડી નથી, કારણ કે આપણે પહેલા કહ્યું હતું કે આપણે ડુપ્લિકેટ વેલ્યુઝ કરવા નથી માંગતા. તેથી, જ્યારે આપણે શામેલ કરીએ છીએ ત્યારે અમે શોધવા માટે પ્રયાસ કરીએ છીએ કે સારા ક્ષેત્રો શામેલ છે કે નહીં, જો સફળ થાય તો અમે કંઈ પણ કરીએ નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 23:35)

તેથી, આ કરવા માટે આ ખૂબ સરળ રીકર્સિવ રીત છે, તેથી સૌ પ્રથમ આપણે એક ખાસ કેસ છે જે તે વૃક્ષો ખાલી છે, પછી આપણે ફક્ત નવો નોડ બનાવીએ અને તે નોડ તરફ ધ્યાન દોરો. તેથી, આ એક અલગ નોડ છે જેનું મૂલ્ય ફક્ત વાચક છે, બાકીના બધાને જમણે ડાબી અને જમણી બાજુ છે, બીજી તરફ જો આપણે તેને શોધીએ તો, આપણે કંઈ પણ નથી કરતા, હવે અમને તે મળ્યું નથી. તેથી, હવે જો વેલ્યુ નાની હોય તો વેલ્યુ જો વર્તમાન નોડ અને જો આપણું બાળક બાકી નહીં હોય, તો આપણે વાસ્તવમાં દાખલ કરીએ છીએ. જો આપણે ડાબે જવું પડે અને ડાબી બાજુ ન જઈએ, તો આપણે ડાબી બાજુએ નવો નોડ બનાવીશું અને તે નોડ પોઈન્ટ બનાવીશું જે તે પેરેન્ટ છે. તેથી, આપણે અહીં નવો નોડ બનાવીએ છીએ અને આપણે કહીએ છીએ કે તે પેરેન્ટ છે. તેથી, ટી ડોટ ડાબ ડોટ પેરેંટ ટી છે; અન્યથા, આપણે ફક્ત 2 વાર શામેલ કરીએ છીએ. તેથી, જો તમારી પાસે ડાબી બાજુ હોય તો આપણે વારંવાર દાખલ કરીએ, જો આપણી પાસે ડાબે ન હોય તો આપણે વી સાથે નોડ બનાવીએ, આ વાસ્તવમાં શામેલ ઓપરેશન છે અને અમે તેને પિતૃ દ્વારા તરફ દોરીએ છીએ. તેવી જ રીતે, જો આપણે જમણી બાજુ જવા માંગીએ છીએ અને કોઈ અધિકાર નથી, તો આપણે તેને જમણી બાજુએ શામેલ કરીએ છીએ અને અમે તે પોતાનું પોતાનું નિર્દેશ કરીએ છીએ કે તે પેરેન્ટ છે; અન્યથા, આપણે વારંવાર જમણી બાજુએ શામેલ કરીએ છીએ, તેથી ખૂબ સીધા આગળના ફંક્શનમાં શામેલ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 24:51)

તો, આપણે નોડ કેવી રીતે કાઢી શકીએ, તેથી કાઢી નાંખે છે, પરંતુ આપણે વેલ્યુ વી આપ્યો છે જે આપણને વૃક્ષમાં વી શોધી કાઢે છે, તમારે તેમાં રહેલા નોડને ડીલીટ કરવું જ પડશે. તેથી, મૂળભૂત કેસ કે જે હેન્ડલ કરવા માટે ખૂબ જ સરળ છે તે છે નોડ એ લીફ નોડ છે, કારણ કે પછી આપણે તેને કાઢી શકીએ છીએ અને પછી તે તળિયે આવેલા ઝાડની માત્રામાં પડે છે. તેથી, ઈન્સ્ટન્ટ્સ માટે જો તમે 65 કાઢી નાખવા માંગતા હો, તો અમે 65 ને શોધવા માટે તેને ડાબી અને જમણી પાથોને અનુસરવાની સામાન્ય પદ્ધતિ દ્વારા શોધીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 25:24)

અને ત્યારબાદ તે એક લીફ નોડ છે, આપણે ફક્ત આ નોડને વૃક્ષમાંથી દૂર કરી શકીએ છીએ અને કંઈ પણ થાય નહીં, તે માન્ય શોધ રહે છે. હવે, કેટલીકવાર કાઢી નાખેલ નોડમાં ફક્ત એક જ બાળક હોઈ શકે છે, દાખલા તરીકે, આપણે વિચારીએ છીએ કે 74 ને કાઢી નાખીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 25:35)

તો, આપણે 74 સુધી નીચે આવીશું, હવે આપણે આ નોડને ડીલીટ કરવા માંગીએ છીએ, પરંતુ તે યોગ્ય બાળક છે , પરંતુ હવે આપણે શું કરી શકીએ, તમે આ બાળકને પ્રોત્સાહન આપી શકો છો, તેથી આપણે ફક્ત એટલું જ કહી શકીએ કે આ લિંક સીધી આની તરફ 91 સુધી જાય છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 25:51)

તેથી, તમે માત્ર તે નોડને વચ્ચે દૂર કરો અને નોડના અનુગામીને સીધા જ 52 જોડો જે કાઢી નાખશે. તેથી, જો આમાં કોઈ એક જ સમસ્યા નથી, તો હવે બાળકને કાઢી નાખો તો, બાળકને 2 કાઢી નાંખો 2 બાળકો તરીકે. તેથી, તમે વિચારી રહ્યા છો કે તમે કાઢી નાખો 37 તે આ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 26:12)

તેથી, આપણે 37 ને ઓળખીએ છીએ 37 ને અહીં જવું જોઈએ, અમે મનસ્વી રીતે વૃક્ષને ફરીથી ગોઠવી શકતા નથી, કારણ કે આપણી પાસે 2 બાળકો હશે અને આપણે નથી જાણતા કે વિચારો સાથે શું કરવું. તેથી, હવે એક વ્યૂહરચના કે જે કામ કરે છે તે 37 ને દૂર કરવા અને તેને સ્થાનાંતરિત કરવા માટે સંપૂર્ણ બનાવવાનું છે, તે પુરોગામી અથવા અનુગામી છે. તેથી, આપણે માનીએ છીએ કે આપણે તેને ઓળખીએ છીએ તે પૂર્વેની સાથે પૂરોગામી છે તે યાદ રાખશે કે તેમાં સૌથી મોટો નોડ હશે જે ઉપ ઉપાડ છોડશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 26:40)

તેથી, અહીં તે 28 થશે, તેથી આપણે શું કરીશું, આપણે 28 ને આ નોડ પર કોપી કરીશું જે કાઢી નાખવું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 26:43)

તેથી, 37 નોડ કાઢી નખાયો છે, તે મૂલ્ય 28 દ્વારા બદલાઈ ગયું છે, હવે મૂલ્ય કેમ માન્ય છે, કારણ કે આપણે ટૂંકા ગાળાને સાચવવા માંગીએ છીએ. તેથી, જો આપણી પાસે ટૂંકા મૂલ્યોની સૂચિ હોય અને હું અહીં કંઈક કાઢી નાંખો, તો પછી શું થાય છે તે છે કે બધું જ પુરોગામીના હકમાં છે. તેથી, મેં મૂળભૂત રીતે આ બિંદુએ પૂર્વગામી ખસેડ્યું છે, જ્યાં ટૂંકા ક્રમમાં આ ઘટકો વચ્ચેનો ક્રમ સાચવો. તેથી, અહીં પુરોગામી ખસેડવું, હું તેમની ઊંચાઈ પર જે કઈ પણ છે તે બાંહેધરી આપું છું, પછી આ નોડ મોટો છે અને ડાબી બાજુની બધી વસ્તુ આ નોડ પછી નાનું રહે છે, કારણ કે તે સૌથી મોટું મૂલ્ય હતું. હવે અલબત્ત, મારી પાસે 28 પર બે કોપી છે, તેથી મને તે 28 ને દૂર કરવામાં આવી હતી, તો પછી હું ડાબી ઉપ ટ્રી પર ધ્યાન

કેન્દ્રિત કરીશ અને હું આ પુરોગામી મૂલ્ય કાઢી નાખીશ. પરંતુ, પુરોગામી મૂલ્યની સારી વસ્તુ એ છે કે તે સૌથી વધુ મૂલ્યવાન છે, યોગ્ય મૂલ્યનો અર્થ એ છે કે તે એક પર્ણ છે અથવા તે માત્ર બાળક જ બાકી છે. તેથી, અમે આ બે કિસ્સાઓમાંના એક સરળ કેસમાં પાછા આવ્યા છીએ. તેથી, આપણે આ બંને કિસ્સાઓમાંના એકનો ઉપયોગ કરવાના પુરાવાને કાઢી નાખી શકીએ છીએ, કાં તો તે એક પાન બનશે, જો હું પણ આ 21 નો પ્રચાર કરવા જઈ રહ્યો છું.

(સ્લાઈડસમયનો સંદર્ભ લો: 27:46)

તેથી, આ કિસ્સામાં 21 જમણી તરફ મળશે, તેથી સામાન્ય રીતે કાઢી નાખવું એ પૂર્વગામીને વર્તમાન મૂલ્ય તરફ ખસેડવાનું અને પછી ડાબી ઉપ ટ્રીમાં પુરોગામી કાઢી નાખવું શામેલ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 27:58)

તેથી, તે કોડનો લાંબો ભાગ છે, કારણ કે અમે ઘણા કિસ્સાઓમાં છીએ, તેથી તમે ખાલી બધા જ વૃક્ષો ખાલી કરો છો તે માટે આપણે કશું પણ કાઢી નાખીશું નહીં. જો વર્તમાન નોડ પર મૂલ્ય ધરાવતી બીજી બાજુ વિરુદ્ધ ન હોય તો તે વી નથી, તેથી જો તે સખત ઓછી હોય, તો તે ડાબી બાજુએ કંઈક છે જે આપણે ત્યાંથી કાઢી નાખીએ છીએ, જો તે કડક રીતે વધારે હોય તો ત્યાં જમણી બાજુ કંઈક છે ત્યાંથી કાઢી નાખો, તેથી આ બંને પુનરાવર્તિત કેસો છે. તેથી, જો તે સખત રીતે ગ્રીટર કરતા ઓછું ન હોય અને તે બરાબર સમાન હોવું જોઈએ અને અમને કેટલાક વાસ્તવિક કાઢી નાખવું પડશે. તેથી, હવે આપણી પાસે આ ત્રણ કેસો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 28:28)

તેથી, પ્રથમ કેસ વાસ્તવમાં બે કિસ્સાઓમાં તૂટી જાય છે, જો તે પાંદડા હોય તો તે પાંદડા હોઈ શકે છે કારણ કે તે રુટ છે, રુટ અને તે માત્ર એક ગાંઠ છે. તેથી, આપણે તે સહેજ અલગ રીતે વર્તશું, તેથી આપણે કહીશું કે જો તેના પેરેન્ટ ન હોય તો તે મૂળ છે, પછી મૂલ્યને કાઢી નાખવું તે ખરેખર વૃક્ષને ખાલી બનાવે છે. તેથી, આપણે ફક્ત ખાલી વૃક્ષની નળી અને વળતર માટે ટી રીસેટ કરીશું. જો બીજી તરફ, તે એક પાંદડાની નોડ છે જે તેના બાળકો નથી, તો પછી આપણે શું કરીએ છીએ તે છે કે અમે ફક્ત પેરેન્ટ મૂલ્યને નિલંબિત કરીને કાઢી નાખવાનો પ્રયાસ કરીએ છીએ. તેથી, હું અહીં આવી રહ્યો છું અને આ મારા પર્ણ નોડ અને હું તેને કાઢી નાખવા માંગુ છું, તો પછી હું જોઉં છું અને પછી હું મૂળભૂત રીતે આનો યોગ્ય નિર્દેશક નિલમ્યો, જે આ લિંકની આવશ્યકતા છે અને કહે છે કે કોઈ અધિકાર નથી પોઈન્ટર. તેથી, જો વર્તમાન નોડ તેના ડાબા બાળકનો છે તો પેરેન્ટ સાથેના ડાબા બાળકને કહ્યું છે કે તે પેરેન્ટ છે. નહિંતર, પેરેન્ટ સાથે યોગ્ય બાળકને નલ કહે છે, તેથી આ કોર્સ, કેટલાક કચરો બનાવો કારણ કે આ નોડ હવે વૃક્ષથી સુલભ છે તે એક સુલભ છે. પરંતુ, અમે માનીએ છીએ કે આ પુનઃપ્રાપ્ત થવું જોઈએ અને અમે તેના વિશે ચિંતા કરતા નથી, જો તમે કોઈ ભાષા પસંદોને જોઈ રહ્યા હોય, તો તમારે આ સ્ટોર પર પાછા ફ્રી સ્પેસ પર ખૂબ કાળજી રાખવી પડશે. પરંતુ, એવી બીજી ભાષામાં જેમાં કચરો સંગ્રહ છે, તે આપમેળે પુનર્સ્થાપિત થશે અને કચરો સંગ્રહ એક કરતા વધુ સમય લેશે, પરંતુ મુદ્દો એ છે કે આપણે તેને ફરીથી સેટ કરીને તેને દૂર કરી રહ્યા છીએ, પેરેન્ટ ત્યાં હોવાનું ધ્યાન દોરે છે, તેથી આ પાંદડાનું કેસ છે .

(સ્લાઈડટાઈમનો સંદર્ભ લો: 29:54)

તેથી, લીફ કેસમાં વિચારણા કરીએ, હવે ચાલો આપણે કેસને ધ્યાનમાં લઈએ જો હું એક માત્ર બાળક સાથે નોડને કાઢી નાખીશ. તેથી, પહેલો કેસ આપણે જોઈશું, જ્યારે આપણે નોડને ડીલીટ કરવાનો પ્રયાસ કરી રહ્યા છીએ કે જેમાં ફક્ત ડાબી ઉપ ટ્રી છે. તેથી, ડાબે નહી, બરાબર નિલ છે, તેથી આપણે સૌ પ્રથમ શું કરીએ છીએ તે આપણે ડાબી અનુગામીની નકલ કરીએ છીએ. તેથી, આમાં તેના પેરેન્ટ ક્યાંક છે, તેથી જો આ ડાબું બાળક સીધી પેરેન્ટ તરફ દોરે. તેથી, ટી ડોટ ડાબું ડોટ પેરેંટ ટી ટૉટ પેરેંટન્ટ છે, તો પછી આપણે તપાસ કરીએ છીએ અને અમે નક્કી કર્યું છે કે આ ડાબું પેરેન્ટ અથવા ડાબું બાળક અથવા પેરેન્ટ જે યોગ્ય છે. તેથી, આપણી પાસે નોડની ધારણા છે કે તે ડાબી બાજુએ બેઠેલી છે તે પેરેન્ટ છે, જો ટી ડોટ પેરેંટ બાકી હોય તો તે છે, તો પછી આપણે શું કરીએ છીએ તે છે કે આપણે આ મુદ્દો સીધો બનાવ્યો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 30:54)

હવે, જો તે આના જેવું ન હતું, પરંતુ તે બીજી રીત હતી, તો આ કઈ પેરેન્ટનું યોગ્ય બાળક બનશે, તો જમણી બાળકને સીધી વાત કરવી જોઈએ નહીં, પેરેન્ટનો સાચો બાળક હવે મારા ડાબા બાળક છે. તેથી, આ વસ્તુને દૂર કરવા માટેનો માર્ગ અને બાળકને પ્રોત્સાહિત કરવાનો તે રસ્તો છે, તે માત્ર બાળક જ છે, જો તે માત્ર યોગ્ય બાળક છે તો આપણે સમપ્રમાણતા કરી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 31:14)

તેથી, આપણે પહેલા યોગ્ય બાળક લઈએ અને પછી ફરી શરૂ કરીએ કે તે પેરેન્ટ છે જે તમારા પોતાના પેરેન્ટ છે અને ત્યાં પછી કેસના આધારે, અમે આ બિંદુએ આ બિંદુએ આ બનાવ્યું છે આની જેમ તેથી, ડાબી બાજુના ટી ડોટ પેરેંટ ટી ટૉટ જમણી છે, જમણી બાજુના ટી ડોટ પેરેંટ ટી ટૉટ બાકી છે. તેથી, જો મારી પાસે ફક્ત એક જ બાળક હોય તો વૃક્ષમાંથી અસરકારક રીતે જો આપણે નોડને દૂર કરીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 31:47)

અને છેલ્લે, જો આપણી પાસે બે બાળકો સાથે નોડ હોય, તો આપણે શું કરીએ છીએ જો આપણે પહેલા પુર્વગામીને વી પર ગણતરી કરીએ અને પછી કહ્યું કે વર્તમાન નોડ્સ એ પુરોગામી મૂલ્ય છે અને હવે આપણે જાણીએ છીએ કે આ મૂલ્ય ડુપ્લિકેટ છે. તેથી, આપણે ડાબી બાજુએ જઈએ છીએ અને કાઢી નાખવામાં આવે છે યાદ છે કે જયારે આપણે અહીં કાઢી નાખ્યા, ડાબી બાજુએ આપણે જે જાણીએ છીએ તે કાઢી નાખવું એ મહત્તમ વેલ્યુ છે. તેથી, તેથી તે ક્યાં તો કોઈ બાળકો હશે નહીં તે એક પાન હશે અથવા તે એકલ બાળક હશે. તેથી, તે આ કેસમાં પાછા આવશે નહીં તે પહેલાના બે કેસોમાં જ અટકી જશે અને તે સફળતાપૂર્વક કાઢી નાખવામાં આવશે. તેથી, ત્યાં ફરીથી કોઈ સમસ્યા નથી જે ફક્ત ફરીથી કાઢી નાખવાનું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 32:22)

તેથી, આ બધા ઓપરેશન્સ કે જેને આપણે વર્ણવવું પડશે હવે એક જ પાથ પર ચાલો. તેથી, આમાંની કોઈપણ કામગીરીમાં સૌથી ખરાબ કેસ સંકુલ એ વર્તમાન આપેલા વૃક્ષની ઊંચાઈ છે. અને જો આપણે કોઈ પ્રકારની સંતુલિત વૃક્ષને જાળવી રાખીએ છીએ જેમ કે આપણે જોયેલી ગરમી, તો આ બાજુઓમાં ઊંચાઈ લઘુગણક છે. તેથી, આપણી પાસે n નો ગાંઠો છે ઓર્ડર લોગ n છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 32:48)

તેથી, તમે આગામી લેક્ચરમાં સંતુલન કેવી રીતે જાળવવું તે જોશો, પરંતુ ધારી રહ્યા છીએ કે આપણે જે સંતુલન જાળવી રાખ્યું છે તેમાં આપણે જાળવી રાખીએ છીએ, આપણે જે પ્રાપ્ત કર્યું છે તેમાં સફળ થયા છે, તે પ્રાપ્ત કરવા માગે છે જે આપણે જોઈએ છે આ બધા 7 ઑપરેશન એકસાથે કાર્યક્ષમ બનશે અને હવે લોગ એન ટાઈમ છે. કારણ કે, તેમાંના પ્રત્યેક વૃક્ષને વૃક્ષના રસ્તા ઉપર અથવા નીચે એક ટ્રાવર્સલમાં પ્રાપ્ત કરી શકાય છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 02
લેક્ચર - 40
સંતુલિત સર્ચ ટ્રી(Search Trees)

અગાઉના લેક્ચરમાં, આપણે સર્ચ ટ્રી(Search Trees) પર કાર્યવાહી જોવી. અમે દાવો કરીએ છીએ કે આ કાર્યક્ષમ હતા કે અમે સંતુલન જાળવી શકીએ. તેથી, ચાલો જોઈએ કે આપણે સર્ચ ટ્રી(Search Trees)ને સંતુલિત કેવી રીતે રાખી શકીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:12)

તેથી, યાદ રાખો કે આપણે આ 7 ઑપરેશંસને જોઈ રહ્યા છીએ, અમે મૂલ્ય શોધવા માટે સક્ષમ છીએ, તમે મૂલ્યો શામેલ કરવા અને કાઢી નાખવા માટે સક્ષમ છો, તમે પણ ગણતરી કરવા માટે સક્ષમ છો ન્યૂનતમ અને વૃક્ષમાં મહત્તમ મૂલ્ય અને આપેલ મૂલ્યના પુરોગામી અને અનુગામી પણ શોધી કાઢો અને અમે દાવો કરેલ આ તમામ ઓર્ડર લોગ n હશે જો ટ્રેસ સંતુલિત હોય.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:34)

આમ, આ રીતે આપણે જે બધા ઓપરેશનો અમલમાં મૂક્યા છે, તે એક જ માર્ગ ઉપર ચાલશે અને તેથી જ સૌથી ખરાબ કેસ વૃક્ષની ઊંચાઈ હશે અને આપણા સંતુલિત વૃક્ષની ઊંચાઈ હશે. વૃક્ષના કદમાં હંમેશાં લઘુગણક બનશે. તેથી, આજે આ પ્રવચનનો ઉદ્દેશ સમજાવવાનો છે, સંતુલન કેવી રીતે જાળવવું તે વૃક્ષને વધે છે અને સંકોચાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:54)

તેથી, ત્યાં વૃક્ષો એક વૃક્ષમાં સંતુલનની ઘણી જુદી જુદી માન્યતાઓ છે, તેથી આવશ્યકપણે સંતુલન આપણે તેને ભૌતિક સંતુલનની જેમ વિચારીએ. તેથી, જો તમે જૂના શૈલીના વનસ્પતિ વિક્રેતા પાસે જાઓ છો, જ્યારે તેઓ પાસે આ પ્રકારનું સંતુલન હશે અને પછી તમે કોઈ પણ બિંદુએ ઈચ્છો છો કે જો તમે તેના દ્વારા એક વૃક્ષ ધરાવો છો તો તે મૂળ છે, બંને બાજુ સંતુલિત હોવી જોઈએ, તે સમાન હોવું જોઈએ . તેથી, સંતુલનની સૌથી સીધી કલ્પના એ છે કે બંને બરાબર છે કે ડાબી બાજુ નોડોની સંખ્યા દરેક નોડ માટે જમણી બાજુ નોડોની સંખ્યા જેટલી છે. પછી, તે જોવાનું સરળ છે કે જ્યારે તમે સંપૂર્ણ દ્વિસંગી વૃક્ષો મેળવો છો, તો ઉદાહરણ તરીકે તમારી પાસે એક વૃક્ષ હોઈ શકે છે જેનું આ માળખું છે અથવા જો તમે તેને એક વધુ સ્તરનો વિસ્તાર કરો છો, તો ડાબી બાજુના દરેક નોડ માટે તમારે જમણી બાજુ , પરંતુ પછી તે સંતુલિત પણ હોવું જોઈએ, તેથી તમારે આને પૂર્ણપણે પૂર્ણ કરવું આવશ્યક છે. તેથી, તમારી પાસે 3 સ્તરો સુધીનું વૃક્ષ હોઈ શકે છે જે સંપૂર્ણ રૂપે ભરાઈ ગયું છે અથવા 4 સ્તર સુધી અને તેથી આગળ છે. તેથી, જો તમે આ ચોક્કસ માપ સંતુલન માંગો છો, તો તે ખૂબ પ્રતિબંધિત છે. તેથી, તમારી પાસે થોડું વધારે સુગમતા હોઈ શકે છે, તમે કહી શકો છો કે અમે બરાબર સમાન નથી બનવા માંગતા હોઈએ, અમે ઈચ્છતા હોઈએ કે તે સૌથી વધુ બંધ હશે, પછી તમે આ જેવા

ઈન્સ્ટન્ટ્સ માટે માળખાં ધરાવી શકો છો, જ્યાં તમારી પાસે ફક્ત ડાબી બાજુ અથવા તમારી પાસે આ બિંદુએ ડાબે અને જમણે તમારી પાસે ફક્ત એરે છે. આ માનક માળખાઓ, જે હવે પૂર્ણ બાયનરી વૃક્ષો નથી, આ વિચારમાં સંતુલિત થઈ જાય છે. તેથી, આ આપણને વધુ લવચીકતા આપે છે અને આપણે આ રીતે વધુ વૃક્ષો મેળવી શકીએ છીએ, પરંતુ ઈન્સર્ટ્સ અને ડિલિટ્સ કરીએ છીએ તેમ આ મિલકતને વધતી જતી રાખવા મુશ્કેલ છે. તેથી, આપણે સંતુલનની જુદી જુદી કલ્પના માટે જઈશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:25)

આપણે જે સંતુલનનો ઉપયોગ કરીશું તે કદના સંદર્ભમાં નથી, પરંતુ ઊંચાઈના સંદર્ભમાં છે. તેથી, આપણે કહીશું કે ઊંચાઈ અસંખ્ય ગાંઠો મૂળથી એક પાંદડા સુધી છે. ઉદાહરણ તરીકે, જો મારી પાસે એક વૃક્ષ જેવો દેખાય છે, તો અહીં ઊંચાઈ 1, 2, 3, 4 છે કારણ કે આ માર્ગ પર આપણી પાસે 4 નોડ છે. તેથી, ઊંચાઈ 4 બની જાય છે, તે ગાંઠોની દ્રષ્ટિએ માપવામાં આવેલ સૌથી લાંબી પાથની લંબાઈ છે, કારણ કે આપણે ગાંઠોની દ્રષ્ટિએ માપીએ છીએ તે કારણ છે, પછી આપણે સહેલાઈથી ભેદ કરી શકીએ છીએ, ફક્ત ઝાડમાંથી ખાલી વૃક્ષ જ મૂળ છે. જો તમે કિનારીઓના માપદંડમાં માપ્યા હો, તો ફક્ત રુટવાળા વૃક્ષની ઊંચાઈ 0 હશે, કારણ કે ત્યાં કોઈ કિનારીઓ નથી અને તેથી ખાલી વૃક્ષ હશે. જો આપણે તેને નોડ્સના સંદર્ભમાં માપીએ તો ખાલી વૃક્ષની ઊંચાઈ 0 હોય છે અને વૃક્ષ સાથે માત્ર વૃક્ષની ઊંચાઈ 1 હોય છે. તેથી, આપણે આ બંનેને અલગ કરી શકીએ છીએ. અને હવે કદની સ્થિતિની અગાઉની રાહતને ધ્યાનમાં રાખીને, ઊંચાઈ સંતુલન વૃક્ષ એક હશે જ્યાં ડાબાની ઊંચાઈ અને જમણેની ઊંચાઈ મોટાભાગના કરતાં અલગ હશે. હવે, તે પાછલી વસ્તુમાં વધુ હળવા છે. ઉદાહરણ તરીકે, હવે અલબત્ત, હું આ જેવા ઊંચાઈ સંતુલન વૃક્ષથી પ્રારંભ કરી શકું છું. અને પછી, હવે હું આને એક ઊંચાઈ સંતુલન વૃક્ષ બનાવવા માટે જોડી શકું છું અને હવે આ ઊંચાઈ છે 3 વૃક્ષ હું ઊંચાઈ 2 વૃક્ષ સાથે જોડાઈ શકું છું અને આ જેવા દેખાતા ઊંચાઈ સંતુલન વૃક્ષ બનાવી શકું છું. તેથી, ડાબી ઉપ ટ્રીની ઊંચાઈ 3 છે, જમણી ઉપ ટ્રીની ઊંચાઈ 2 છે, આમાં વારંવાર ડાબી ઉપ ટ્રીની ઊંચાઈ 2 છે, જમણીની ઊંચાઈ એ ઉપ ટ્રી 1 છે અને તેથી આગળ. તેથી, અમારી પાસે એવી વસ્તુઓ હોઈ શકે છે જે તફાવત છોડી દે છે, તેથી ત્વરિત કદ માટે અહીં કદ 4 છે અને કદ 2 છે. પણ, તમે ગણતરી કરી શકો છો કે આ કિસ્સામાં કદ પણ ઊંચાઈમાં ઘાતાંક હશે અથવા ઉંચાઈ કરશે કદમાં લોગરિધમિક હોવું. તેથી, આ વૃક્ષોને એ.વી.એલ. વૃક્ષો કહેવામાં આવે છે, જેનું નામ બે લોકોના નામ પરથી બનાવવામાં આવ્યું છે, જેમણે સ્વતંત્ર રીતે તેમને એક વ્યક્તિ એડ્રેલ્સ-વેલ્સકી (Adelson-Velsky) અને સ્વતંત્ર રીતે લેન્ડિસ તરીકે ઓળખાવ્યા હતા. તેથી, એ.વી.એલ. વૃક્ષ એક ઊંચાઈ સંતુલિત વૃક્ષ છે જે કહે છે કે દરેક નોડ પર ડાબેની ઊંચાઈ અને જમણેરી ઉપ ઝાડની ઊંચાઈ મોટાભાગે 1 થી અલગ હોય છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:53)

તેથી, ચાલો આપણે જોઈએ ઊંચાઈ વચ્ચેની માત્રા જ ઢાળની વચ્ચેનો તફાવત, તેથી આપણી ચિત્રોમાં અંતર્જ્ઞાન છે. તેથી, જો તે અસંતુલન હોય તો વસ્તુની સારવાર કરવામાં આવે છે, તેથી આ રીતે અથવા આ રીતે સુધી આપણે આવી શકીએ, તેથી આપણે આ ઢાળને બોલાવી શકીએ. તેથી, આપણે કહીએ કે આ ઢાળ ડાબી માપની ઊંચાઈની ઊંચાઈ છે, તેથી ડાબીની ઊંચાઈ જમણીની ઊંચાઈ કરતાં ઓછી છે, પછી તમારી પાસે હકારાત્મક ઢાળ છે. જો ડાબીની ઊંચાઈ જમણીની ઊંચાઈ કરતાં મોટી હોય, તો તમારી પાસે જમણી બાજુ છે, તમારી પાસે નકારાત્મક ઢાળ હોય તેના કરતાં ડાબે નાના છે,

ડાબે જમણેથી વધારે તમે હકારાત્મક ઢાળ છો. તેથી, સંતુલિત વૃક્ષમાં ઊંચાઈના તફાવતથી સંપૂર્ણ મૂલ્ય 1 હોવું આવશ્યક છે, તમારી પાસે માત્ર વૃક્ષની ત્રણ શક્ય ઢોળાવ હોઈ શકે છે, કાં તો ત્યાં કોઈ ઢાળ સમાન નથી અથવા તે 1 અથવા વત્તા 1 ની વસ્તી છે. હવે, જો તમે ખૂબ જ સરળતાથી દલીલ કરી શકે છે કે જો વર્તમાન મૂલ્ય ઢાળના કેટલાક ઓછા 1 વત્તા 1 છે, જ્યારે તમે નોડને કાઢી નાખો છો, ત્યારે તમે 1 થી 1 ઊંચાઈને ઘટાડી શકો છો. તેથી, ઉંચાઇ તફાવત 1 થી 2 અથવા જ્યારે તમે વધશો તમે ઊંચાઈનો તફાવત કરી શકો છો, ફરી 1 થી 2 સુધી જઈ શકો છો. તેથી, એક જ શામેલ અથવા એકલ વિતરણ પછી નવી ઢાળ મોટા ભાગે 1 અથવા વત્તા 2, બાદબાકી 2 અથવા વત્તા 2 હોઈ શકે છે. તેથી, આપણે શું સમાપ્ત કરીશું તે કરવા માટે આપણે જે કરવાનો પ્રયાસ કરીશું તે છે કે જ્યારે પણ આપણે કોઈ દાખલ અથવા કાઢી નાખીએ ત્યારે આપણે તાત્કાલિક વૃક્ષને ફરીથી કરવાનો પ્રયાસ કરીશું. તેથી, માઈનસ 2 અથવા પ્લસ 2 માંથી એક જ મુશ્કેલી ઊભી થશે તે ક્યારેય ખૂબ ખરાબ અસંતુલન બનશે નહીં અને અમે તરત જ બેલેન્સને 1 થી વત્તા 1 સુધીમાં પુનઃસ્થાપિત કરીશું. તેથી, તમે આ રીબેલેન્સિંગ કરશો, અમે આ રીબેલેન્સિંગ તળિયે પણ કરીશું. અપ, તો શું થાય છે અમે એક શામેલ કરીશું, જો તમને યાદ છે કે અમે નીચે જઈએ છીએ અને અમને શામેલ કરવા માટે સ્થાન મળે છે. તેથી, આ બિંદુએ આપણે નવો નોડ ઉમેરીએ, તેથી હવે આ બિંદુએ કેટલાક અસંતુલન હોઈ શકે છે, તેથી આપણે તેને ઠીક કરીએ, પછી આપણે આ પાથ ઉપર જઈશું અને આપણે ત્યાં જઈશું અને અહીં પાથને ઠીક કરીશું, પરંતુ આ સમયે તમે ધારેશો કે નીચેનું વૃક્ષ સંતુલિત છે. તેથી, જ્યારે પણ આપણે રેંજની બહારના ઢાળને ફરીથી બંધ કરીએ છીએ, ત્યારે તમે એમ માનશો કે નીચેનાં પેટા વૃક્ષો પહેલાથી સંતુલિત છે, કારણ કે આ સંતુલન અમે જોઈશું તે તળિયે કરવામાં આવશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:59)

તેથી, અહીં એક સામાન્ય પરિસ્થિતિ છે કે અમે એક જ ઓપરેશન પછી પહોંચીશું જે સંતુલનને દૂર કરે છે. તેથી, આપણી પાસે નોડ હોઈ શકે છે જે સ્લોપ પ્લસ 2 અથવા બાદબાકી 2 ધરાવે છે, તો ચાલો આપણે પ્લસ 2 બાદ 2 ને સમપ્રમાણતા તરફ વળીએ. તેથી, આપણી પાસે નોડ છે જેને આપણે એક્સ કહીએ છીએ જેમાં ઢાળ વત્તા 2 છે અને તેનો અર્થ શું છે, તે ડાબે વૃક્ષ અને જમણો વૃક્ષ ધરાવે છે. આ રીતે, ડાબા વૃક્ષની ઊંચાઈ જમણી વૃક્ષની ઊંચાઈ કરતાં 2 વધુ છે, યાદ રાખો કે આ ઢાળ જમણી કે ડાબી બાજુની જમણા અથવા ડાબે છે, તેથી એચ પ્લસ 2 ઓછા એચ હશે 2. હવે, વારંવાર આપણે ધારે છે કે અહીં અને અહીંની બધી ઢોળાવ વધુ વત્તા 1 અથવા બાદબાકી 1 છે. તેથી, આપણે ધારી રહ્યા છીએ કે આની નીચેની દરેક વસ્તુ નિશ્ચિત કરવામાં આવી છે અને x પરના આ પેટા વૃક્ષમાં ફક્ત એક જ સંતુલન છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:50)

તેથી, હવે ડાબી બાજુ ઊંચી એચ પ્લસ 2 હોવાથી, તેની ઊંચાઈ ઓછામાં ઓછી 2 હોઈ શકે છે, એચ ઓછામાં ઓછું નાના હોઈ શકે છે, તેથી તેની ઊંચાઈ ઓછામાં ઓછી 2 હોઈ શકે છે, તેથી અહીં ઓછામાં ઓછા 1 નોડ છે. મારો મતલબ એ છે કે અહીં ઓછામાં ઓછા 2 નોડો છે, તેથી અમારી પાસે ઓછામાં ઓછા 1 નોડ છે. તેથી, આપણે તેને રુટ કરીને વિસ્તૃત કરીશું અને રુટ સામાન્ય રીતે 2 પેટા વૃક્ષોમાં હશે, તેથી હવે આ સમગ્ર વસ્તુ ઊંચાઈ h પ્લસ 2 છે. તેથી, હવે આપણે આ નવો નોડ જોઈશું જે આપણે અપેક્ષિત છે. તેથી, આ ઢાળ ઓછો 1 0 અથવા વત્તા 1 છે અને અમે કેટલાક તળિયે રીબેલેન્સિંગ કરવા જઈ રહ્યા છીએ, અમે તે બધું નીચે ધારી રહ્યા છીએ તે કેસ છે. તેથી, હું વાય ની ઢાળ શું છે તેના આધારે કેટલાક કેસ વિશ્લેષણ કરવા જઈ રહ્યો છું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:41)

તેથી, ચાલો સૌ પ્રથમ પરિસ્થિતિને જોઈએ જ્યાં y ની ઢાળ 0 અથવા વત્તા 1 છે, તેથી જો તે 0 અથવા વત્તા 1 નો અર્થ છે તો આનો અર્થ એ છે કે આખી વસ્તુ h plus 2 હતી જેમાંથી 1 નોડ અહીં છે, તેથી તે બાકી છે બાળક ઓછામાં ઓછું એચ પ્લસ 1 હોવું જોઈએ અને કારણ કે તે ઢાળ 0 અથવા વત્તા 1 છે, યોગ્ય બાળક ક્યાં તો H વત્તા 1 છે, ઢાળ 0 હોય અથવા તે ઢાળને ઢાંકી દે છે પ્લસ 1 છે. તેથી, હવે આ વર્તમાન સ્થિતિ છે કારણ કે અમારી પાસે અસંતુલિત નોડ x સાથે તે બધું સંતુલિત છે. પરંતુ આપણે ફક્ત એવી પરિસ્થિતિમાં આવી ગયા છીએ કે જ્યાં આપણે પાછળની પરિસ્થિતિનું વિશ્લેષણ કરવાનો પ્રયાસ કરીએ છીએ. તેથી, x ને પ્લસ 2 નું સંતુલિત અસંતુલન મળ્યું છે અને તેના નીચે આપણી પાસે શામેલ છે કે કેમ તે 0 અથવા વત્તા 1 છે. તેથી, હવે આપણે આ પરિભ્રમણ કરીએ છીએ, તેથી આપણે આ વૃક્ષ લઈએ છીએ અને આપણે વાય દ્વારા તેને અટકીએ છીએ અને અમે વસ્તુઓ ફરીથી જોડે છે. તેથી, આ પરિભ્રમણમાં જ્યારે તમે વાય દ્વારા તેને અટકી જાઓ છો, ત્યારે x નીચે આવે છે અને હવે આપણે આ પેટા વૃક્ષો પર ધ્યાન આપીએ છીએ, તેથી આપણી પાસે 3 પેટા વૃક્ષો છે, અમારી પાસે TLL, TLR અને TR છે. તેથી, TR એ x ની જમણી બાજુએ છે અને તે y ની જમણી બાજુ પણ છે, તેથી તે બંનેનો અધિકાર છે, ટી ની ડાબી બાજુએ x ની જમણી બાજુ છે, ટીએલએલ y ની ડાબી બાજુએ છે, x ની ડાબી બાજુએ છે .

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:57)

તેથી, જો તમે ત્યાં જાઓ છો તો અમને લાગે છે કે ટીઆર બંનેની જમણી બાજુ છે, ટીએલઆર વાયની જમણી બાજુએ ડાબી બાજુએ છે અને ટીએલએલ બંનેની ડાબી તરફ છે. તેથી, અમે આ વૃક્ષો અટકી ગયા છીએ, તેથી હવે અમે ચકાસી શકીએ તે બધા મૂલ્યોને હાલમાં એક સર્ચ ટ્રી(Search Trees) સમસ્યા તરીકે સંગઠિત કરવામાં આવશે. પરંતુ, હવે જો તમે ઢોળાવ પર નજર નાખો, તો આપણે આ ઢોળાવને વારસામાં મેળવી છે, આપણે જાણીએ છીએ કે ટીએલએલની એચ પ્લસ 1, ટીએલઆર એ એચ પ્લસ 1 અથવા એચ અને ટીએચ છે. તેથી, આનો અર્થ એ છે કે જો હું આ બિંદુએ આ બધી ઊંચાઈ પર જોઉં છું, તો તે ક્યાં તો એચ પ્લસ 1 અથવા મહત્તમ એચ પ્લસ 2 છે, તેથી આ એચ પ્લસ 2 અથવા એચ પ્લસ 1 છે. જો તે એચ પ્લસ 2 છે, તો પછી વાય પર ઉંચાઈ ઢાળ 1 ની બાદબાકી છે, જો તે એચ પ્લસ 1 હોય તો બંને બાજુઓ પર એચ વત્તા 1 સ્લો પર વાય 0 છે અને જો તમે x ને જુઓ છો, તો આપણી પાસે h plus 1 છે અને અહીં આપણી પાસે h છે, તેથી તફાવત 0 છે. અથવા વત્તા 1. તેથી, એક્સ હવે સંતુલિત છે, વાય સંતુલિત છે અને ધારણા દ્વારા અનુકૂલનપૂર્વક બધા ગ્રે પેટા વૃક્ષો સંતુલિત છે. તેથી, એક જ પરિભ્રમણથી, આપણે વૃક્ષને ફરીથી બંધ કરી દીધા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:55)

તેથી, ત્રીજી પરિસ્થિતિ એ છે કે તે 0 અથવા વત્તા 1 નથી, પરંતુ સ્લોપ માઈનસ 1 છે. તેથી, આપણે કેસ સાથે પહેલાથી જ વ્યવહાર કર્યો છે જ્યાં 0 અથવા પ્લસ 1 શક્યતા છે. તેથી, જો તે 1 ઓછા છે, તો એનો અર્થ એ છે કે જમણે સબ વૃક્ષ આગામી ડાબા ઉપ ટ્રી કરતાં સખત લાંબી હોવી આવશ્યક છે. તેથી, આખી વસ્તુ ફરીથી યાદ રાખવી એ છે કે એચ પ્લસ 2 ઊંચી ધારણા છે, કારણ કે આખી વસ્તુ વત્તા 2 છે, તેથી આખી વસ્તુ h પ્લસ 2 છે અને આ વસ્તુ h છે. હવે, આ એચ પ્લસ 2 હવે તે વાયનું વિભાજન કરે છે અને બાકીનું, એચ પ્લસ 1 જમણી તરફ આવવું આવશ્યક છે અને 1 ડાબી બાજુએ

આપણે ધારણા દ્વારા ઢાળ મેળવીએ છીએ, આપણે માનીએ છીએ કે ઢોળાવ ઓછો છે. તેથી, હવે આપણે જઈ રહ્યા છીએ આ નોડને વિસ્તૃત કરવા માટે હવે તે એચ પ્લસ 1 છે, તેથી અહીં ઓછામાં ઓછો 1 નોડ પણ એક રસ્તો છે જે 0 ની ઓછામાં ઓછી રુટ નોડ છે. તેથી, આપણે રુટ નોડને એક્સપોઝ કરીશું કારણ કે આપણે પહેલા y ખુલ્લું પાડ્યું હતું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:48)

તેથી, હવે આપણે આ TLR ને 2 પેટા વૃક્ષો TLRL સાથે ઝેડ તરીકે વિતાવીએ છીએ. તેથી, આ સૂચવવાનું માનવું જોઈએ કે આપણે તેમના મૂળ વૃક્ષને જઈએ અને ડાબે જમણે જાઓ અને ડાબી બાજુ જાઓ, તેથી ટીએલઆરએલ ડાબી બાજુ જાઓ, જમણે જાઓ, જમણી બાજુ જાઓ. તેથી, તે TLRR છે, તેથી તે પેટા વૃક્ષો માટે સંકેત છે, તેથી આખી વસ્તુ એચ પ્લસ 1 હતી. તેથી, આ એચ પ્લસ 1 ક્યાં તો બાજુ આવી શકે છે, તેથી આ એચ છે અથવા એચ ઓછા 1 અથવા આ એચ છે. અથવા એચ અવતરણ 1 વાસ્તવમાં તેમાંથી એક હોવું જ જોઈએ. કારણ કે, આપણે બંને ઓછા 1 ઓછા છે, તો આખી વસ્તુ એચ પ્લસ 1 હોઈ શકે નહીં. તેથી, ઓછામાં ઓછું 1 એચ છે અને તે સંતુલિત છે, તેથી મને ખબર નથી કે તે કઈ રીતે છે, પરંતુ ઓછામાં ઓછા આમાંની એક ટ્રેસ છે. હું બીજાને h અથવા h માઈનસ 1 મેળવી શકું અને તે શબ્દ હશે તે કોઈ વાંધો નહીં. તો, હવે મારી પાસે બે પગલાની પ્રક્રિયા છે જે હું કરીશ, હવે હું સમાન રોટેશન કરીશ, પણ ડાબી બાજુ આ બે નોડોનો સમાવેશ કરશે. તેથી, હું z દ્વારા અહીં આ પેટા વૃક્ષોને અટકીશ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:50)

તેથી, જો હું આ પેટા વૃક્ષો લગાવીશ તો શું થાય છે કે નીચે આ 3 વૃક્ષો મને ફરીથી જોડાયા છે અને આપણે બંનેને યોગ્ય રીતે ફરીથી જોડવું પડશે, ટી.એલ.એલ. બંનેની ડાબી બાજુએ બીટ છે. અને ઝેડ ટીએલઆરઆર, વાય અને ઝેડ અને ટીએલઆરએલ બંનેની જમણી બાજુએ છે અને 1 ની જમણી બાજુએ છે. અને હવે, જો તમે પાછા જાઓ અને બધી ઊંચાઈઓ તપાસો, તો તમને લાગે છે કે y પર મને ક્યાં ઊંચાઈ છે 0 અથવા મારી પાસે ઊંચાઈ વત્તા 1 છે. હવે, જો હું z ને જોઉં તો ડાબું બાજુ બાજુ હવે એચ પ્લસ 1 છે અને જમણા હાથ બાજુ h અથવા h ઓછા 1 છે. તો, જો એચ h plus 1 છે અને આ h છે મને પ્લસ 1 મળે છે, જો તે એચ પ્લસ 1 હોય તો આ એચ માઈનસ 1 મને પ્લસ 2 મળશે. તેથી, આ પ્રક્રિયામાં હવે મારી પાસે એક્સ નથી જે અસંતુલિત હતું અને અસ્થાયી રૂપે બનાવેલો તે ઝેડ છે જે સંભવતઃ અસંતુલન છે. તેથી, હું ખાતરીપૂર્વક કહી શકતો નથી કે તે અસંતુલન છે, પરંતુ તે અસંતુલન હોઈ શકે છે, કારણ કે મને ખબર નથી કે h શું છે અને જે h ઓછા 1 છે, પરંતુ આ માત્ર મધ્યવર્તી પગલાં છે, તેથી હવે મેં જે કર્યું છે, તેથી અગાઉ આપણે જે કર્યું તે, આપણે અહીં ડાબી રોટેશન પર કરીશું. તેથી, હું x પર જમણી ફેરવણી દ્વારા આને અનુસરવા જઈ રહ્યો છું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:08)

તેથી, આપણે લખાણને ઝેડ અપ કરીએ, x વધે છે x જાય છે અને હવે ફરીથી આપણે 4 વૃક્ષો અટકીએ છીએ અને ત્યાં અટકી જવાનો એકમાત્ર રસ્તો છે, તે ચકાસી શકે છે કે તે પાછું જશે અને તપાસ કરશે તેમની અટકી. તેથી, ટીએઆરઆર ઝેડ અને એક્સ કરતાં મોટું હોવું જ જોઈએ, ટીએલએલ વાય અને ઝેડ કરતા નાની હોવી આવશ્યક છે. અને હવે અમારી પાસે ખૂબ જ સુખી પરિસ્થિતિ છે કે જો તમે y ની ઢાળ જુઓ છો, તો તે 0 અથવા તે વત્તા 1 હોઈ શકે છે કારણ કે આ h

અવતરણ 1 હોઈ શકે છે. જો તમે x ને જુઓ તો તે 1 ઓછા થઈ શકે છે , કારણ કે આ h ઓછા 1 હોઈ શકે છે આ ચોક્કસપણે એચ છે અથવા આપણે 0 હોઈ શકીએ છીએ. પરંતુ, આ બધી વસ્તુ ચોક્કસપણે એચ પ્લસ 1 છે કારણ કે મારી પાસે કદ એચ ના ઓછામાં ઓછા 1 પેટા વૃક્ષ છે, આ સંપૂર્ણ વસ્તુ ફરીથી વત્તા 1 છે, કારણ કે મારી પાસે કદ એચ ના ઓછામાં ઓછા 1 પેટા વૃક્ષ છે અને તેથી, 0 પર z પર ઢાળ 0 છે. તેથી, ત્રણેય ગાંઠો x, y અને z બંને યોગ્ય રેન્જમાં ઢોળાવ ધરાવે છે અને તેથી હું ફરીથી સંતુલન સંગ્રહિત કરીશ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:59)

તેથી, આ સારાંશ માટે આપણે જે કહ્યું છે તે સારૂ છે કે જો આપણી પાસે વત્તા 2 હોય, તો આપણે x ની ડાબી બાજુએ બાળકને y કહીશું, જો y ની ઢાળ 0 અથવા 1 હોય તો આપણે ફેરવીશું x પર જમણી બાજુએ, જો y પર ઢાળ અવતરણ 1 હોય તો પહેલા તેને ફેરવો અને પછી આપણે વાય ફેરવો. ક્યાંક કેસ આપણે x પર ફેરવો, પરંતુ પ્રથમ આપણે વાયુઓને 1 થી ઓછા ગુણોમાં ફેરવી દીધી. હવે, તે ચાલુ થશે કે જો તમારી પાસે અન્ય સેટ કેસ છે, તો અન્ય આત્યંતિક જ્યાં x પર સ્લો ઢોળાવો 2 છે, તો પછી સમપ્રમાણ ચિત્ર, તેથી આપણને જમણી બાજુ ખુલ્લી કરવી પડશે, તેથી હું તેને y કહીશ. તેથી, મૂળભૂત બેઝિક્સ ઓપરેશન એ છે કે જો y ની અવતરણ 1 અથવા 0 ની નીચે હોય તો તે સમપ્રમાણતા યાદ કરે છે, તો પછી આપણે એક ડાબી બાહ્ય પરિભ્રમણ x કરીશું. નહિંતર, આપણે y પર યોગ્ય પરિભ્રમણ પ્રથમ કરીશું, જો વાય પર ઢાળ વત્તા 1 હોય અને પછી આપણે ડાબા પરિભ્રમણને જ કરીશું. તેથી, અમે તેમના ઓછા 2 કેસને વિગતવાર ન જોઈએ, પરંતુ તે જોવાનું સરળ છે કે તે પ્લસ 2 કેસમાં સમપ્રમાણ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:58)

તો, તમે આ રોટેશન કેવી રીતે કરો છો, તમે આ ચિત્રો ક્યાંથી દોરી શકો છો અને બહાર નીકળે છે. તેથી, તમે ફક્ત બધું જ નામો આપો, તેથી આપણે કહીએ છીએ કે આપણે આ x ને y ઉપર ફેરવવા માંગીએ છીએ, હવે આપણે અહીં મૂળ યાદ રાખીએ છીએ. તો, આપણે નોડમાં ફેરફાર કરી શકતા નથી, જે t એ 2 તરફ છે, તેથી આપણે યાદ રાખીએ છીએ કે t ની કિંમત આપણે અહીં મૂલ્યને યાદ કરીએ છીએ. તેથી, આપણને આ નામ x અને y ની જરૂર છે જેમાં તે શામેલ છે અને પછી આ ત્રણેય વૃક્ષો TLL, TLR અને TR તરફ નિર્દેશ કરે છે અને પછી અમે તેને ફરીથી કનેક્ટ કરીએ છીએ. તેથી, આપણે જે કરીએ છીએ તે આપણે પહેલા આ મૂલ્યને y દ્વારા બદલીએ છીએ, તો આપણે શું કરીએ છીએ તે છે કે આપણે આ નોડને જમણી બાજુએ આવીએ અને આપણે તેને x થી મૂલ્ય આપીએ તે ફરીથી સેટ કરીએ. તેથી, આપણે અહીં એક એક્સ મુક્યો છે અને આપણે ત્યાં તે નોંધ્યું છે અને પછી ડાબી બાજુએ ટોપ નોડ નીચે અટકી ગયા છે. અમે ટી.એલ.એલ. મૂકીએ છીએ અને જમણા જમણી બાજુ TLR જો ડાબી બાજુએ જમણી બાજુએ જમણે નવો જમણો નોડ નીચે છે. તેથી, તમે માત્ર આ પ્રકારનો પુન: જોડાણ કરો છો જેમ કે તમે જોડાણ છોડવું અને ફરી જોડાણ કરો છો અને માત્ર અમે બધા નામો ટ્રેક રાખે છે. તેથી, તે બધું અનૂકુળ છે અને બરાબર હૂક છે. તેથી, તે ખૂબ જ સરળ છે, તેથી તમે નોંધ્યું છે કે આ પ્રકારનાં સતત સેટ ઑપરેશનમાં આમાંથી કેટલાક ડાબે અને જમણે પોઈન્ટર સામેલ છે. તેથી, તે વૃક્ષના કદને ધ્યાનમાં લેશે, તે ખૂબ જ સ્થાનિક કામગીરી છે. તેથી, આપણે આને એક સતત એકમ ઑપરેશન તરીકે કરી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:25)

અને ડાબી બાજુની સમાન વસ્તુ, અમે નામો આપીએ છીએ અને આપણે આ અપડેટ કરીએ છીએ તે બરાબર આપણે પહેલા કર્યું છે. તેથી, ફરીથી આ ઓપરેશન્સના સતત સેટ છે જે ડાબી બાજુ ફેરવશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 17:38)

તેથી, હવે તે બે રોટેશન ધરાવે છે, તો પછી રીબલેન્સિંગ સેટ શું કરે છે, રીબેલેન્સિંગ સેટ કરે છે કે જો મારી પાસે નોડ છે જે પ્લસ 2 ધરાવે છે, તો હું ખુલ્લી કરીશ કે તે બાળક બાકી છે અને હું તેને તપાસું છું ઢાળ, જો તે ઢોળાવ 1 ની વટાણા છે, તો પ્રથમ ત્યાં હું ડાબું ફેરબદલ કરીશ અને પછી ભલે હું ટોચ પર જમણી ફેરબદલી કરીશ. સમપ્રમાણતાના કિસ્સાઓમાં, જો મારી પાસે ટોચની માત્રા 2 ની માત્રા હોય, તો તે એક યોગ્ય બાળક હશે અને પછી જો તે વત્તા 1 તરીકે હશે ત્યારે હું અહીં પ્રથમ જમણી પરિભ્રમણ તરફ આગળ વધું છું અને પછી ભલે હું ડાબી બાજુના રોટેશન કરું ટોચ. તેથી, રીબેલેન્સિંગ એ આ બે પગલાં છે અને પરિભ્રમણ આ અન્ય પગલાંઓનું નિર્માણ કરે છે, તેથી મૂળભૂત રીતે રીડલેન્સિંગ આપેલા નોડ પર તે નોડ પર સતત સંખ્યાબંધ ઓપરેશન્સ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:24)

તેથી, હવે આપણે જે કરીએ છીએ તે દરેક વખતે આપણે આપણા વૃક્ષમાં પરિવર્તન લાવીએ છીએ જે નોડની ઢાળના સંતુલનને અસર કરી શકે છે, અમે સંતુલન કરીએ છીએ. તેથી, જ્યારે આપણે આપણા પહેલાના કોડમાં દાખલ કરવા માટે, જ્યારે આપણે વૃક્ષના ડાબા ભાગમાં ફરી વળેલું શામેલ કર્યું હતું, અમે પુનર્જીવિત કર્યું છે, તેવી જ રીતે આપણે વૃક્ષના જમણે શામેલ કરીએ છીએ જે અમે રીબાલન્સ કર્યું છે, અમે ફક્ત આ રીબાલન્સ કોડ રજૂ કરીએ છીએ અને નોંધ એ છે કે આ આ નોડ માટે સતત કામગીરી. તેથી, તમે પાથ સાથે આ બધી રીતે કરશો. તેથી, આ સતત સંખ્યાના ઓપરેશન્સમાં લોગ એન વખત કરશે, તેથી તે આપણા ઑપરેશનની અસિમ્પ્ટોટિક જટિલતાના સંદર્ભમાં કંઈપણને અસર કરશે નહીં, તે દરેક નોડ માટે સતત સમય સંચાલનની લઘુગણક સંખ્યા હશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:04)

ડીલીટ સાથે તે જ છે, જ્યાં ઉપર આપણે પુનરાવર્તન કરીએ છીએ તે કાઢી નાખીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:10)

અને પછી ત્યાં એક કેસ હતો જ્યાં અમે મહત્તમ મૂલ્ય અથવા પુરોગામી કાઢી નાખ્યું, તેથી ફરીથી અમે પુનર્વિચાર કરીએ છીએ. તેથી, આપણે વૃક્ષની માળખુંને અસર કરતી વખતે શામેલ અને શામેલ કરી શકીએ છીએ તે અમે માત્ર ખાતરી કરીએ છીએ કે અમે તે વૃક્ષને ફરીથી બંધ કરી દીધી છે

((સમયનોસંદર્ભ: 19:22)).

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:23)

તેથી, એક મુદ્દો છે કે આપણે આ બાબતમાં સાવચેત રહેવું જોઈએ, તેથી અમે કહ્યું કે આપણે આ બધા રીબેલેન્સિંગ કરવું પડશે. તેથી, જો તમે રીબેલેન્સિંગ પર પાછા જાઓ છો, તો રીબેલેન્સિંગને ઢાળ અને ઢાળની ગણતરી કરવાની જરૂર છે જે આપણે જમણી બાજુની ડાબા માળની ઊંચાઈની ઊંચાઈને વ્યાખ્યાયિત કરી છે. હવે, ફ્લાય પર ઊંચાઈની ગણતરી કરવી શક્ય છે, વૃક્ષની ઊંચાઈ વારંવાર ગણતરી કરવામાં આવે છે, જો તે ન હોય તો તેની ઊંચાઈ 0 હોય છે; અન્યથા, તમે વારંવાર ડાબેની જમણી બાજુની જમણી બાજુની ગણતરી કરો છો અને પછી આ નોડ માટે એકાઉન્ટિંગ કરો છો, તમે તેમાં એક ઉમેરો છો. તેથી, તમે મહત્તમ બે પેટા વૃક્ષો લો અને એક ઉમેરો, પરંતુ આ દુર્ભાગ્યે વૃક્ષમાંના દરેક નોડની તપાસ કરવી શામેલ છે. તેથી, આ વૃક્ષના કદના ક્રમમાં પ્રાયોગિક હશે. તો, આ અમુક અર્થમાં ઘોષણાત્મક કામગીરી હશે તેના પર આધાર રાખીને મારો અર્થ પાથની સરખામણીમાં થાય છે, તેથી અમે લોગિન ઓપરેશન કરવાનો પ્રયાસ કરી રહ્યા છીએ, આ ઓર્ડર એન ઓપરેશન હશે. તેથી, આ બધું કાર્યક્ષમ રીતે કરવાના આપણા બધા પ્રયાસો હત્યા કરશે, કારણ કે ઊંચાઈની ગણતરી કરવા માટે આપણે ખરેખર આખા વૃક્ષને જોવું જોઈએ જે આપણે કરવા નથી માંગતા.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 20:28)

પરંતુ, આપણે તેની ઊંચાઈના વર્તમાન મૂલ્યને જાળવી રાખીને અને પછી દરેકને અપડેટ કરીને આની આસપાસ જઈ શકીએ છીએ. તેથી, અમારી પાસે વધારાની વસ્તુઓ છે, તેથી આપણી વૃક્ષ નોડમાં છે, આપણી પાસે હાલમાં મૂલ્ય, પેરેન્ટ, ડાબે અને જમણે છે. તેથી, હવે આપણે ફક્ત એક વધુ ફીલ્ડ ઉમેરીએ છીએ, તેથી જ્યારે પણ આપણે કોઈપણ રીબેલેન્સિંગ કરીએ છીએ, ત્યારે આ વૃક્ષની ઊંચાઈ વર્તમાન નોડને બદલી શકે છે. તેથી, અમે ફક્ત ઈન્ટરેક્ટિવ રીતે જોઈએ છીએ કે આપણે ધારીએ છીએ કે નીચે ઉંચાઈ યોગ્ય રીતે સેટ થઈ ગઈ છે. તેથી, અમે બે ઊંચાઈઓ જોઈએ છીએ જે સ્થાનિક રીતે ત્યાં મહત્તમ છે ઉમેરવા માંગો છો).

((સમયનોસંદર્ભ: 21:07)

હવે, આ ફક્ત એક મૂલ્ય શોધી રહ્યું છે અને બે પાડોશીઓ નીચે બે બાળકો છે. તેથી, હવે તે સતત સમય ઓપરેશન્સ બને છે, તે માટે જરૂરી નથી કે આખા વૃક્ષો તરફ આગળ વધવું. તેથી, જેમ આપણે પુન: સંતુલન કરી રહ્યા છીએ તેમ અમે ઊંચાઈને ફરીથી ગોઠવીએ છીએ અને દર વખતે જ્યારે તમે ઢોળાવ પર તપાસ કરવા માંગો છો, ત્યારે અમારે ફક્ત બે વૃક્ષનું મૂલ્ય ચકાસવું છે અને તેની ઊંચાઈના તફાવતને તપાસો. કારણ કે, તે નોડમાં આ ઊંચાઈવાળા ક્ષેત્ર દ્વારા સ્થાનિક રીતે આપવામાં આવશે. તેથી, આપણે સાવચેતી રાખવી જોઈએ કે વારંવાર ઊંચાઈની ગણતરી ન કરવી, પરંતુ વૃક્ષના ભાગ તરીકે ઊંચાઈને સંગ્રહિત કરવી અને તે દરેક અપડેટ સાથે અપડેટ કરવું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 21:39)

તેથી, સારાંશ આપવા માટે તમે ઊંચાઈ સંતુલિત દ્વિસંગી સર્ચ ટ્રી(Search Trees) જાળવવા માટે રોટેશનનો ઉપયોગ કરી શકો છો અને પછી ઊંચાઈ સંતુલન સર્ચ ટ્રી(Search Trees) અમે દાવો કર્યો છે કે ઊંચાઈ કદમાં લઘુગાણક બનશે. અને કારણ કે અમારા બધા ઓપરેશન્સ ઉંચાઈ માટે પ્રસ્તાવના છે, કારણ કે તેઓ બધા એક સાથે અને એક પાથ જાય છે, આ બધા ઓપરેશન્સ જેમ કે શોધો, શામેલ કરો, કાઢી નાખો, ન્યૂનતમ, મહત્તમ, પુરોગામી અને ઉત્તરાધિકારી બધા લઘુગાણક સમયમાં કરી શકાય છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 03
લેક્ચર - 41
ગ્રીડી એલ્ગોરિધમ્સ: ઈન્ટરવલ શેડ્યુલિંગ (Greedy algorithms: Interval scheduling)

ચાલો ગ્રીડી એલ્ગોરિધમ્સ: પર બીજું નજર લઈએ.
(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:05)
તેથી, અમે એલ્ગોરિધમ્સ જોઈ રહ્યા છીએ જ્યાં આપણે પસંદગીની શ્રેણી બનાવીને વૈશ્વિક ઈષ્ટતમ પ્રાપ્ત કરવાની જરૂર છે. તેથી, ગ્રીડી(Greedy) વ્યૂહરચનામાં આપણે શું કરીએ છીએ તે આપણે સ્થાનિક પસંદગીના આધારે આગામી પસંદગી કરીએ છીએ. તેથી, ત્યાં ઘણી પસંદગીઓ હોઈ શકે છે, પરંતુ અમે હમણાં જ તેમાંથી કંઈક પસંદ કરીએ છીએ જે આ ક્ષણે સારી લાગે છે અને હવે અમે ક્યારેય પાછા નહીં જઈએ અને પહેલાંના નિર્ણયને સુધારીશું નહીં. તેથી, આપણે દરેક પગલું પર સારી પસંદગીઓ પસંદ કરીને નિરાકરણની આ સ્થાન દ્વારા નિશ્ચિતપણે શોધી કાઢીએ છીએ અને આ તે સ્થાનને ભારે ઘટાડે છે જેમાં આપણે શોધ કરવી પડે છે. તેથી, સૌથી યુક્તિજનક વસ્તુ એ છે કે, આ વ્યૂહરચના ઘણી વખત કામ કરતી નથી. તેથી, જો તમારી પાસે ગ્રીડી(Greedy) ની વ્યૂહરચના છે, તો તમારે પાછા જવાની અને સાબિત કરવાની જરૂર છે કે અમે જે રીતે સ્થાનિક પસંદગી કરી છે તે ખરેખર વૈશ્વિક શ્રેષ્ઠતમ પ્રાપ્ત કરે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:53)

તેથી, આપણે અત્યાર સુધી ત્રણ એલ્ગોરિધમ્સ જોયા છે જે આ 3 ડી પેરાડિગનું અનુસરણ કરે છે. પ્રથમ સ્ત્રોત ટૂંકા પાથ સમસ્યા માટે ડીજેક્સ્ટ્રા(Dijkstra's) એલ્ગોરિધમ હતું. તેથી, યાદ રાખો કે આ અલ્ગોરિધમમાં આપણે શિરોલંબને બાળી રાખીએ છીએ અને દરેક તબક્કે અમે નજીકના અદ્રશ્ય શાખા સુધી અંતરને સ્થિર કરી દઈએ છીએ અને દાવો કરીશું કે તે વાસ્તવમાં સ્રોતથી તે શિરચ્છેદની સૌથી ટૂંકી અંતર હશે. તેથી, વૈશ્વિક સ્તરે આ અલ્ગોરિધમમાં આપણે પ્રાપ્ત કરેલ શ્રેષ્ઠતમ એ છે કે આ ગ્રીડી(Greedy) વ્યૂહરચના દ્વારા સોંપાયેલ અંતર એ સ્રોતથી સૌથી ટૂંકી અંતર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:30)

નજીકથી સંબંધિત એલ્ગોરિધમ ન્યૂનતમ કિંમત ફેલાવવાના વૃક્ષ માટે પ્રાઈમની અલ્ગોરિધમનો છે. તેથી, અહીં આપણે વધતા જ એક વૃક્ષ બનાવીએ છીએ અને દરેક તબક્કે આપણે આ ફેલાયેલી ઝાડમાં ઉમેરીએ છીએ, નજીકના શિરચ્છેદ જે વૃક્ષમાં નથી. અને અહીં વૈશ્વિક શ્રેષ્ઠતમ જે આપણે પ્રાપ્ત કરી છે તે એ છે કે આપણે સ્પૅનિંગ વૃક્ષ બનાવ્યું છે જે લઘુત્તમ ખર્ચ છે. ન્યુનતમ ખર્ચ વિસ્તારવાના વૃક્ષ માટેનું બીજું ઍલ્ગોરિધમ ક્રુસ્કલનું એલ્ગોરિધમ (Kruskal’s algorithm) છે. અહીં, આપણે સીધા વૃક્ષ બનાવતા નથી, પરંતુ આપણે કિનારી એકઠા કરીએ છીએ અને એક જોડાયેલ ઘટક બનાવે છે જે એક વૃક્ષ બને છે. તેથી, અહીં આપણે આપણા સમૂહમાં કિનારોના વર્તમાન સમૂહમાં ઉમેરી રહ્યા છીએ, આગળનો સૌથી નાનો કિનારો જે તે ચક્ર બનાવતો નથી જેની સાથે આપણે પહેલાથી જ પસંદ કરી દીધી છે અને હવે વૈશ્વિક ઈષ્ટતમ એ છે કે આપણે આ રીતે એકત્રિત કરેલી ધાર લઘુત્તમ ખર્ચ વૃક્ષ ફેલાવો.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:31)

તો, હવે ચાલો એક સંપૂર્ણપણે અલગ સમસ્યા જોઈએ, અંતરાલ સુનિશ્ચિત તરીકે ઓળખાતી સમસ્યા. તેથી, ધારો કે અમારી પાસે એક વિશિષ્ટ વિડિઓ ક્લાસ રૂમ છે, જ્યાં અમે ઑનલાઈન ભાષણો આપી શકીએ છીએ. હવે, વિવિધ શિક્ષકો વર્ગને પહોંચાડવા માટે વર્ગ ખંડ બુક કરવા માંગે છે અને દરેક પ્રશિક્ષક પાસે એક સ્લોટ છે જે તે આ ભાષણને પહોંચાડવા માંગે છે. તેથી, પ્રશિક્ષક પાસે મારી પાસે સ્લોટ છે, ચાલો આપણે si સમયે એક સમયે શરુ કરીએ અને તેને f પર સમાપ્ત કરીએ. . તેથી, તમારી પાસે એક સ્લોટ છે જે સી થી શરૂ થાય છે અને અંતે સમાપ્ત થાય છે, હવે બે પ્રશિક્ષકો સ્લોટને લપેટ્યા કરતા હોઈ શકે છે. તેથી, કદાચ કોઈક જે સ્લોટ ઈચ્છે છે, તેથી લાલ સ્લોટ સમાપ્ત થાય તે પહેલા વાદળી સ્લોટ શરૂ થાય છે, તેથી દેખીતી રીતે બંને સ્લોટ બુકિંગમાં હોઈ શકતા નથી, કારણ કે એકબીજા સાથે દખલ કરવામાં આવી હતી. તેથી, અમારું કાર્ય એ બુકિંગ્સના સેટને જોવું અને સબસેટ પસંદ કરવું એ શક્ય છે જે બે બુકિંગ્સ છે જે અમે એકબીજા સાથે દખલ કરવાનું પસંદ કરીએ છીએ. તેથી, ત્યાં અમે રૂમનો ઉપયોગ કરવા માટે સંખ્યાબંધ શિક્ષકોને મહત્તમ કરીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 03:40)

તો, જો આપણે ગ્રીડી(Greedy) અભિગમને અનુસરીએ તો મોટે ભાગે આપણે આ કરીશું. બધી બુકિંગ્સમાં જે હજી ફાળવેલ નથી અને જે ફાળવવા માટે ઉપલબ્ધ છે. અમે કેટલીક સ્થાનિક વ્યૂહરચનાના આધારે એક પસંદ કરીશું, ત્યારબાદ અમે આ બુકિંગ સાથે ઓવરલેપ કરાયેલ બુકિંગ સાથેની બધી વિરોધાભાસી બુકિંગ્સ, બુકિંગ્સ જે અમે ફાળવેલ સ્લોટ સાથે અને કોઈપણ રીતે એવી દલીલ કરવી પડશે કે બુકિંગનો આ ક્રમ જેને આપણે પસંદ કરી રહ્યા છીએ તે નંબરને મહત્તમ કરે છે રૂમનો ઉપયોગ કરવા શિક્ષકોનો સમાવેશ થાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:12)

તેથી, ચાલો આપણે એવી કેટલીક લાક્ષણિક ગ્રીડી(Greedy) વ્યૂહરચનાઓ જોઈએ કે જે ઈચ્છે છે. તેથી, એક વ્યૂહરચના જે પ્રારંભિક પ્રારંભનો પ્રારંભ છે તે બુકિંગ પસંદ કરી શકે છે, પરંતુ કાઉન્ટર ઉદાહરણ સાથે આવવું મુશ્કેલ નથી. તેથી, જો તમે આ ચિત્રને જુઓ છો, તો ત્યાં એક લાંબી લીલી બુકિંગ છે જે પ્રારંભિક પ્રારંભ થાય છે અને વાસ્તવમાં અન્ય બુકિંગ્સ કર્યા પછી તે સમાપ્ત થાય છે. તેથી, જો આપણે આ ગ્રીડી(Greedy) વ્યૂહરચનાનો ઉપયોગ કરીએ તો અમે આ ખૂબ લાંબી બુકિંગ ફાળવીશું અને આખી અવધિ તેને માત્ર એક શિક્ષકને જ ફાળવવામાં આવશે, જ્યારે જો આપણે થોડીવાર પછી બુકિંગ પસંદ કરીશું, તો પછી આપણે ખરેખર છ શિક્ષકોની બુકિંગને સંતોષી શકીએ છીએ અને ત્યારથી અમારું લક્ષ્ય શિક્ષકોની સંખ્યા વધારવાનો છે, જે ઓરડામાં વધુ સારી વ્યૂહરચના હોઈ શકે છે. તેથી, આ ગ્રીડી(Greedy) વ્યૂહરચના સ્પષ્ટ રીતે ફ્લોપ થઈ ગઈ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:00)

અને બીજી ગ્રીડી(Greedy) વ્યૂહરચના જે આપણે વિચારીએ છીએ તે બુકિંગ પસંદ કરવાનું છે જેની અંતરાલ ટૂંકી છે. એકવાર ફરીથી અહીં એક ઉદાહરણ છે, મધ્યમાં અંતરાલ સૌથી ટૂંકું છે, પરંતુ જો આપણે આ પસંદ કરીએ તો તે અન્ય બુકિંગ્સ સાથે વિરોધાભાસમાં છે, તેથી અમારે બંનેને મહત્તમ અનુકૂળ કરવું પડશે. તેથી, જો આપણે ટૂંકા અંતરાલ પસંદ કરીએ તો આપણે માત્ર રૂમમાં એક શિક્ષક જ ફાળવી શકીએ છીએ, જ્યારે આપણે વ્યૂહરચનાને જાણીએ છીએ અને જો આપણે ખૂબ લાંબુ અંતરાલો પસંદ કરીએ, તો આપણે વાસ્તવમાં બે શિક્ષકો માટે રૂમનો ઉપયોગ કરી શકીએ છીએ અને તેના માટે વધુ સારું મેળવી શકીએ છીએ. અમે પસંદ કરેલ સમસ્યા છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:40)

તેથી, અગાઉના ઉદાહરણ સૂચવે છે કે તકરાર કરવા માટે કંઈક છે, તેથી કદાચ અમે બુકિંગ્સ પસંદ કરી શકીએ કે તેઓએ કેટલી બુકિંગ કરી હતી. તેથી, હવે એક એવી વ્યૂહરચના કે જે આપણે વિચારી શકીએ છીએ કે બુકિંગ પસંદ કરવાનું છે જે ઓછામાં ઓછા અન્ય બુકિંગ્સ સાથે ઓવરલેપ થાય છે. બીજા શબ્દોમાં કહીએ તો, આ બુકિંગ પસંદ કરીને આપણે નકારી કાઢીએ છીએ કે અન્ય બુકિંગ શક્ય છે. તેથી, ચાલો આપણે આ ઉદાહરણ જોઈએ, અહીં કેન્દ્ર બુકિંગ ફક્ત બે જ સાથે ઓવરલેપ થાય છે, આ એક અને આ, દરેક અન્ય બુકિંગ ઓછામાં ઓછા ત્રણ સાથે ઓવરલેપ થાય છે. તેથી, જો આપણે આ બુકિંગ પસંદ કરીએ, તો અમે તેની બાજુના બુકિંગ્સને નકારી કાઢીએ છીએ અને તેનો અર્થ એ છે કે, આપણે પણ ત્યાં છીએ, અમે આમાંથી એક અથવા એક કરી શકીએ છીએ. તેથી, જો આપણે આ કેન્દ્ર બુકિંગ લઈએ તો અમે એકંદરે ત્રણ બુકિંગ્સ પર કરી શકીએ છીએ, અમે બંને બાજુએ તે કરી શકતા નથી. તેથી, અમે ક્યાં તો બે આત્યંતિક કરી શકીએ છીએ અથવા આમાંના કોઈપણ અને આપણે કોઈપણ કરી શકીએ છીએ. તેથી, અમે કુલ ત્રણ કરી શકીએ છીએ, અમે આમાં કુલ ત્રણ બુકિંગ ફાળવી શકીએ છીએ. જો કે, જો આપણે આ ન કરીએ તો, જો આપણે વધુ સારી વ્યૂહરચના પસંદ કરીએ છીએ, તો સારી વ્યૂહરચના એ સ્પષ્ટપણે ટોચ પર ચાર લેવી પડશે. તેથી, જો આપણે આ વ્યૂહરચનાનો ઉપયોગ ન કરીએ તો, અમે આ રૂમમાં ચાર શિક્ષકોને ફાળવી શકીએ છીએ, પછી આપણે ઓછામાં ઓછા સંઘર્ષ સાથે એક પસંદ કરવો પડશે. તેથી, આ ગ્રીડી(Greedy) વ્યૂહરચના પણ સુધારાઈ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:17)

તેથી, અહીં એક ચોથી વ્યૂહરચના છે, જેમ કે આપણે પ્રારંભ કરીએ છીએ, જેમની શરૂઆતનો સમય પ્રારંભ થાય છે તે પસંદ કરવાને બદલે, ચાલો એક પસંદ કરીએ જેનો પ્રારંભનો સમય વહેલો હોય. તેથી, આપણે કાઉન્ટર ઉદાહરણ સાથે આવી શકીએ અથવા આપણે તેને સાબિત કરવાનો પ્રયત્ન કરવો જોઈએ કે આ સાચું છે. તેથી, હકીકતમાં આ વ્યૂહરચના કાર્ય કરે છે અને ચાલો જોઈએ કે આપણે તેને કેવી રીતે સાબિત કરી શકીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:41)

આપણે તેને સાબિત કરતા પહેલા, ચાલો આપણે થોડા વધુ સ્પષ્ટ રીતે એલ્ગોરિધમને લખીએ. તેથી, અમે બુકિંગ્સ બીના સેટથી પ્રારંભ કરીએ છીએ અને અમે આ સેટમાંથી, એક સ્વીકૃત બુકિંગ્સનો સબસેટ એ બનાવવા માંગીએ છીએ. તેથી, શરૂઆતમાં અમારી પાસે કોઈ સ્વીકૃત બુકિંગ નથી, કારણ કે અમે હમણાં જ આ સેટ બનાવવાની શરૂઆત કરી છે અને હવે

અમે નીચે મુજબ કરીએ છીએ. તેથી, જ્યાં સુધી અમારી પાસે બાકી બુકિંગ બાકી છે, ત્યાં સુધી અમે તે બુકિંગ પસંદ કરીએ છીએ જે સેટમાં સૌથી નાનો ફાઇનિંગ સમય છે

((સમય:08:14))

અને અમે તે ઉમેરીએ છીએ, તે A અને A હવે તેણે એમાં ઉમેર્યું છે, અમે આ બુકિંગ સાથે વધુ પડતી બુકિંગ શેડ્યૂલ કરી શકતા નથી. તેથી, અમે અમારા સેટ કેપિટલ બીમાંથી દૂર કરીએ છીએ, બૂકિંગ બ સાથેના બધા બુકિંગ્સઅમે ફક્ત પસંદ કરીએ છીએ. તેથી, દર વખતે અમે આગલી બુકિંગ પસંદ કરીએ છીએ જે હજી પણ નાના અંતિમ સમય સાથે ઉપલબ્ધ છે અને અમે તેની સાથે સંઘર્ષ કરતો બધું દૂર કરીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 08:44)

તેથી, અહીં પાવર એલ્ગોરિધમનો દાખલો છે જે કાર્ય કરે છે. તેથી, અહીં આપણી પાસે નવ બુકિંગ છે, વાદળી રેખાઓ બુકિંગ સૂચવે છે અને બુકિંગની સંખ્યા તેના ઉપર આપવામાં આવે છે. તેથી, આમાં, એક સાથે ... તેથી, શરૂઆતમાં અમારા સેટ બી પાસે આ નવ નવ બુકિંગ્સ છે અને અમારું સેટ એ શરૂઆતમાં ખાલી છે. તેથી, હવે આપણે જે જોઈએ છીએ તે નવ બુકિંગ્સમાં એક નાનો અંતિમ સમય છે અને તે 1 થાય છે. તેથી, આપણે 1 પસંદ કરીએ છીએ અને પછી 1 પસંદ કર્યા પછી, તે બધી બુકિંગ્સ જે તેની સાથે ઓવરલેપ થાય છે. તેથી, 2 સાથે 1 ઓવરલેપ થાય છે અને 6 પર સંગ્રહિત થાય છે, તેથી અમે અમારા સેટમાંથી 2 ખસેડીશું અને અમે અમારા સેટમાંથી 6 ને દૂર કરીશું, તેથી હવે બીને 3, 4, 5, 6, 7, 8, 9 અને એમાં ફેંકવામાં આવ્યા છે. બુકિંગ નંબર 1 છે. તેથી, હવે આ સંભવિત સેટ 3, 4, 5, 6, 7, 8, 9 વચ્ચે અમે તે પસંદ કરીએ છીએ જે સૌથી પહેલા સમાપ્ત થાય છે 3 અને પછીથી 4 એ 3 સાથે વિરોધાભાસ છે, આપણે 4 ને દૂર કરીએ છીએ. તેથી, આ રીતે ચાલુ રાખીએ, આપણે હવે 5 પસંદ કરીએ, કારણ કે 5 એ સમાપ્ત થવા માટે સૌથી પહેલું છે અને પછી 7 કારણ કે 5 સાથે સંઘર્ષ છે, આપણે 7 ને દૂર કરીએ છીએ. અને હવે આપણી પાસે બે બાકી છે, 8 અને 9, પણ 9 પહેલા 8 સમાપ્ત થાય છે, આપણે વાસ્તવમાં કોઈ એક પસંદ કરી શકે છે, પરંતુ અમારા એલ્ગોરિધમ 8 પસંદ કરશે, કારણ કે 8 ટૂંકા ગાળાનો સમય ધરાવે છે. તો, આપણે 8 પસંદ કરીશું અને પછી આપણે કહીશું કે 9 શક્ય નથી, તેથી આપણે તેને પસંદ નથી કરતા અને હવે આપણી પાસે ખાલી જગ્યા છે અને એ 1, 3, 5, 8 છે અને બી ખાલી છે, તેથી આપણી પાસે હવે વધુ નથી શેડ્યૂલ કરવા માટેની નોકરી, સન્માન માટે વધુ બુકિંગ્સ, તેથી અલ્ગોરિધમનો અંત થાય છે. તેથી, અમે આ યાદીમાં ચાર બુકિંગ્સનો સંભવિત સમૂહ શોધી કાઢ્યો છે જેનો સમાવેશ કરી શકાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:46)

તેથી, અમારું ધ્યેય એ બતાવવાનું છે કે એલ્ગોરિધમ, સોલ્યુશન એ આપણા અલ્ગોરિધમ દ્વારા પેદા થાય છે તે ખરેખર સાચું છે. તેથી, ધારો કે બુકિંગ ઓનું શ્રેષ્ઠતમ સેટ છે, હવે આપણે સામાન્ય રીતે એવું માની શકતા નથી કે આપણું સોલ્યુશન એ ઓ સમાન છે, કારણ કે ત્યાં કદાચ સમાન કદના ઉકેલો ઉત્પન્ન કરવાના અનેક રસ્તાઓ છે. યાદ રાખો કે આપણે જે જોઈએ છે તે એક ઉકેલ છે જે ઘણા બધા શિક્ષકોને રૂમમાં ફાળવે છે. તેથી, શિક્ષકોની સમાન સંખ્યા ફાળવવા માટે કદાચ બે અલગ અલગ માર્ગો હોઈ શકે છે, તેથી આપણે એવી દલીલ કરી શકતા નથી કે A અને O એક સમાન છે, પરંતુ તે

બતાવવું આશ્ચર્યજનક છે કે A અને O એ સમાન કદના છે. બીજા શબ્દોમાં કહીએ તો, બીજી કોઈ વ્યૂહરચના દ્વારા કઈ શ્રેષ્ઠ બુકિંગ ઉત્પન્ન થાય છે, અમારી વ્યૂહરચના અમારી ગ્રીડી(Greedy) વ્યૂહરચના એ એક જ કદનું ઉત્પાદન કરે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:36)

તેથી, એ બુકિંગનો સમૂહ બનશે કે જે અમારી વ્યૂહરચના પસંદ કરે છે અને આ તે ઓર્ડર હોઈ શકે છે અને જે અમે પસંદ કરી શકીએ છીએ, તેથી હું 1 પસંદ કરું છું અને પછી હું 2 અને બીજું, તેથી જ્યારે હું 1 પસંદ થયેલ છે અને i 2 હજી પણ શક્ય છે અને કારણ કે હું 1 સંપૂર્ણરૂપે પ્રારંભિક અંતિમ સમય હતો, અમારું કહેવું છે કે i નું સમાપ્તિ i 2 ના પ્રારંભના સમય પહેલા છે, i 2 નું સમાપ્ત થવાનું સમય એ પ્રારંભિક સમય પહેલા છે. 3 અને તેથી. તેથી, એમાં આ બુકિંગ સૉર્ટ કરેલ ક્રમમાં છે, હવે ચાલો ધારીએ કે અમારી પાસે m બુકિંગ j1 થી jm સાથે સૉર્ટ કરેલ ક્રમમાં ફરીથી અનુકૂળ ઉકેલ છે. તેથી, j 2 શરુ થાય તે પહેલાં j 1 સમાપ્ત થાય છે, જે 2 જ 3 પ્રારંભ થાય તે પહેલાં જ અંત થાય છે અને બીજું. તેથી, અમારું ધ્યેય એ બતાવવાનું છે કે કે હકીકતમાં તે કે સમાન છે, અન્ય શબ્દોમાં, શ્રેષ્ઠતમ ઉકેલ એ ગ્રીડી(Greedy) વ્યૂહરચનાને ઉત્પન્ન કરેલા સોલ્યુશન જેટલું જ માપ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:30)

તેથી, આપણે ખરેખર બતાવીશું કે ક્રમ અને i માં દરેક જોબ માટે, એ અનુક્રમમાં અનુરૂપ જોબ એ અનુક્રમમાં સમાન અનુક્રમણિકા કરતાં પછીથી સમાપ્ત થતું નથી. તેથી, પ્રત્યેક આર અપ ટુ કે માટે, ir ની f એ j r ની પહેલા અથવા તેના બરાબર બરાબર છે. તેથી, આ અર્થમાં આપણે એવી દલીલ કરવાનો પ્રયાસ કરી રહ્યા છીએ કે ગ્રીડી(Greedy) ઉકેલ કોઈપણ શ્રેષ્ઠ ઉકેલથી આગળ રહે છે, અમે કોઈપણ અન્ય પદ્ધતિ દ્વારા પેદા કરી શકીએ છીએ. તેથી, આ દાવાની પુરાવા આર પર રજૂઆત દ્વારા છે. તેથી, જ્યારે આપણે પ્રથમ જોબ 1 પર નજર કરીએ છીએ, ત્યારે આપણે જાણીએ છીએ કે 1 એ બધી જ નોકરીઓ વચ્ચેનો પ્રારંભિક સમાપ્તિ સમય છે, અમારી સૂચિમાંની તમામ બુકિંગ્સ, કારણ કે મારી પાસે 1 બુકિંગ પરનો પ્રારંભિક સમાપ્તિ સમય છે, તે આવશ્યક છે ચોક્કસપણે j 1 ની f કરતાં ઓછી અથવા બરાબર હોવી જોઈએ, કારણ કે j 1 એકંદર સમાપ્તિ સમય કરતાં નાના હોઈ શકતા નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:30)

હવે, ચાલો ધારીએ કે અમે ઈન્ડક્શન દ્વારા સ્થાપિત કર્યું છે, જે 1 ની બાદબાકી 1 સુધી છે, હું ઓછા 1 ની બુકિંગ, માઈનસ 1 નો સંપૂર્ણ સમય ધરાવે છે જે પહેલાથી જ જે 1 આર ની બાદબાકી 1 બુકિંગ કરતા પહેલા છે. પછી, અમે દાવો કરીએ છીએ કે તે જરુરી છે જે જુનિયર પહેલા પૂર્ણ થાય છે, કારણ કે જો આપણી પાસે ન હોત તો આપણી પાસે નીચે ચિત્ર હશે. તેથી, આપણી પાસે જે.આર. માઈનસ 1 કરતા પહેલાં તે આઈઆર ઓછા 1 સમાપ્ત થાય છે. હવે, માની લો કે આપણે જુનિયર ઈઆર પહેલા ખરેખર સમાપ્ત થાય છે, તો પછી આ તબક્કે આપણો અલ્ગોરિધમ જોશે કે j r હજી પણ શક્ય છે, કારણ કે તે આઈઆર 1 બાદ ઓવરલેપ કરતું નથી અને જુનિયર રહેવાની નોકરીમાં આર કરતા પહેલાનો પ્રારંભિક સમય છે. તેથી, આપણી ગ્રીડી(Greedy) વ્યૂહરચના આઈઆર કરતા જુનિયર પસંદ કરશે, તેથી આ હકીકત છે કે આપણે આઈઆર લીધી છે અને j r નથી એટલે કે આપણે આ જેવી ચિત્ર મેળવી શકીએ નહીં. તે હોઈ શકે નહીં કે ir જુનિયર પછી કડક રીતે સમાપ્ત થાય છે, તે j r ની અંત પહેલા અથવા તે જ સમયે સમાપ્ત થવું આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:37)

તેથી, હવે બતાવ્યું છે કે ગ્રીડી(Greedy) વ્યૂહરચના હંમેશાં આગળ છે, હવે આપણે દાવો કરીશું કે ખરેખર અમારું સોલ્યુશન શ્રેષ્ઠ હોવું જોઈએ. તો, માની લો કે m ખરેખર k કરતાં કડક છે, તો આપણે જાણીએ છીએ કે જ્યારે આપણે ik સુધી પહોંચીએ છીએ, તે j કે પહેલા છે. હવે, કારણ કે આપણી પાસે ઉકેલ કરતાં લાંબા સમય સુધી એક ઉકેલ છે, જે પછી જેકે પ્લસ 1 કહેવાતી બીજી નોકરી છે, એમ ધારે છે કે એમ સખત રીતે છે, કારણ કે તે નોકરીની બુકિંગમાં જાય છે. તેથી, કારણ કે આ થાય છે ત્યાં જેકેનું અનુક્રમ હોવું જ જોઈએ, જેકે પછી બુકિંગની શ્રેણી, તેથી ચાલો આપણે અનુક્રમણિકા જોઈએ. હવે, દાવા એ છે કે આ બિંદુએ આ ચોક્કસ બુકિંગ પહેલાં થયું છે તે કોઈપણ વસ્તુ દ્વારા નકારી કાઢવામાં આવ્યું નથી, તેથી જો આપણે ik સુધી 1 ને જોવું જોઈએ, આમાંના કોઈ પણ જેકે પ્લસ 1 સાથે ઓવરલેપ કરશે નહીં, કારણ કે જેકે પ્લસ 1 જે પછી છે કે. તેથી, જેકે પ્લસ 1 પ્રારંભ થાય તે પહેલા, ik જેકે અને જેકે સમાપ્ત થાય તે પહેલા સમાપ્ત થાય છે. તેથી, ik જેકે પ્લસ 1 સાથે સુસંગત છે, આનો અર્થ આ તબક્કે બી ખાલી નથી. જ્યારે આપણે અમારી ગ્રીડી(Greedy) એલ્ગોરિધમ પ્રોસેસિંગ 1 માં ik ને ખાલી કરવા માટે આને સમાપ્ત ન કરવું જોઈએ, પરંતુ અમે દાવો કરીએ છીએ કે અમે ik સાથે પ્રારંભ કરીએ છીએ અને અમારું ગ્રીડી(Greedy) એલ્ગોરિધમ રોકશે તે એકમાત્ર કારણ છે કારણ કે બી ખાલી છે. તેથી, જો કોઈ નોકરી અથવા બુકિંગ જેકે પ્લસ 1 હોય, તો તે હોઈ શકે નહીં કે આ સમયે અમારી અલ્ગોરિધમ બંધ થઈ ગઈ છે, તેથી વિરોધાભાસ છે. તેથી, તેથી અમારી પાસે શ્રેષ્ઠતમ ઉકેલમાં કોઈ બુકિંગ ન હોઈ શકે કે જે k થી આગળ જાય અને તેથી, m એ k સમાન હોવું આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:24)

તેથી, બતાવ્યું કે તે સાચું છે, ચાલો આપણે તરત જ જોઈએ કે આપણે આ કેવી રીતે અમલમાં મૂકીશું અને જટિલતાના ઉપરના બાઉન્ડનો અંદાજ કરીશું. તેથી, શરૂઆતમાં, અમે સમય સમાપ્ત કરીને એમ બુકિંગ્સને સૉર્ટ કરીએ છીએ, આ એન બુકિંગ માટે ટાઇમ એન લોગ એન લે છે અને હવે આપણે ધારીએ કે જો આ બુકિંગ ક્રમમાં 1, 2 અપ ટુ એનને બુકિંગ કરવામાં આવે છે. તેથી, બુકિંગ 1 નું પ્રારંભિક પૂર્ણાહુતિ સમય છે, બુકિંગ 2 નું બીજું પ્રારંભિક અંતિમ સમય છે અને બીજું. હવે, અમે એક ઓર્ડર n સ્કેન માં સેટ કર્યું છે, એરે એસટી જેમ કે ST માં i બુકિંગ એરેનો પ્રારંભનો સમય છે. હવે, અમે બુકિંગ 1 થી પ્રારંભ કરીએ છીએ અને દર વખતે જ્યારે અમે બુકિંગ જે પસંદ કરીએ છીએ, ત્યારે આપણે j plus 1 થી પ્રારંભ કરીએ છીએ અને બુકિંગના શરૂઆતના સમયને સ્કેન કરવાનું ચાલુ રાખીએ છીએ જ્યાં સુધી આપણે પ્રારંભિક કે જેનો પ્રારંભ સમય j ની આગળ ન હોય. બીજા શબ્દોમાં કહીએ તો, આપણે જોઈ રહ્યા છીએ ... તેથી, આપણે જાણીએ છીએ કે આ બુકિંગ અંતિમ સમય માટે છે. તેથી, આપણે જાણીએ છીએ કે જે પછી બુકિંગ જે આગામી સમાપ્ત થાય છે તે j plus 1 છે, પરંતુ જો તે સમય શરૂ થાય છે તે j ના અંતિમ સમયની બહાર નથી, તે ઓવરલેપિંગ છે, તેથી તે સુસંગત હોઈ શકતું નથી. તેથી, અમે આ એરે એસટીને સ્કેન કરીએ ત્યાં સુધી આપણે સૌથી નાના કે જે FJ સમાપ્ત થયા પછી શરૂ થાય ત્યાં સુધી શોધે છે. તેથી, આ રીતે એક ઓર્ડર એન સ્કેન અમે અમારી તમામ બુકિંગ્સમાંથી પસાર થઈ શકીએ છીએ અને ગ્રીડી(Greedy) ઇષ્ટતમ સેટ લઈ શકીએ છીએ. તેથી, આ ઓર્ડર n સ્કેન સૉર્ટિંગ ઑર્ડર n લોગ ઇન લે છે, તેથી એકંદરે આ ગ્રીડી(Greedy) વ્યૂહરચના સાચી છે અને તે સમય લે છે O n લોગ n.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 06
મૉડ્યૂલ - 04
લેક્ચર - 42
ગ્રીડી એલ્ગોરિધમ્સ (Greedy algorithms): લેટનેસ ઘટાડે છે

હવે આપણે એક અલગ ગ્રીડી(Greedy) એલ્ગોરિધમ જુએ છે જે સાચાંપાના ના વધુ જટિલ સાબિતી સાથે છે. તેથી, આપણે જે સમસ્યાની તપાસ કરી રહ્યા છીએ તેને લઘુતમ ઘટાડવા કહેવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:12)

તેથી, છેલ્લા ઉદાહરણમાં આપણી અંતરાલ શેડ્યૂલિંગ સમસ્યાની જેમ, અમારી પાસે એક જ સ્રોત છે અને આ સ્રોતનો ઉપયોગ કરવાની વિનંતી છે. તેથી, હવે અગાઉની પરિસ્થિતિથી વિપરીત જ્યાં અમારી પાસે પ્રારંભનો સમય અને સમાપ્તિનો સમય હતો અને આ સમય દરમિયાન સંસાધનનું નિર્ધારણ કરવું પડ્યું હતું. અહીં આપણે ફક્ત જાણીએ છીએ કે દરેક વિનંતીને પૂર્ણ કરવા માટે હું સમયનો સમય માંગું છું અને હું પ્રત્યેક વિનંતીને હું સમયસીમા સાથે આવું છું, અહીં અમે દરેક વિનંતીને શેડ્યૂલ કરવા જઈ રહ્યા છીએ. તેથી, દરેક વિનંતી જે જે શરૂઆતના સમયે શરૂ થશે, તેને એસજે કહેવામાં આવે છે અને તે સમય લેશે TJ, તેથી તે j ની f પર સમાપ્ત થશે, જે j ના સમાપ્ત થાય છે જે શરૂઆતનો સમય હશે અને તે સમય લેશે પ્રક્રિયા વિનંતી હવે, જો આ સમાપ્તિ સમય સીમાચિહ્ન કરતા મોટો છે, તો તે મોડું થઈ ગયું છે, તેથી તે વિલંબની રકમ વિલંબ અને સમાપ્તિ સમય વચ્ચેના તફાવત દ્વારા આપવામાં આવે છે અને લક્ષ્ય એ શેડ્યૂલ શોધવાનું છે જે મહત્તમ અક્ષમતાને ઘટાડે છે. તેથી, આપણે બધા જજે પર આ એલજેની મહત્તમ કિંમત ઘટાડવા માંગીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:28)

તેથી, આપણે જાણીએ છીએ કે આપણે ગ્રીડી(Greedy) વ્યૂહરચનાઓ શોધી રહ્યા છીએ, ચાલો આપણે આ સમસ્યા માટે કેટલીક ગ્રીડી(Greedy) વ્યૂહરચનાઓ અજમાવવા અને સૂચવવાનું શરૂ કરીએ. તો, માની લો કે અમે શક્ય તેટલી ઝડપથી જોબ સમાપ્ત કરવા માંગીએ છીએ, તેથી અમે સૌપ્રથમ ટૂંકી જોબ પસંદ કરીએ છીએ, તેથી અમે લંબાઈના ઓર્ડરમાં જોબ પસંદ કરીએ છીએ. તેથી, આ ગ્રીડી(Greedy) વ્યૂહરચના હોઈ શકે છે, પરંતુ કમનસીબે એકદમ સરળ કાઉન્ટર ઉદાહરણ છે. તો, માની લો કે આપણી પાસે 2 જોબની જોબ 1 ટાઈમ યુનિટ લે છે અને જોબ 2 એ 10 સમય એકમો લે છે, પરંતુ સમય સીમા અનુક્રમે 110 અને 10 છે. બીજા શબ્દોમાં કહીએ તો, પ્રથમ જોબમાં ઘણો લાંબો અંતર હોય છે જેમાં તેને કોઈપણ દંડ વિના સુનિશ્ચિત કરી શકાય છે, જ્યારે આ બીજી જોબએ પૂર્ણ થવાનું શરૂ કરવાનું ધારી લીધું છે. તેથી, હવે જો તમે આ સૌથી ટૂંકી જોબ પસંદ કરો છો, તો આપણે 1 ની અક્ષમતાનો અંત લાવીશું, કારણ કે આપણે 1 થી જઈશું અને પછી આપણે 2 થી 11 સુધી જઈશું. તેથી, બીજી જોબ સમાપ્ત થશે. 1 ની એકમ મોડું થાય છે, બીજી તરફ જો આપણે 1 થી 10 કરીએ તો આપણે 11 કરીએ, પછી આપણે કોઈ લેટનેસ નહી મેળવીએ, આપણે લેટિનેસ 0 મેળવીએ. તેથી, અહીં સૌથી નાનો જોબ પસંદ કરવો એ આપણને શ્રેષ્ઠ જવાબ આપતું નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:44)

તેથી, બીજી વ્યૂહરચના તે જોબ પસંદ કરી શકે છે, તેથી અગાઉ આપણે જોયું કે અમારી પાસે જોબ છે જેની પાસે 10 સમય એકમો છે અને તેની પણ 10 ની સમયસીમાની જરૂર પડશે. તેથી, અમે તે જોબ પસંદ કરવાની જરૂર છે, જેની સમય સીમા સમાપ્ત થઈ જાય છે. તેથી, અમે અમારી જોબમાં શરૂઆતમાં વિલંબ કરવા માટે કેટલો સમય પ્રયાસ કરી શકીએ તે વિશે વિચારીએ છીએ, ડીજે ઓછા ટીએજે અને નાના કદના લોકોની પસંદગી કરો. તેથી, અહીં આપણી પાસે પહેલી જ એક ખૂબ જ સમાન ઉદાહરણ છે, સિવાય કે પ્રથમ કામની સમય સીમા જે હવે ... તેથી, અહીં આપણી પાસે બીજા જોબ માટે 0 ની સંખ્યા છે અને પહેલી જોબ છે, તેથી બીજી વ્યક્તિ પાસે તે સમયની સમકક્ષ સમય મર્યાદા છે, પ્રથમની સમયરેખા છે જે સમયે એક નોડ છે. તેથી, પછી આ વ્યૂહરચના દ્વારા આપણે પ્રથમ 2 પસંદ કરીશું અને જો તમે t 1 પછી t2 પસંદ કરશો, તો પછી શું થાય છે કે આ લેટનેસ 11 ઓછા 2 થશે, કારણ કે આપણે પ્રથમ t થી 2. તેથી, આપણે t 1 જોબ 1 ફક્ત એક સમય 11 છે, તેથી તે સમય 11 સમાપ્ત કરે છે, પરંતુ તે 2 સમાપ્ત હોવો જોઈએ, તેથી લાપતા 9 છે. બીજી બાજુ, જો આપણે t 1 પછી t 1 કરીએ, તો આપણી પાસે તે લેટનેસ તે ફક્ત 1 છે, કારણ કે બીજી જોબ 10 ની સમાપ્ત થઈ ગઈ હોવી જોઈએ, તેના બદલે તે 11 વાગ્યે સમાપ્ત થાય છે. તેથી, 11

((સમયનોસંદર્ભ: 04:15))

1 છે, તેથી અત્યારે અહીં આપણી અંતર્જ્ઞાન અમને આ સૌથી નાનો શાંત સમય પસંદ કરવા માટે કહે છે ખરેખર તે સારું નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:25)

તેથી, તે બતાવે છે કે કામ કરવાની લાલચવાળી વ્યૂહરચના એ જ પહેલાની પ્રથમ અંતિમ તારીખ ડીની પસંદગી કરવાનું છે, આ પડકાર સાબિતી આપે છે કે આ વ્યૂહરચના ખરેખર સાચી છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:40)

તેથી, સાબિતી સાબિત કરવા માટે આપણે સૌ પ્રથમ ધારીશું કે આપણે સમયસરના ક્રમમાં અમારી બધી જ જોબનો આંક આંક્યો છે. તેથી, અમે અમારી જોબ 1, 2, 3 ને n સુધી ક્રમાંકિત કરીએ, તેથી તે 1 ની સમય સીમા ઓછી છે અથવા 2 ની સમય સીમાની સમાન છે અને તેથી આગળ. હવે, આ અમારા શેડ્યૂલ કર્યા પછી ખૂબ સીધી આગળ છે, અમે માત્ર જોબ 1, પછી જોબ 2, પછી જોબ 3 અને તેથી જ શેડ્યૂલ કરીએ છીએ. એક વખત અમે સમયસર દ્વારા જોબને ટૂંકાવી દીધા પછી, આપણે તે ક્રમમાં ગોઠવીએ છીએ, અમે તેને તે ક્રમમાં શેડ્યૂલ કરીએ છીએ જે 1 થી શરૂ થાય છે તે સમય 0 થી શરૂ થાય છે, જે સેટ 1 તરીકે બોલાવે છે, તે 1 ના f પર બંધ થાય છે જે t ની છે 1 0 વત્તા ટી 1. હવે, 2 એ જોબ 2 માટેનો પ્રારંભનો સમય જલ્દી 1 ના રોજ છે, તેથી એફ 1 પર આપણે 2 નો પ્રારંભ કરીએ છીએ અને તે 2 ની વત્તા ટી 2 પર સમાપ્ત થશે. તેથી, હવે પણ 3 એફ 2 હશે, આપણે ટાઈમ 2 સમયે 3 જૉબ શરૂ કરીશું અને આપણે એસ 3 પ્લસ ટી 3 પર

જઈશું અને આ એફ 3 ને બોલાવીશું. તેથી, આ સમયરેખાના પાછલા એક જેટલા જલદી જ આપણે દરેક જોબને શેડ્યૂલ કરીશું. .

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:46)

તેથી, આપણે રાહ જોયા વિના એક પછી એક પછીની જોબ સુનિશ્ચિત કરી રહ્યા છીએ, તે ખૂબ જ સ્પષ્ટ છે કે આ શેડ્યૂલમાં કોઈ અંતર નથી, તેમાં નિષ્ક્રિય સમય નથી. જે સ્રોત અમે ફાળવવાનો પ્રયાસ કરી રહ્યા છીએ તે સતત ઉપયોગમાં છે, જ્યાં સુધી બધી n વિનંતીઓ સમાપ્ત થાય નહીં. તેથી, હવે એવો દાવો છે કે ત્યાં એક અનુકૂળ શેડ્યૂલ છે જેમાં નિષ્ક્રિય સમય નથી, કારણ કે ધારો કે તમારી પાસે મહત્તમ અનુક્રમણિકા છે જેમાં તમારી પાસે આ પ્રકારનાં બ્લોક્સ છે જ્યાં સંસાધનોનો ઉપયોગ કરવામાં આવે છે અને આ વચ્ચેનો અંતર નિષ્ક્રિય હતો. જ્યારે, હું ખૂબ જ સ્પષ્ટ કરી શકું છું કે હું આ વસ્તુઓને આગળ ખસેડી શકું છું, આ જુઓ, જ્યારે હું આને છોડી શકું ત્યારે કોઈ અવરોધ નથી, જ્યારે વસ્તુ સમાપ્ત થઈ જાય ત્યારે જ મને અવરોધ છે. તેથી, જ્યારે વસ્તુઓને પહેલા ખસેડવામાં આવે ત્યારે હું માત્ર અક્ષમતાને ઘટાડી શકું છું, તેથી જો અંતર સાથેનો વાદળી શેડ્યૂલ શ્રેષ્ઠ હતો, તો હું તેને ખસેડી શકું છું, જેથી તેમાં અવરોધો ન હોય અને ચોક્કસપણે મારા નવા શેડ્યૂલમાં વાદળી રંગમાં કોઈ વધુ અક્ષમતા હોતી નથી. તેથી, અમે હંમેશાં ધારી લઈએ છીએ કે શ્રેષ્ઠતમ શેડ્યૂલ પાસે નિષ્ક્રિય સમય નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:51)

તેથી, હવે અમારું લક્ષ્ય વાસ્તવમાં એવી દલીલ છે કે આ શેડ્યૂલ કે જે અમે સમયસમાપ્તિના સંદર્ભમાં સૉર્ટ કરીને નિર્માણ કર્યું છે અને પછી તે ક્રમમાં આંખનો ઉપયોગ કરીને, આ કોઈપણ શ્રેષ્ઠતમ શેડ્યૂલ તરીકે હોઈ શકે છે. તેથી, અહીં પાછલા અંતરાલ સુનિશ્ચિત સમસ્યામાં, અમે કહ્યું કે અમે તે શેડ્યૂલને ગેરેંટી આપી શકતા નથી કે જે અમને મળ્યું છે તે શ્રેષ્ઠ ઑપ્ટિમ શેડ્યૂલની બરાબર છે, પરંતુ અમે ફક્ત તે જ કદ બતાવીશું. હવે, અહીં આપણે થોડું અલગ કરીએ છીએ, અમે કોઈ અન્ય વ્યૂહરચના દ્વારા ઉત્પન્ન થયેલ મહત્તમ અનુક્રમણિકા લઈશું અને આપણે તેને એક પગલું દ્વારા તબદીલ કરીશું જે આપણે ઉત્પન્ન કર્યા છે. તેથી, આને એક્સ્ચેન્જ દલીલ કહેવામાં આવે છે, આપણે કેટલાક શેડ્યૂલથી શરૂઆત કરીએ છીએ અને ત્યારબાદ આપણે તે શેડ્યૂલમાં વસ્તુઓને શ્રેષ્ઠતા જાળવી રાખીને ચાલુ રાખીએ છીએ, ત્યાં સુધી આપણે આખરે શેડ્યૂલ O ને આપણા શેડ્યૂલ એમાં ફેરવીશું, જે ગ્રીડી(Greedy) વ્યૂહરચનામાં આવશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:52)

તેથી, અમારી વ્યૂહરચના પ્રક્રિયાઓ અને સમયરેખાના આધારે જોબ શેડ્યૂલ કરે છે, તેથી અમે કહી શકીએ કે આ શેડ્યૂલ O, તમારા શ્રેષ્ઠ સમયપત્રકમાં એક બદલાવ છે, જો તેની પાસે વાસ્તવમાં બે જોબ છે જે હુકમની બહાર દેખાય છે અન્તિમ રેખા. તેથી, ત્યાં જોબ છે જે જોબ જે અને ઓ પહેલા દેખાય છે, પરંતુ j ની અંતિમ તારીખ i ની સમયરેખા પહેલા સખત રીતે છે. તો, ધ્યાન રાખો કે અમારું સોલ્યુશન, કારણ કે ગ્રીડી(Greedy) ઉકેલ પ્રક્રિયાઓ સમયરેખાના ક્રમમાં બદલાતી રહે છે, ત્યાં અમારા શેડ્યૂલમાં કોઈ બદલાવ હોઈ શકતું નથી, પરંતુ અમે જે ઑપ્ટિમાઈઝ ઈષ્ટતમ અનુવર્તી શેડ્યૂલ રજૂ કરી છે તે શ્રેષ્ઠ હોઈ શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:37)

તેથી, હવે પ્રથમ મુદ્દો એ છે કે જો તમારી પાસે કોઈ વ્યુત્ક્રમ અને નિષ્ક્રિય સમય ન હોય તો, લેટનેસ સમાન હોવી આવશ્યક છે. તો, સૌ પ્રથમ, જો તમારી પાસે કોઈ બદલાવ ન હોય અને નાનિષ્ક્રિય સમય પછી, અમારી પાસે એકમાત્ર લવચીકતા પુનઃક્રમાંકિત કરવાનો છે, કારણ કે અમે અગાઉની સમયસીમા સાથેની વસ્તુઓની આગળની સમય સીમા સાથે વસ્તુઓ મૂકવાની મંજૂરી આપતા નથી. અમારી પાસે એકમાત્ર સુગમતા છે જે સમાન સમયરેખા સાથે વસ્તુઓને ફરીથી ગોઠવવાનું છે, અમારી પાસે સમાન સમયરેખામાં બહુવિધ જોબ હોઈ શકે છે અને અમે આ જ સમયસીમાની વિવિધ શ્રેણીબદ્ધ પુનરાવર્તન અથવા અલગ રેન્ડરિંગ પસંદ કરી શકીએ છીએ, તેઓ ઈનવર્ઝનને માન્ય કરશે નહીં, કારણ કે તે સમાન છે. પરંતુ, માત્ર ત્યારે જ બદલાવ થાય છે જ્યારે સખત રીતે બનેલી વસ્તુ પછી કંઈક કડક રીતે આવે છે. તેથી, દાવા એ છે કે આવી પરિસ્થિતિમાં, અમે અલગ જવાબ ધરાવી શકતા નથી, કારણ કે તમે અમારા સ્વયંને સમાન સમયમર્યાદામાં જોબને શફલ કરવાની મંજૂરી આપો છો. તેથી, અહીં એક ચિત્ર છે, તો ધારો કે આ ત્રણ જોબ, વાદળી જોબ, પીળી જોબમાં, લાલ કામની બધી જ મૂત રેખા છે. તો, અહીં એક ક્રમ છે જ્યાં આપણે પહેલા વાદળી રંગ કરીએ, પછી પીળો, પછી લાલ અહીં બીજું અનુક્રમ છે, તેથી આપણે પહેલા લાલ, પછી વાદળી અને પીળો રંગ કરીએ. હવે, બધા પાસે તે જ સમયસીમા છે, તેથી આ જ અંતિમ મુદત આ બિંદુએ છે. તેથી, સમય સીમા અહીં છે, હવે આ કામની છેલ્લી બાબતો ધ્યાનમાં રાખીને આપણે તેને કેવી રીતે શફલ કરી શકીએ તે જ મુદ્દાને સમાપ્ત કરશે. કારણ કે, અમારી પાસે કુલ સમયનો સરવાળો છે અને તે અંત હશે અને આમાં છેલ્લું જોબ હશે, આ સમયરેખાને ધ્યાનમાં રાખીને મહત્તમ વિલંબ. તેથી, કારણ કે આપણે મહત્તમ અક્ષમતા ગણીએ છીએ, મહત્તમ અક્ષમતા બદલાઈ શકે તેમ નથી, હું આ જોબ કેવી રીતે શફલ કરું છું, જેનો કોઈ પણ સમય સમાપ્ત થાય છે તે જ સમયે સમાપ્ત થશે, કારણ કે આ બધા એક જ લંબાઈનાં છે અથવા મારો અર્થ એ છે કે આ સમાન લંબાઈને ધ્યાનમાં લીધા વગર હું કેવી રીતે શફલ કરું છું. અને તેથી, લેટનેસ બદલાતું નથી, તેથી કોઈકમાં જો તમારી પાસે બે શેડ્યૂલ હોય છે જેમાં વધુ વ્યુત્પત્તિ હોય છે અને નિષ્ક્રિય સમય નથી, તો પછી આપણે ઉત્પન્ન થતા અક્ષમતાના સંદર્ભમાં આ જવાબ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:41)

તેથી, હવે જો તમે દાવો કરી શકો છો કે કોઈ ઈનવર્ઝન અથવા નિષ્ક્રિય સમય વિના શ્રેષ્ઠ અનુક્રમણિકા છે. હવે, યાદ રાખો કે અમારા શેડ્યૂલ એ પાસે આ મિલકત છે; એ બાંધકામ દ્વારા કોઈ બદલાવ નથી અને નિષ્ક્રિય સમય નથી. અને હવે હું દાવો કરું છું કે ત્યાં એક શ્રેષ્ઠતમ શેડ્યૂલ છે, અમે ઓ સાથે પ્રારંભ કરવા જઈ રહ્યા છીએ અને અમે આ કંઈપણ પ્રાઈમમાંથી ઉત્પન્ન કરવા જઈ રહ્યા છીએ જેમાં કોઈ બદલાવ નથી, નિષ્ક્રિય સમય નથી. અને પહેલાની ટિપ્પણી દ્વારા, જ્યારે ઓ પ્રાઈમ અને એ બંને પાસે કોઈ નિષ્ક્રિયતા નથી, તે નિષ્ક્રિય સમય નથી, તે વાસ્તવમાં તે જ અક્ષમતા ઉત્પન્ન કરે છે. તો, આપણે આ કેવી રીતે કરીએ? તેથી, સૌ પ્રથમ, આપણે જાણીએ છીએ કે અમે ધારે છે કે શ્રેષ્ઠતમ શેડ્યૂલ પાસે નિષ્ક્રિય સમય નથી. કારણ કે, આપણે પહેલેથી જ કહ્યું છે કે નિષ્ક્રિય સમય નકામું છે, આપણે હંમેશાં બાકી રહેલી કોઈપણ વસ્તુને પાળી શકીએ છીએ, અંતરને સંકોચો શકીએ છીએ, જેથી ત્યાં નિષ્ક્રિય સમય ન હોય. તેથી, પ્રથમ નિરીક્ષણ એ છે કે હવે આપણી પાસે કોઈ નિષ્ક્રિય સમય નથી, તેથી આ જરૂરિયાતનો એક ભાગ ગ્રહણ કરવામાં આવે છે, તેથી એકમાત્ર વસ્તુ જેને આપણે ચિંતા કરવાની જરૂર છે તે છેવટે છે. તેથી, પ્રથમ દાવા એ છે કે જો ઓમાં બદલાવ હોય તો, હકીકતમાં આપણી પાસે સતત બે ઘટકો વચ્ચેનો ભ્રમ છે, ત્યાં જોબનો એક જોડી છે અને j જેવો છે કે જે j પછી તરત જ છે, પરંતુ j ની સમય સીમા

કરતાં નાની છે હું સમયસીમા. તેથી, અમારી પાસે નાની સમયરેખા સાથે કંઈક છે જે મોટી સમયરેખા માટે કંઈક પછીથી આવે છે અને તે ખૂબ જ સ્પષ્ટ છે, કારણ કે જો કોઈ વ્યુત્ક્રમ હોય તો, સમય મર્યાદા સામાન્ય રીતે વધશે અને પછી ક્યાંક આ એક બદલાવ થશે, તેથી તે નીચે આવે છે . તેથી, તે બિંદુએ જ્યાં તે નીચે આવે છે, ત્યાં આપણી પાસે બે નજીકની વસ્તુઓ હોવી જોઈએ, જ્યાં મોટી વ્યક્તિ નાના કરતા પહેલા આવે. તેથી, જ્યારે પણ આપણે અનુક્રમમાં ગમે ત્યાં બદલાવ કરીએ છીએ, ત્યારે આપણે કોઈ મુદ્દો શોધી શકીએ છીએ જ્યાં સતત બે વસ્તુઓ એક બદલાવ ધરાવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:37)

હવે, આગળનું અવલોકન એ છે કે આપણે આ બે જોબને સ્વેપ કરીને આ વ્યુત્પત્તિ દૂર કરી શકીએ છીએ. તો, આપણી પાસે j અને i જે છે, જેનો વ્યુત્ક્રમ છે, તો જો આપણે i અને j નું વિનિમય કરીએ તો તે પહેલા હું j ને મુકું છું, તો હવે j ની d એ d કરતાં ઓછી છે અને આ વ્યુત્પત્તિ ગઈ છે. તેથી, તે સ્પષ્ટ છે કે આપણે શા માટે વ્યુત્પત્તિ દૂર કરીએ છીએ. પરંતુ, હવે તે સ્પષ્ટ છે કે આ સતત જોબને આ સ્વચાલિત રીતે દૂર કરીને આ ક્રમમાં દૂર કરવાની આ પ્રક્રિયા ખરેખર આ વિચારની ગુણવત્તાને અસર કરશે નહીં. તેથી, આપણે જે પૂછવાની જરૂર છે તે છે કે i અને j ને સ્વેપ કર્યા પછી આપણને એક ઉપાય મળે છે જેની મહત્તમ લંબાઈ O કરતાં વધુ મોટી નથી. તેથી, આપણી પાસે મહત્તમતમ ઉકેલ છે, આપણી પાસે વ્યુત્ક્રમ અને સતત અનુવર્તન છે, આપણે આ કરવા માંગીએ છીએ આ સતત પરિવર્તનને તે બે સતત તત્વોને સ્વેપ કરીને પૂર્વવત્ કરો, પરંતુ અમે શ્રેષ્ઠતાને બદલવા નથી માંગતા.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:38)

તો, આ ફરીથી આપણે આકૃતિ દ્વારા જોઈ શકીએ છીએ, તેથી યાદ રાખો કે આ અવ્યવસ્થાએ કહ્યું છે કે હું j પહેલા આવ્યો હતો, પરંતુ j ની ડી ખૂબ ઓછી હતી ત્યારબાદ હું ડી હતો તેથી તે વર્ણન હતું . તેથી, અમારી પાસે આ પ્રકારની પરિસ્થિતિ છે, તેથી અમારી પાસે આ વાદળી ઈતિહાસનો કેટલોક ઈતિહાસ હતો અને પછી આ સમયે અમારી પાસે અને પછી j. તેથી, આ મારું મૂળ છે અને હવે હું વિનિમય દ્વારા જઈને ઓ પ્રાઈમ પર જઈશ

((સમયનોસંદર્ભ: 14:08)).

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:10)

તેથી હવે અવલોકન કરો કે j ની ડી એ ધારણા દ્વારા d ની ડાબી બાજુએ છે, તેથી j ની ડી ખૂબ ઓછી છે પછી i ની ડી. તો, હવે ચાલો આપણે j ની લેટનેસ ને જોઈએ, તેથી તે અંતથી છે, હું વત્તા જે સમાન સમય લે છે, પછી હું j અથવા j પહેલા હું કરીશ. તો, જો હું j ની સમયની સીમાચિહ્નથી જુ એન વખત જ્યાં આની લંબાઈ જોઈતો હોય, તો આ લાકડા વધારે હોવી જ જોઈએ, પછી નીચે લાંબું લાંબું હોવું જોઈએ નહીં. કારણ કે હું d ની j ની ડીની જમણી તરફ જ છું અને n પોઈન્ટ સમાન છે. તેથી, જો હું એન પોઈન્ટ પર ધ્યાન આપું છું અને સમયરેખા બિંદુને બાદ કરું છું, તો હું અંતિમ મુદત માટે n બંધ છું, પછી j માટે અંતિમ મુદત છે, કારણ કે ડીજે ડી પહેલા છે. તેથી, તેથી હું અને j ને માત્ર વિનિમય દ્વારા

વ્યુત્ક્રમો પરની ગતિએ ખાતરી આપી છે કે આ નોટિસને લીધે મોટું કોઈ અન્ય જોબ નથી, આ બિંદુ સુધી દરેક અન્ય જોબ એ જ સમયે જવાબ આપે છે જ્યારે આ બિંદુ પછીની દરેક વખતે પણ તે જ મુદ્દાને સમાપ્ત કરે છે. તેથી, અન્ય કોઈ જોબ નરમ દ્વારા અસર કરેલા લેટિએશનને ફક્ત જોબ માટે જ અસર કરે છે, જે લેટનેસ changes અથવા i અને j એ આ રીતે બદલાવ દ્વારા થાય છે કે સમગ્ર લૈંગિકતા ફક્ત ત્યારે જ ઘટાડે છે, તે વધતી નથી. તેથી, આ શ્રેષ્ઠતા જાળવવાના સંદર્ભમાં સલામત નરમ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 15:34)

તેથી, હવે આપણે આપણા દાવા પર પાછા આવીએ છીએ કે કોઈ ઈનવર્ઝન અને નિષ્ક્રિય સમય વિના શ્રેષ્ઠતમ શેડ્યૂલ છે, અમે જાણીએ છીએ કે અમારી પાસે કોઈ નિષ્ક્રિય સમય ન હોવાથી મહત્તમ અનુક્રમણિકા છે, કારણ કે તે સામાન્ય સિદ્ધાંત . હવે, અગાઉના દલીલથી આપણે લેટનેસને વધ્યા વગર દરેક સતત ઈનવર્ઝનને દૂર કરી શકીએ છીએ. તેથી, શ્રેષ્ઠતા સચવાય છે, હવે જો તમે તેમાંની દરેક જોડી હુકમની બહાર ન હોય તો પણ તમે n જોબ ઉમેરો છો, તો તેની સાથે પ્રારંભ કરવા માટે ફક્ત 2 થી 2 બાદ 2 અવરોધો છે. તેથી, અમે શ્રેષ્ઠતાને પ્રભાવિત કર્યા વિના, દરેકમાં વ્યવસ્થિત ઈનવર્ટ્સ કરી શકીએ છીએ, જ્યાં સુધી અમને કોઈ વ્યુત્પન્ન અને નિષ્ક્રિય સમય ન હોય ત્યાં સુધી શ્રેષ્ઠતમ શેડ્યૂલ મળે. અને આપણે પહેલેથી જ જોયું છે કે કોઈપણ બે શેડ્યુલ્સ કોઈ બદલાવ નથી અને નિષ્ક્રિય સમય કોઈ નિષ્ક્રીયતા સમાન હોવું જોઈએ નહીં. તેથી, અમારા શેડ્યૂલ એ કે જેની પાસે મિલકત નથી કે જેમાં તેની કોઈ વ્યુત્પત્તિ નથી અને O નું ટ્રાન્સફોર્મર સંસ્કરણ તરીકે સમાન અવ્યવસ્થિતતા તરીકે કોઈ નિષ્ક્રિય સમય નથી, O ના પરિવર્તન સંસ્કરણથી તે જ લેટનેસ O પોતે જ છે અને O એ અમારા એલ્ગોરિધમનો શ્રેષ્ઠ ઉકેલ છે. પણ શ્રેષ્ઠ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:45)

તેથી, આ સમસ્યામાં યુક્તિ ખરેખર સાબિત થઈ હતી કે ગ્રીડી(Greedy) વ્યૂહરચનાઓ સાચી હતી, અમલીકરણ અને જટીલતા ગણતરી કરવા માટે ખૂબ જ સરળ છે, અમારે ફક્ત ટૂંકા સમય સુધી જ કામ ટૂંકાવી છે અને પછી આ શેડ્યૂલમાં તે જ ક્રમમાં વાંચો. તેથી, જોબને ટૂંકાવીને n લોગ n કરવું સામાન્ય છે અને આ શેડ્યૂલને વાંચવું એ ઑર્ડર એન ટાઈમ લે છે, કારણ કે આપણે ફક્ત ટૂંકા થયા પછી જ જોબ 1 ને વાંચીએ છીએ. તેથી, એકંદરે આપણી પાસે n લોગ n એલ્ગોરિધમ છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 06
મૉડ્યૂલ - 05
લેક્ચર - 43
ગ્રીડી એલ્ગોરિધમ્સ (Greedy algorithms): હફમેન કોડ્સ(Huffman Codes)

આ કોર્સમાં ગ્રીડી(Greedy) એલ્ગોરિધમના છેલ્લા ઉદાહરણ માટે, આપણે એક કોમ્યુનિકેશન થિયરીમાં એક સમસ્યા જોશું, આપણે જોશું હફમેન કોડ્સ (Huffman Codes)ની સમસ્યા પર.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:10)

તેથી, જ્યારે આપણે વાતચીત કરીએ છીએ, ત્યારે આપણે એક સ્થળેથી બીજી જગ્યાએ માહિતી પ્રસારિત કરવી પડે છે. તેથી, અમે કેટલીક ભાષામાં અંગ્રેજી, હિન્દી અથવા ગમે તે રીતે કામ કરી શકીએ છીએ, પરંતુ જો આપણા ડેટાને ટ્રાન્સમિશન કરવા માટે, ઉદાહરણ તરીકે કમ્પ્યુટર્સનો ઉપયોગ કરીએ છીએ, તો આપણે જાણીએ છીએ કે તેઓએ આ માહિતીને દ્વિસંગી શબ્દમાળાઓ માં મોકલવી આવશ્યક છે. તેથી, અમારું લાક્ષણિક લક્ષ્ય મૂળાક્ષરો લેવાનું છે અને પછી તેને 0 અને 1 ની સ્ટ્રીંગ પર એન્કોડ કર્યું છે, જેથી બીજી બાજુ, અમે સંદેશને ડીકોડ કરી અને પુનઃપ્રાપ્ત કરી શકીએ છીએ. તેથી, જો તમે 26 નીચલા કેસ અક્ષરોને z થી કહો છો, તો તે જોવાનું સરળ છે કે જો તમારે દરેક અક્ષરને 0 અને 1 ના નિયત અનુક્રમ તરીકે નિયત અનુક્રમણિકા તરીકે એન્કોડ કરવા માંગતા હોય, તો અમારે ઉપયોગ કરવાની જરૂર પડશે પત્ર માટે 5 બિટ્સ, કારણ કે જો તમે ફક્ત 4 બિટ્સનો ઉપયોગ કરો છો, તો ફક્ત 16 જુદા જુદા સંયોજનો મેળવી શકે છે, જેમાં 5 બિટ્સનો સમાવેશ થાય છે જે 32 જુદા જુદા સંયોજનો મેળવી શકે છે. તેથી, હવે એક કુદરતી પ્રશ્ન એ છે કે, આપણે વિવિધ અક્ષરો માટે વિવિધ લંબાઈ એન્કોડિંગનો ઉપયોગ કરીને હોંશિયાર કંઈક કરી શકીએ છીએ, જેથી વધુ વારંવાર અક્ષરો નાના ઈનપુટ્સ સાથે મોકલવામાં આવે. તેથી, આપણે એક સ્થાનેથી બીજા સ્થાને સંદેશ મોકલવા માટે ખરેખર કેટલી માહિતીને સ્થાનાંતરિત કરીએ છીએ તેને અમે ઑપ્ટિમાઈઝ કરી શકીએ?

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:24)

તેથી, આ આપણને વેરિયેબલ લંબાઈ એન્કોડિંગ ધરાવતી વિચાર પર લાવે છે, જ્યાં અમે મૂળાક્ષરોમાં વિવિધ અક્ષરો માટે વિવિધ લંબાઈના વિવિધ સ્ટ્રીંગ્સનો ઉપયોગ કરીએ છીએ. તેથી, વેરિયેબલ લંબાઈ એન્કોડિંગના સૌથી પ્રસિદ્ધ ઉદાહરણોમાંનો એક શાસ્ત્રીય મોર્સ કોડ છે, જેનો વિકાસ સેમ્યુઅલ મોર્સે દ્વારા ટેલિગ્રાફ દ્વારા કરવામાં આવ્યો છે જેનો શોધ કરવામાં આવી છે. તેથી, આ સંપર્ક પર ક્લિક કરીને યાંત્રિક ઉપકરણનો ઉપયોગ કરીને કરવામાં આવ્યો હતો અને તે લાંબી અને ટૂંકી ક્લિક્સ બનાવશે. તેથી, ટૂંકા ક્લિક્સને બિંદુઓ કહેવામાં આવે છે અને ડેશ્સ તરીકે ઓળખાતા લાંબા ક્લિક્સ, આપણે તેમનો તેમજ બીટ્સ 0 અને 1 નું પ્રતિનિધિત્વ કરવા વિચારી શકીએ છીએ. તેથી, મોર્સ કોડ એન્કોડિંગમાં, જુદા જુદા અક્ષરોમાં વિવિધ એન્કોડિંગ્સ હોય છે અને અંગ્રેજીમાં તે છે મોટાભાગના વારંવાર પત્ર અને ટી એક અન્ય પ્રકારનું વારંવારનું પત્ર છે. તેથી, મોર્સે તેમને ડોટના કોડ્સ આપ્યાં, જે ઈ અને ડેશ માટે 0 છે, તે ટી માટે 1 છે, પછી મોર્સે

અન્ય વારંવારના અક્ષરો લેતા હતા અને તેમને બે અક્ષર એન્કોડિંગ આપી હતી. તેથી, ડોટ ડૅશ તરીકે એન્કોડેડ છે, જ્યાં 0 અને 1. હવે, મોર્સના એન્કોડિંગ સાથે સમસ્યા એ છે કે તે અસ્પષ્ટ છે, જ્યારે તમે ડીકોડિંગમાં આવો છો. તેથી, ઉદાહરણ તરીકે, જો આપણે શબ્દ 0, 1 જુઓ, તો આપણે જાણતા નથી કે આપણે આમાંના દરેકને એક અક્ષર કોડ તરીકે અર્થઘટન કરવી જોઈએ અને ઈટીટ મેળવવું જોઈએ, ઉદાહરણ તરીકે આપણે આને 2 બે અક્ષર કોડ્સ અને aa અને so on. તેથી, આપણે તેને બંધ કરીએ કે નહીં તે 0 અથવા 0 થી 0 સુધી વિસ્તરે છે તેના આધારે, આપણે ઘણા જુદા જુદા અર્થઘટન મેળવી શકીએ છીએ. હવે, મોર્સ કોડમાં વ્યવહારમાં, આપણે જે બનવા માટે ઉપયોગ કરીએ છીએ તે એ છે કે ઑપરેટર, અક્ષરના અંતને સૂચવતી સ્લાઈડ વિરામ આપે છે. તેથી, તેથી, અસરકારક રીતે મોર્સ કોડ બાઈનરી કોડ નથી, પરંતુ તે ત્રણ અક્ષરનો કોડ 0 1 છે અને થોભો, હવે અમે અલબત્ત ડિજિટલ કમ્પ્યુટર્સનો ઉપયોગ કરી રહ્યા છીએ, અમે ત્રણ અક્ષરો પર જવા નથી માંગતા. તો, આપણે ફક્ત બે અક્ષરોનો ઉપયોગ કરીને કાર્યક્ષમ રીતે આ કરવા માંગીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:20)

તેથી, વેરિયેબલ લંબાઈ કોડને નિષ્ક્રીય ડીકોડેબલ બનાવવા માટે, આપણને પ્રીફિક્સ જથ્થા કહેવામાં આવે છે. જ્યારે આપણે 0 અને 1 ના ક્રમ દ્વારા વાંચીએ છીએ, ત્યારે આપણે અસ્પષ્ટતાપૂર્વક સ્પષ્ટ હોવું જોઈએ, પછી ભલે આપણે કોઈ અક્ષર વાંચ્યું હોય અથવા વાંચવા માટે વધુ હોય. આપણે પહેલાના કિસ્સા જેવું હોવું જોઈએ, જ્યાં આપણે 0 વાંચ્યું છે અને આપણે જાણીએ છીએ કે, આપણે 0 પર રોકાઈએ છીએ અને તેને મોર્સ કોડ સેટિંગમાં ઈ કહીએ છીએ અથવા આપણે તેને 0 જે છે તે કહીએ છીએ. 1. તેથી, આપણે ફંક્શન એન્કોડિંગને સૂચવવા માટે આ મૂડી પત્ર E નો ઉપયોગ કરવા જઈ રહ્યાં છો. તેથી, x ની E એ અક્ષરની એન્કોડિંગ છે. તેથી, અહીં એક ઉદાહરણ છે, તેથી આપણી પાસે પાંચ અક્ષરો a, b, c, d, e અને now, તમે ચકાસી શકો છો કે આમાંની કોઈપણ એન્કોડિંગને કંઈપણ પર વિસ્તૃત કરી શકાશે નહીં. મારી પાસે નથી 1, જો હું 1 જોઉં, તો તે હેડર હોવું જોઈએ, બીજું કોઈ કોડ જે 1 થી શરુ થાય. 1 જો હું 0 જોઉં તો તે કોડ નથી, પરંતુ 0 0 એ કોડ છે, તેથી 0 1 હોઈ શકતું નથી વિસ્તૃત અને તેથી. તેથી, આમાંના દરેકને અન્ય કોઈપણ અક્ષરનો કોડ ગણાવી શકાય નહીં. તેથી, હવે જ્યારે આપણે આ પ્રમાણે અનુક્રમે આવીશું, ત્યાં કોઈ શંકા નથી, જ્યારે હું પત્ર પૂર્ણ કરું ત્યારે પ્રથમ બિંદુ 0 0 છે અને આ એક સી છે. આગલું બિંદુ જ્યારે હું અક્ષર પૂર્ણ કરું છું તે ત્રણ 0 છે અને તે એક છે, તો પછી હું બીજું c વાંચું છું અને પછી એક અને પછી બી. તેથી, જો તમારી પાસે પ્રીફિક્સ કોડ પ્રોપર્ટી હોય, તો તે કોઈ શબ્દ નથી જે શબ્દમાળામાં એન્કોડેડ છે જે એન્કોડિંગ અથવા અન્ય કોઈ અક્ષરનો ઉપસર્ગ છે, પછી આપણી પાસે અસમર્થ ડીકોડિંગ શક્ય છે અને તે ખૂબ જ મહત્વપૂર્ણ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:50)

તેથી, અમારું લક્ષ્ય શ્રેષ્ઠ ઉપસર્ગ કોડ્સ શોધવાનું છે. તેથી, અમને શ્રેષ્ઠતા દ્વારા જેનો અર્થ છે તે વિશે વાત કરવાની જરૂર છે. તેથી, યાદ રાખો કે અમે કહ્યું છે કે અમારું લક્ષ્ય ટૂંકા કોડ્સને વધુ વારંવાર અક્ષરો પર અસાઈન કરવું છે. તેથી, કોઈક રીતે આપણે નિર્ધારિત કરવું પડશે, વધુ વારંવાર અને ઓછા વારંવાર અક્ષરો શું છે? તેથી, લોકોએ દરેક અક્ષર અને વિવિધ ભાષાઓની ઘટનાની આવર્તનને માપ્યું છે, તેથી આ ખૂબ જ વિશિષ્ટ ભાષા છે. તેથી, આ શ્રેષ્ઠતા એ કંઈક છે જે અંગ્રેજી માટે અનુકૂળ છે, ફ્રેન્ચ અથવા અન્ય કોઈ સ્પેનિશ અથવા કંઈક કામ કરી શકશે નહીં. તેથી, તમે કોઈ ચોક્કસ ભાષામાં ટેક્સ્ટનો મોટો ભાગ લે છે અને તમે A ની b c d અને e ની સંખ્યાને ગણાવી શકો છો અને પછી તમે બધા પગલાઓ

પરના અક્ષરોની કુલ સંખ્યામાંથી માત્ર અપૂર્ણાંકને જુઓ, કેટલા છે ઈ, કેટલા બી, કેટલા સી છે. તેથી, આ આની સરેરાશ આવર્તનનો આંકડાકીય અંદાજ છે. તેથી, આ આવર્તન એક અપૂર્ણાંક હશે, મોટાભાગના પાઠ્ય લખાણ પર અક્ષરનો કેટલો ભાગ હશે, તે કેટલો અંશે સી હશે અને આ અપૂર્ણાંક એક સાથે ઉમેરાશે, કારણ કે દરેક અક્ષર તેમાંથી એક હશે. તેથી, એક અંશે પ્લસનો ભાગ છે, b નો અપૂર્ણાંક વત્તા, c નો ભાગ છે અને તેથી તે 1 થી ઉમેરવામાં આવશે અને તેના કારણે, આપણે આ આંકડાકીય અંદાજ અથવા સંભાવનાની સંભાવના પણ વિચારી શકીએ છીએ. ચાલો જો હું તમને રેન્ડમ લેટર આપીશ, જો હું ટેક્સ્ટનો ભાગ જોઉં છું અને ટેક્સ્ટમાંથી રેન્ડમ લેટર પસંદ કરું છું, તો એક્સની સંભાવના શું છે, તે x ની તમામ ટેક્સ્ટ f માં માત્ર x ની આવર્તન છે. તેથી, આ 1 માં ઉમેરવામાં આવશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:16)

તો, હવે, આપણી પાસે એક સંદેશ છે, તેમાં કેટલાક એન ચિન્હો છે. તેથી, આપણી પાસે એમ 1, એમ 2 અપ ટુ એમ, તેથી આ એન ચિન્હો છે. હવે, આપણે જાણીએ છીએ કે જો હું કોઈ ચોક્કસ અક્ષર x લઈશ, તો આનું એફએક્સ અપૂર્ણાંક x છે, તેથી બીજા શબ્દોમાં કહીએ તો, જો હું n લે અને હું અપૂર્ણાંક દ્વારા ગુણાકાર કરું, તો જો હું ઠીક કરું તો એક તૃતીયાંશ, પછી ત્રીજો ના. તેથી, આ 3 પ્રતીકો દ્વારા n એ અક્ષર x અને હવે હશે, આ x માં પ્રત્યેક x ને રજૂ કરીને તે એન્કોડિંગ છે. તેથી, તે 0 0 0 હોવાનું માનવું પછી પ્રત્યેક x 3 બિટ્સ દ્વારા રજૂ થવાનું છે, તો પછી fx માં n એ fcx ની સંખ્યા છે અને આ એન્કોડિંગની લંબાઈમાં મને લેવાયેલા બિટ્સની કુલ સંખ્યા આપવામાં આવશે. આ સંદેશમાં બધા x ને એન્કોડ કરો. હવે, જો હું દરેક અક્ષર માટે આ કરું છું, તેથી જો હું એન વાયના દરેક મૂળાક્ષરમાં દરેક વાય અથવા દરેક x પર સારાંશ લે તો અક્ષરની આવર્તન તે અક્ષરની એન્કોડિંગ લંબાઈને ગુણશે. તેથી, આ મને કહે છે કે મને તે ચોક્કસ અક્ષરને એન્કોડ કરવાની કેટલી બિટ્સની જરૂર છે, બધા અક્ષરો ઉમેરો, મને એન્કોડેડ સંદેશની કુલ લંબાઈ મળે છે. અને જો હું આ n નો સમાવેશ કરતો નથી, તો તે કહેવાનું નથી કે તે સારાંશનો ભાગ નથી, તે એક સ્વતંત્ર વસ્તુ છે, તે કોઈપણ n નો અપૂર્ણાંક છે. જો હું એન્કોડિંગની બે લિંક્સની કુલ વેઇટ્ડ એવરેજ જોઉં છું, તો આ છે જો તમે સંભાવના સિદ્ધાંતનો અભ્યાસ કરો છો, તો એન્કોડિંગની અપેક્ષિત લંબાઈ કહેવામાં આવે છે. તેથી, આ બિટ્સની સરેરાશ સંખ્યા છે, હું એક અક્ષર માટે ઉપયોગ કરું છું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:06)

તેથી, ચાલો આપણે કેવી રીતે આ કાર્ય કરીએ, તેથી ધારો કે અમે અમારા 5 અક્ષરોની અગાઉની ઉદાહરણ લઈએ છીએ, હવે આપણે ફ્રીક્વન્સીઝ વિશે કેટલીક કાલ્પનિક માહિતી શામેલ કરીએ છીએ. તેથી, આ બધા પાંચ મૂલ્યો 0 અને 1 ની વચ્ચેના અપૂર્ણાંક છે, તમે 1 માં ઉમેરી શકો છો. પછી જો હું x ની x ની લંબાઈ x ની લંબાઈ x ની લંબાઈ પર કરું, તો મારી પાસે 0.32 ગુણ્યા 2 છે, કારણ કે એન્કોડિંગ એ એ બે અક્ષરો છે, પછી મારી પાસે 0.25 ગુણ્યા 2 છે, કારણ કે એન્કોડિંગ ઈ એ બે અક્ષરો છે અને બીજું. તેથી, મારી પાસે આ પાંચ શબ્દો a, b, c, d, e છે અને પછી મેં તેને ઉમેર્યું અને મને 2.25 મળે છે, તેથી તે કહે છે કે મને પ્રત્યેક પત્ર સરેરાશ 2.25 બિટ્સની જરૂર છે. અલબત્ત, હું પત્ર દીઠ 2.25 બિટ્સનો ઉપયોગ કરતો નથી, પરંતુ ઉદાહરણ તરીકે જે કહે છે, જો મારી પાસે 100 અક્ષરો છે, તો હું આઉટપુટ એન્કોડિંગમાં 225 બીટ્સ જોવાની અપેક્ષા રાખું છું. હવે, એક ખૂબ જ વિશિષ્ટ પ્રકારનો ઉપસર્ગ કોડ એ નિયત લંબાઈ કોડ છે, જ્યાં માત્ર તે હકીકત દ્વારા કે દરેક નિયત લંબાઈ પર કોડ છે, હું જાણું છું કે દરેક જ ક્યાં છે. તેથી, માની લો કે હું આ

કિસ્સામાં 3 બિટ્સનો ઉપયોગ કરું છું, જો મારે આનો લંબાઈનો કોડ ઠીક કરવો હોય તો, પાંચ અક્ષરો છે, હું તેને 2 બિટ્સથી કરી શકતો નથી, કારણ કે મને માત્ર ચાર જુદા જુદા સંયોજનો મળે છે. તેથી, મને 3 બિટ્સની જરૂર છે, જો હું 3 બીટ કોડનો ઉપયોગ કરું છું, તો દરેક 3 બિટ્સ એક અક્ષર હશે. તેથી, આમાં નિયત લંબાઈ એન્કોડિંગમાં, હું અક્ષરો માટે 3 બિટ્સનો ઉપયોગ કરીશ, તેથી તેથી સ્પષ્ટપણે અક્ષરો દીઠ બિટ્સની સંખ્યા 3 છે, કારણ કે હું તેમાંના દરેક માટે 3 નો ઉપયોગ કરી રહ્યો છું. અને તેથી વેરિયેબલ લંબાઈ કોડ એન્કોડિંગ પર જઈને જે ફ્રીક્વન્સી પર લે છે અને ખરેખર બચત તે 100 અક્ષરો માટે 300 બિટ્સ મોકલવા માટે છે, તે 225 માં મોકલે છે. તેથી, અમારી પાસે 25 ટકા બચત છે, તેથી આ તે છે જે આપણે પર વિચાર કરવાનો પ્રયાસ કરો. તેથી, આ ઉદાહરણમાં પહેલાની વસ્તુમાં આપણી પાસે બે ફ્રીક્વન્સીઝ 0.2 અને 0.18 છે, 0.18 એટલે કે d એ c કરતા ઓછી આવર્તન છે, પરંતુ કોઈક રીતે આપણે c ને લાંબી કોડ તરીકે સોંપી દીધી છે. તેથી, આ અમારા મૂળભૂત સિદ્ધાંતનું ઉલ્લંઘન કરે છે કે ટૂંકા કોડને વધુ વારંવાર અક્ષરોમાં સોંપવામાં આવે છે. તેથી, જો તમે અક્ષરોની જોડી જોશો કે જ્યાં એક બીજા કરતાં વધુ વારંવાર હોય, તો મને ટૂંકા કોડ મળવાની આશા છે અને હું આમ કરું છું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:18)

તેથી, જો હું તેને રદ કરીશ, તો હું ધારું છું કે હવે હું ડીને ત્રણ અક્ષરોનો કોડ અને બે અક્ષર કોડ સીને સોંપીશ, પછી આ બે શરતો અન્ય શરતોને બદલે છે, તેમ છતાં એન્કોડિંગમાં ભિન્નતા હોઈ શકે છે લંબાઈ બદલાતી નથી. તેથી, પછી મને 2.25 ની જગ્યાએ મળે છે, હું 2.23 પર પહોંચ્યું છું, તેથી આનો અર્થ એ છે કે જુદા જુદા એન્કોડિંગ્સ જોઈને મને અલગ એબીએલ મૂલ્યો મળી શકે છે, આ સરેરાશ બિટ્સ પ્રતિ અક્ષર. તેથી, હવે, અમારું ધ્યેય એ અસાઈનમેન્ટ કેપિટલ ઈ શોધવાનું છે જે આ જથ્થાને ઘટાડે છે. તેથી, અમારા કોડિંગમાં સરેરાશ કાર્યક્ષમ શક્ય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:56)

તેથી, આને મેળવવા માટે, આ એન્કોડિંગમાં બાઈનરી વૃક્ષોનો વિચાર કરવો ઉપયોગી છે, તેથી દ્વિસંગી વૃક્ષમાં હું 0 અને 1 ની દિશાઓને અર્થઘટન કરી શકું છું, તેથી સામાન્ય રીતે ડાબી બાજુ 0 છે અને જમણી બાજુ છે 1. તેથી, હવે, જો હું બાઈનરી વૃક્ષમાં પાથ વાંચું છું, તો તે દ્વિસંગી ક્રમ પણ હશે. તો, આ પાથ 1 1 1 બરાબર છે અને ઉદાહરણ તરીકે આ પાથ પ્રોગ્રામ 0 છે અને આ પાથ 0 0 0 છે. હવે, જો હું પાથ વાંચીશ અને પછી તે લેબલનો અક્ષર શોધીશ, તો તે તે પાથ દ્વારા લખેલું પાથ લેબલ કહેવાનું સારું છે. તેથી, અહીં છે કારણ કે ઈ પાસે બાઈનરી વૃક્ષમાં એન્કોડિંગ 0 0 0 છે, હું પાથ 0 0 નું અનુસરણ કરીશ અને તે જ અક્ષરને લેબલ કરીશ જે ઈક્વિટી દ્વારા ઈ. હવે, કારણ કે તે એક ઉપસર્ગ કોડ છે, જો 0 0 લેબલ્સ ઈ છે, તો તે કોઈ કોડ નથી જે 0 0 0 કહેશે અને બીજું કંઈક, તે નીચે અન્ય લેબલ હશે નહીં. તેથી, આ લેબલો અન્ય લેબલ સુધી વિસ્તૃત થશે નહીં, મને નીચે ડી f મળશે નહીં, કારણ કે તે ઉપસર્ગ કોડ નથી, કારણ કે હું 1 0 કરું છું અને હું આગળ વધું છું, પરંતુ હું 1 0 ને કંઈક બીજું કરું છું, તો મને એક એફ મળે છે. . તેથી, f માટેનો કોડ ડી માટે કોડ વિસ્તરે છે, તે પૂરતું નથી. તેથી, ઉપસર્ગ કોડમાં આ થઈ શકતું નથી, તેથી, તમામ લેબલ્સ ખરેખર લીફ ગાંઠો પર હોય છે, ત્યાં તે નોડ્સ છે જે વધુ સફળતાઓ ધરાવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:23)

તેથી, અહીં બીજી યોજના માટે એક એન્કોડિંગ છે જે આપણી પાસે છે, જ્યાં આપણે મૂલ્યો વિનિમય કર્યા છે c અને d. તેથી, હવે C પાસે બે અક્ષર કોડ છે અને ડી પાસે ત્રણ અક્ષર કોડ છે, જે હવે ઊંડાણ દ્વારા સૂચવે છે, ઊંડાઈ આ પાથ છે, પાથની લંબાઈ નોડની ઊંડાઈ છે. તેથી, સીમાં પહેલાની વસ્તુ ઊંડાઈ 3 હતી, અને હવે તે ઊંડાઈ 2 છે અને ડી ઊંડાઈ 2 હતી અને હવે તે ઊંડાઈ 3 છે. તેથી, આપણે મૂળભૂત રીતે વસ્તુને મુકવા માંગીએ છીએ જેમાં ઉચ્ચ ફ્રીક્વન્સીઝ હોય છે કરતાં ઓછું વૃક્ષ

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:59)

તેથી, એન્કોડેડ દેખાવને દ્વિસંગી વૃક્ષને એન્કોડિંગ કરવું છે, હવે આપણે થોડા અવલોકન કરીશું, તળિયેના ત્રણ નિરીક્ષણ, જે ઉપયોગી હશે અલ્ગોરિધમનો વિકાસ કરવા અને તે શ્રેષ્ઠ છે તે સાબિત કરે છે. તેથી, પહેલી વસ્તુ એ છે કે આ પ્રકારના વૃક્ષમાં, જો તે શ્રેષ્ઠ હોય, તો દરેક નોડમાં ક્યાં તો કોઈ બાળક હશે નહીં અથવા તે બે બાળકો હશે. તેથી, આ આપણે પૂર્ણ કહેવાય છે. તેથી, દરેક શ્રેષ્ઠ ઝાડ ભરેલું છે, હવે તેને જોવાનું સરળ છે, કારણ કે દાવાને ધ્યાનમાં રાખીને, અમે અન્ય શ્રેષ્ઠ વૃક્ષ કે જેમાં ક્યાંક વચ્ચે, એક નોડ હતો જેનો એક જ બાળક હતો. પછી, આ બાળક અસરકારક રીતે પ્રમોટ કરવામાં આવશે, અમે આ નોડને સંપૂર્ણપણે દૂર કરી શકીએ છીએ અને અમે આ દિશામાં વૃક્ષને સ્ટ્રિંગ કરી શકીએ છીએ, કંઈ પણ બદલાશે નહીં, સિવાય કે નોડની ઊંડાઈ ઓછી થઈ જાય. તેથી, વાસ્તવમાં, આપણે સંભવતઃ ટૂંકા સરેરાશ એવરેજ બીટની લંબાઈ મેળવીશું, તેથી, સિંગલટન હોવાને લીધે, ફક્ત એક ડાબું બાળક અથવા જમણો બાળક હોઈએ, અમે શ્રેષ્ઠ ન હોઈ શકીએ. તેથી, દરેક નોડને 0 અથવા બે બાળકો હોવા જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:11)

આગળની પ્રોપર્ટી એ બરાબર છે જે આપણે પહેલાની વસ્તુ જોયું, જે છે, જો મારી પાસે બે ગાંઠો x અને y છે, જેમ કે, x વાય કરતા વધારે છે, તો x એ કેટલાક સ્તરે છે અને વાય અલગ સ્તર છે. પછી, x પાસે ટૂંકા એન્કોડિંગ છે, પછી y નો અર્થ એ છે કે x નું f ઓછામાં ઓછું વાય છે, બીજા શબ્દોમાં, જ્યારે હું વૃક્ષ નીચે જાઉં છું ત્યારે મારી આવર્તન ઈન્ક્રીથી કરો, કારણ કે જો વાય વાય x ની તુલનામાં મોટું હશે, તો જો હું વધારે જ્ઞાત હોત તો x એ mentax છે. પછી, હું ફક્ત 2 નું વિનિમય કરીશ, હું અહીં y અને x ને ઉમેરીને વધુ સારું એન્કોડિંગ મેળવીશ. કારણ કે, હવે જો હું y ની લંબાઈ વાય અને આ વૃક્ષની ઊંડાઈ વાય કરીશ, તો તે ઘટાડે છે, કારણ કે y ની ઊંડાઈ ઓછી થાય છે અને આ વધારે ગુણાકાર છે. તેથી, જો મારી પાસે નીચે ઊંચી વસ્તુ હોય, તો હું પત્ર બદલી શકું અને વધુ સારુ વૃક્ષ મેળવી શકું અને તે પછી શ્રેષ્ઠ વૃક્ષમાં ન થાય, તો પછી શ્રેષ્ઠ વૃક્ષ, જો હું વૃક્ષ નીચે જાઉં તો, મને માત્ર ઓછી આવર્તન મળશે અક્ષરો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:09)

અંતિમ મિલકત પાંદડાઓથી નીચલા સ્તર પર છે, તેથી મને લાગે છે કે મારા પાંદડા નીચલા પાંદડા પર છે, તેથી આ તળાવની સૌથી નીચી સપાટી છે. તેથી, હું જાણું છું કારણ કે તે એક શ્રેષ્ઠ વૃક્ષ છે, તે એક અલગ બાળક હોઈ શકતું નથી, તેના માટે એક ભાઈને જવું અને નીચે આવવું આવશ્યક છે, તે એક ભાઈ હોવા જ જોઈએ. હવે, ત્યાં બે શક્યતાઓ છે, આ બે શક્યતાઓ એ એક પૂર્ણ છે અથવા બીજી શક્યતા એ છે કે, તે પાંદડા નથી, દાવો કરે છે કે જો તે પાંદડા નથી, તો તે

બાળકો હોવી જ જોઈએ. તેથી, અહીં પાંદડાઓ છે જે નીચલા સ્તર પર છે, પછી x, પરંતુ x એ મહત્તમ ઊંડાઈ ડી હોવાનું માનવામાં આવે છે. તેથી, મહત્તમ ઊંડાઈ પાંદડા ભાઈ-બહેન ન હોઈ શકે, જે પાંદડા નથી, કારણ કે તે ભાઈ-બહેનોને બાળકો હશે, જે ઉચ્ચ ઊંડાઈ ધરાવશે. તેથી, જો મારી પાસે મારા શ્રેષ્ઠ વૃક્ષમાં મહત્તમ ઊંડાઈ પર્ણ હોય, તો અમારે એક જોડી બીજા મહત્તમ ઊંડાઈવાળા પાંદડા સાથે આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:19)

અને તેથી આ એક નિષ્કર્ષ છે કે જોડીમાં મહત્તમ ઊંડાઈના પાંદડા જોવા મળે છે અને પછી આપણે જાણીએ છીએ કે, કારણ કે આપણે ઊંડાણમાં વધતા જતા ફ્રીક્વન્સીઝ ઘટતા જતા હોવાથી, આ જોડીમાં સૌથી નીચી આવર્તન હોવી આવશ્યક છે સૌથી નીચો ફ્રીક્વન્સીઝ. તેથી, ઉકેલને વિકસાવવા માટે, આપણે રિકર્ઝનનો ઉપયોગ કરીશું, તેથી આપણે શું કરીશું, આપણે કહીશું, ચાલો આપણે એકંદર કોષ્ટકમાં જોઈએ જે આપણે શરૂ કરીએ અને બે અક્ષરો પસંદ કરીએ, જેમાં સૌથી નીચો આવર્તન છે. તેથી, અમે તેમને સૌથી લાંબી કોડ અસાઈન કરી શકીએ છીએ, તેથી તેમને સૌથી નીચલા સ્તર પર મૂકી શકાય છે અને પછી આપણે જાણીએ છીએ કે જોડીમાં સૌથી નીચલા સ્તરની પાંદડાઓ. તેથી, ચાલો ધારીએ કે આ જોડી બનાવશે, તેથી અમે આ સૌથી નીચલા આવર્તન અક્ષરો x અને y ને, પાંદડાઓની જોડીમાં મહત્તમ ઊંડાઈ, ડાબે અને જમણે વાંધો નહીં આપીએ, કારણ કે તે ઊંડાઈ જે મહત્વપૂર્ણ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 17:10)

તેથી, હવે, એક પુનરાવર્તિત ઉકેલ કહેશે કે, બાકીનું વૃક્ષ જેવો દેખાય છે તે કેવી રીતે લાગે છે, જો મારી પાસે કોઈ પરિસ્થિતિ હોય તો, જ્યાં મેં એક્સ અને વાય બંને નક્કી કર્યા છે અહીં પછી, હું વૃક્ષનો પ્રકાર કરીશ, આ એકમ છે અને એક નવું અક્ષર કોલ x, y બનાવે છે અને તેને સંચયિત આવર્તન અસરો વત્તા x, y ને જૂના બે અક્ષરોનો આપે છે. તેથી, નવું મૂળાક્ષર બનાવો અને જે હું એક્સ અને વાય છોડું છું અને હું હાઈબ્રિડ અક્ષર x, y નું નવું સંયોજન ઉમેરું છું. જેની ફ્રીક્વન્સીઝ એફ્એક્સ પ્લસ એફ્ વાય હશે. હવે, રિકર્ઝન ફિક્શન, મારી પાસે એક માર્કસ 1 અક્ષર મૂળાક્ષરો છે, તેથી મેં વારંવાર શોધ્યું છે અને તેના માટે શ્રેષ્ઠ એન્કોડિંગ કર્યું છે. હવે, ઉકેલને કેવી રીતે અનુકૂલિત કરવું તે પહેલાં, પુનરાવર્તિત અંત માત્ર ત્યારે જ બે અક્ષરો ધરાવે છે, કારણ કે બે શ્રેષ્ઠ ઉકેલ એ વૃક્ષ બનાવવાનું છે જે બરાબર બે પાંદડા, લેબલ 0 અને 1 પાથ પર છે. તો, આ મૂળભૂત કેસ છે, જો મારી પાસે બે કરતા વધારે અક્ષરો છે, તો હું નાના કદમાં વૃક્ષને ફરીથી બનાવશે અને પછી હું પાછો આવીશ, જે વૃક્ષ મેં બનાવ્યું છે તેમા મારી પાસે પર્ણનું લેબલ x વાય હશે. હવે, xy એ અક્ષર નથી, તેથી હું શું કરું છું, હું આ બદલીશ, x અને y નામની બે નવી પર્ણ લખીશ. તેથી, હું વૃક્ષોથી એ એ પ્રાઈમ પર એ કાર્યો દ્વારા વૃક્ષ પર જઈશ. તેથી, આ એલ્ગોરિધમ છે જે હફ્મેનને વિકસિત કરે છે અને આ પ્રકારના કોડિંગને હફ્મેન કોડિંગ કહેવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 18:36)

તેથી, ચાલો આપણે આ ઉદાહરણ જોઈએ જે આપણે પહેલા કર્યું હતું, તેથી અહીં બે સૌથી આવર્તન અક્ષરો d અને e છે. તેથી, આપણે તેમને નવા અક્ષર d, e માં મર્જ કરીએ છીએ અને આ એક આવર્તન 0.23 છે, કારણ કે તે 0.18 વત્તા 0.05 છે. હવે, આ બે બે નીચલા અક્ષરો છે, તેથી આપણે તેમને મર્જ કરીએ છીએ અને આપણને એક નવી અક્ષર સી, ડી, સંચયી

આવર્તન 0.43 ની મળે છે, જે બધી ફ્રીક્વન્સીઝ બે મૂલ્યો છે. હવે, તે શબ્દ છે, આ બે નાના છે. તેથી, હું તેમને એક અક્ષર છે, બી અને હવે, હું મારા બેઝ કેસને તોડુ છું જ્યાં બરાબર બે અક્ષરો છે. તેથી, હું આ બે અક્ષરોના નાના વૃક્ષને 0 અને 1 લેબલ સેટ કરી શકું છું. અને હવે હું પાછળથી કામ કરું છું, તેથી મેં જે છેલ્લું કર્યું તે એ અને બીને મર્જ કરવું હતું, હવે હું એ અને બી વસ્તુને અલગ કરીશ અને તેને વિભાજીત કરીશ. એક એ અને બી છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 19:30)

હું A અને B ને A અને B તરીકે વિભાજીત કરીશ, મને આ પ્રિન્ટ મળશે, તો પહેલાનો પગથિયું સીને જોડવાનો હતો, ડી અને સી માં ડી, ઈ. તેથી, હું આ સી અને ડી વિભાજિત કરવા જઈ રહ્યો છું, ઈ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:40)

અને અંતે, હું આને વિભાજીત કરવા જઈ રહ્યો છુ અને ડી છે. તેથી, આ હફમેનનું એલ્ગોરિધમ છે અને બે વાર ન્યૂનતમ ફ્રીક્વન્સી ગાંઠોનું મિશ્રણ કરીને, અને ત્યારબાદ સંયુક્ત નોડ લે છે અને તેને પાછળથી વિભાજિત કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:56)

તેથી, બતાવવા માટે કે આ અલ્ગોરિધમ શ્રેષ્ઠ છે, અમે અલ્ગોરિધમનો કદના અંત સુધી જઈએ છીએ. હવે, સ્પષ્ટપણે જ્યારે મારી પાસે માત્ર બે અક્ષરો છે, હું તેમને 0 અને તેમાંથી એકને અસાઈન કરતાં વધુ સારી રીતે કરી શકતો નથી, તેથી બેઝ કેસ શ્રેષ્ઠ છે. તેથી, આપણે ધારીશું કે આ કે ઓછા 1 અક્ષરો માટે શ્રેષ્ઠ છે અને હવે તે બતાવે છે કે કે અક્ષરો માટે પણ શ્રેષ્ઠ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 20:17)

તેથી, યાદ રાખો કે, આપણે બે ઓછા આવર્તન અક્ષરોને 1 જેટલા સંયોજિત કરીને, આ નાના મૂળાક્ષર માટે એક શ્રેષ્ઠ વૃક્ષ બનાવતા અને પછી x ને વેતન આપીએ તો, ન્યુન્યુસ 1 માં ગયા. તેથી, દાવા એ હતો કે જ્યારે હું કે અક્ષરોથી વૃક્ષ પર વૃક્ષથી કે ઓછા માપદંડો પર વૃક્ષથી જાઉં, ત્યારે ખર્ચ વધશે, પરંતુ આ કોન્ટ્રેક્ટ માટે હું પસંદ કરું છું તે કોઈપણ અક્ષર દ્વારા ઠીક કરવામાં આવશે. તેથી, જો હું ટી અને ટી પ્રાઈમમાંથી જવા માટે મર્જ કરવા માટે એક્સ અને વાય પસંદ કરું છું, તો જે રકમ દ્વારા પ્રત્યેક અક્ષર દીઠ સરેરાશ બિટ્સ બદલાવવાનું છે તે બરાબર આ સંયુક્ત અક્ષર fx વાય નું આવર્તન છે. તેથી, હું જે પસંદ કરું છું તે નિર્ધારિત રીતે નક્કી કરું છું, પછી ભલે મને વૃક્ષોનો ખર્ચ સીધો જ ખબર હોય, પણ હું તમને કહી શકું છું કે આ પછી કેટલાક ખર્ચ અલગ હશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 21:08)

આ સાબિત કરવું મુશ્કેલ નથી, તેથી તે સારાંશમાં આપણે હતા, તેથી વૃક્ષની નોંધમાં યાદ રાખીએ કે z ની આ ઊંડાઈ સેટના એન્કોડિંગની લંબાઈ જેટલી જ છે, કારણ કે ઊંડાઈ એન્કોડ કરવામાં આવતી સ્ટ્રીંગની લંબાઈ બરાબર એક રીફ્લેક્સ. તેથી,

આ અમારી એબીએલ ગણતરી છે. હવે, xy અને xy સિવાયના દરેક નોડ માટે, ટી અને ટી પ્રાઈમ્સમાં સમાન સ્થિતિ બરાબર છે. તેથી, આ શબ્દો સાથે આ સારાંશ સંમિશ્રાણ બદલાતું નથી, તેથી આ ત્રણ નોડ્સ માત્ર એક જ ફેરફાર છે. તો, હું શું કરીશ અને ટી પ્રાઈમ ટુ ટી પર જઈશ, હું આ સંક્ષિપ્ત અક્ષર x, y નો આ ફાળો દૂર કરીશ. અને પછી આગલા સ્તર પર જે xy ની 1 વત્તા ઊંડાઈ છે, હું નોડ એફએક્સ ઉમેરીશ અને હું નોડ એફએક્સ ઉમેરીશ અને હું ઘટકો xy, fx times જે વત્તા xy વખત મળશે, હું આ રકમને બાદ કરીશ. બાકી, પછી હું આ રકમ જમણી બાજુ ઉમેરી રહ્યો છું. તેથી, હવે, વાસ્તવમાં xy નું f એ કંઈપણ નથી પરંતુ એફએક્સ વત્તા એફ વાય છે. તેથી, આ ડાબી બાજુની બાજુ વાસ્તવમાં આ જમણી બાજુ xy ની ઘટક ઊંડાઈ છે, x વત્તા વાય ના સમય છે. તેથી, હું આને બાદ કરી રહ્યો છું અને તેને પાછું ઉમેરી રહ્યો છું, તેથી એકબીજાને રદ્દ કરો, તેથી, બધુ જ બાકી રહ્યું છે એક વખત તે x વત્તા વાય જે fy અથવા fx વાય દ્વારા fx છે. તેથી, તેથી હું ટી પ્રાઈમ થી ટી જાઉં છું, નિર્ણાયક વસ્તુ ફક્ત મારા માટે કયા અક્ષરોનો કરાર છે તેના ઉપર આધાર રાખે છે, તે ફિક્સ એ અનન્ય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 21:51)

હવે, ચાલો ધારીએ કે, આપણે બધા કે ઓછા 1 ને કેવી રીતે ઉકેલવું તે જાણીએ છીએ કે મૂળાક્ષરોને અસરકારક રીતે કહો અને આપણે માપ કર્યો છે કે કદ કદ માટે વૃક્ષનું નિર્માણ કરવા. ધારો કે, આ બીજી વ્યૂહરચના કદ કદ માટે વધુ સારા વૃક્ષનું ઉત્પાદન કરશે, તેથી આ અન્ય વૃક્ષના ઉમેદવાર વૃક્ષની કેટલીક અલગ વ્યૂહરચના દ્વારા ઉત્પન્ન થાય છે, જે પત્ર માટે સરેરાશ બિટ્સ એ એક છે જે આપણે પુનરાવર્તિત બનાવતા કરતા વધુ કડક છે. હવે, એસમાં, આપણે ખાતરીપૂર્વક જાણીએ છીએ કે આ બે અક્ષરો કે જે આપણે પુનરાવર્તિત બાંધકામ x અને y માં ઉપયોગ કરીએ છીએ. તેથી, તેઓ ક્યાંક આ વૃક્ષના તળિયે આવે છે. તેથી, આ મારું વૃક્ષ એસ છે, આ નોડ્સ છોડવા જ જોઈએ, કારણ કે તેમની પાસે સૌથી ઓછા આવર્તન છે, તે બધા અક્ષરો ઉપર છે, તેથી મહત્તમ ઊંડાઈ હોવી જોઈએ, જેથી તેઓ એક બીજાની નજીક નહીં હોય. પરંતુ, તે કોઈ વાંધો નથી, કારણ કે તેઓ બંને અને મહત્તમ ઊંડાઈ છે, તેથી હું તેમને આજુબાજુ ખસેડી શકું છું, આ પગલું, હું આસપાસ અક્ષરો ખસેડી શકું છું, હું પ્રેષકને બે અન્ય પર્ણ ફરીથી સોંપવા કરી શકું છું. આ રીતે, હું ગોઠવણી સાથે આવીશ જે હું નવા એસ પર આવીશ, હું તેને ફરી એસ કહીશ, જ્યાં વાસ્તવમાં x અને y એકસાથે હોય. હું ધારું છું કે એસ, જેની પાસે આ શ્રેષ્ઠ ગુણધર્મ છે, જે વૃક્ષમાં વધુ સારી છે, વાસ્તવમાં x અને ya ભાઈ અન્ય મહત્તમ ઊંડાઈને છોડે છે. હવે, હું જે કરવા જઈ રહ્યો છું તે આ એસ છે, હું કોંક્રિટલી રીતે આને ફ્યૂઝ કરીશ અને એસ પ્રાઈમ મેળવીશ. તેથી, આ સમજૂતી આપણી કે બાદબાકી 1 અક્ષરો હશે સિવાય કે તે ફરીથી આવતી કોલ દ્વારા કરવાને બદલે, હું વાસ્તવમાં કે અક્ષર પર કૉંક્રિટ ટ્રી કૉલ કરી રહ્યો છું અને હું વાસ્તવમાં બે નોડ્સમાં 1 માં સંકોચાઈ રહ્યો છું અને તેના ઉપર વૃક્ષ પર કૉલ કરું છું. બાદબાકી 1 ડેટા સમૂહ. પરંતુ, કારણ કે તે કે ઓછા 1 અક્ષરોથી વધુ છે અને આ એન્કોડિંગનું પ્રતિનિધિત્વ કરે છે, તે એ કેનિંગ કરતા વધુ સારું ન હોઈ શકે કે જે મેં કે બાદબાકી 1 અક્ષરો માટે પુનરાવર્તન કર્યું છે, કારણ કે મને તે અલ્ગોરિધમનો સમાવેશ કરીને ગ્રહણ કરવામાં આવે છે જે કે ઓછા 1 અક્ષરો માટે કાર્યક્ષમ છે . તેથી, તેથી એસ પ્રાઈમ ટી સમય કરતાં વધુ સારી નથી, પરંતુ xy નો પ્રાઈમ પ્લસ એફ એ એસ છે, ટી x એ xy નો પ્રાઈમ પ્લસ એફ છે. તેથી, વિવિધ ટી પ્રાઈમ અને ટી અને એસ પ્રાઈમ એસ અને બરાબર તે જ છે. તેથી, એસ પ્રાઈમ એ પછીના ટી પ્રાઈમ છે, તો એસ ટી કરતા વધુ સારી નથી. તેથી, તે એમ માનવું એક વિરોધાભાસ હતું કે હું ટી કરતાં સખત રીતે વધુ સારી રીતે આ 2 ની છે કે જે ફરીથી કમ્પ્યુટિંગ ટીની વ્યૂહરચના બધા કે માટે શ્રેષ્ઠ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 25:21)

અમલીકરણ વિશેનો શબ્દ, તેથી આપણે જે કરવાનું છે તે k માઈનસ 1 સમય છે, આપણને આ બે ન્યૂનતમ મૂલ્યોને મર્જ કરવું પડશે અને પુનરાવર્તન સોલ્યુશન, બોટલ ગરદનની ગણતરી કરવી જોઈએ, તે શું શોધશે ન્યૂનતમ મૂલ્યો. જો તમે એરેનો ઉપયોગ કરો છો, તો પછી અમે લઘુતમ મૂલ્યો શોધવા માટે એરે ઈન્સ્ટન્ટ સ્કેન જાણીએ છીએ, યાદ રાખો કે ન્યૂનતમ મૂલ્યો બદલાતા રહે છે, હું તે બધા માટે એક મોકલી શકતો નથી. કારણ કે, દર વખતે હું બે અક્ષરો ભેગા કરું છું, હું મારા એરેમાં એક નવો ન્યૂનતમ મૂલ્ય સાથે નવી અક્ષરનો ઉપયોગ કરી શકું છું જે નવો મૂલ્ય પહેલાં ન હતો અને નહીં તે નવો હોઈ શકે છે, જે તે પહેલાં ન હોઈ શકે અથવા તે ન્યૂનતમ ન પણ હોઈ શકે. તેથી, દરેક રાજ્યને મને ન્યૂનતમ શોધવું પડે છે, તેથી તે દરેક સમયે ઓર્ડર કેસ હોઈ શકે છે, તેથી રેખીય સ્કેન અને હું આ સમયને યોગ્ય રીતે કરું છું. તેથી, મને ઓર્ડર કેસ બે મળે છે, પરંતુ તે એક હોપનો ઉપયોગ કરીને આ બોટલ ગરદન આસપાસ મળી શકે છે, જ્યાં ઓછામાં ઓછા શોધવા માટે હીપ્સ સારી છે તે ચોક્કસ છે ત્યાં જોવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 26:13)

તેથી, જો હું ફ્રીક્વન્સીઝને જાળવી રાખું છું, તે એક ઢગલાની જેમ નથી, તો ઓર્ડર લોગ કે સમય, હું ન્યૂનતમ વેલ્યુ શોધી શકું છું અને પછી હું નવા સંયુક્ત અક્ષરોને પણ હેપમાં પાછું દાખલ કરી શકું છું. કે સમય લોગ. તેથી, દરેક પુનરાવર્તન કેટલાક લોગ લે છે, અને તેથી હું કે ચોરસથી k લોગ કેમાં સુધારો કરી રહ્યો છું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 26:33)

તેથી, યાદ રાખો કે, આપણે ઉલ્લેખ કરીએ છીએ કે શરૂઆતથી આ એક ગ્રીડી(Greedy) અલ્ગોરિધમ બનશે. તેથી, વાય ગ્રીડી(Greedy) છે, સારું કારણ કે દરેક વખતે, આપણે એક રિકર્સિવ કૉલ કરીએ છીએ, અમે બે નોડોને એક અને બે અક્ષરોમાં ભેગા કરવાનો નિર્ણય લઈ રહ્યા છીએ જે હંમેશાં સૌથી નીચલી આવર્તન સાથે પસંદ કરે છે. હવે, શું કહેવાનું છે કે, અમે દર ત્રીજા પંક્તિઓ પછી સૌથી નીચો પસંદ કરીને વધુ સારી રીતે કરી શક્યા નથી, પરંતુ હવે અમે પ્રયાસ કરીએ છીએ, અમે ફક્ત બીજા પંક્તિઓ સુધી સૌથી નીચો પ્રયાસ કરીએ છીએ. તેથી, અમે સ્થાનિક રીતે શ્રેષ્ઠ પસંદગી કરીએ છીએ અને અમે પસંદગી સાથે ચાલુ રાખીએ છીએ, ક્યારેય મુલાકાત લેવાનું ચાલુ રાખતા નથી, અને આખરે આપણને વૈશ્વિક ઉકેલ મળે છે. હવે, આપણે સાબિત કરીએ છીએ કે આ વૈશ્વિક ઉકેલ વાસ્તવમાં શ્રેષ્ઠ છે અને આપણે તે કરવું પડશે, કારણ કે ત્યાં બીજું કોઈ કારણ નથી કે હું ટૂંકા ગમતી પસંદગીની અપેક્ષા રાખું છું, વર્તમાન સમયે બે ખરાબ ફ્રીક્વન્સીઝ લો અને તેમને ભેગા કરો, કે તમે હંમેશાં શ્રેષ્ઠ ઉકેલ મેળવશે. તેથી, આ ખૂબ ગ્રીડી(Greedy) પત્ર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 27:25)

તેથી, છેલ્લે એક ટૂંકી ઐતિહાસિક નોંધ, તેથી ક્લોટ શૅનન માહિતી સિદ્ધાંતના પિતા છે અને જ્યારે આપણે 1950 ની આસપાસ આ સમસ્યાઓ પર ધ્યાન આપીએ છીએ, તેથી જ્યાં તેઓ આ સમસ્યાનો સામનો કરે છે અને કાર્યક્ષમ હોય છે . તેથી, શૅનન અને ફેનોએ, વિભાજનનો પ્રસ્તાવ મૂક્યો અને વસ્તુનો વિજય મેળવ્યો, તેથી અમને તે શું છે તે અમને મૂળાક્ષરોના એન્કોડિંગને જોવા દો. તેથી, તેમાંના કેટલાક 0 થી શરૂ થવાનું છે, કેટલાક અન્ય સાથે શરૂ થવાનું છે. 1. આ

કોડિંગ વૃક્ષનાં ડાબી ઉપ ટ્રીમાં જે બધું છે તે છે જે આપણે શોધી કાઢેલ 0 ની એક કોડ છે કંઈક, કંઈક, જમણી બાજુની દરેક વસ્તુ એકવાર મળી આવે તેવું કંઈક હશે. તેથી, તે તેમના માટે સાહજિક લાગે છે, પછી વિભાજીત કરો અને વ્યૂહરચનાને જીતી લો, તમે આ બાજુના અક્ષરો શામેલ કરી શકો છો, જેમ કે કુલ મૂળાક્ષરોની ફ્રીક્વન્સી વેઈટના આશરે. તે બધા અક્ષરોની આવર્તન છે, જે એન્કોડિંગ 0 થી પ્રારંભ કરે છે, તેમની ફ્રીક્વન્સીઝ લગભગ સખત રીતે ઉમેરવામાં આવે છે અને બીજી એક પણ મુશ્કેલ હોય છે. તો, બે સમાન વજનમાં મૂળાક્ષરને વિભાજિત કરો, તેમાંના કેટલાકને 0 થી શરૂ કરવા માટે, બીજું 1 થી પ્રારંભ કરવા માટે, પછી હું ફરી વાર કહીશ કે આ ટીને હલ કરી શકાય છે. તેથી, એક ભાગ એ 1, એ 2, દરેક અન્ય સેટમાં ફ્રીક્વન્સીઝનો જથ્થો મોટે ભાગે બરાબર છે, તેને વારંવાર હલ કરે છે. કમનસીબે, આ એક શ્રેષ્ઠ એન્કોડિંગ જનરેટ કરવાની ગેરંટી નથી, તમે ઉદાહરણ ઉપર આવી શકો છો, જ્યાં તમે આ કરી શકો છો અને તે પછી તે કંઈક સમાપ્ત કરી શકો છો જે તમે બીજાને કરી શકો છો. તેથી, તે બહાર આવ્યું કે હફમેન ફેનોના કોર્સમાં ગ્રેજ્યુએટ વિદ્યાર્થીઓ હતા, તેમણે આ સમસ્યા વિશે સાંભળ્યું અને અમે તેના વિશે વિચાર્યું, અને થોડા વર્ષો પછી તે આ ચપળ એલ્ગોરિધમ સાથે આવ્યા જે અમે તેમાં કર્યું છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 07
મોડ્યુલ - 01
લેક્ચર - 44
ડાયનેમિક પ્રોગ્રામિંગ

આગામી કેટલાક વ્યાખ્યાનમાં, અમે ડાયનેમિક પ્રોગ્રામિંગ તરીકે ઓળખાતા ઍલ્ગોરિધમ્સ ડિઝાઈન કરવા માટે એક ખૂબ જ શક્તિશાળી તકનીક જોશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:08)

તેથી, ડાયનેમિક પ્રોગ્રામિંગનો પ્રારંભિક મુદ્દો ધ્યાનમાં લેવાનો છે, જેને આપણે ઈન્ડક્ટિવ વ્યાખ્યાઓ કહીશું. હવે, ઘણા બધા સરળ કાર્યો છે જે આપણે પાર પાડીએ છીએ. તેથી, આપણે બધા જાણીએ છીએ કે ગાણિતિક રીતે n ફેક્ટોરિઅલ એ n times n minus 1 ફેક્ટોરિઅલ છે. તેથી, આપણે ફેક્ટોરીયલની એક ઈન્ટરેક્ટિવ વ્યાખ્યા લખી શકીએ છીએ. મૂળ કેસ એ છે જ્યારે n એ 0 છે અને આ કિસ્સામાં ફેક્ટોરિઅલ 1 છે, તેથી 0 નું એફ 1 છે. અને સામાન્ય રીતે, જો આપણી પાસે 0 થી વધારે હોય, તો n નો f n n ઓછા 1 ના f times દ્વારા આપવામાં આવે છે. બીજા શબ્દોમાં કહીએ તો, આપણે નાના ઈનપુટ પર લાગુ પાડવામાં આવેલા સમાન ફંકશનના સંદર્ભમાં n ના પરિભાષાને વ્યક્ત કરીએ છીએ, હવે આ પ્રકારની ઈન્ટરેક્ટિવ વ્યાખ્યા ફક્ત આંકડાકીય સમસ્યાઓને જ મર્યાદિત નથી, તમે તેને માળખાકીય સમસ્યાઓ માટે પણ કરી શકો છો. તેથી, સૂચિ અથવા અરેમાં ઉદાહરણ તરીકે, તમે કોઈ નાની સમસ્યા તરીકે પેટા સૂચિ અથવા સબ એરેને ધ્યાનમાં લઈ શકો છો. તેથી, નિવેશ સૉર્ટનું વર્ણન કરવાનો એક ખૂબ જ સરળ માર્ગ અહીં છે. તેથી, જો હું એન તત્વોને સૉર્ટ કરવા માંગું છું, તો માર્ગ દાખલ કરવાની રીત તે છે કે, મૂળ કેસને સૉર્ટ કરવા માટે કંઈ નથી, તો પછી અમે કર્યું છે. નહિંતર, અમે બીજા ઘટકથી શરૂ થતી બાકીની સૂચિને જુએ છે અને અમે તેને વારંવાર સૉર્ટ કરીએ છીએ અને પછી અમે સૉર્ટ કરેલ સૂચિમાં પ્રથમ તત્વ શામેલ કરીએ છીએ. તેથી, નિવેશ સૉર્ટ X 1 થી X n પર લાગુ થાય છે, અમને X 2 થી X n માં વારંવાર સૉર્ટમાં મૂલ્ય X 1 દાખલ કરવાની જરૂર છે. તેથી, ફરીથી આપણે તે જ ફંકશન લાગુ કરી રહ્યા છીએ જે આપણે નાના ઈનપુટ અને મૂળ કેસમાં વ્યાખ્યાયિત કરવાનો પ્રયાસ કરી રહ્યા છીએ, ખાલી એક નાનકડો ઈનપુટ એટલે કે આપણી પાસે એક જવાબ છે જે આપણા માટે સરળતાથી ઉપલબ્ધ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:57)

તેથી, આકર્ષણોમાંની એકમાં ઈન્ડેક્ટિવ વ્યાખ્યાઓ છે તે એ છે કે એક ખૂબ જ કુદરતી રીકર્સિવ પ્રોગ્રામ્સમાં ઉપજ. તેથી, આપણે ઘણું વિચારવાની જરૂર નથી, અમે લગભગ પ્રારંભિક વ્યાખ્યા લઈ શકીએ છીએ અને તેને પ્રોગ્રામ તરીકે સીધી અનુવાદિત કરી શકીએ છીએ. તેથી, અહીં ફૉક્ટોરિયલ માટેનું અનુવાદ છે જે અગાઉથી બંધારણની પ્રતિક્રિયા આપે છે, યાદ રાખો કે જે માળખું તે પહેલાં હતું તે 0 ની બરાબર 1 નું છે અને n ની f બરાબર n ની બરાબર n ગુણ્યા છે. 1. તેથી, આપણે કહીએ છીએ કે જો એન ... હવે, ફક્ત તેને થોડું વધારે મજબૂત બનાવવા માટે, જેથી લોકો આપે, તેઓ નકારાત્મક

નંબરો આપે, આપણને સમજદાર જવાબો મળે છે, તેથી ફક્ત n બરાબર 0 ની ચકાસણી કરી રહ્યા છે, તમે કોઈપણ માટે તપાસ કરી શકો છો મૂલ્ય 0 અથવા ઓછું, આપણે ફક્ત 1 પરત કરીશું. જો કોઈ વ્યક્તિ પૂછે છે કે બાદબાકી 7 નું પરિમાણ શાબ્દિક છે, તો આપણે ફક્ત 1 પરત કરવા માંગીએ છીએ. પરંતુ અપેક્ષિત વસ્તુ એ છે કે તેઓ 0 થી શરૂ થશે અને પછી જો તેઓ અમને એક સકારાત્મક નંબર આપશે જે 0 કરતા મોટો, પછી આપણે પુનરાવર્તન કીઓ પર જઈશું. તેથી, આપણે n minus 1 માટે ફેક્ટોરિઅલની ગણતરી કરીશું n એ ગુણાકાર કરીને અને પછી આ જવાબ પાછો આપીશું. તેથી, આ ઈન્ટરેક્ટિવ વ્યાખ્યા અને પુનરાવર્તિત પ્રોગ્રામ વચ્ચે એક પત્રવ્યવહાર માટે એક ખૂબ જ સીધો એક છે અને તે એક ફંકશનને વર્ણવવાના દૃષ્ટિકોણથી ઈન્ટરેક્ટિવ વ્યાખ્યાઓ ખૂબ આકર્ષક બનાવે છે. કારણ કે, અધિષ્ઠાપિત વ્યાખ્યા ગાણિતિક રીતે ન્યાયી હોઈ શકે છે અને પછી કાર્યક્રમ દેખીતી રીતે સાચી છે; કોડ્સમાં સ્પષ્ટ છે, કારણ કે તે ઈન્ડક્ટિવ વ્યાખ્યાના પુનરાવર્તનથી સીધા જ અનુસરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:17)

તેથી, આવી પ્રાસંગિક વ્યાખ્યાઓ શોષણ કરે છે તે ઘણી વખત શ્રેષ્ઠ પેટા માળખાકીય મિલકત કહેવામાં આવે છે. તેથી, જટિલ આધાર શું છે તે મૂળભૂત રીતે તે છે જે તમે ઈન્ટરેક્ટિવ ડેફિનેશનથી અપેક્ષા કરો છો કે તમે સબ સમસ્યાઓના ઉકેલને જોડીને મૂળ સમસ્યાને હલ કરો છો. તેથી, મૂળ સમસ્યાના ઉકેલો સબ સમસ્યામાં ઉકેલોના સંદર્ભમાં લેવામાં આવ્યા છે અને ખાસ કરીને આ સબ સમસ્યા એ જ પ્રકારની છે, તો તે સમાન પ્રકારના જવાબની ગણતરી કરી રહી છે. હવે, ફેક્ટોરીયલ જેવા આંકડાકીય પ્રશ્નાવલિમાં, કંઈક શ્રેષ્ઠ હોવાનું કહેવાનું અર્થપૂર્ણ નથી, પરંતુ ઉદાહરણ તરીકે, જ્યારે તમે નિવેશ સૉર્ટ કરો છો અને ચોક્કસપણે જ્યારે તમે સબ સૂચિને સૉર્ટ કરો છો ત્યારે તે પરિણામ તે પેટા સૂચિ માટે તમે ઇચ્છો છો. તેથી, આ સબ સમસ્યાની કલ્પનામાં વધારો કરે છે. તેથી, n માઈનસ 1 નો પરિભાષા સ્પષ્ટપણે n ના ફેક્ટોરિઅલની ફેક્ટોરિઅલ સબ સમસ્યા છે, પરંતુ તે માત્ર n ની ફેક્ટોરિઅલ પર થાય છે, તે માત્ર n માઈનસની માત્ર આવશ્યક ફેક્ટોરિઅલ છે. હવે, અમને સમસ્યાઓ આવી શકે છે જેને એકથી વધુ તાત્કાલિક જરૂર છે. નાની પેટા સમસ્યા, દાખલા તરીકે ફૉક્ટોરિઅલ એન માઈનસ 2 એ ઉપ સમસ્યા પણ છે, n માઈનસ 3 એ ઉપ સમસ્યા પણ છે. તેથી, કોઈપણ નાની ઇનપુટ મૂળ વસ્તુની ઉપ સમસ્યા તરીકે માનવામાં આવે છે. તેવી જ રીતે, જ્યારે આપણે વાસ્તવમાં સૉર્ટ દાખલ કરીએ છીએ, ત્યારે આપણે તેને X n માં ઇનપુટ X 1 આપીએ છીએ અને આપણે X 2 થી X n ને સૉર્ટ કરવા માટે કહીએ છીએ. તેથી, આ ઉપ સમસ્યા છે જે રાજ્યનો સીધો ભાગ છે. પરંતુ, સમાન્ય રીતે સૉર્ટિંગની સબ સમસ્યા તરીકે સૉર્ટ કરવા માટે કોઈપણ સેગમેન્ટ X i થી X j નો વિચાર કરી શકાય છે, આ એક ઉદાહરણ માટે થાય છે જેમ કે મર્જ સૉર્ટ અથવા ઝડપી સૉર્ટ, ખાસ કરીને સૉર્ટ કરો. પછી, તમે એરેને છિદ્રમાં વિભાજીત કરો અને પછી ક્વાર્ટર્સમાં ભરો અને તેથી, કોઈપણ સમયે તમે એ આઇ થી એજેના કેટલાક સેગમેન્ટમાં સમાન ઍલ્ગોરિધમનો ઉપયોગ કરી રહ્યા છો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:05)

તેથી, ચાલો હવે એવી સમસ્યાની તપાસ કરીએ જે આપણે ગ્રીડી(Greedy) અલ્ગોરિધમ્સના સંદર્ભમાં પહેલાં જોયેલી હોય. તેથી, આપણે અંતરાલ સુનિશ્ચિત તરીકે ઓળખાતી સમસ્યાને જોશું. તેથી, એકીકૃત સુનિશ્ચિતતાએ કહ્યું કે અમારી પાસે એક એવો સંસાધનો છે જે સમયાંતરે ઉપલબ્ધ છે અને હવે લોકો આવી આવશે અને આ સંસાધનો માટે બુકિંગ કરશે. તેથી, કોઈક આ સેગમેન્ટ દરમિયાન તેને બુક કરવા માંગે છે, કોઈ બીજું તેને આ સેગમેન્ટમાં બુક કરવા માંગે છે, બીજું કોઈ

પણ આ સેગમેન્ટ દરમિયાન ઈચ્છે છે અને બીજું. હવે, આ ઓવરલેપ થતી વસ્તુઓ દરમ્યાન, તમે બે લોકોને સંસાધન આપી શકતા નથી, તેથી તમે પ્રારંભિક સમય અને અંતિમ સમય માટે વિનંતીનો સમૂહ આપ્યો છે. તેથી, અમારી પાસે પ્રારંભ અને અંત અથવા સમાપ્તિનો સમય છે અને હવે તમે શું કરવા માંગો છો તે નક્કી કરો કે તમે આમાંથી કયા વિનંતીઓ ફાળવી શકો છો તે નક્કી કરો, જેથી બુકિંગની મહત્તમ સંખ્યા ખરેખર આપવામાં આવે. તેથી, ધ્યેય એ બુકિંગની સંખ્યા વધારવાનો છે, બુકિંગની લંબાઈ નહીં, પરંતુ બુકિંગની સંખ્યા.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:03)

તેથી, આ ખાસ કિસ્સામાં, શું થાય છે જ્યારે તમે બુકિંગનું સન્માન કરો છો, હવે જો બુકિંગ અન્ય બુકિંગ્સ સાથે ઓવરલેપ થાય તો, પછી જો મેં આ બુકિંગ લેવાનું નક્કી કર્યું હોય, તો આ દૂર જાય છે. તેથી, આ બે બુકિંગ્સ જે તેની સાથે ઓવરલેપ થઈ ગઈ છે, તે હવે સુનિશ્ચિત કરી શકાશે નહીં, કારણ કે તેઓ આમાં થોડો અંતરાલ સાથે સંઘર્ષ કરે છે. તેથી, તેથી હવે બુકિંગની સમસ્યાના કેટલાક ઉપગ્રહ માટે બાકીની બુકિંગ ફાળવવાનો માર્ગ ઉકેલવો અથવા શોધવાનો છે. તેથી, આ કિસ્સામાં બુકિંગનો દરેક પેટા ઉપાય એ ઉપાય છે અને જે વ્યૂહરચના અમે જોયેલી તે એક ગ્રીડી(Greedy) હતી, જેણે પ્રારંભિક અંતિમ સમય પસંદ કરવાનો નિર્ણય લીધો હતો. તેથી, તે બધી બુકિંગ્સમાં જે હજુ પણ અમને ફાળવવા માટે ઉપલબ્ધ છે, અમે એક ગ્રીડી(Greedy) રીતે એક પસંદ કરીએ છીએ તે ફક્ત તે જ જોવાનું છે જે તે પહેલાના અંતિમ સમય રહેશે. હવે, જેમ આપણે કહ્યું હતું કે, જ્યારે આપણે ઉમેરીશું, ત્યારે તે કેટલીક બુકિંગ્સને દૂર કરશે જે વિરોધાભાસ છે જે આપણને ઉપ સમસ્યા આપે છે અને તમે આ ઉપ સમસ્યાને હલ કરશો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:06)

તો, ત્યાં કેટલી પેટા સમસ્યાઓ છે? હવે, અમારી પાસે એન બુકિંગ છે અને આનો દરેક સબસેટ ઉપ ઉપસ્થિતિ છે, તેથી અમારી પાસે પેટા સમસ્યાઓની ઘાતક સંખ્યા છે. સામાન્ય રીતે, અમારી પાસે કોઈ શક્ય સબસેટ જવાબ હોઈ શકે છે. તેથી, આપણે સર્વશ્રેષ્ઠ ફાળવણી શોધવા માટે, આ બધી ઘાતાંકીય વસ્તુઓને સિદ્ધાંતમાં જોવું જોઈએ, જે સૌથી વધુ બુકિંગને સંતોષવા માટે આપે છે. હવે, ગ્રીડી(Greedy) વ્યૂહરચના શું છે તે અસરકારક રીતે આ રેખાંશ જગ્યાને રેખીય જગ્યામાં કાપી નાખે છે, કારણ કે તે શું કરે છે તે પસંદ કરે છે, તેથી અમે શરૂઆતમાં બુકિંગ્સ 1 ને એન પર લઈએ છીએ. પછી, તેમાંથી તે પસંદ કરશે, આપણે તેને ધારીએ વાસ્તવમાં પ્રારંભિક અંતિમ સમયના ક્રમમાં ગોઠવવામાં આવે છે. તેથી, તમે પહેલો જ લો, પછી તમે ત્યાંથી થોડામાંથી બહાર નીકળશો. અને હવે તમારી પાસે થોડા બાકી હશે અને તેમાંના એકમાં આપણે સૌથી પહેલા એક પસંદ કરીશું અને તમે તમારામાંથી વધુ ને વધુ નકારી કાઢશો. તેથી, મોટાભાગે તમે તે બધા એનને ફાળવશો, પરંતુ દરેક વખતે જ્યારે તમે નિયમનો સમાવેશ કરો છો તેમા 1 શામેલ છે, તો તમે થોડામાંથી બહાર નીકળશો. તેથી, ચોક્કસપણે રેખીય સ્કેનમાં, તમે ફક્ત તે જ સબ સમસ્યાઓને જોશો. તેથી, તમે માત્ર એન પેટા સમસ્યાઓનો ઓર્ડર જુઓ અને શોધવાનું, તમે જે દાવો કરશો તે એક શ્રેષ્ઠ જવાબ છે. અને ત્યારથી, તમે 2 થી એન અને ઓર્ડર N માંથી આવા કડક કપાત કરી રહ્યા છો; દેખીતી રીતે, ત્યાં પૂછવાની એક પ્રશ્ન છે કે તમે અયોગ્ય રીતે તેમને તપાસતા ન હોવાને કારણે કેટલીક ઉપ સમસ્યાઓની અવગણના કરી છે કે કેમ. તેથી, તમારે સાબિતીની જરૂર છે, તેથી જ ગ્રીડી(Greedy) વ્યૂહરચનામાં, તમારે સાબિત કરવાની જરૂર છે કે તમે જે કરી રહ્યા છો તે વાસ્તવમાં સોલ્યુશન બહાર આવે છે, કારણ કે તમે ખરેખર મોટી સંખ્યામાં નથી જોઈ રહ્યા છો, પેટા સમસ્યાઓની સંખ્યા.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:39)

તો, માની લો કે આપણે અંતરાલ સમયપત્રક સમસ્યાને ખૂબ જ સહેજ બદલીએ છીએ, અમે દરેક વિનંતીને એક વજન સાથે સાંકળીએ છીએ, ઉદાહરણ તરીકે વજન હોઈ શકે છે, જે રકમ કોઈને ચૂકવવા માટે તૈયાર છે, તેથી લોકો પ્રયાસ કરી રહ્યા છે બુક કરવા માટે, તેથી અમારી પાસે એક ઑડિટોરિયમ છે જે અમે પ્રદર્શન અને અન્ય પ્રવૃત્તિઓ માટે દાખલ કરીએ છીએ. અને લોકો, તેનો ઉપયોગ કરવા માટે આવે છે તે તેનો ઉપયોગ કરવા માટે ચુકવણી કરવા તૈયાર છે. અલબત્ત, ફક્ત એક જ ઓડિટોરિયમ છે, તેથી બે લોકો એક જ સમયે ઉપયોગ કરી શકતા નથી. હવે, અમારું અગાઉનું લક્ષ્ય બુકિનની સંખ્યાને મહત્તમ બનાવવું હતુંઅમે જે આપ્યું છે, પરંતુ હવે, અમારી પાસે બીજું માપદંડ છે જે વધુ તાત્કાલિક છે, જે દરેક વ્યક્તિ કેટલી ચૂકવણી કરવા તૈયાર છે. તેથી, જો માત્ર એક જ વ્યક્તિને આપે તો પણ, તે વ્યક્તિ બીજા બધા કરતા વધારે ચૂકવણી કરે છે અને તે આપણા માટે શ્રેષ્ઠ હશે. તેથી, હવે અમારો ધ્યેય કુલ વજન વધારવાનો છે. તેથી, અમે અમારા ફાળવણીથી શક્ય એટલું આવક મેળવવા માંગીએ છીએ કે બુકિંગની સંખ્યા નહીં, પરંતુ કુલ વજન.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:40)

તેથી, અગાઉના કિસ્સામાં ગ્રીડી(Greedy) વ્યૂહરચનાને યાદ કરો, અમે પ્રારંભિક સમાપ્તિ સમય ઈચ્છતા હતા. તેથી, જો આપણે આ ત્રણ વિનંતીઓની ચોક્કસ પસંદગી જોયેલી હોય, તો પછીનો પૂર્ણાહુતિનો સમય આ હશે. તેથી, આપણે સૌ પ્રથમ આને લઈશું, જે આને નકારી કાઢશે અને પછી ત્રીજી વિનંતી પ્રથમ પૂર્ણ થાય પછી શરૂ થાય છે, તેથી તમે આ લેશે અને તેથી આપણને બે કીઓ મળશે. અને 3 માંથી 2 એ શ્રેષ્ઠ છે જે આપણે કરી શકીએ છીએ અને તે બિન વેઇટ્ડ કીઓમાં મળી આવ્યું છે. પરંતુ, કમનસીબે, અમારી પાસે જે છે તે છે કે આપણે વજન ધરાવીએ છીએ, તેથી અમારી પાસે આ વજન સંકળાયેલું છે. તેથી, આપણે કંઈક વધુ ચપળ કરવું પડશે, કારણ કે હવે જો આપણે પહેલો એક અને ત્રીજો પસંદ કરીએ, તો કુલ વજન ફક્ત 2 છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:44)

તેથી, નોકરી, નોકરી નહી, પરંતુ ચાલો આપણે તેને બુકિંગ કરો 1 વત્તા 3 એ 2 નું વજન આપે છે, જ્યારે એકલા બુકિંગ 2 નું વજન આપે છે, કારણ કે તેનું વજન 3 છે. તેથી, આ સ્થિતિમાં આદર્શ રીતે, આપણે સ્વીકારવું જોઈએ કે મધ્યમ વિનંતીને દંડને દૂર કરવા માટે પૂરતા વજન છે તે માત્ર એક જ હશે જે સુનિશ્ચિત કરવામાં આવશે. તેથી, જો કે અમે કોઈ 3 વિનંતી શેડ્યૂલમાંથી બહાર નીકળીશું, વાસ્તવમાં અમને ખર્ચ સંભવિતથી મહત્તમ લાભ મળે છે. તેથી, તેથી બીજા શબ્દોમાં જેનો અર્થ છે તે એ ગ્રીડી(Greedy) વ્યૂહરચના છે જે અમે ભારાંક વગરના કેસ માટે સાબિત કરી છે તે કમનસીબે વધુ માન્ય નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:27)

તો, આપણે શું કરીશું? તેથી બીજી ગ્રીડી(Greedy) વ્યૂહરચનામાં જોવાની એક વ્યૂહરચના છે, અમે બીજી ગ્રીડી(Greedy) વ્યૂહરચના શોધી શકીએ છીએ અને દલીલ કરવાનો પ્રયત્ન કરીએ છીએ કે તમે કામ કરશો અને એવી દલીલ કરો કે તે જે

કાર્ય કરે છે તે કામ કરે છે, તે થોડો પ્રયાસ લે છે. કારણ કે, અમને એવી કોઈ વસ્તુની વિનિમય દલીલનો ઉપયોગ કરવો પડશે જે પુરાવા આપે છે કે કોઈપણ અન્ય વ્યૂહરચના દ્વારા તમને પ્રાપ્ત થઈ શકે તેવા કોઈપણ ઉપાય કરતાં વધુ સારું છે. તેથી, બીજા અભિગમ જે આપણે આગામી થોડા વ્યાખ્યાનમાં વધુ વિગતવાર તપાસ કરવા જઈ રહ્યા છીએ તે છે એક પ્રાયોગિક સોલ્યુશનનો પ્રયાસ કરવાનો પ્રયાસ કરો જે દેખીતી રીતે સાચી છે કે જેની કાર્યક્ષમતાપૂર્વક મૂલ્યાંકન કરી શકાય. તેથી, ધ્યેય બચાવવા માટે છે, તેથી આપણે જે બચાવવા જઈ રહ્યા છીએ તે એ સાબિત કરવાનો આ પ્રયાસ છે કે ફક્ત કેટલાક કિસ્સાઓમાં જ, આપણે ખરેખર શ્રેષ્ઠ જવાબ આપી રહ્યા છીએ. અમે દરેક કિસ્સામાં કેટલાક અર્થમાં જોશું, પરંતુ આપણે દરેક કેસને ચપળતાપૂર્વક જોવું જોઈએ અને તે જ છે, આપણે જઈ રહ્યા છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:22)

તેથી, આ સમસ્યા માટે આપણે આ કેવી રીતે કરી શકીએ તે માટે આ સમય છે, ચાલો આપણે કંઈક વધુ સીધી રીતે કરીએ, પછી આપણે છેલ્લા સમયને જોવા માટે છેલ્લી વાર શું કર્યું, ફક્ત જુઓ પ્રારંભિક પ્રારંભિક સમયે. તેથી, ચાલો ધારીએ કે કે કાર્યો વિનંતી છે કે આના જેવા કૉલ ઓર્ડર. તેથી, અમે આ ક્રમમાં તેમને પસંદ કરો. તેથી, અમે પહેલી બુકિંગ સાથે પ્રારંભ કરીએ છીએ જેને તમે બી 1 કહે છે. હવે, અવલોકન કરો કે અંતિમ જવાબમાં, ક્યાં તો બી 1 છે, બી 1 ત્યાં નથી. તો, આપણે બે વિકલ્પો લઈશું. તેથી, હા બી 1 અને ના બી 1, હવે જો આપણે બી 1 ને દૂર કરીએ, તો પછી અથવા ઉપ સમસ્યામાં ફક્ત બી 2 ને સમાવશે. તેથી, અમારી પાસે ફક્ત એક ઉપ સમસ્યા છે જે બી માટે બી છે. તેથી, જો આપણે બી 1 ને બાકાત રાખીએ, તો આપણે ફક્ત બી 2 નો બી નો ઉપયોગ કરીએ. તેથી, બી 1 ને બાકાત રાખવાનો અર્થ એ છે કે ફક્ત તેને બહાર ફેંકી દો અને તમને દર્શાવશે કે તમે માત્ર એન બાદ 1 ની શરૂઆતથી નજીક છે. બીજી તરફ, જો આપણે બી 1 નો સમાવેશ કરીએ, તો તમારે થોડું સાવચેત રહેવું જોઈએ, કારણ કે હવે જો હું આ વિશિષ્ટ વસ્તુમાં બી 1 શામેલ કરું છું, તેથી જો આપણે બી 1 શામેલ કરીએ, તો અમને કંઈક કે જે સંઘર્ષ તેથી, અમે ગ્રીડી(Greedy) કેસમાં કરેલા તમામ વિવાદાસ્પદ વિનંતીઓને દૂર કરીએ છીએ અને પછી અમારી પાસે બીજી પેટા સમસ્યા છે જે જરૂરી નથી કે બી 2 થી બી એન, તે બી 2 નું બી પેટા સેટ હશે, પરંતુ હવે આપણે બન્ને વિકલ્પો લેવામાં આવ્યા છે, આપણે બી 1 શામેલ છે અને બાકાત બી 1 શામેલ છે, તેથી એવી અપેક્ષા રાખવી વધુ વાજબી છે કે અમારી પાસે બી 1 સાથે અથવા બી 1 વિના ઉકેલ છે, ત્યાં ત્રીજો વિકલ્પ નથી, સોલ્યુશનમાં બી 1 હોય છે અને કરે છે બી 1 નથી, અમે બંનેનું મૂલ્યાંકન કરવાનો પ્રયાસ કરી રહ્યા છીએ અને પછી અમે શ્રેષ્ઠ વિકલ્પ પસંદ કરવાનો પ્રયાસ કરી રહ્યા છીએ. તેથી, આ બે ઉપ-કેસો સાથે સમસ્યાના એક અપવાદરૂપ ઘૂસણખોરી છે, બી 1 વિના બી 1, આપણે કોઈ આગાહી કરી રહ્યા નથી જે વધુ સારું છે, અમે બંનેનું મૂલ્યાંકન કરીએ છીએ અને વધુ સારી રીતે લઈએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:21)

તેથી હવે ચાલો દલીલ કરીએ કે આ પ્રકારની વ્યૂહરચના વાસ્તવમાં વિકલ્પોની ગણના કરે છે. તેથી, કોઈપણ બીજે માટે બી 1 ની જેમ, સોલ્યુશનમાં બીજે હોય છે અથવા પાસે બીજે નથી, આ ખૂબ સ્પષ્ટ છે. તો, ત્યાં સંભવિત 2 ને શક્ય ઉકેલ છે, મારી પાસે બી 1 હોઈ શકે અથવા બી 1 ન હોય, બી 2 હોય અથવા બી ન હોય 2. તેથી, હું દરેક સંભવિત સબસેટનો પ્રયાસ કરી શકું જે જૂથ બળ દલીલ હશે, આપણે દરેક સંભવિત સબસેટનો પ્રયાસ કરવાનું ટાળો. હવે, બી 1 માટે આપણે સ્પષ્ટપણે બન્ને કેસો સ્પષ્ટપણે ચકાસી લીધા છે, બી 2 વિશે શું, આપણે ખાતરી કરી શકીએ કે આપણે બધા કેસો ચકાસી

રહ્યા છીએ બી 2 એ આપણને b પર જોવા દો. હવે, જો બી 2 અને બી 1 માં નથી સંઘર્ષ કે જે બી 1 અને બી 2 છે અથવા વિવાદિત અંતરાલમાં, પછી બી 1 પસંદ થયેલ છે કે નહીં તે સ્વતંત્ર છે અથવા બી 2 પસંદ કરેલું નથી. આનો અર્થ એ છે કે આપણે બી 1 અથવા બી 2 ને પસંદ કરીએ છીએ કે પરિણામી પેટા સમસ્યા હજી પણ બી 2 પસંદ કરવાની મંજૂરી આપે છે. તેથી, જો આપણે બી 1 અથવા બી પસંદ ન કરીએ કે શરૂઆતમાં બી 1 નહીં, તો તે ધ્યાનમાં લેવામાં આવશે બંને પેટા સમસ્યા અને જ્યારે, અમે હલ કરીએ છીએ કે તમે બંને પસંદગીઓ લો. બીજી બાજુ, બી 1 અને બી 2 તુલનાત્મક નથી અથવા તુલનાત્મક નથી, તે બી 2 અથવા બીના બી 1 નિયમો છે, કારણ કે તેઓ ઓવરલેપ થયા છે. પછી, જ્યારે બી 1 પસંદ કરવામાં આવે છે બી 2 ત્યાં હોઈ શકતું નથી, તેથી બી 1 ત્યાં હોઈ શકે છે જો બી 2 હોઈ શકે, ફક્ત બી 1 ત્યાં ન હોય. તેથી, જ્યારે બી 1 પસંદ કરવામાં આવે છે ત્યારે આપણે બી 2 ને ધ્યાનમાંશું નહીં, પરંતુ બી 1 પસંદ નથી કરતું યાદ રાખીએ કે આપણને પરિણામી પેટા સમસ્યા બી 2 થી બી એન મળે છે. તેથી બી 2 ફરીથી પસંદ કરવામાં આવશે અથવા પસંદગી આપવામાં આવશે, તેથી બી 2 અમે હાજરી અથવા ગેરહાજરીમાંના બધા વિકલ્પોને ધ્યાનમાં રાખીએ છીએ. એ જ રીતે, આપણે એવી દલીલ કરી શકીએ છીએ કે બી 3 ની હાજરી અથવા બી 1 ની ગેરહાજરીમાં ધ્યાનમાં લેવામાં આવશે અને શું થઈ રહ્યું છે, કારણ કે આપણે વધુ પર વધુ બનાવવા સાથે આગળ વધી રહ્યા છીએ. વચનો, અમે અસંખ્ય અસંગત સંયોજનોને ચૂકાદા આપી રહ્યા છીએ જે આપણે અન્યથા અંધકારપૂર્વક વિચારણા કરીશું, અમને એનને 2 મળશે. હવે, અસરકારક રીતે મૂલ્યાંકન કરવું પડશે, પરંતુ ઓછામાં ઓછું એવું માનવું મુશ્કેલ નથી કે આપણે ખરેખર પ્રયાસ કરી રહ્યા છીએ દરેક શક્ય વિકલ્પ બહાર. અમે અગાઉથી નક્કી નથી કરી રહ્યા છીએ કે ગ્રીડી(Greedy) વ્યૂહરચના જેવી કેટલીક સ્થાનિક માપદંડો ચોક્કસ પેટા સમસ્યાઓને નકારી કાઢવા માટે પૂરતી છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:33)

તેથી, ગણતરીત્મક પડકાર એ હકીકત પરથી આવે છે કે આપણે જે ઉપ સમસ્યાઓ પેદા કરીએ છીએ તે વારંવાર દેખાય છે. તેથી, ચાલો આપણે એક સરળ કેસ જોઈએ, તમે ધારો છો કે અમે તે ચિત્ર છે જે નીચે બતાવેલ છે. તેથી, આપણી પાસે બી 1 અને બી 2 છે જે સંઘર્ષમાં છે, પરંતુ ધ્યાન આપો કે બી 1 અને બી 2 બન્ને શબ્દો જે શબ્દો પછી આવે છે તેનાથી સુસંગત છે. તેથી, જો આપણે બી 1 પસંદ કરીએ, તો આપણે બી 2 ને નકારીશું અને તેથી ઉપ સમસ્યા, આપણને બી 3, બી 4 અને બી એન મળશે. બીજી બાજુ, જો તમે બી 1 ને નકારી કાઢો તો તમે પ્રયાસ કરો તે પહેલાં અમે કહ્યું હતું જે બધું બાકી છે તે બી 2 થી બી એન છે. હવે, તમારે બી 2 ને એન બી પ્રયાસ કરવો પડશે, તેથી હવે, જ્યારે તમે બી 2 બી એન આવે ત્યારે, તમારે બી 2 ને નાશ કરવો પડશે અથવા તમારે બી રાખવા પડશે. 2. તેથી, તમે બી 2 કાઢી નાખો છો, તો પછી જ્યારે તમે અહીંથી બી 2 ને અવગણશો ત્યારે શું થાય છે, તમે બાકીના ભાગને B 3 થી B N સુધી મેળવી શકો છો. તેથી, તમે ફરીથી બી 3 ની સમસ્યા ઉભી કરો જે પહેલાથી પૂછવામાં આવી હતી. એક વખત બી પસંદ કરવાના ડિસ્કના સંદર્ભમાં. તેથી, હવે તમારી પાસે છે કે તમે બી 1 પસંદ કર્યું છે, હા, ના કહી દો; જો તમે બી 1 પસંદ કરો છો તો મને આ સમસ્યા મળે છે જે બી 3 થી બી એન છે. પછી, જો હું કોઈ પસંદ કરું, તો મને બી 2 ફરીથી પસંદ કરવાની તક મળે છે, હા અને ના, જો હું બી પસંદ ન કરું તો 2. પછી, હું. બી 2 ને ફરીથી કાઢી નાખો, મને બી 3 થી બી એન મળે છે, તેથી હું અહીં એક વાર અને એક વાર અહીં આ સમસ્યાને ઉકેલવા માંગું છું, સિવાય કે હું કોઈ હોંશિયાર કરું.

(સ્લાઈડસમયનો સંદર્ભ લો: 18:09)

તેથી, આ અભિગમ સાથેની સંપૂર્ણ સમસ્યા એ છે કે ઈન્ડેક્સિવ સોલ્યુશન વિવિધ તબક્કે સમાન સમસ્યાનું ઉદભવ કરી શકે છે. અને જો આપણે ફક્ત પુનરાવર્તનનો ઉપયોગ કરીએ છીએ જેમ કે અમે જાસૂસી વ્યાખ્યા પરના એક મહાન ફાયદાઓને ધ્યાનમાં લેતા પહેલાં તમે કહ્યું છે કે તમે ફક્ત પુનરાવર્તિત સોલ્યુશન લખી શકો છો જે સમસ્યાની આંતરિક વ્યાખ્યાને ફક્ત મિરર કરે છે. પરંતુ જો તમે નૈતિક રીતે કરો છો, તો તમે દરેક વખતે કાર્યમાં આવવા માટે આવો ત્યારે, તમે તે જ કાર્યને વારંવાર બોલાવો છો, પછી ભલે તમે પહેલા કર્યું હોય અને તે ખૂબ જ ખર્ચાળ કાર્યક્ષમતા હોઈ શકે. તેથી, ડાયનેમિક પ્રોગ્રામિંગનો ધ્યેય આ કચરાને ટાળવા માટે છે. તેથી, ત્યાં બે તકનીકો છે જે આપણે જોઈશું, ત્યાં સ્મૃતિ કહેવાતી તકનીક છે, જે રિકર્ઝનમાં કેટલીક હોંશિયારતામાં નિર્માણ કરવાનો એક રસ્તો છે. તેથી, તમે આ જ ફંક્શનને ફરી વારંવાર બે વખત કૉલ કરશો નહીં. અને ડાયનેમિક પ્રોગ્રામિંગ એ આ એક સાથે આવતીકાલિક કોલ્સને એકસાથે ટાળવા માટેનો એક રસ્તો છે. તેથી, ડાયનેમિક પ્રોગ્રામિંગ એ સબ સમસ્યાઓને સીધી રીતે સમજાવવા અને તેમને હલ કરવાનો એક રસ્તો છે, કેમ કે તે જાણવું કે ઉપ સમસ્યાઓમાં કેટલીક નિર્ભરતા છે જેની તમે આગાહી કરી શકો છો. તેથી, અમે આ બે તકનીકોને ધ્યાનમાં લઈશું, આગામી કેટલાક પ્રવચનો અને મેમોઈઝેશન અને ડાયનેમિક પ્રોગ્રામના આ વિચારોથી પરિચિત થવા માટે કેટલાક ઉદાહરણો જોઈએ છીએ, જે ઉકેલવા માટે કાર્યક્ષમ આવર્તક અમલીકરણમાં આવશ્યક રૂપે વ્યાખ્યાયિત છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
મોડ્યુલ - 02
લેક્ચર - 45
સંસ્મરણાત્મક

ચાલો આપણે પ્રેરક વ્યાખ્યાઓની ચર્ચા ચાલુ રાખીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:05)

તેથી, યાદ રાખો કે ફેક્ટોરિયલ અને ઈન્સર્શન સૉર્ટ જેવા ફંકશન નાના પેટા સમસ્યાઓના સંદર્ભમાં કુદરતી પ્રેરક વ્યાખ્યાઓ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:16)

અને પ્રેરક વ્યાખ્યાઓને જોવાનું આકર્ષણ એ છે કે, આપણે ખૂબ જ સરળતાથી એવા રિકર્સિવ પ્રોગ્રામ્સ સાથે આવી શકીએ જે તેમને ખૂબ જ સ્પષ્ટ રીતે અમલમાં મૂકશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:25)

પરંતુ, ઉપ સમસ્યાઓની ઓળખ કરવામાં પડકારો છે, હું ખાતરી કરી રહ્યો છું કે તેઓ ઓવરલેપડ નહીં થાય. તેથી, હકીકતદર્શી કિસ્સામાં યાદ રાખવું કે કોઈપણ નાની ઈનપુટ ફેક્ટોરિઅલ એ ઉપ સમસ્યા છે, સૉર્ટિંગ માટે તમે સબ સમસ્યા તરીકે સૉર્ટ કરવા માટે સૂચિના કોઈપણ ભાગને વિચારી શકો છો. અને સામાન્ય રીતે, આપણે પુનરાવર્તિત સોલ્યુશન્સ અથવા ઈન્ડિક્યુટિવ સોલ્યુશન્સ પર ધ્યાન આપીએ છીએ, જ્યાં આપણે જે ફંકશન એફ વ્યાખ્યાયિત કરવાનો પ્રયાસ કરી રહ્યા છે તેના ઉકેલને મૂળ સમસ્યાની પેટા સમસ્યાઓના ઉકેલ દ્વારા ઉદ્ભવ્યું છે અને તેથી આપણે y કરતાં નાના ઈનપુટ્સ પર f નો ગણતરી કરીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:00)

તેથી, ચાલો જોઈએ કે આ કેવી રીતે પરિચિત શ્રેણી સાથે કામ કરે છે જે તમને ફિબોનાકી નંબર્સ દ્વારા જાણવામાં આવી શકે છે. તેથી, ચાલો આપણે ફિબોનાકી નંબરોને નીચે પ્રમાણે વ્યાખ્યાયિત કરીએ, પ્રથમ બે ફિબોનાકી નંબર્સ 0 અને 1 છે અને પછી આ બંનેને ઉમેરીને દરેક અનુગામી ફિબોનાકી સંખ્યા પ્રાપ્ત થાય છે. હવે, આપણે ફિબોનાકી નંબર્સને કેવી રીતે ક્રમાંકિત કરીશું તે પ્રારંભ કરીએ તે પહેલાં તે જોવાનું સૂચન આપવામાં આવે છે. આપણે જાણીએ છીએ કે પ્રથમ બે 0 અને 1 છે અને આપણે જાણીએ છીએ કે આગળનો આ બેનો સરવાળો છે. તેથી, પછીનો એક ફરીથી થશે, પછીનો એક 2 હશે, આ 1 વત્તા 1 માટે, પછીનો એક 3 હશે, પછીનો એક 5 હશે, પછીનો એક 8 થશે, પછીનો એક 13 થશે અને બીજું. તેથી,

તે ખૂબ જ સ્પષ્ટ છે કે આ નંબરો પ્રેરક વ્યાખ્યા તરીકે હોવા છતાં અને ત્યાં એક સ્પષ્ટ પુનરાવર્તન કાર્ય છે જે તેની સાથે જાય છે જે મેં હમણાં જ એક મિનિટમાં ઉલ્લેખ કર્યો છે. આ નંબરોની ગણતરી કરવાની ખૂબ જ અસરકારક રીત છે, સીધી લગભગ રેખીય સમયમાં. તો, ફંક્શન કોડની વ્યાખ્યા કેવી રીતે કરે છે? N ની Fibonacci એ 0 છે, જો n 0 છે, 0 પરત કરો, જો n એ 1 રીટર્ન 1 છે. તો, જો n 0 અથવા 1 છે, તો મૂલ્ય પોતે જ છે. નહિંતર, તમે આ માપદંડનો ઉપયોગ કરીને પુનરાવર્તિત મૂલ્યની ગણતરી કરો, n માઈનસ 1 વત્તા ફિનાકાસી ની n માઈનસ 2 ના અંતમાં અને છેલ્લે, તમે જે પણ મૂલ્યનું મૂલ્ય કરો છો તે પરત કરો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:21)

તેથી, કેચ ક્યાં છે? તેથી, ચાલો જોઈએ કે આ કેવી રીતે નાના ઈનપુટ પર કામ કરશે જેમ કે આપણે ફક્ત ફિબોનાકી જોયું છે. તેથી, આપણે ફક્ત 5 ની ફિબોનાસી જોયું, આ 5, આપણે જાણીએ છીએ કે આપણે 0, 1, 1 સાથે પ્રારંભ કરીએ છીએ, તેથી આ દલીલો છે. તેથી, મૂલ્યો 0, 1, 1, 2, 3 અને પછી 5 છે. તેથી, આપણે ગણતરી કરવાની કોશિશ કરી રહ્યા છીએ, જો આ એક ફિબોનાકી નંબર છે જેનો આપણે ખૂબ જ ઝડપથી ગણતરી કરીએ છીએ. પરંતુ, ચાલો જોઈએ કે આ પુનરાવર્તિત વસ્તુ ખરેખર કેવી રીતે મળી શકે, જો હું 5 ના ફિબોનાસીને બોલાવીશ, તો તે ફરીથી મને 4 અને 3 ના ફિબોનાસીની ગણતરી કરવા કહેશે, 4 ની ફાઈબોનાસી બદલામાં મને 3 અને 2 ની ગણતરી કરવા કહેશે.

(સ્લાઈડજુઓ સમય: 02:55)

હવે, 3 બદલામાં મને 2 અને 1 ની ગણતરી કરવા માટે પૂછશે, તો 2 બદલામાં મને 1 અને 0 ની ગણતરી કરવા કહેશે. હવે, સદભાગ્યે 1 અને 0 માટે, મારી પાસે મૂળ કેસ છે. તેથી, હું તે વેલ્યુ અનુક્રમે 1 અને 0 મેળવીશ. તેથી, કારણ કે મને 1 અને 0 મળ્યું છે અને હવે હું તેમાંથી ફિબોનાકી 2 ની ગણતરી કરી શકું છું અને મને જવાબ મળી શકશે. હવે, હું ફિબોનાકી 1 માં ફેરવાઈ ગયો છું, ફરીથી તે બેઝ કેસ છે, મને મળે છે. જવાબ 1. તેથી, હવે મારી પાસે બંનેના જવાબો 3 ની ફિબોનાસી માટે જરૂરી છે. તેથી, મને જવાબ 2 મળે છે, હવે હું પાછો જાઉં છું અને હું 2 ની ફિબોનાકીની ગણતરી કરું છું, પરંતુ યાદ રાખો કે મેં પહેલાથી તેનું ગણતરી કરી છે, પણ કારણ કે મેં નથી બનાવ્યું આનો કોઈ અર્થમાં નોંધ કરો, હું આંખે રિકરશન કરું છું. 2 ની ફિબોનાસી ફરીથી મને 1 અને 0 ના ફિબોનાસીની ગણતરી કરવા કહેશે, જે ફરીથી 1 અને 0 સાથે બેઝ કેસ ઉત્પન્ન કરશે. તેથી, હું ફરીથી ફિબોનાકીની 2 ની 1 ઘાત ગણું છું અને હવે બંને 3 અને 2 ની ફિબોનાસી ઉપલબ્ધ થશે, હું 4 અને 3 ની ફિબોનાસીનો જવાબ મેળવીશ. અને હવે, હું પાછો જઈશ અને ફરીથી ફિબોનાકી 3 નું ગણતરી કરીશ. તેથી, હું આ આખું વૃક્ષ ફરીથી કરીશ, 3 કોલ 2 અને 1, 2 કોલ 1 અને 0, આ બેઝ કેસ આપશે, જે મને 2 નું મૂલ્ય આપશે, 1 બેઝ કેસ આપશે, તે મને 3 ની વેલ્યુ આપશે. છેવટે, આ બધા પછી, મને 5 નું મૂલ્ય ફિબોનાસી મળે છે. તેથી, આપણે જોઈ શકીએ છીએ તે સમસ્યા એ છે કે ફિઓનાકસી 3 નું પૂર્ણ અને સંપૂર્ણ જટિલતામાં ગણતરી કરવામાં આવે છે, અન્ય કાર્યો જેમ કે ફિબોનાકી 2 ની ગણતરી કરવામાં આવી છે 1, 2 , 3 વખત અને તેથી વધુ. અને અલબત્ત, તેના તરફ 1 ની ફિબોનાસી એ મૂળ કેસ છે 1, 2, 3, 4, 5 વખત અનેક વખત કૉલ કરવામાં આવ્યો છે અને 0 ની ફિબોનાકીને 3 વખત કહેવામાં આવે છે. તેથી, આ પુનરાવર્તિત ગણતરી સાથે આ એક મોટી સમસ્યા છે, પરંતુ આપણે 5 ની ફિબોનાકીની ગણતરી કરવા માટે આદર્શ રીતે ઘણી બધી વસ્તુઓનું ગણતરી કરી રહ્યા છીએ અને મને ખબર છે કે મને 4, 3, 2, 1 અને 0. ની જરૂર છે તેથી મને જરૂર છે ફક્ત છ મૂલ્યોની ગણતરી કરી શકાય છે અને હું ઘણું વધારે અને છ ગણતરી કરું છું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:03)

તેથી, સમસ્યાની ક્રેક્સ એ છે કે ત્યાં સબ સમસ્યાઓ છે જે વિવિધ સંદર્ભમાં પુનરાવર્તિત ગણતરીમાં ઉદ્ભવે છે, કારણ કે ફિબોનાકી 3 નું બંને મૂલ્ય મૂળ પેદા થાય છે જેને ફિબોનાકી 4 નું કહેવાય છે અને નેસ્ટેડ રિકર્સિવ કહેવાય છે ફિબોનાકી, તેથી મૂળ 5 નું ફિબોનાસી કહેવાયું અને નેસ્ટેડ પુનરાવર્તિત 4 નું ફિબોનાસી કહેવાતું. તેથી, અમારી પાસે ગણતરીનું આખું વૃક્ષ છે જે ડુપ્લિકેટ છે અને આ પ્રકારના કચરોને પુન: વિતરણ માટે લીધે, બધા ગણતરીઓના વૃક્ષો ઝડપથી વધે છે. તેથી, તમે વાસ્તવમાં સામાન્ય રીતે શોધી શકો છો, કે nth ફિબોનાકી નંબરની ગણતરી કરવા માટે, તમે ખરેખર એક પગલામાં કેટલાક ઘાતાંકીય રીતે કરો છો. તેમછતાં પણ, આપણે જોઈ શકીએ છીએ કે તમે માત્ર રેખાંશ સમયમાં nth ફિબોનાકી નંબરની ગણતરી કરી શકો છો, અમે હમણાં જ સૉર્ટ કર્યું છે, તમે હમણાં જ 0, 1, 2, 3, 4 અને આ રીતે ગણતરી કરેલ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:53)

તેથી, આની આસપાસ જવાનો એક રસ્તો એ છે કે તમે ક્યારેય ઉપ સમસ્યાને ફરીથી મૂલ્યાંકન ન કરો. તેથી, તમે તે કોષ્ટક બનાવો છો જેને કેટલીકવાર મેમરી ટેબલ કહેવામાં આવે છે અને તે કોષ્ટક શું છે તે ફક્ત તે દરેક મૂલ્યનો ટ્રૅક રાખે છે જેની તમે પહેલા ફંક્શનની ગણતરી કરી છે. તેથી, જાવા જેવી ભાષામાં તે જ જોવા મળે છે, તે હેશ નકશો અથવા પાયથોન જેવી ભાષા હોઈ શકે છે, તે શબ્દકોશ હોઈ શકે છે. દર વખતે જ્યારે તમે મૂલ્ય x પર ફોન કરો છો, ત્યારે તમે આ કોષ્ટકમાં ફક્ત એક્સ કૉમા FX સંગ્રહિત કરો છો. તેથી, તમે ઉપર જોઈ શકો છો અને જોઈ શકો છો કે અમે પહેલાથી તેની ગણતરી કરી છે. તેથી, તેને સંસ્મરણાત્મક કહેવામાં આવે છે. તેથી, આ શબ્દ સંસ્મરણાત્મક શબ્દ મેમોમાંથી આવે છે, જેમાં કેટલાક થોડા થાય છે. તેથી, મેમો એ એક નોંધ છે જે હજુ પણ છે કે તમે કંઈક બીજું કે કંઈક છે. તેથી, સંસ્મરણાત્મક, મેમો ટેબલ અથવા મેમરી કોષ્ટક તમને યાદ કરાવે છે કે તમારા ગણતરીના સમયનું મૂલ્ય પહેલેથી જ ગણતરી સાથે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:50)

તો, સંસ્મરણાત્મક કેવી રીતે કાર્ય કરે છે? તેથી, આપણી પાસે આ મેમો કોષ્ટક છે, તેથી આ આપણું મેમો ટેબલ છે, તેથી આ મેમો ટેબલમાં તે માત્ર એક કોષ્ટક છે જ્યાં આપણે k ની અલગ કિંમતને ફિટ કરીએ છીએ અને fib k એ આપણે તેનું ગણતરી કરીએ છીએ. અમે ધારી લીધું છે કે આપણે કંઈપણ ગણતરી કરી નથી, આપણે બેઝ કેસ પણ નથી જાણતા, અમે ફક્ત પુનરાવર્તિત વ્યાખ્યા લાગુ કરવા જઈ રહ્યા છીએ. પરંતુ, દર વખતે જ્યારે આપણે કંઈક ગણતરી કરીએ છીએ, ત્યારે આપણે વારંવાર ગણતરી કરતા પહેલા આપણે કોષ્ટકને જોશું અને જો આપણે વારંવાર ગણતરી કરીશું તો, આપણે કોષ્ટકના દરેક નવા ગણતરીના પાછલા ભાગને સંગ્રહિત કરીશું. તેથી, ચાલો આપણે પહેલાથી શરૂ કરીએ કે આપણે 5 ની ફિબોનાકીનું કમ્પ્યુટિંગ શરૂ કરવાનું શરૂ કર્યું. તેથી, પુનરાવર્તિત વ્યાખ્યાઓ કહે છે કે, આપણે 4 અને 3 ની ફિબોનાકીને કૉલ કરીશું. હવે, અમે 4 પર જઈએ છીએ, જ્યાં માત્ર રિઝર્ઝન જમણી બાજુથી ડાબે છે, તેથી 4 3 અને 2 ને કૉલ કરશે, 3 2 અને 1 કૉલ કરશે, 1 અને 0 ને કૉલ કરશે, આ બિંદુએ આપણે પ્રથમ વખત ફિબોનાકી 1 નું મૂલ્યાંકન કરવાનો પ્રયાસ કરીશું. જ્યારે, અમે પ્રથમ વખત ફિબોનાકી 1 નું મૂલ્યાંકન કરીએ છીએ, ત્યારે અમને બેઝ કેસ મળે છે અને તે આપણને જણાવે છે કે આ મૂલ્ય 1 છે. તેથી, અમે આ મૂલ્ય પરત કરીશું, પણ અમે તેને કોષ્ટકમાં સંગ્રહિત કરીશું. તેથી, હવે, આપણે ટેબલમાં પહેલી એન્ટ્રી બનાવી છે, 1 નું ફિબોનાસી 1 છે, તો આપણે 0 ની ફિબોનાસી જુઓ અને ફરીથી, તે એક નવું મૂલ્ય છે,

આપણને તેને મૂળ કેસમાંથી મળ્યું છે, પરંતુ આપણે ક્યારેય ગણતરી કરેલ નથી પહેલાં તેથી, હવે, આપણે બેઝ કેસની ગણતરી કરીને તેને ગણવામાં આવે છે, આપણે કોષ્ટકની બાજુમાં એન્ટ્રી મૂકીએ છીએ, ફિબોનાસી 0 ની ચિન્હ. તેથી, આની સાથે આપણે પહેલા ફિબોનાકી 2 મેળવ્યું છે. તેથી, આ બિંદુ સુધી, આપણે કંઈપણ બચાવ્યું નથી, સિવાય કે આપણે હવે પણ છે, કારણ કે આપણે 2 ની ગણતરી કરેલ ફિબોનાકી હોઈએ, આપણે ઉમેર્યું છે, આપણે નવું મૂલ્ય કર્યું છે. ગણતરી કરી અમે તેને કોષ્ટકમાં મુક્યા. હવે, 3 ની ફિબોનાસી સાથે ચાલુ રાખવા માટે, મને 1 ની ફિબોનાસી પાછા જવાની જરૂર છે, અલબત્ત, તે મૂળ કેસ છે, પરંતુ મારી ટેબલ વસ્તુ અમને શું કહે છે, તમને કહે છે કે તમારે પહેલા કોષ્ટકને જોઈએ છે. તેથી, અમે બંને ટેબલ પર આશા રાખીએ છીએ કે તે કેટલાક કાર્યક્ષમ રીતે સંગઠિત છે, પરંતુ સૌથી ખરાબ કિસ્સામાં, આપણે નૈતિક રીતે તેને સ્કેન કરીએ, આપણે જોયું છે કે મેં ક્યારેય પહેલા ફિબોનાકી 1 કર્યું છે, આ મેં ફિબોનાકી 1 નું કર્યું છે. તેથી, તે મૂળ કેસને બોલાવ્યા વગર તે મૂલ્યને જોશે, આપણે ફંક્શન્સ વ્યાખ્યા પર ધ્યાન આપવું પડશે અને માત્ર 1 નું ફિબોનાસી પરત કરવું જોઈએ. તેથી, અમે તેને નારંગી ફેરવીને સૂચવ્યું છે કે આ કૉલ ખરેખર ટેબલમાંથી પાછો આવ્યો હતો , તે ખરેખર જરૂરી ગણતરીમાં છે. તેથી, હવે, તે 1 અને 1 ધરાવે છે, 3 નું ફિબોનાસી હવે 2 છે. તેથી, હવે મને 4 ની ફિબોનાસી સાથે ચાલુ રાખવું પડશે. તેથી, હવે મને ફરીથી ફિબોનાકી 2 નું કૉલ કરવું પડશે. તેથી, હવે તે નીચે જશે અને કહેશે કે મેં ક્યારેય ફિબોનાકી 2 નું જોયું છે અને કહેવું છે કે હું અને તેથી, ફરીથી ગણતરી કર્યા વિના, તે આ મેમો ટેબલને જુએ છે અને ફરીથી આપણે તેને નારંગી મૂલ્ય તરીકે ચિલ્લિનત કરીએ છીએ, કારણ કે તેમાંથી આવ્યું છે કોષ્ટક, તે વેલ્યુ 1 આપે છે જે આપણે કોષ્ટકમાં સંગ્રહિત કર્યું છે. તેથી, હવે મેં આ ગણતરી કરી છે, મને 4 ની ફિબોનાસી મળી છે. તેથી, આપણે 4 ની ફિબોનાકીની ગણતરી પુનરાવર્તિત વ્યાખ્યાથી 3 થાય છે અને ત્યાં ફરીથી આપણે તેને કોષ્ટકમાં સંગ્રહિત કરીએ છીએ. તેથી, અમે ગણતરી કરીએ છીએ તે દરેક નવા મૂલ્ય, અમે તે કોષ્ટકમાં સંગ્રહિત કરીએ છીએ જે તે ફરીથી ઉપયોગ કરવામાં સક્ષમ થઈ શકે છે અથવા નહીં ખબર નથી, પરંતુ જો આપણે ફરીથી તેનો ઉપયોગ કરવો હોય તો, તે મેળવવું સારું છે. તેથી, આપણે ભવિષ્યમાં ઉપયોગ માટે આપણે જે કંઈપણ ગણતરી કરી છે તે બધું જ આંખે મૂકીએ. હવે, આપણે 5 ની ટોચની ફિબોનાકી પર પાછા આવીએ છીએ, આપણને ડાબી બંદૂકો મળી છે, આપણને બંદૂકોની જરૂર છે, આપણે 3 ની ફિબોનાસીમાં જઈએ છીએ, અગાઉ આપણે 3 ની ફિબોનાકી મેળવવા માટે નીચે પ્રમાણે 3 સંપૂર્ણ રીતે પુનર્નિર્માણ કર્યું હતું, પરંતુ આ વખતે આપણે વધુ સ્માર્ટ છીએ, આપણે ટેબલ નીચે જઈએ છીએ અને આપણને લાગે છે કે 3 ની ગણતરી કરવામાં આવી છે અને મૂલ્ય 2 છે, તેથી આપણને આ વેલ્યુ મળે છે. અને હવે, આપણી પાસે 5 ની ફિબોનાસી માટે જરૂરી જવાબો છે. અને તેથી આપણે 5 ની ફાઈબોનાસી મેળવીએ 5 અને ફરીથી, આપણે તેને કોષ્ટકમાં મૂકીએ, જો કે આ ચોક્કસ કિસ્સામાં, આપણે યુઝર પાસે ગણતરી કરવા માટે જઈ રહ્યા નથી, પરંતુ તે મોટી ગણતરીઓનો એક ભાગ હોઈ શકે છે, તેથી અમે ફક્ત આ કરવાનું ચાલુ રાખ્યો. તેથી, તમે આમાં શું જોઈ શકો છો તે છે કે ગણતરીનું વૃક્ષ જે ઘાતાંકનો ઉપયોગ કરે છે તે હવે રેખીય છે. શા માટે તે રેખીય છે? કારણ કે, દરેક મૂલ્ય વાદળી રંગમાં દેખાય છે અને દરેક મૂલ્ય જે વાદળી રંગમાં નથી તે કોષ્ટકમાં એક નજર છે, તેથી, તેની કિંમત ઓછી નથી, તેથી આપણે હવે બનાવેલ છે, તેથી આ વધુ અથવા ઓછા આ મેમોઈઝ્ડ ફિબોનાકી આપણે શું કરીએ છીએ, ભલે આપણે તે અલગ રીતે કરીએ, જ્યારે આપણે વાસ્તવમાં હાથ દ્વારા ગણતરી કરીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:01)

તો, ફક્ત એ જોવા માટે કે મેમોઈઝ્ડ ફિબોનાસી શું લાગે છે? તો, લીલો કોડ એ જે આપણે અગાઉ લખ્યો છે, n ના રેસ, n એ 0 અથવા 1 છે, તો n ને વેલ્યુ સુયોજિત કરો, else n minus 1 ની fib પર value, n minus 2 ની fib અને return

ને સુયોજિત કરો. તેથી, હવે આપણે મેમો બિટ્સ રજૂ કર્યા છે જે લાલ છે. તેથી, અમારી પાસે એક કોષ્ટક છે જેને આપણે ફિબ કોષ્ટક કહીએ છીએ, જેની મૂલ્યો સ્થિતિ દ્વારા અનુક્રમિત થાય છે. તેથી, જો હું ગણતરી કરું છુ, તો 6 કોષ્ટક ફિબ કરવા જઈ રહ્યો છે. તેથી, જ્યારે મને ભરવા માટે દલીલ મળે છે, ત્યારે હું જે પહેલી વસ્તુ કરું છું તે એ છે કે મને લાગે છે કે ફીબ ટેબલ તેના માટે એન્ટ્રી છે કે નહીં. તેથી, જો n ની Fib કોષ્ટક અસ્તિત્વમાં હોય તો તે જે પણ કહે છે તે વળતર આપે છે, જો તે અસ્તિત્વમાં નથી, તો આપણે તેને ગણતરી કરીએ તે પછી, આપણે તેને ગણતરી કરીએ તે પહેલાં, આપણે તે કંઈપણ કરવા પહેલા, આપણે તેને કોષ્ટકમાં પાછું મુકીશું. તેથી, તે આગલી વખતે આપણે તેનું ગણતરી કરીશું નહીં. તેથી, દરેક નવા મૂલ્યને તે ટેબલમાં પાછું મુકવામાં આવે છે અને પછી, અમે તેને પરત કરીએ છીએ. તેથી, જ્યારે આપણે અહીં મૂલ્ય પરત કરીએ છીએ, તે પ્રથમ વખત ગણવામાં આવ્યું છે, પરંતુ તે કોષ્ટકમાં સ્ટોર કરવામાં આવ્યું છે. તેથી, આગલી વખતે, હું તે જ મેળવીશ, હું અહીં આવીશ અને હું અસ્તિત્વમાં આવીશ તે જ છે. તેથી, આ આપણું સંસ્મરણીય ફિબોનાસી છે, તે શરૂઆતમાં કોષ્ટક માટે ખૂબ સરળ તપાસ ધરાવે છે અને જ્યારે તમે મૂલ્યની ગણતરી કરો છો, તે તેને પાછું મૂકે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:15)

તો, આ ખૂબ જ સરળ વસ્તુ છે જે તમે કરી શકો છો, તમે કોઈપણ કાર્યો માટે આ કરી શકો છો. તેથી, ધારો કે તમારી પાસે કેટલાક રિકર્સિવ ફંક્શન અથવા ત્રણ દલીલો સાથે ઈન્ડેક્ટિવ ફંક્શન છે, તો પછી તમારી પાસે ફક્ત ત્રણ સૂચકાંકોવાળી એક કોષ્ટક છે. તેથી, ચાલો આપણે itd af ટેબલ પર કૉલ કરીએ, તેથી કમ્પ્યુટિંગ f. તેથી, તમારી પાસે પેટા સમસ્યાઓમાંથી મૂલ્યની ગણતરી કરવાની કેટલીક પ્રગત રીત છે. તેથી, સૌ પ્રથમ તમે આ એક્સ, વાય, ઝેડ પહેલા ક્યારેય જોયું છે, જો તેમ હોય તો તેને પાછું કરો. નહિંતર, આ આપેલા x, y અને z માટેના મૂલ્યને વ્યાખ્યાયિત કરતી સમસ્યાના ઈન્ડેક્ટીવ માળખાથી પુનરાવર્તિત વ્યાખ્યાના સંદર્ભમાં નવી મૂલ્યની ગણતરી કરો, તેને ફરીથી કોષ્ટકમાં મૂકો, કે તમે ક્યારેય સ્પષ્ટપણે ગણતરી કરી નથી અને તમે પાછા આવો છો. તેથી, આ એક જિનેસિક્સ કી છે જે તમે લખી શકો છો તે કોઈપણ પુનરાવર્તિત ફંક્શન પર લાગુ થઈ શકે છે, એક માત્ર વસ્તુ જે તમે કરો છો તે છે કે તમારે ખાતરી કરવી પડશે કે તમે યોગ્ય દેખાવ સાથે ટેબલને ડિઝાઈન કરી શકો છો અને તેને જોશો . કારણ કે, જો ટેબલના મૂલ્યને જોવામાં લાંબા સમય લાગે, તો તે ખોવાઈ જાય છે. તો, જો તમે તેને અરેના અંતમાં બનાવી શકો છો તો તે એક આદર્શ છે, તેથી આ યાદશક્તિ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:17)

તેથી, છેલ્લા શબ્દોમાં આપણે જે પરિભાષા રજૂ કરીએ છીએ તે ડાયનેમિક પ્રોગ્રામિંગ છે. તેથી, ડાયનેમિક પ્રોગ્રામિંગનું મૂલ્યાંકન કરવાના પુનરાવર્તિત ભાગને દૂર કરવાનો પ્રયાસ તે કયા ડાયનેમિક પ્રોગ્રામિંગ કરે છે. તેથી, યાદશક્તિમાં આપણે શું કરીએ છીએ, આપણે સામાન્ય રીતે જેમ જેમ આપણે સામાન્ય રીતે પરિપૂર્ણ રીતે વ્યાખ્યાયિત વ્યાખ્યાનું મૂલ્યાંકન કરીએ છીએ, તમે માત્ર એક જ વસ્તુ તે કોષ્ટક રાખો છો જે સમાન વસ્તુને બે વાર ગણતરી કરવાથી ટાળવામાં મદદ કરે છે. ડાયનેમિક પ્રોગ્રામિંગમાં, તમે અપેક્ષા કરો કે કોષ્ટક જેવો હોવો જોઈએ અને કોષ્ટકનાં મૂલ્યો કેવી રીતે નિર્ભર છે. તેથી, આપણે 5 ની ફિબોનાકીનું કમ્પ્યુટિંગ કર્યું છે, ત્યારબાદ ફિબોનાસીના કેટલાક સરળ વિશ્લેષણ દ્વારા, આપણે જાણીએ છીએ કે જો ફિબોનાકીની મને કંઈપણની જરૂર હોય તો તેને નાની વસ્તુઓની જરૂર છે અને નાની વસ્તુ તમને 0. નું ફિબોનાકી મળી શકે છે. તેથી 5 ના ફિબોનાસી, અમે તરત જ જાણીએ છીએ કે ઉપ સમસ્યા જે અમને અથવા

ફિબોનાકીની 0 અને ફિબોનાકી વચ્ચેની બધી પેટા સમસ્યાઓમાં રસ હોઈ શકે છે. 5 નું ફિબોનાકી જરૂરી નથી જઈ રહ્યું છે 6, 7, 8 ચિલ્ડનની ફિબોનાસી કૉલ કરી શકાય છે નકારાત્મક સંખ્યાના ફિબોનાસી કારણ કે તે વ્યાખ્યાયિત નથી. તેથી, ત્યાં એક ખૂબ નાનો અનંત સેટ અપ મૂલ્યો છે જેને આપણે ધ્યાનમાં લેવાની જરૂર છે. હવે, આગળનું અવલોકન એ છે કે આપણે નિર્ભરતાને સમસ્યાની રચનાથી ગણતરી કરી શકીએ છીએ; આ પ્રેરક વ્યાખ્યા છે. તેથી, 4 ની ફિબોનાસી એ બરાબર છે, મારો અર્થ એ છે કે n એ મીન માઈનસ 1 પ્લસ મી ઓછા 2 છે. તેથી, 5 4 અને 3 પર આધાર રાખે છે. તેથી, હું કહું છું કે, 5 નું ગણતરી કરવા માટે ક્રમમાં છે, હું પ્રથમ ગણતરી 4 અને 3 ની. તેથી, તે શું છે તેના પર આધાર રાખે તેમાંથી હું તીર દોરીશ. તો, 5, 5 પર આધાર રાખે છે 5, 5 પર આધાર રાખે છે. તેથી, મેં 4 થી 3 અને 4 થી 5 અને 3 થી 5 સુધી એરે મૂકી છે, ત્યાં એક નિર્ભરતા છે તે મોકલો. બદલામાં 4, 1, 3, 2 અને 3 આધાર રાખે છે અને 2 અને 1 અને 2 1 અને 0 પર આધાર રાખે છે. અને અલબત્ત, 1 અને 0 આધાર કેસ છે અને તે કંઈપણ પર આધારિત નથી. તેથી, ડિપેન્ડન્સીઝ ડગ બનાવશે, તેમનું ફોર્મ શા માટે ડગ લાગે છે, બરાબર ચોંટાડવામાં આવે છે તે સ્પષ્ટ છે, કારણ કે તેમાં ચક્રીય અવલંબન છે, તો તમે અન્યની તુલનામાં એકની ગણતરી કરી શકતા નથી. તેથી, તે કોઈ પણ પ્રકારની નિર્ભરતામાં ખૂબ જ કુદરતી વસ્તુ છે, પછી ભલે આપણે જે જોયું તે પાસું વચ્ચેની અવલંબન હોય, ત્યાં ઘણા લોકો હોય તો, બીજું કંઈક કરવું તે ત્યાં આગેવાની લે છે, તે કહે છે કે, જો હું ન કરી શકું તો 2 ની ફિબોનાસીની ગણતરી કરો, હું 3 ની ફિબોનાકીની ગણતરી કરી શકતો નથી, કારણ કે પહેલા કરવામાં આવ્યું છે. હવે, હું આ નિર્ભરતાને જાણું છું, હવે હું તેને કોઈ પણ સ્થાનિક સ્તરે ગણતરી કરી શકું છું, જેમ કે જ્યારે હું ગણતરી કરવા માટે મૂલ્ય સુધી પહોંચું છું, ત્યારે તે જે બધું તેના પર નિર્ભર છે તે 4 ધોરણ છે. તેથી, ઉદાહરણ તરીકે અહીં કુદરતી લૉગ આઉટ કરો અલબત્ત, 0, 1, 2, 3, 4, 5, કારણ કે આ બંને આવનારી વયના તત્વો નથી. તો, હું ક્યાં તો સાથી સ્થાનાંતરિત ક્રમમાં શરુ કરી શકું જે 0 અથવા તે 1. તેથી, ચાલો આપણે 0 ની શરૂઆતથી ધારીએ, તેથી મને ખબર છે કે 0 નું ફિબોનાસી કંઈક પર આધાર રાખે છે, તે એક મૂળ કેસમાં ફેલાય છે, તેવી જ રીતે હું જાણું છું કે ફિબોનાકી ના 1 કશું પર આધાર રાખે છે. તેથી, તે બેઝ કેસ છે, જો હું તે માં છું, તો આ બિંદુએ આ બંને કર્યું છે, કારણ કે આ બંને કર્યું છે, આ કિનારીઓ જતી રહી છે, તેથી ત્યાં તે નિર્દેશાંકિત નથી. 2. તેથી, આ એક ટોપોલોજિકલ સૉર્ટ છે, મૂળભૂત ટોપોલોજિક સમૂહ પણ છે. તેથી, ત્રણમાં હજુ પણ 2 નું નિર્ભરતા છે, પરંતુ 2 ને 1 અને 0 બંને માટે વધુ નિર્ભરતા છે, તેથી હું 2 ની ફિબોનાકીને, પછી 3, 4 અને 5 ના ફિબોનાસીને ફાયબર કરી શકું છું અને આ બરાબર છે જે આપણે પહેલા કર્યું હતું, જ્યારે અમે હાથ દ્વારા ફિબોનાકીની ગણતરી કરવાનો પ્રયાસ કરીએ છીએ. તેથી, તમે વાસ્તવમાં એક્ઝિક્યુટ કરી રહ્યા છો, જ્યારે તમે જમણી બાજુથી ફિબોનાસી નંબરો લખી રહ્યા છો, જ્યાં આગળની તરફ જોઈ રહ્યા છે અને પછીની ગણતરી કરી રહ્યા છે તે વાસ્તવમાં ડાયનેમિક પ્રોગ્રામ કહેવામાં આવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:11)

તેથી, ફિબોનાકી ફંક્શન માટે ડાયનેમિક પ્રોગ્રામિંગમાં માત્ર તે જ રીતે કોષ્ટક ભરવામાં આવે છે, અમે કિંમત 0 અને 1 ને શરૂ કરીશું, પછી 2 થી n માટે, જો કોર્સ એન કરતાં ઓછું હશે આપણે આ સમીકરણને 2 થી એન માટે નહીં લઈશું, અમે 2 ઓછા એન્ટ્રી દ્વારા i ઓછા 1 મી એન્ટ્રીમાંથી ith એન્ટ્રી ભરીશું, પરંતુ ત્યાં આ બે મૂલ્યો પહેલાં આપણે જોયું તે પહેલાં, તે પહેલાથી ભરાશે . તેથી, મેં પ્રારંભિકથી અંત સુધી આંખે ભરવા માટે નેટવર્ક તરીકે માંગને ભરીને એક રિકર્સિવ ગણતરીને ભરીને મારા મેમો કોષ્ટકને રૂપાંતરિત કર્યું છે, જે ડાયનેમિક પ્રોગ્રામ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:57)

તેથી, સારાંશ માટે, અમે બે વ્યૂહરચનાઓ જોયા છે જેમાં ઈન્ડક્ટિવ ફંક્શનો વધુ કાર્યક્ષમ ગણતરીઓ વધુ કાર્યક્ષમ બનાવવા માટે છે. પ્રથમ સ્મૃતિપત્ર છે. તેથી, જ્યારે પણ તે સબ સમસ્યામાં ફંક્શનના મૂલ્યની ગણતરી કરે છે ત્યારે સ્મૃતિપત્ર શું કરે છે, તે તેને ઊંડામાં મૂકે છે. અને તે કિંમતને ગણતરી કરવા માટે ફંક્શનને વારંવાર બોલાવે તે પહેલા, તે એ જ મૂલ્યને જોશે કે જેનું મૂલ્ય આપણે પહેલેથી જ સમાન મૂલ્ય પર ગણ્યું છે તે ક્યારેય બે ગણવામાં આવતું નથી. બીજી બાજુ ડાયનેમિક પ્રોગ્રામિંગમાં, આપણે સમસ્યાના માળખાને વિશ્લેષણ કરીએ છીએ, ઉપ સમસ્યાઓની ઓળખ કરી રહ્યા છીએ અને આપણે જાણીએ છીએ કે આ ઉપ સમસ્યા માળખું એક ગુંદરને સંતોષી શકે છે, ડિપેન્ડન્સીઝ એક ડેગ બનાવવી જ જોઈએ, કારણ કે એક ડગ ન હોવું જોઈએ, પછી આપણી પાસે ચક્ર હશે અને વસ્તુઓ કઈ પણ હલ કરી શકતી નથી. અને આ કરવાથી, આપણે ડાયનેમિક પ્રોગ્રામિંગમાંથી જે મેળવીએ છીએ તે તેનું મૂલ્યાંકન છે. તેથી, આ વાસ્તવમાં મોટી બચત છે, કારણ કે મોટા ભાગની પ્રોગ્રામિંગ ભાષાઓમાં રિકર્ઝન માટે છુપાવેલી કિંમત છે કારણ કે પ્રત્યેક ફંક્શન કોલ માટે ખરેખર અમુક ઓપરેટિંગ સિસ્ટમ અને પ્રોગ્રામિંગ ભાષા, અમલદારશાહી, કેટલાક વહીવટી કાર્યની આવશ્યકતા હોય છે, તેમને કેટલીક મેમરી જાહેર કરવા માટે લે છે સ્ટેક પર અને તેથી આગળ અને આ ડાયનેમિક પ્રોગ્રામ રીઝોલ્યુશનમાં સંપૂર્ણપણે ટાળી શકાય છે. તેથી, સ્મૃતિ દૂર કરવા માટે તે મોટી બચત છે, જો કે સંસ્મરણાત્મક આપેલ સમસ્યા માટે જરૂરી પુનરાવર્તિત કૉલ્સની મહત્તમ સંખ્યા આપશે, જો નહીં, તો બે વાર કૉલ કરવા જઈ રહ્યું છે. તેમાં હજી પણ રિકર્સિવ કૉલ્સ શામેલ હશે અને રીકર્સિવ કૉલ્સ તેમના પોતાના લેખમાં વિસ્તૃત હોઈ શકે છે. ઍલ્ગોરિધમ્સ માટે, અમે સામાન્ય રીતે ફંક્શન કૉલને એકમના ભાવના ઓપરેશન તરીકે વાંચીએ છીએ, પરંતુ વ્યવહારમાં, તે કેસ નથી. તેથી, ડાયનેમિક પ્રોગ્રામિંગ મોટી બચત હોઈ શકે છે, કારણ કે તે વાસ્તવમાં રૂપાંતરણ કરે છે, પુનરાવર્તિત વસ્તુને પુનરાવર્તિત યોજનામાં ઑપ્ટિમાઈઝ કરે છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 07
મોડ્યુલ - 03
લેક્ચર - 46
ગ્રીડ પાથ્સ

તેથી, કાર્યક્ષમ રીતે પુનરાવર્તિત કાર્યોની ગણતરી વિશેની અમારી ચર્ચા ચાલુ રાખીએ, ચાલો ગ્રીડ પાથની ગણતરી કરવાની સમસ્યાને ધ્યાનમાં લઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:09)

તેથી, આપણી પાસે ગ્રીડ, લંબચોરસ ગ્રીડ છે, જ્યાં આપણે નીચે ડાબા ખૂણે વૉકિંગ કરવાનું પ્રારંભ કરીએ છીએ, અને નિયમ એ છે કે આપણે ફક્ત ઉપર અથવા જમણી જઈ શકીએ છીએ. તેથી, આપણે તળિયેથી પ્રારંભ કરવા માંગીએ છીએ અને આપણે ઉપરના જમણે ખૂણે પહોંચવા માંગીએ છીએ. તેથી, આપણે કોઓર્ડિનેટ્સની સંખ્યા કરી શકીએ છીએ. તેથી, નીચે જમણે ખૂણે આપણે (0,0)કૉલ કરીએ છીએ. આ વિશિષ્ટ ગ્રીડ પાસે 5 કૉલમ અને 10 પંક્તિઓ છે. તેથી, ઉપરનો જમણો ખૂણો છે (5,10). અને આપણે જે પ્રશ્ન પૂછીએ છીએ તે છે કે કેટલા જુદા જુદા માર્ગો ચાલે છે, તે (0,0) થી (5,10) સુધીની છે. તેથી, અમે વિવિધ રીતે શું અર્થ છે. ઠીક છે, ઉદાહરણ તરીકે અહીં એક ગ્રીડ પાથ છે. આ વાદળી રેખા અમને (0,0),પછી જમણે, પછી ઉપર, પછી જમણી બાજુએ અને આગળ લઈ જાય છે. તેથી, આ ગ્રિડ સાથેના કિનારીઓના એક ખાસ અનુક્રમને ટોચની ડાબે ખૂણેથી ઉપર જમણી તરફ લઈ જાય છે. તેથી, આપણે અલબત્ત, એક અલગ પસંદ કરી શકીએ છીએ, જે આ વિશિષ્ટમાં લાલ અને વાદળી એક સંપૂર્ણપણે અસહિષ્ણુ છે. તેઓ કોઈ એક જ ધારનો ઉપયોગ કરતા નથી, પરંતુ સામાન્ય રીતે, હું એવા પાથ્સ ધરાવી શકું છું જે અન્ય લોકો સાથે ઓવરલેપ કરે છે. તેથી, આ પીળો અંશ વાદળી સાથે આંશિક રીતે ઓવરલેપ્સ કરે છે, અહીં, અને પછી તે અહીં લાલ રંગની સાથે ઓવરલેપ કરે છે, જમણી બાજુ. આપણે જાણવા માંગીએ છીએ કે કેટલા (0,0)થી (એમ,એન)સુધી આવા ઘણા જુદા પાથો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:28)

હવે, તે બહાર આવે છે કે આ સમસ્યા વાસ્તવમાં કોમ્બિનેટોનિક્સ ગણતરીમાં એક શાસ્ત્રીય સમસ્યા છે. તેથી, આનો વિશ્લેષણ કરવાનો માર્ગ એ જોવા માટે છે કે, જો હું તળિયેથી ઉપરની તરફ જવા માંગું છું, તો એક પરિમાણમાં, હું 0 થી 5 સુધી જવું જોઈએ. બીજા પરિમાણોમાં, મારે આવશ્યક છે 0 થી 10 ની ઉપર જાઓ. તેથી, તદ્દન, મારે 15 પગલાઓ જ બનાવવી જોઈએ. ત્યાં કોઈ પસંદગી નથી, મારે 15 સેગમેન્ટ્સ ચાલવું પડશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 01:51)

સામાન્ય રીતે, જો હું કેટલાક(એમ,એન)પર જાઉં છું, તો મારે એમ પ્લસ એન સેગમેન્ટ્સ જવું પડશે. હવે, જો મારી પાસે આ 15 સેગમેન્ટ્સ છે અને હું (5,10)જાઉં છું, તો તેનો અર્થ છે, મને જમણી બાજુ જવું પડશે, 5 વખત, બરાબર, અને મારે 10 વાર વધવું પડશે. હવે, જે અનુક્રમમાં હું આ જમણી બાજુ અને અપ્સ કરું છું તે નક્કી કરે છે કે હું કયા પાથને લઈ શકું છું. પરંતુ દરેક પાથમાં બરાબર 5 જમણી ચાલ હશે અને બરાબર 10 ઉપર ચાલ ચાલશે. તો, 10 ઉપર ચાલ અને 5 જમણી ચાલ ચાલશે. તેથી, હવે જો હું આ કરું અને હું આને એકંદર વસ્તુ કહું છું કે આ મારો પહેલો પગલું છે, આ મારો બીજો ચાલ છે, આ મારો ત્રીજો ચાલ છે અને તેથી, 15 મી ચાલ સુધી. પછી જો હું તમને કહું કે તમે જમણી બાજુએ ગયા છો, તો હું પાંચ સ્થાનો પર, આ સ્થાનો પર જમણી બાજુએ ખસેડ્યો, પછી આપમેળે બાકીની સ્થિતિઓ પર ખસેડવું પડશે કારણ કે મને જમણી બાજુ 5 અને જમણી બાજુ 5 જમણે અને 10 અપ કરવું પડશે. તેથી, હું જે રસ્તો લઈ રહ્યો છું તે નક્કી કરવા માટે માત્ર મને જરુર છે કે કુલ 15 જમણી બાજુમાં 5 જમણી બાજુ ચાલવાની સ્થિતિને ઠીક છે. તેથી, 15 સ્થિતિઓમાં હું 5 પસંદ કરું છું. આ સામાન્ય રીતે 15 પસંદ 5 તરીકે લખાય છે, સાચું. તેથી, આ એક ખૂબ જ પ્રમાણભૂત, સંમિશ્રણ વસ્તુ છે, 15 પસંદ કરો 5 n પસંદ કરો n એ ફેક્ટોરિયલ છે, જે n માયન સ કે કે ફેક્ટોરિયલમાં, કે ફેક્ટોરિઅલ દ્વારા વિભાજિત છે. તેથી, 15 ફેક્ટોરિઅલ 10 ફેક્ટોરિઅલ વખત 5 ફેક્ટોરીયલ દ્વારા વિભાજિત થાય છે, તે 3003

((સમયનોસંદર્ભ: 03:24))

થાય છે. તેથી, આ વિશિષ્ટ ગ્રીડ માટે 3003 માર્ગો છે. હવે, અલબત્ત, યોગ્ય સ્થાનો પસંદ કરવાને બદલે હું તમને 10 સ્થાનો પણ જણાવી શકું છું જ્યાં મેં ખસેડ્યું છે, જે 5 ખુલ્લા સ્થાનો છોડે છે જ્યાં હું

((સમયનોસંદર્ભ: 03:39)).

તેથી, આ આપણને 15 પસંદ કરશે, અને તેથી તે કોઈ સંયોગ નથી કે 15 પસંદ કરો 10 અને 15 પસંદ કરો 5 તે જ અભિવ્યક્તિ છે. તેથી, જો તમે તેને ગણતરી કરવા પસંદ કરો છો, 15 તો 5 અથવા 15 પસંદ કરો 10 પસંદ કરો, તે કોઈ વાંધો નથી. તેથી, સામાન્ય રીતે તે m પ્લસ n હશે, m અથવા m plus n પસંદ કરો n ને પસંદ કરો જ્યાં હું (0,0)થી ઉપર (એમ,એન),જમણી બાજુએ જાઉં છું. મારે જમણી ચાલ અને એન અપ ખસેડવાનું છે. તેથી, મારે એમ પ્લસ એન કુલ ચાલમાંથી એમ પસંદ કરવું પડશે અથવા એમ વત્તા એન ની બહાર એમ પસંદ કરવું પડશે, તે બંને મને સમાન અભિવ્યક્તિ આપશે, કારણ કે, n પસંદ કરો, n પસંદ કરો કે n એ ફેક્ટોરિઅલ કે ફેક્ટોરિઅલ n છે. માઈનસ કે. તેથી, જો તમે હમણાં જ કે અને n માઈનસ k નું વિનિમય કરો છો તો તમને સમાન અભિવ્યક્તિ મળે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:31)

તેથી, તે બધું બરાબર છે, પરંતુ ઉદાહરણ તરીકે, જો આ સંપૂર્ણ ગ્રીડ નથી. ધારો કે અમારી પાસે કેટલાક અંતરાયો છે, જે અવરોધિત છે. તેથી, આ ચોક્કસ કિસ્સામાં, જો તમે આ આંતરછેદને જુઓ છો, જે (2,4)છે, તો તે 1, 2 અને પછી 1, 2, 3, 4 છે. તેથી, આપણે તે સૂચવવા માટે એક કાળો ચિહ્ન મુક્યો છે. ક્ષણ માટે, તમે તેના દ્વારા પસાર કરી શકતા નથી, અધિકાર. તેથી, (2,4)
દ્વારા પસાર થતો કોઈપણ ભાગ,(0,0)થી (5,10)સુધીનો માન્ય પાથ વચ્ચે ગણાશે નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:57)

તો, આ વાદળી માર્ગ જે આપણે આ આંતરછેદથી પસાર થતાં પહેલાં દોરો હતો. તેથી, આ પાથ હવે માન્ય પાથ નથી. લાલ પાથ બરાબર છે, કારણ કે તે બાયપાસ કરે છે, પરંતુ પીળી પાથ દુર્ભાગ્યે પણ આ આંતરછેદથી પસાર થાય છે. તેથી, ત્રણ રસ્તાઓ, જે આપણે અત્યાર સુધી જોયા હતા, બે વાસ્તવમાં પસાર થતા નથી. તેથી, હવે આ પ્રશ્ન છે કે, 3003 પાથોમાંથી જે અમે કહ્યું હતું કે,(0,0)
થી (5,10),જો તમે આ આંતરછેદમાંથી પસાર થવાની મંજૂરી ન આપો તો તેમાંના કેટલા હજી પણ માન્ય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:26)

તેથી, તે તેના પર વળે છે તે વાસ્તવમાં એક સંયુક્ત જોડાણ ધરાવે છે. તેથી, તમે શું કહી શકો છો કે અહીંથી જવા માટે અને જો હું બ્લોક આંતરછેદ ટાળવા માંગુ છું. તેથી, ચાલો જોઈએ કે તેમાંથી પસાર થવા માટે કેટલા માર્ગો છે અને તેને દૂર કરો, જમણી બાજુ. તો, હું શું કરીશ, હું કહું છું કે વર્તમાન બ્લોક આંતરછેદ મારફતે જે પણ તે પસાર થાય છે તે (0,0)થી 92, 4 નો માર્ગ છે (2,4)થી (5,10)કારણ કે તે (2,4)પસાર થાય છે. તેથી, આપણે આનાથી નાના ગિદ

(0,0)

થી

(2,4),

જમણી તરફ વિચારી શકીએ છીએ. તેથી, જો આપણે હલ કરીએ, તો આ સામાન્ય એમ પ્લસ એનએમ પ્લસ n પસંદ કરો. તેથી, મને 6 પસંદ થાય છે તેથી, મને ત્યાં જવાના 15 માર્ગો છે

(0,0)

થી

(2,4).

તેવી જ રીતે, જો હું આ ગ્રીડનો વિચાર કરું, તો આ 3 છે અને આ 6 છે કારણ કે તે 10 અને 5 હતું, મેં અનુક્રમે 2 અને 4 કર્યું છે. તેથી, 2 પછી હજી પણ 3 આડી બાકી છે; 4 પછી હજુ પણ 6 ડાબી બાજુએ છે. તેથી, આ નવું

(0,0)

થી

(3,6),

જમણી તરફ જવા જેવું છે. તેથી, મને 6 વત્તા 3 પસંદ 3 મળે છે અને આ 84 થાય છે. હવે, કોઈપણ ભાગ, જે કોઈપણ ભાગ પછી 2 થી 4 ની નીચે આવે છે, તે ત્યાંથી ટોચ પર જાય છે તે એક માન્ય માર્ગ છે જે 2 થી પસાર થાય છે તેથી 4, હું આ બે ક્રમાંકોને ગુણાકાર કરું છું, મારે 15 ગુણ્યા 84 અને ફરીથી 1260 લેવું પડશે. તો, આ સંખ્યાઓની સંખ્યા છે, જે પસાર થઈ રહી છે

(2,4),પણ

મને નથી લાગતું કે પાથ પસાર થાય

(2,4).હું કહું છું, 2 થી 4 ની તરફ જવાના રસ્તાઓને મંજૂરી નથી. તેથી, મારે આ બધા ભાગોને અમાન્ય હોવા જોઈએ. તેથી, હું મૂળ 3003 બાદબાકી લઈશ. તેથી, હું 3003 લે છે અને હું 1260 બાદ કરું છું અને મને 1743 મળે છે, બરાબર. તેથી, સંયોજનની કસરતને આગળના પગલામાં લઈ જવું. મૂળમાંથી જમણી બાજુએ આપેલ બિંદુ સુધી કેટલા પાથ જાય છે તે જોવું જોઈએ જો એક પાથ એક પોઝિશન અવરોધિત હોય.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:29)

તેથી, આ અપૂર્ણતા વધુ જટિલ હોઈ શકે છે. મારી પાસે બે સ્થિતિ અવરોધિત થઈ શકે છે, જમણી બાજુ. આ કિસ્સામાં, વાદળી, પીળા અને લાલ પાથ બધા અમાન્ય છે કારણ કે લાલ પાથ અવરોધિત થવા માટે પણ થાય છે. તેથી, હવે મેં

(2,4)અને

(4,4)પર

પણ અવરોધિત કર્યું છે. તેથી, જો હું બેમાંથી પસાર થતાં બધા પાથોને ગણું છું,

(2,4),જે મેં પહેલા કર્યું છે, હું આ બાદ કરી શકું છું. એ જ રીતે, હું (4,4)પસાર થતા બધા પાથોની ગણતરી કરી શકું છું અને તે બાદ કરી શકું છું. પરંતુ હવે શું થાય છે, આ યલો પાથ બે વખત બાદ કરવામાં આવે છે કારણ કે તે (2,4)પસાર થતા રસ્તાઓ અને (4,4)પસાર થતા પાથનો ભાગ છે, જમણી બાજુ. તેથી, મેં આકસ્મિક રીતે મારા કુલથી બે વાર દૂર કર્યું છે, તેથી મારે તેને પાછું મૂકવું પડશે. તમારે હવે તે પાથોની ગણતરી કરવી પડશે, જે આંતરછેદ દ્વારા પસાર થાય છે અને તેમને પાછા ઉમેરશે અને આ સંયોજનને સમાવવા અને બાકાત કહેવામાં આવે છે. જ્યારે તમે સેટ કરો છો ત્યારે તમે વેન ડાયાગ્રામ કરો છો ત્યારે પણ ઘણીવાર તે મળે છે. જો તમે શોધવા માંગતા હો, તો હવે તમે કહો કે, કેટલા લોકો છે, એક તત્વ કેટલા સેટ છે.

((સમય 08:33 નો સંદર્ભ લો))

આંતરછેદ, તમારી પાસે કેટલા સેટ્સ છે, તમે જાણો છો, તમે છો, વિદ્યાર્થીઓ ત્રણ વિષયો, ઇંગલિશ, ઇતિહાસ અને ભૌતિકવિજ્ઞાન લઈ રહ્યા છે, અને પછી ઘણા ઇંગલિશ અને ઇતિહાસ લેતા, ઘણા ઇતિહાસ અને ભૌતિકશાસ્ત્ર લઈ અને તેથી. તેથી, જ્યારે તમે તે પ્રકારની ગણના કરો છો ત્યારે તમે આ સમાધાન બાકાત બરાબર કરો છો. તેથી, જેમ આપણે વધુને વધુ અવ્યવસ્થિત ગ્રિડ મેળવે છે, આ સંયોજન પ્રશ્ન આ રીતે હલ કરવા માટે વધુ જટિલ બને છે.

(સ્લાઇડટાઇમ નો સંદર્ભ લો: 09:00)

તેથી, ચાલો આપણે વધુ સારા ઉકેલ માટે જોઈએ. તેથી, તમે અનુમાન કરી શકો છો, કેમ કે અમે આ પ્રકારના ઇન્ટરેક્ટિવ ફોર્મ્યુલેશન્સ અને રીકર્સિવ પ્રોગ્રામ્સ તરફ જોઈ રહ્યા છીએ, જે આપણે ખરેખર શોધી રહ્યા છીએ તે છે ગ્રીડ પાથની પ્રેરણાત્મક રચના. તેથી, ચાલો આપણે પોતાને નીચેના પ્રશ્ન પૂછીએ. હું ગ્રીડ પર બિંદુ

(i,j)

કેવી રીતે મેળવી શકું? તેથી, હું માનું છું કે, આપણી માળખું આપવામાં આવ્યું છે, જે છે, કે આપણે જમણી બાજુ જઈ શકીએ છીએ અથવા આપણે ઉપર જઈ શકીએ છીએ, અહીં માત્ર બે રીતો છે જે હું અહીં આવી શકું છું. હું ક્યાં તો મારા ડાબા પર પાડોશી પાસેથી જઈ શકું છું. તેથી,

(i,j માઈનસ 1)

થી હું એક પગલું યોગ્ય બનાવી શકું છું અને

(i,j)

અથવા

(i માઈનસ 1, j)

પર આવી શકું છું, હું એકવાર ત્યાં જઈ શકું છું. તેથી, જો મારી પાસે કોઈ રસ્તો છે, જે

(0,0)થી

શરૂ થાય છે, જમણે, કોઈપણ પાથ, જે કઈ રીતે આવે છે

(i,j -1),

પછી એક લઈને, એક પગલું બરાબર, તે રસ્તો એક માર્ગ બને છે

(આઈ, જે)

તેથી,

(0,0)થી

હું ઓછા દરેક પાથ

(i,j-1)

ને અનન્ય રૂપે વિસ્તૃત કરી શકાય છે

(i,j).

તેવી જ રીતે, કોઈપણ પાથ, જે

(0,0)થી

(i-1,j)

થી આવે છે તે અનન્ય રૂપે, વિસ્તૃત રીતે વિસ્તૃત કરી શકાય છે. તેથી, જો હું બંને પાડોશીઓને આવું કાર્ય ગણું છું, તો તે દરેક પાથને વર્તમાન નોડ સુધી પહોંચાડવા માટે વિસ્તૃત કરી શકાય છે. તેથી, તેથી, હું નીચે પ્રમાણે ઈન્ડેક્ટીવ ફોર્મ્યુલેશન મેળવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:13)

તેથી, ચાલો

(0,)થી

વર્તમાન બિંદુ

(i,j)

સુધીના પાથોની સંખ્યાને સૂચવવા માટે પાથો

(i,j)

લખીએ. તેથી, સમસ્યા પરના અમારા ઈન્ટરેક્ટિવ વિશ્લેષણમાંથી આપણે જે હમણાં જ જોયું છે તે છે કે, જો તમે

(i,j)

પર છો, તો પછી પાથ ડાબે અથવા નીચેથી આવે છે. તેથી,

(i,j)

ના રસ્તાઓ

(i માઈનસ 1, j)

અને પાથો

(i,j માઈનસ 1)

નો પાથોનો સરવાળો છે. તેથી, અલબત્ત, કેટલીક સરહદ પરિસ્થિતિઓ છે. તો, જો તમે અમારા ગ્રીડને જોશો, સામાન્ય રીતે, તો જો આપણે ડાબી બાજુના સ્તંભ તરફ જોવું જોઈએ, તો ચાલો નીચેની પંક્તિ શરૂ કરીએ. તેથી, આપણે નીચે પંક્તિને શરૂ કરીએ, જમણી બાજુએ, પછી આપણે જાણીએ છીએ કે આ ફોર્મ (0,0),પછી (1,0)અને તેથી (5,0),બરાબર છે. તેથી, ફોર્મ

(i,0)

ની કોઈપણ વસ્તુ ફક્ત તેના ડાબેથી જ મૂલ્ય પ્રાપ્ત કરે છે કારણ કે ડાબેથી કોઈ અનુરૂપ પંક્તિ નથી. એ જ રીતે, ડાબી બાજુના સ્તંભથી, આ

(0,0),

(0,1)અને

તેથી ઉપર

(0,15).અને

હવે, ડાબેથી કશું જ નથી. તો, હું તેને નીચે પંક્તિમાંથી જ -1 માંથી જ મેળવી શકું છું. અને છેવટે, તમારે પોતાને પૂછવું પડશે કે પ્રારંભિક સ્થિતિમાં શું થાય છે. તેથી, જો હું (0,0)પર છું અને હું (0,0)પર જવા માંગુ છું, તો ત્યાં કેટલા માર્ગો છે? ઠીક છે, આ એક નાનો માર્ગ છે, ત્યાં ફક્ત એક જ રસ્તો છે, ત્યાં જ રહેવાથી. તે મહત્વનું છે કે તે 0 નથી કારણ કે યાદ

રાખો, જો તે 0 છે, તો દરેક જગ્યાએ તમે ફક્ત વસ્તુઓ ઉમેરી રહ્યા છો અને કશું જ આવશે નહીં, તમને કોઈ માર્ગ મળશે નહીં. તેથી, તે મહત્વપૂર્ણ છે કે,

(0,0)થી

એક જ રસ્તો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:42)

તો, આપણે આ સેટઅપમાં છિદ્રો સાથે કેવી રીતે કામ કરીશું? તે સરળ છે, આપણે ફક્ત કહીએ છીએ કે, જ્યારે પણ આપેલ બિંદુ પર છિદ્ર હોય ત્યારે આપણે ફક્ત ભાગો

((સમય:11:52))

નો ભાગ જાહેર કરીએ. બીજા શબ્દોમાં, જો કોઈ બિંદુએ છિદ્ર હોય તો ઉપર અથવા નીચે શું છે તે ધ્યાનમાં લીધા વગર આ વસ્તુ 0 ફાળો આપવા જઈ રહી છે. તેથી, જો હું આવીશ, જો મારી પાસે અહીંથી કંઈક આવી રહ્યું છે અને અહીં હું તેને અવગણશો અને કહું છું કે આ 0 છે. અને તેવી જ રીતે, જ્યારે હું વસ્તુની ગણતરી કરું છું ત્યારે ડાબી બાજુથી અમુક વેલ્યુ x આવશે. તેથી, આ ફક્ત x વત્તા 0 હશે. અને તે જ રીતે, અહીંથી કંઈક નીચે આવી રહ્યું છે, y ને કહો, અને આ 0 + y હશે, બરાબર. તેથી, કોઈપણ છિદ્ર માત્ર ઘોષણા પાથો

(i,j)

થી સમાન હશે કારણ કે તેના દ્વારા કંઈપણ પસાર થઈ શકતું નથી અને તે આપમેળે તેના પડોશીઓને પ્રચાર કરશે. બાકીના ઈન્ટરેક્ટિવ

((સંદર્ભસમય: 12:27))

પહેલા જેવું જ બરાબર. તેથી, જો તે છિદ્ર નથી, તો તે તેના બે પાડોશીઓ પર નિર્ભર છે. પછી આપણી પાસે નીચે પંક્તિ હરોળ છે અને મૂળ મૂળ સ્થિતિઓ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:38)

તેથી, આ ગણતરીમાં મુશ્કેલી એ ફિબોનાસી સાથે સંકળાયેલું જ છે, જે છે કે, જો મારી પાસે ચોક્કસ ગણતરી હોય તો, ઉદાહરણ તરીકે, માની લો કે આપણે પુનરાવર્તિત ગણતરી

(5,10),

પછી આ

(4,10)અને

(5,9)

માટે પૂછશે. હવે, આ બદલામાં બંને (4,9) માટે પૂછશે, તેથી (4,9).જો હું આ માર્ગોનું પુનરાવર્તિત રીતે મૂલ્યાંકન કરું છું, જે રીતે તે પરત આવી રહ્યું છે, તે ઈન્ટરેક્ટિવ ડેફિનેશન છે, તે (4,9) ના કોલિંગ પાથોને આમાંથી ઓછામાં ઓછા બે વખત સમાપ્ત કરશે

(સંદર્ભસમય: 13: 12))

અને આ બિનઉપયોગી પુન: વિવાદ સમગ્ર દરમિયાન થશે અને અમને પાથ પર કૉલની ઘોષણાત્મક સંખ્યા મળશે. તેથી, આપણે પહેલા જોયું છે, આની સાથે વ્યવહાર કરવા માટે અમારી પાસે બે તકનીકો છે. તેથી, એક યાદશક્તિ છે જ્યાં આપણે ફક્ત ખાતરી કરીએ છીએ કે અમારી પાસે ક્યારેય

(i,j)

પાથ્સ એક જ વાર કરતા વધુ અને i કરતાં વધુ નથી. બીજી રીત એ છે કે પેટા સમસ્યાઓનો અંદાજ કાઢવો, તે કેવી રીતે તેઓ એકબીજા પર આધાર રાખે છે તે નક્કી કરો, પછી યોગ્ય ક્રમમાં તેને સીધા જ હકારાત્મક રીતે હલ કરો અને આ તે છે જેને આપણે ડાયનેમિક પ્રોગ્રામિંગ કહીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:42)

તો, આપણે ગ્રીડ પર ડાયનેમિક પ્રોગ્રામિંગનો ઉપયોગ કેવી રીતે કરીશું? તેથી, આ (2) અને (4,4)
(2,4)પરના બે છિદ્રો સાથેની આ ગ્રિડ છે. પ્રથમ વસ્તુ એ આધારભૂતતાની ડી.એ.જી.ની માળખું ઓળખવા માટે છે, તેથી, તે છે, આપણે જાણીએ છીએ કે, દરેક (i,j) તેના પર આધાર રાખે છે, તે ડાબી અને નીચેનો પાડોશી છે. તેથી, આ રીતે આપણે ડીએજી કરીએ છીએ. જો તમને યાદ છે, જો તે બે મૂલ્યો પર આધારિત છે, ડાબી અને નીચે, અમે ડાબેથી આ નોડ અને નીચેથી આ નોડ સુધીના ધારને દોરીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 14:10)

તેથી, આ ડીએજી બંધારણ છે, જમણી બાજુ. તેથી,

((સમયનોસંદર્ભ લો: 14:11)),

ડી.એ.જી.નું માળખું. તેથી, ડી.એ.જી. નું માળખું કુદરતી રીતે તળિયે ડાબેથી ટોચ સુધી જાય છે, તેથી, આ માત્ર ડિગ્રી નોડમાં 0 ડીએગ છે. તેથી, જો તમે ગતિશીલ પ્રોગ્રામિંગનો સીધો ઉપયોગ કરીને ગ્રીડ પાથ પર આ ગ્રિડની ગણતરી કરવા માંગો છો, તો એકમાત્ર જગ્યા જે આપણે શરૂ કરી શકીએ તે છે (0,0)

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:35)

તેથી, આપણે (0,0)થી શરૂ કરીએ છીએ અને ત્યાં મૂલ્યને ભરીએ છીએ, જે 1 છે. અને હવે, આપણે જોયું છે કે અમારી પાસે બે શક્યતાઓ છે. આપણે જમણી બાજુ જઈ શકીએ છીએ અથવા આપણે ઉપર જઈ શકીએ છીએ કારણ કે આ બે ડિપેન્ડન્સીઝ હવે દૂર થઈ ગઈ છે. તો, ચાલો આપણે હરોળથી હરોળ કરીએ. તેથી, જો આપણે આ મૂલ્ય કરીએ, તો આપમેળે આ નિર્ભરતા જાય છે. તેથી, આપણે આ મૂલ્ય કરી શકીશું, જ્યારે આ જશે, તેથી અમે આ મૂલ્ય કરી શકીએ છીએ અને બીજું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:55)

તેથી, આપણે સમગ્ર તળિયે પંક્તિને ગણતરી કરી શકીએ છીએ અને આ બધા અમારા બેઝ કેસનો ઉપયોગ કરીને ડાબી બાજુથી વારસાગત છે. તેથી, અને તે ખૂબ જ સ્પષ્ટ છે, મારો મતલબ એ છે કે, આમાંથી કોઈ પણ સ્થાન મેળવવા માટે ફક્ત એક જ રસ્તો છે, જે સીધા જ ડાબેથી જમણે જાય છે, કોઈ વિચલન શક્ય નથી. તેથી, નીચે નોડ પર દરેક નોડનો એક જ રસ્તો છે. હવે, આપણે એક પંક્તિ ઉપર જઈ શકીએ છીએ. આપણે જોઈ શકીએ છીએ કે, બીજી પંક્તિમાંથી પહેલી પંક્તિમાં આ નોડ હવે ઉપલબ્ધ છે. તે પહેલેથી જ ઉપલબ્ધ છે, પરંતુ જો આપણે તે કરીએ તો પણ આપણે તે કરી શકીએ

((સમય:15:22))

તો, આપણે બીજી બીજી પંક્તિને ભરી શકીએ છીએ અને દરેક બિંદુએ આપણે ફક્ત ઉમેરી રહ્યા છીએ. તેથી, 2, 2 વત્તા 1 એ 2 છે, 1 વત્તા 3 એ 4 છે અને તેથી આગળ. તેવી જ રીતે, અમે આગામી છિદ્ર કરી શકો છો. તેથી, ઉદાહરણ તરીકે, 6 વત્તા 4 10 અને તેથી આગળ છે. તેવી જ રીતે, તમે આગળની હરોળ કરી શકો છો, 20 વત્તા 15 35 છે. હવે, આપણે છિદ્ર પર આવીએ, જમણે. તેથી, આપણે જે છિદ્રો પર કહ્યું છે, તે નીચેથી આવે છે તે ધ્યાનમાં લીધા વગર 0 હશે. નીચેથી 10 આવ્યાં હોવા છતાં, આ પ્રથમ છિદ્ર 0 થી નીચે હોવું જોઈએ, નીચેથી 35, છિદ્ર 0 હોવું આવશ્યક છે. તેથી, આપણે આ પંક્તિને બરાબર પહેલાની ગણતરી કરીએ સિવાય કે જે 0 માં આપણે જે ઈનપુટ દાખલ કરીએ છીએ તે છૂપાવે છે તે આપણે તેને 5 વત્તા 10 તરીકે ગણતા નથી, આપણે ફક્ત 0 મુકો. આપણે આ છિદ્રને 20 વત્તા 35 તરીકે ગણતા નથી, આપણે 0 મુકો.

(સ્લાઈડસમયનો સંદર્ભ લો: 16:04)

હવે, જ્યારે આપણે આગળની પંક્તિ પર જઈશું આ 0 માત્ર 0. નું યોગદાન આપે છે. તેથી, તેના ઉપર, કારણ કે આ દ્વારા કોઈ માર્ગો આવી શકતા નથી, આ દિશામાં, અહીં આવતા પાથોની સંખ્યા ડાબી બાજુથી માત્ર 6 છે. તેથી,

((સમયનોસંદર્ભ લો: 16:12))

અહીં આવતા પાથની સંખ્યા ડાબી બાજુથી ફક્ત 26 છે. તેથી, પાથની સંખ્યા જે આ બિંદુએ આવી રહી છે તે ત્યાંથી આવી શકતી નથી. તેથી, આ 20 નકલ થયેલ છે, આ 56 નકલ થયેલ છે. તેથી, આપણે પંક્તિ દ્વારા આ પંક્તિ કરી શકીએ છીએ. અને જો તમે કૉપિ કરીને આની જેમ દરેક હરોળની ગણતરી કરી શકો છો, તો ડાબી અને નીચેની કિંમતોને ઉમેરી રહ્યા છે અને અમને મળશે કે, આ બે અવરોધો સાથે ખરેખર 1363 પાથ છે. અને જો આપણે આ અવરોધોને આસપાસ ખસેડીએ, તો આપણે આ ગણતરીઓ ફરીથી કરી શકીએ છીએ, તે કાર્યક્ષમ હશે, આપણે આ સમાવેશ અને બાકાતતા વિશે ચિંતા કરવાની જરૂર નથી. તેથી, ગતિશીલ પ્રોગ્રામિંગ કરવાની આ એક રીત છે, પરંતુ યાદ રાખો કે કોઈપણ ટોપોલોજિકલ સૉર્ટ કાર્ય કરશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 16:48)

તેથી, આપણે તેના બદલે 1 થી પ્રારંભ કરી શકીએ અને કૉલમ દ્વારા સ્તંભ પર જાઓ. તેથી, આપણે જઈને તેને ભરી શકીએ છીએ અને પછી આપણે સંપૂર્ણ પ્રથમ કૉલમ ભરી શકીએ છીએ. પ્રથમ કૉલમ ફરી એક વાર થશે કારણ કે આ કરવા માટેનો એકમાત્ર રસ્તો હશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 17:01)

પ્રથમ કૉલમ ભરીને, હવે આપણે બીજા સ્તંભને ભરી શકીએ છીએ અને પછી તમે ત્રીજી કૉલમ અને ચોથા કૉલમ ભરી શકો છો અને તેથી આગળ. અને દેખીતી રીતે, આ મૂલ્યોને સમજાવવાનો આ એક માત્ર અલગ રસ્તો છે. કિંમત બદલાશે. તેથી, આપણે દરેક બિંદુએ આ જ મૂલ્યો સાથે અંતમાં સમાપ્ત થવું જોઈએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 17:17)

અને અંતે, તમે ડીએગ દ્વારા પણ કરી શકો છો. તેથી, નોંધ લો, કે જ્યારે હું આ કરું છું, તો આ બે બિંદુઓ હવે 0 અને ડિગ્રી નોડ્સ છે. હું તે બંને કરી શકું છું. હવે, મારી પાસે આ ત્રણ બિંદુઓ છે, 0 અને ડિગ્રી મોડ. તેથી, હું આ બધા ત્રણ કરી શકું છું. પછી મારી પાસે આ ચાર બિંદુઓ, 0 અને ડિગ્રી નોડ છે. તેથી, હું આ બધું કરી શકું છું. તેથી, આ ફક્ત ભાર આપવા માટે છે, કે ટોપોલોજિકલ સૉર્ટની પસંદગી તમારા ઉપર સંપૂર્ણ છે. ઘણી વાર, આ પ્રકારની ત્રિકોણીય વસ્તુઓને પ્રોગ્રામ કરવા માટે તે વધુ જટીલ બનશે તેથી, સામાન્ય રીતે જે કરવાનો પ્રયાસ કરે છે, તે એક પંક્તિ અને કૉલમ છે, પરંતુ મૂળ મૂલ્યોના કોઈપણ સ્થાનિક ક્રમમાં તેનો ગણતરી કરવા માટે ઉપયોગ કરી શકાય છે. તમારે જાણવાની જરૂર છે, જ્યારે તમે નોડ આવે છે, તે મૂલ્ય જેને તમે ગણતરી કરવા માંગો છો, તેની બધી નિર્ભરતા પહેલાથી જ ગણતરી કરવામાં આવી હોવી જોઈએ, તે જ આ DAGS માળખું તમને આપે છે. તેથી, પેટા સમસ્યાઓ એ ડીએજી બનાવે છે અને તમારે તેને પ્રોગ્રામિંગ સુધી સૌથી અસરકારક રીતે ડીએજી દ્વારા નેવિગેટ કરવું પડશે. તે સામાન્ય રીતે કોષ્ટકમાં પંક્તિ અથવા એક કૉલમ દ્વારા હોય છે, પરંતુ તે પણ ત્રાંસા હોઈ શકે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:18)

તેથી, અંતે, આપણે સમાપ્ત થાય તે પહેલા, ચાલો આપણે મેમોઈઝેશન અને ડાયનેમિક પ્રોગ્રામિંગ વચ્ચેના તફાવતના એક સૂચક દૃષ્ટાંતને જોઈએ. તેથી, જો આપણી પાસે સરહદની છિદ્રોનો સમૂહ હોય તો, પછી તે જે કહે છે તે સરળતાથી, જો હું અહીંથી શરૂ કરું છું અને હું ગમે તે રીતે જાઉં છું, તો હું અટવાઈ જઈશ, બરાબર. તો, જો હું ત્યાંથી મારી યાદશક્તિ શરૂ કરું, તો દરેક જગ્યાએ તે માત્ર 0 જોવા જઈ રહ્યું છે. તેથી, આ બધા મૂલ્યો 0 થશે, અને મેમોઈઝેશન આજુબાજુ આગળ જોશે નહીં. તેથી, યાદ રાખવું, હું દાવો કરું છું કે, આ બાહ્ય સીમા સાથે જ જોવાનું છે, ફક્ત આ ગ્રિડ બિંદુઓને યાદ રાખીને જ ગણવામાં આવશે. જો કે, જો હું ડાયનેમિક પ્રોગ્રામિંગ કરું, તો હું અહીં ફોર્મ શરૂ કરીશ અને હું આંખે ભરાઈશ. ફક્ત જ્યારે હું 0 સુધી પહોંચું ત્યારે, મને સમજાયું છે કે ગ્રીડની અંદર મૂલ્યોની ગણતરી કરવામાં આવી છે, ચિંતા કરશો નહીં, બરાબર.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:14)

તેથી, આ ગ્રિડમાં આ સંપૂર્ણ શેડ થયેલ ક્ષેત્ર છે જેની મૂલ્યોની જરૂર નથી કારણ કે તેઓ તળિયેથી ટોચ સુધી કોઈપણ પાથમાં યોગદાન આપી શકતા નથી કારણ કે તેમના દ્વારા પસાર થતા પ્રત્યેક પાથ, તમે અવરોધિત થશો આ છિદ્રોમાંના એક દ્વારા. જ્યારે, ગતિશીલ પ્રોગ્રામિંગ હવે આ બિંદુઓ માટે મૂલ્યનું અંધ ગણવામાં આવે છે, તેમછતાં પણ તે નિરર્થક યાદગીરી છે, કારણ કે તે માત્ર જરૂરિયાતની ગણતરી કરે છે અને તે આ મુદ્દા સુધી ક્યારેય પહોંચી શકશે નહીં.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:36)

તેથી, તેથી, મેમો ટેબલમાં એન્ટ્રીની રેખીય સંખ્યા હશે. તે કહેશે, 2 મી પ્લસ 2 એન પ્રવેશો. તેની પાસે ફક્ત આ ગ્રિડની બાહ્ય સીમા હશે

((સમયનોસંદર્ભ: 19:46)).

તેથી, તેની પાસે બે પરિમાણોના સંદર્ભમાં રેખાંશ સંખ્યા હશે, જ્યારે ગતિશીલ પ્રોગ્રામિંગમાં એન ટાઈમ્સ એન્ટ્રી હશે. દરેક ગ્રિડ પોઈન્ટ કોષ્ટકમાં ગણવામાં આવશે, તેમ છતાં તેમાંના મોટા ભાગના નકામું છે. તેથી, એક અર્થમાં, આ ઉદાહરણમાં, ગતિશીલ પ્રોગ્રામિંગ વેલ્યુઝ કમ્પ્યુટિંગ કમ્પ્યુટિંગ છે, જેનો આપણે થોડો વિશ્લેષણ કરી શકીએ તે સમજવામાં આવશે નહીં તેનો ઉપયોગ ક્યારેય કરવામાં આવશે નહીં કારણ કે તે ફક્ત મૂળથી ટોચ પરની દરેક પેટા સમસ્યાને ગણતરીમાં રાખે છે. જો કે, આપણે પહેલા કહ્યું હતું કે, ગતિશીલ પ્રોગ્રામિંગ પુનરાવર્તિત છે. તે ફક્ત સરળ બનશે, યાદશક્તિ પુનરાવર્તિત થઈ રહી છે. તે પુનરાવર્તિત કૉલને બે વાર નહીં બનાવવાની ઑપ્ટિમાઈઝ થઈ રહ્યું છે, પરંતુ તેમ છતાં તે પુનરાવર્તિત છે અને પ્રોગ્રામિંગ પ્રોગ્રામિંગ ભાષામાં એક્ઝેક્યુશનના સંદર્ભમાં ખર્ચ કરે છે. તેથી, જો છિદ્રો ખરાબ રીતે વહેંચવામાં આવે તો આ એક નકામી ગતિશીલ પ્રોગ્રામીંગ વ્યૂહરચના જેવી લાગે છે. ખરેખર, તે કોઈ વાંધો નથી. તે સામાન્ય રીતે બહાર આવે છે, કે ગતિશીલ પ્રોગ્રામિંગ અમલીકરણ સામાન્ય રીતે મેમોઈઝેશન અમલીકરણ કરતાં વધુ કાર્યક્ષમ રહેશે. તેથી, યાદગાર અમલીકરણ કરવાનું સરળ છે કારણ કે આપણે જાણીએ છીએ કે, આપણે એક ઈન્ટરેક્ટિવ ડેફિનેશનથી સીધા જ એક પુનરાવર્તિત પ્રોગ્રામ પર જઈ શકીએ છીએ. અને અમે છેલ્લા સમય જોયું, કે ત્યાં એક સામાન્ય ફોર્મ્યુલા છે, કોઈપણ પુનરાવર્તિત પ્રોગ્રામ યાદ રાખવા માટે રેસીપી. તમારે માત્ર બે પગલાંઓ શામેલ કરવી પડશે, ટેબલને જુઓ અને ટેબલ ફીડ

કરો, જમણી બાજુ. તેથી, તેથી, મેમોઈઝ્ડ અમલીકરણ મેળવવું એ ખૂબ જ સરળ છે, ગતિશીલ પ્રોગ્રામિંગ અમલીકરણ મેળવવા માટે પેટા સમસ્યાઓના કેટલાક વિશ્લેષણની જરૂર છે અને સબ સમસ્યાઓનું મૂલ્યાંકન કરવા માટે સારો ઑર્ડર શોધી કાઢવો આવશ્યક છે. તેથી, તે વધુ કાર્ય છે, પરંતુ આ વધારાનો કાર્ય સામાન્ય રીતે ચૂકવે છે, કારણ કે તે પછી તમે પુનરાવર્તિત કરતાં બધી સબ સમસ્યાઓનો ઈન્ટરેક્ટિવ ગણતરી મેળવી શકો છો.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક-07
મોડ્યૂલ - 04
લેક્ચર - 47
સામાન્ય સબડવર્ડ્સ અને ઉપસંહરણો

ચાલો હવે આપણે સામાન્ય સબવર્ડ્સ અને ઉપસંશોરણો શોધવા માટે બે અનુક્રમ વચ્ચેની સમસ્યાની ફેરબદલ કરીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:09)

તેથી, આપણે જે પ્રથમ સમસ્યાને જોવી તે સૌથી સામાન્ય સબવર્ડ સમસ્યા કહેવામાં આવે છે. તો, ચાલો ધારીએ કે આ ક્ષણ માટે આપણને બે શબ્દો આપ્યા છે, ચાલો ધારીએ કે ત્યાં અંગ્રેજી અથવા તે જેવી ભાષા જઈ રહી છે, આપણે બે શબ્દમાળાઓ આપ્યા છે અને આપણે તેમની વચ્ચે સૌથી લાંબી સામાન્ય સબડવર્ડની લંબાઈ શોધવા માંગીએ છીએ. સબવર્ડમાં, તે ફક્ત એક સેગમેન્ટ છે, અત્યાર સુધીમાં જો મારામાં ગુપ્ત અને સચિવ હોય અને સૌથી લાંબી સામાન્ય સબવર્ડ માત્ર ઉપસર્ગ રહસ્ય હોય અને તેની લંબાઈ 6 હોય, તો ભાગલા અને ત્રિજ્યા વચ્ચે, મારી પાસે આ સામાન્ય સેગમેન્ટ છે. જો હું બીજી તરફ દ્વિભાષી અને ગુપ્ત રહું છું, તો સામાન્ય સબવર્ડ શબ્દ લંબાઈ 3 છે, એટલે સેકંડ અને જો મારી પાસે ડિરેક્ટર અને સેક્રેટરી હોય, તો વાસ્તવમાં ફક્ત બે લંબાઈવાળા સબવર્ડ્સ છે અને તેમાંના બે છે, તેથી બંનેમાં ઈસી થાય છે તેમને અને તેથી ફરીથી કરે છે. તેથી, અમારી પાસે સૌથી લાંબી સામાન્ય સબડવર્ડ માટે એકથી વધુ ઉમેદવાર હોઈ શકે છે, પરંતુ આ ક્ષણે લક્ષ્ય દ્વારા મુખ્યત્વે સબવર્ડ શબ્દની ગણતરી કરવી નહીં, પરંતુ લંબાઈ, આપણે જોશું કે આ ઉપનામ કેવી રીતે આપણે લાંબું ગણતરીથી સરળતાથી બહાર નીકળી શકીએ છીએ . તેથી, આ ક્ષણે ધ્યાન કેન્દ્રિત કરવાની લંબાઈ મેળવવાનું છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:28)

તેથી, ચાલો આ વસ્તુને વધુ ઔપચારિક રીતે જોઈએ, તેથી આપણી પાસે 0 થી 0 ની સંખ્યા છે અને બી 0 થી બીન થાય છે, અને હવે આપણે શું કહેવા માંગીએ છીએ કે જો હું 0 લખું છું , 1, પછી પોઝિશન Ai, Ai Plus 1 અને તે પછી, એમ. તેથી, જો હું Ai અને BJ થી શરૂ થતા કેટલાક સેગમેન્ટ્સ શોધી શકું, જેમ કે જો હું અહીંથી પ્રારંભ કરું છું અને હું કે અક્ષરો વાંચું છું, તો બીજેથી પ્રારંભ કરો અને હું કે અક્ષરો વાંચું છું. તેથી, મેં એઆઈઆઈ પ્લસ 1 અપ ટુ કે ઓછા 1, હું પ્લસ કે ઓછા 1 અને બીજે, બીજે પ્લસ 1 અને બીજે પ્લસ કે માઈનસ 1 વાંચ્યું. પછી આ બંને કે વાક્ય સેગમેન્ટ છે જે સમાન છે, પછી હું કહીશ કે તમે અને v એ લંબાઈ કેનનું સામાન્ય સબવર્ડ છે. તેથી, ત્યાં શબ્દનો મેળ ખાતો ઉપભાગ હોવો આવશ્યક છે અને હવે, તેનો ઉદ્દેશ લાંબાગાળાના આવા સેગમેન્ટની લંબાઈ શોધવાનો છે. તેથી, યુ અને ડબલ્યુનો સૌથી લાંબો સામાન્ય સબવર્ડ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:25)

તેથી, અહીં બ્રુટ ફોર્સ એલ્ગોરિધમ છે, તમે દરેક સ્થાનને અજમાવી જુઓ, તમે દરેક Ai અને BJ ને જુઓ છો, તેથી તમે દરેક I અને J ને જુઓ છો, હું 0 અને એમ અને j વચ્ચે 0 અને એન. અને પછી હું જોઈ રહ્યો છું, જો હું આ બે મેચ જોઉં, તો હું આગળનો પત્ર જોઉં છું અને આગળ વધું છું અને જ્યાં સુધી હું બે સ્થિતિ શોધી શકતો નથી ત્યાં સુધી હું જતો રહ્યો છું. પછી મને ખબર છે કે આ બે સ્થિતિઓથી શરૂ થતું સૌથી લાંબું સબવર્ડ માત્ર બેવકૂફ છે, તેથી હું અબિયા પ્લસ 1 બાય પ્લસ 1 થી મેચ કરી શકું છું, શક્ય તેટલું લાંબી મેચનો ટ્રેક રાખી શકું છું, હું દરેક ai અને b j માટે આ કરું છું. તેથી, જો એમ n કરતા વધારે છે, તો પ્રથમ અને ij ની બધી પસંદગીઓની સંખ્યા m વખત n છે અને સામાન્ય રીતે, હું ટૂંકા શબ્દને બંધબેસતા દરેક શબ્દની લંબાઈ પર જઈશ. તેથી, હું આ સ્કેનના અંત સુધી પહોંચવા પહેલા n શબ્દ ઓર્ડર કરી શકું છું, ટૂંકા શબ્દના અંત સુધી દરેક મેચ મેળ ખાય છે, પછી મેં ઓર્ડર n શબ્દ આપ્યો છે. તેથી, તેથી, આ m ના n ના અથવા n mn ચોરસમાં હશે. તેથી, હવે, અમારું ધ્યેય જોવાનું છે, ભલે આપણે કંઈક વધુ આકર્ષક બનાવીને આ વધુ કાર્યક્ષમ બનાવી શકીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:41)

તેથી, પ્રસ્તાવના અવલોકનમાં પ્રથમ અવલોકન એ છે કે જો મારી પાસે એઆઈઆઈ પ્લસ 1 હોય તો, જો મારી પાસે બીજેબીજે પ્લસ 1 હોય. તો, જો મારી પાસે લંબાઈ કે અહીંથી સબવર્ડ શબ્દ હોય, તો પછી જુઓ આગળની સ્થાને અને આગળ વધવું, તે લંબાઈ કે ઓછા 1 નું સબડવર્ડ હોવું આવશ્યક છે. તેથી, આઈજે પર લંબાઈ k નું સામાન્ય સબવર્ડ છે, જો અને લંબાઈ કે ઓછા 1 નો લાંબા સામાન્ય સબવર્ડ હોય તો જ અને વત્તા 1 અને જે પ્લસ 1. તેથી, બીજા શબ્દોમાં હું આઈજે જોઉં છું, જો ij બરાબર છે, તો Ai અને b j પર સામાન્ય સબવર્ડ પર શબ્દ શરૂ કરવો શક્ય છે. તેથી, જો Ai બીજે જેટલું હોય, તો હું સૌથી લાંબો સામાન્ય સબવર્ડ જોઉં છું જે હું પ્લસ 1 અને જે પ્લસ 1 ખરીદ્યો હોત અને તેમાં 1 ઉમેરી શકું. બીજી બાજુ, જો Ai BJ સમાન નથી, તો તેમાં એક સામાન્ય સબડવર્ડ ઉમેરી શકતો નથી, કારણ કે પ્રથમ તે પહેલાથી મેળ ખાતું નથી. તેથી, પછી હું ફક્ત 0 ની શરૂઆતથી શરૂ થતા સૌથી લાંબી સામાન્ય સબવર્ડની લંબાઈ. તેથી, આ અમને ઈન્ડેક્ટીવ માળખું પર હેન્ડલ આપે છે. તેથી, જો હું સમાન ન હોત તો, હું બે પોઝિશન્સ જોઉં છું, હું જાહેર કરું છું કે ત્યાં કોઈ સામાન્ય સબવર્ડ્સ શરૂ થતા નથી, જો તે સમાન હોય, તો પછી હું સમસ્યાને આગળની સ્થાને પોસ્ટપોસ્ટ કરીશ અને પછીની સ્થાને જે પણ મળશે તે 1 ઉમેરીશ. સીમાની સ્થિતિ એ છે કે જ્યારે કોઈ એક શબ્દ ખાલી હોય છે, ત્યારે આપણે કોઈ છોડીએ નહીં, તેથી આપણે કોઈ એક શબ્દનો વિસ્તાર કરી શકતા નથી, આપણે આગળ વધી શકતા નથી, તો આપણે કહી શકીએ કે ત્યાં સામાન્ય સબડવર્ડ હોઈ શકતું નથી, કારણ કે સામાન્ય સબવર્ડ્સમાંથી એકનો ઉપયોગ કરવા માટે કોઈ અક્ષરો નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:14)

તેથી, સગવડ માટે, આપણા બધા શબ્દો 0 થી છું અને બી 0 થી બીએમ સુધી જાય છે, અમે એમ વત્તા ઑનથ પોઝિશન અને એન પ્લસ ઑનથ પોઝિશનને મંજૂરી આપીશું. તો, તમારી પાસે 0 માં m વત્તા 1 અને 0 થી n વત્તા 1 માં સ્થાનો હશે, તેથી m વત્તા 1 નો અર્થ એ છે કે હું તમારામાં છેલ્લી સ્થાનેથી આગળ ગયો છું અને બધા અક્ષરોને તૂટી ગયો છું. તેવી જ રીતે બી માં n વત્તા 1 એનો અર્થ એ છે કે આપણે v શબ્દની અંત સુધી પહોંચી ગયા છે. તેથી, આપણે જે કહ્યું છે તે છે

કે જો આપણે તમારામાં શબ્દના અંત સુધી પહોંચીશું, તો પછી આપણે કોઈ જગ્યાએ હોઈએ, ત્યાં કોઈ કોમન્સ નથી શબ્દનો, તેથી સામાન્ય સબડવર્ડની લંબાઈ 0 છે. તેવી જ રીતે, જો આપણે v માં શબ્દના અંત સુધી પહોંચી ગયા હોય, તો આપણે જ્યાં પણ હોઈએ ત્યાં કોઈ સામાન્ય સબવર્ડ લંબાઈ 0 નથી. તેથી, જો આપણે આ બેમાં નથી કિસ્સાઓ, આપણે શબ્દના અંતમાં નથી, તો પછી આપણે એ ચકાસીશું કે એઆઈ બીજે સમાન છે, જો Ai બીજે સમાન નથી, તો આપણે ઘોષણા કરીએ છીએ કે આપણે પહેલા જોયું છે કે સૌથી લાંબી સામાન્ય સબવર્ડ સાથે લંબાઈ 0 છે. નહિંતર, અમે મૂલ્યપૂર્વક ગણતરી કરીએ બંને શબ્દોમાં આગળના સ્થાને અને તેને 1 ઉમેરો, કારણ કે આપણે તે શબ્દ 1 થી વધારી શકીએ છીએ, કારણ કે ai equal to b j.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:24)

તેથી, આપણે તેના માટે યાદગાર રિકર્સિવ ફંક્શન લખી શકીએ છીએ, પરંતુ અમે આ સમસ્યા માટે ડાયનેમિક પ્રોગ્રામિંગ સોલ્યુશનની ગણતરી કરવાની સીધી રીતે પ્રયાસ કરીશું. તેથી, અમારી પાસે સામાન્ય રીતે ફોર્મની એલસીડબ્લ્યુ, આઈજે; આ મૂળભૂત ઉપ સમસ્યા છે. તો, પહેલાની વસ્તુમાંથી આપણે જે જોયું છે તે એ છે કે જો મારી પાસે Ai સમાન બીજે હોય, તો હું Ai Plus 1, Bj Plus 1 ને જોવાની જરૂર છે, તેથી LC, IW, LCW ij LCW i plus 1 અને j plus 1 પર આધાર રાખે છે. અમે આ પ્રકારની ઉપપ્રોબ્લેમ નિર્ભરતા છે, દરેક બિંદુએ મૂલ્યને નીચે આપેલા એક તરફ ધ્યાન આપવાની જરૂર છે. તેથી, આપણી પાસે સ્પષ્ટ છે કે આ અમારું છે અને આ આપણું જ છે, તેથી તેને જોવું અને લખવાનું છે. તો, તેથી, આપણા સંમેલન સાથે આપણે એરે પાછળની તરફ દોરીએ છીએ જે છે, જો આ ચોરસ પર આધાર રાખે છે, તો આપણે એક તીર દોરો, જે નિર્ભરતાને ઉલટાવી સૂચવે છે. તેથી, હું તે નીચે તીર પર જવા પહેલા, તીરના તળિયે ગણતરી કરવાની જરૂર છે, હું આની ગણતરી કરી શકું તે પહેલાં, તીર, મને આની ગણતરી કરવાની જરૂર છે. તેથી, હવે, આપણે આ ખૂણાથી પ્રારંભ કરી શકીએ છીએ, કારણ કે તે કંઈક પર આધાર રાખે છે અને પછી કામ કરવાનું શરૂ કરે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:43)

તેથી, જેમ આપણે સીમા પર જોયું, જ્યાં મેં એક શબ્દ થાક્યો છે, જ્યાં ક્યાંતો પ્રથમ શબ્દ અથવા બીજો શબ્દ છે. તેથી, આપણે આ દ્વિભાષી અને રહસ્યથી આ શબ્દને પાછો મોકલી દીધો છે, અમને જણાવો કે Ai અને b j ની કિંમતો શું છે. તો, આ એકનાં મૂલ્યો છે અને આ બીજો મૂલ્યો છે, તેથી આ મારા બે શબ્દો છે. તેથી, શરૂઆતમાં સીમા પર દરેક લાંબી સામાન્ય સબવર્ડ 0 છે, કારણ કે બે શબ્દોમાંથી એક ખાલી છે અને હવે, હું પંક્તિ દ્વારા અથવા કૉલમ દ્વારા પંક્તિ દ્વારા કૉલ કરી શકું છું. તેથી, ચાલો તેને બદલી માટે કૉલમ દ્વારા કરીએ. તેથી, હવે, આ બિંદુએ આપણી પાસે 5 બરાબર બી 5 છે, તેથી આપણને 1 વત્તા એલસીડબ્લ્યુ 6, 6 મળે છે અને તેથી આપણને 1 મળે છે, બીજે ક્યાંક સી અને ટી, ઈ અને ટી, પછી ટીમાં હોય છે. . તેથી, બાકી દરેક જગ્યાએ, તે ફક્ત 0 છે, કારણ કે મૂલ્ય મેળ ખાતું નથી. તેથી, જો હું પહેલાનાં અથવા સ્તંભ પર જાઉં છું. અહીં, માત્ર એક જ જે કહેવું છે તે ઈ અને ઈ છે, પરંતુ ત્યાં કારણ કે પછીનું 0 છે. તેથી, મને 1 વત્તા 0 મળે છે. જો હું પાછલા એક પર જાઉં, તો વાસ્તવમાં વેલ્યુ નો, ત્યાં છે કોઈ r માં ભાગ્યે જ, તેથી, સબવર્ડ 0 ની બધી કિંમતો છે, જો હું આગલી વસ્તુ પર આવીશ, તો ફરીથી મારી પાસે એક જગ્યા છે જ્યાં સી અને સી મેચ છે. તેથી, હું કહું છું કે અહીંથી શરૂ થતો સૌથી લાંબો સામાન્ય સબવર્ડ 1 વત્તા જે છે તે ટી અને આર થાય છે, પરંતુ ટી અને આર 1 વત્તા 0 થી મેળ ખાતું નથી અને હવે, કારણ કે આ સી હવે ઈ દ્વારા આગળ વધી રહ્યું છે. તેથી, આ પર મને લાગે છે કે તે

1 વત્તા તળિયે જે પણ થાય છે, તેથી તે 1 વત્તા 1 એ 2 છે અને તે જ રીતે, મને લાગે છે કે તે S અને S નું મેચ છે અને તે 1 વત્તા જે પણ નીચે છે તે 3 થાય છે. તેથી, આપણે આ રીતે ઓળખીએ છીએ, આપણે 3 એન્ટ્રી મૂકી છે. તેથી, આ ચોક્કસ કોષ્ટકમાં, તે થોડું રહસ્યમય છે, જ્યાં જવાબ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:33)

તેથી, આપણે શું કરવું છે તે સૌથી મોટી વેલ્યુ સાથે એન્ટ્રી મળી છે, આ કિસ્સામાં 2 કોમા 0, આ આપણું સૌથી મોટું મૂલ્ય છે. તેથી, આ આપણને કહે છે કે આ બે શબ્દો વચ્ચે, સૌથી લાંબી સામાન્ય સાર્વભૌમ લંબાઈ 3 છે. જેમ મેં પહેલા કહ્યું હતું તેમ, આપણે ફક્ત સૌથી લાંબી સામાન્ય સબવર્ડની ગણતરી કરી રહ્યા છીએ, તેથી આપણે વાસ્તવમાં સબવર્ડ શબ્દ કેવી રીતે શોધી શકીએ. ઠીક છે, આપણે ખરેખર આ કિસ્સામાં, સરળતાથી વાંચી શકીએ છીએ, આપણે જાણીએ છીએ કે ત્યાં લંબાઈ 3 નું સૌથી લાંબું સામાન્ય સબવર્ડ છે; તેનો મતલબ એ છે કે, તે લંબાઈ 2 અને તેથી વધુ શબ્દ દ્વારા સફળ થવું આવશ્યક છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:06)

તેથી, આ વિશિષ્ટ ત્રિકોણ આપણને કહે છે કે શબ્દોમાં કયા અક્ષરો છે અને જો હું આ કર્ણને ફક્ત એક સ્તંભ બંને પર મુકું છું, તો મને c, c માં શબ્દ મળે છે. તેથી, આપણે આ ઉદાહરણમાં જોયું કે, આપણે લંબાઈની ગણતરી કર્યા પછી, આ ઉકેલને વાંચવું ખૂબ સરળ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:22)

તેથી, અહીં આ ચોક્કસ અલ્ગોરિધમનો કેટલાક કોડ છે, ડાયનેમિક પ્રોગ્રામિંગ વર્ઝન, ફક્ત તેને ઓછા ગૂંચવણમાં લાવવા માટે, મેં i અને j ની જગ્યાએ, r અને c ની ચલોનો સ્પષ્ટ ઉપયોગ કર્યો છે. તેથી, r એ પંક્તિ છે, તે આ રીતે હતી અને c આ રીતે જાય છે. તેથી, આપણે જે કહીએ છીએ તે છે કે શરૂઆતમાં જો હું છેલ્લા સ્તંભને જોઈ રહ્યો છું, તો આ બધા 0 છે અને તમે નીચેની પંક્તિ જોઈ રહ્યા છો, યાદ રાખો કે ક્રમાંક 0 થી n વત્તા 1 અને 0 થી એમ વત્તા 1 થી છે. તેથી, અમે નીચે પંક્તિ જોઈ રહ્યા છીએ, જ્યારે પંક્તિ સંખ્યા n વત્તા 1 છે, દરેક સ્તંભ 0 છે. તેથી, આ 0 ની મૂકે છે. તેથી, આ બે રેખાઓ માત્ર વસ્તુઓને વહન કરે છે, હવે આપણે શોધવું પડશે, યાદ રાખવું છે કે આખરે જવાબ આપવા માટે આખી એરેમાં મહત્તમ પોઝિશન શોધવાનું ધ્યેય છે, તેથી અમે અહીં મૂલ્ય સાથે તેનો ટ્રૅક રાખીએ છીએ. તેથી, આપણે પ્રારંભિક ધારીએ છીએ કે લંબાઈ 0. નું મહત્તમ સામાન્ય સબવર્ડ. હવે, આપણે શું કરીએ છીએ, આપણે ફક્ત સ્તંભ દ્વારા પંક્તિ અને પંક્તિ દ્વારા સ્તંભ પર જાઓ, તેથી આપણે n થી 0 સુધી દરેક સ્તંભ પર જઈએ છીએ, તેથી આપણે કૉલમ દ્વારા સ્તંભ કરીએ છીએ અને દરેક સ્તંભ, અમે પંક્તિથી નીચેથી ટોચ સુધી, એમથી 0 સુધી જઈએ છીએ. અમે શબ્દ પોઝિશન આર અને પોઝિશન સી એ યુ જુઓ, આર ઈક્વલ બી, સી. જો એમ હોય તો, હું 1 ઉમેરી શકું, એલસીડબલ્યુ હોવું જોઈએ, તેથી જો મારી પાસે ua બાર c ની સમાન છે, તો લાંબી સામાન્ય શબ્દ, લંબાઈ લાંબું સામાન્ય સબવર્ડ આરસી 1 વત્તા ઉમેરો પ્લસ 1 સી વત્તા 1. નહિંતર, 0 જો, કારણ કે ત્યાં કોઈ સામાન્ય શબ્દ નથી અને જો ગણતરી કરેલું નવું મૂલ્ય અત્યાર સુધી જેટલા મૂલ્ય કરતાં મૂલ્યવાન છે, તો હું વર્તમાન મૂલ્યને અપડેટ કરીશ અને અંતે, અંત આ લંબાઈ લાવશે.

તેથી, આ કૉલમ દ્વારા આ વસ્તુ સ્તંભને પ્રક્રિયા કરવા માટેનો એક સીધો આગળનો માર્ગ છે, તમે આ બંને પ્રેરણાઓને અને પંક્તિ દ્વારા પંક્તિને રોકી શકો છો, જેથી તે એક કસરત છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:27)

તેથી, આપણે અગાઉ જોયું છે કે જો આપણે દરેક સ્થાન પર આંખે જુએ છે અને તે સ્થિતિને શરૂ કરતા શબ્દને સ્કેન કરવાનો પ્રયાસ કરીએ છીએ, તો અમને કંઈક મળે છે જે ઓર્ડર m, n ચોરસ છે. હવે, આ ઉકેલ જ્યારે કદ એમ ટાઈમ n ની કોષ્ટક ભરવા માટે જરૂરી છે, તો દેખીતી રીતે, એમ ટાઈમ્સ n ટેબલમાં દરેક એન્ટ્રીઝ, આપણે ફક્ત તેને ભરવા માટે પાડોશીઓને જોવું જોઈએ. તેથી, તે સતત સમય ઑપરેશન છે. તેથી, એમ ટાઈમ્સ એન એન્ટ્રીઝ, આપણે તેને m વખત n વખત ભરીએ છીએ. તેથી, આપણી પાસે ઓર્ડર n1, m n છે. જો ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરીએ તો, અમે તે કર્યું છે, પરંતુ જો મેમોઈઝેશનનો ઉપયોગ પણ કરવામાં આવે છે, તો તમને સમાન જવાબ મળશે, યાદ રાખવું જોઈએ, ત્યાં એક પુનરાવર્તિત કોલ્સનો ખર્ચ થઈ શકે છે અને વાસ્તવિક અમલીકરણ સમયની શરતો હોઈ શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:04)

તેથી, હવે આપણે એક કરતાં વધુ સામાન્ય સબવર્ડ શબ્દની તુલનામાં થોડી વધારે સામાન્ય સમસ્યા જોઈ શકીએ છીએ જે વધુ રસપ્રદ રીતે ગણતરીત્મક છે. તેથી, જો આપણે કોઈ ચોક્કસ મેળ ન શોધીએ, પરંતુ આપણે સ્વયંને કેટલાક અક્ષરો છોડી દેવાની મંજૂરી આપીએ છીએ. તેથી, આપણી પાસે ઉપ-શબ્દ નથી, તે આપણને કેટલાક અક્ષરો અને હવે છોડવાની છૂટ આપે છે, જો તમે જાણવા માંગો છો, કેટલાક અક્ષરોને છોડ્યા પછી, સૌથી લાંબી મેચ આપણે શોધી શકીએ છીએ. તેથી, હવે, આપણું અગાઉનું ઉદાહરણ, તેમાંના કેટલાક સમાન છે, જેમ કે કોઈ પણ અક્ષરને છોડ્યાં વિના હું 6 મેળવી શકું છું, હું તેને સુધારી શકું તેમ નથી, આપણે તેને વિભાજીત કરીશું, હું તેને સુધારી શકું નહીં. પરંતુ, હવે જો હું દ્વિભાષી અને રહસ્યને જોઉં, તો પહેલા આપણે ફક્ત 3 મેચ સેકન્ડ, સેકન્ડની લંબાઈ ધરાવતી હતી, પરંતુ હવે હું લંબાઈ 4 ને મેચ કરી શકું છું, કારણ કે અહીં જો આપણે તેને ઉમેરીશું, તો અહીં હું બે અક્ષરો મૂકી શકું છું અને મેળવી શકું છું ટી. તેથી, હું ખરેખર મેચ મેળવી શકું છું જેલંબાઈ 4 છે, તેવી જ રીતે આ બે ડિરેક્ટર અને સેક્રેટરીમાં, અગાઉ આપણે ફરી આવ્યા હતા, અને અમારી પાસે ec, ec હતી. પણ, હવે આ મુદ્દા દ્વારા હું ઈક્ટર મેળવી શકું છું, જેમ કે ectr, સમાન રીતે હું retr મેળવી શકું છું, હું retr મેળવી શકું છું. તેથી, જો આપણે સ્વયંને પત્રને છોડી દેવાની મંજૂરી આપીએ છીએ કારણ કે મેચ કરતા વધુ ક્રમ મળે છે અને આને સૌથી લાંબી સામાન્ય અનુગામી કહેવામાં આવે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:32)

તેથી, આપણે તેના વિશેના સૌથી લાંબી સામાન્ય સબવર્ડ શબ્દના સંદર્ભમાં વિચારી રહ્યા છીએ કે હું હવે કરી શકું છું, જો હું ચોક્કસ સેગમેન્ટને મેચ કરું છું, તો હું જમણી બાજુથી મેચ કરવાનું ચાલુ રાખી શકું છું. તેથી, હું બંને સૂચકાંકો આગળ વધવા દો. તો, હું ગ્રીડમાં નીચે અને જમણી બાજુ જઈ શકું છું અને મેચ જ્યાં ત્યાં છે ત્યાં બીજા સ્થાને જોઈએ છીએ. તો, પહેલાં, આપણી પાસે સેકંડ વચ્ચે ત્રણ સ્તરની મેચ હતી, અને પછી આપણે d વચ્ચે એક લેવલ મેચ ઉમેરીએ છીએ. તેથી, હું આ બંનેને ચાર સ્તરની સૌથી લાંબી સામાન્ય અનુગામીની જેમ જોડી શકું છું, પરંતુ આપણે અનુગામી ગણતરીને ત્યાં વધુ સીધી રીતે કરીશું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:04)

તેથી, આપણે લાંબા સમય સુધી સામાન્ય અનુગામી સમસ્યા રસપ્રદ કેમ છે તે જોવા માટે ઉપયોગી થઈએ તે પહેલાં. તેથી, આ વિસ્તારનો એક બાયો ઈન્ફોર્મેટીક્સ છે. તેથી, જીવવિજ્ઞાની એ ઓળખમાં રસ ધરાવે છે કે આનુવંશિક ભાવનામાં બે જાતિઓ કેટલી નજીક છે. તેથી, જો આપણે ડી.એન.એ. જુઓ, તો આપણું ડીએનએ મુખ્યત્વે કદ 4 ના મૂળાક્ષર પર લાંબી શબ્દમાળા છે. તેથી, આ 4 પ્રોટીન છે, તે એક ડીએનએ બનાવે છે, એ, ટી, જી, સી સંક્ષિપ્તમાં છે તેથી હવે, જો આપણે ડીએનએના બે શબ્દમાળાઓ અને સરખાવવાની કુદરતી રીતને જુએ છે, તો તેઓ એકબીજા પાસે કેટલું નજીક છે તે પૂછવા છે, તેઓએ અહીં અને ત્યાં કેટલીક વસ્તુઓની ડ્રોપને કેવી રીતે સરસ રીતે ગોઠવી છે. તેથી, તે એક વધારાની જાતિઓ અને અન્ય જાતિઓ હોઈ શકે છે જે થોડા વધારાના જીન તરીકે બીજું કંઈક છે અને જો તે જીન્સ છોડે છે, તો બીજું બધું એક જ છે, પરંતુ આ જનીનો થઈ શકે છે, તે બે જાતિઓમાં જુદા જુદા સ્થાનો હોઈ શકે છે. તેથી, એવું નથી કે ત્યાં એક સામાન્ય ભાગ છે, અને તે પછી એક વધારાનો ભાગ છે, તેના બદલે નવા જનીનો અન્ય જીન્સમાં ફેલાયેલો છે. તેથી, આપણે જાણવાની જરૂર છે કે, જો આપણે થોડા જનીનોમાંથી એક અને થોડા જીન્સમાં ઘટાડો કરી શકીએ તો બીજી બાજુ લીપ પર હોઈ શકે છે. તેથી, તે UNIX અથવા LINUX કોઈપણ સંબંધિત ઑપરેટિંગ સિસ્ટમનો ઉપયોગ કરતી સૌથી લાંબી સામાન્ય અનુગામી સમસ્યા છે, ત્યાં કમાન્ડ કોડ DIFF છે, જે તમને બે ટેક્સ્ટ ફાઈલો સાથે તપાસ કરવાની મંજૂરી આપે છે અથવા તેમની વચ્ચે ન્યૂનતમ તફાવત શોધે છે. તેથી, ડીઆઈએફએફ આદેશ પણ લાંબી સામાન્ય અનુક્રમણિકા કરે છે, તે એક અક્ષર તરીકે દરેક લાઈન સુધી પહોંચે છે અને તે કહે છે, ઓછામાં ઓછા રેખાઓ શું છે, હું આ બંને ફાઈલો વચ્ચે છોડી શકું છું, 4 હું તેમને મેચ કરી શકું છું, અને પછી તે તમને કહે છે કે કેવી રીતે પાછા વસ્તુઓ દાખલ કરવા માટે. પરંતુ, મૂળભૂત રીતે તે તમને કહેવા માટે સૌથી લાંબી સામાન્ય અનુગામી ગણતરી કરી રહ્યું છે, એકબીજાને કેટલી નજીકની ટેક્સ્ટ ફાઈલો છે. તેથી, આ લાંબી સામાન્ય અનુગામી સમસ્યા માટે પુષ્કળ એપ્લિકેશન છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:52)

તેથી, ચાલો આપણે આ સૌથી લાંબી સામાન્ય અનુગામી સમસ્યાના ઈન્ડેક્ટીવ સ્ટ્રક્ચરને સીધા જ સમજીએ, સૌથી લાંબી સામાન્ય સબવર્ડ ફેંકીએ નહીં. તેથી, નોંધ કરવાની પ્રથમ વસ્તુ એ છે કે જો હું આ બંને વચ્ચે આ સૌથી લાંબી સામાન્ય સબડવર્ડ શોધી રહ્યો છું, તો મને લાગે છે કે 0 એ હકીકત છે કે બી 0 ની બરાબર છે. હવે, હું દાવો કરું છું કે, હું તેને ઉકેલમાં જોડવું જોઈએ, અને પછી બાકીના સમાધાન માટે જુઓ. તેથી, મારે કંઈક કરવું જોઈએ, જ્યાં હું કહું છું કે એક મેચ 0 સમાન છે, અને પછી મને સૂચિ જ જોઈએ. તેથી, આને દલીલની થોડીક જરૂર છે, કારણ કે તે હોઈ શકે છે કે આ શ્રેષ્ઠ નથી. તેથી, યાદ રાખો કે આ એક લાલચ જેવી વસ્તુ છે; આપણે કહી રહ્યા છીએ કે, કારણ કે 0 બરાબર બી 0 મેચ થાય છે, અને પછી આગળ વધો.

(સ્લાઈડસમયનો સંદર્ભ લો: 17:44)

તેથી, કોઈ દલીલ કરી શકે છે કે, આ રસ્તો નથી, હું જવા માંગું છું, 0 ની કલ્પના કરવી, વાસ્તવમાં, બી 2 થી મેળ ખાવું જોઈએ, તે શ્રેષ્ઠ ઉકેલ હશે. તેથી, સારો વિચાર એ 0 થી 0 થી મેળ ખાતો નથી, 0 એ 0 થી મેળ ખાય છે, પરંતુ હવે ધ્યાન આપો કે જો 0 બી 2 સાથે મેળ ખાતું હોય, કારણ કે તે અનુગામી છે, તો પછી લખવા માટે કંઈક, કહે છે કે 1 એ કંઈક હોવું

જોઈએ જમણી તરફ, આ રેખાઓ, મારી પાસે એવી કોઈ વસ્તુ નથી જે આના જેવી પ્રક્રિયા કરે છે, હું પ્રક્રિયા સાથે કોઈ મેચ કરી શકતો નથી. કારણ કે, તે જ અનુક્રમમાં આવશ્યક છે, તેથી જો 0 થી બી 2 મેચ, જમણી બાજુએ એક વસ્તુ મેળ ખાતી હોય, તો 2 ને હજી પણ જમણી બાજુએ કંઈક મળવું જોઈએ અને બીજું. તેથી, આ તમામ મેચો એકબીજાને પ્રોસેસ કર્યા વિના નીચેથી ઉપરના શબ્દથી તળિયા સુધી જાય છે. તેથી, હવે, જો હું આ સોલ્યુશન લઈશ અને હું 0 અને બી 0 ને જાણું છું, તો હું ખરેખર આ તીરને અહીં ખસેડી શકું છું અને આ રીતે મેચ કરી શકું છું અને આને દૂર કરીશ, આ શું કરશે, તે મૂળ ઉકેલનો ઉપયોગ કરીને લાંબા સમય સુધી વિક્ષેપિત કરશે 0 મેચ અબ 2, પરંતુ, 0 થી 0 ની મેચ. પરંતુ, સોલ્યુશનની ગુણવત્તાના સંદર્ભમાં, મેચોની સંખ્યા, પહેલા 0 મેચો બીજા ક્યાંક મેચ કરે છે, હવે તે બી 0 થી મેચ થાય છે, લંબાઈ સૌથી લાંબી સામાન્ય અનુગામી બદલાતી નથી. તેથી, દલીલો પાછળ પાછી ફેરવી, તે કહે છે કે તેથી 0 જો 0 ની બરાબર છે, તો એ ધારવું ખૂબ સલામત છે કે 0, બી 0 સમાધાનનો ભાગ બનાવે છે અને બાકીની સમસ્યા તરફ આગળ વધે છે. તેથી, આપણે 1 અને બી 1 થી ઉપપ્રોબ્લેમ જોઈ શકીએ છીએ. તેથી, આ પહેલો કેસ છે, પ્રથમ કેસ કહે છે કે, 0 ની બરાબર 0 ની બરાબર છે, તો આપણે આ ઉપપ્રોબ્લેમને 1 થી જોઈ શકીએ છીએ. બી 1.

(સ્લાઈડસમયનો સંદર્ભ લો: 19:15)

હવે, જો તે સમાન નથી, તો તમે શું કરો છો તેની ખાતરી કોઈ નથી, તે બન્નેને છોડવા માટે એક સારો વિચાર નથી. તેથી, તાત્કાલિક માનીએ કે મારી પાસે સ્ટ્રો અને રસ્તાની જેમ કંઈક છે, તેથી માત્ર એસ કારણથી મેળ ખાતું નથી, તેનો અર્થ એ નથી કે હું બંનેમાં હોવું જોઈએ, તો તે એસને જીવંત રાખવા માટે હું આગલા એસ સાથે મેચ કરવું જોઈએ. તેથી, બન્ને ત્યાં હોઈ શકતા નથી, કારણ કે તેઓ એકબીજા સાથે નથી હોતા અને એસ જમણી બાજુએ કંઈક મેળવે છે અને તેના માટે કંઈક મળતું નથી. જો તેઓ બન્ને જુદા હોય, જો તેઓ સમાન નથી, તો આ કંઈક ત્યાંથી મેળ ખાવું જોઈએ અને આ અહીં કંઈક મેળ ખાવું જોઈએ અને બંને થઈ શકશે નહીં. કારણ કે, તેઓ પાર કરશે, આ અમે મંજૂરી આપી ન હતી, તેથી તેમાંના એક માત્ર બીજા શબ્દની જમણી બાજુથી કંઈક મેળ કરી શકે છે, પરંતુ આપણે તે જાણતા નથી. તેથી, આ અનિવાર્ય ઉકેલોનું સામાન્ય સિદ્ધાંત એ છે કે તમે એક અથવા બીજાને પસંદ કરવા માટે લખતા નથી. તો, જો હું બી છોડીશ અને 0 ને છોડીશ, તો મને એક ઉકેલ મળે છે, મને એક ઉપપ્રોબ્લેમ મળે છે જેનો 1 થી છું અને બી 0 થી બી એમ હોય છે. જો બીજી તરફ, હું 0 અને એક ડ્રોપ બી 0 રાખું છું, તો પછી 0 અવશેષ છે, પરંતુ બી 1 થી જાણી શકું છું, આ મારા બે સંભવિત પેટાપ્રવાહ છે જે જાણીને છે કે 0 અને બી બંને ત્યાં હોઈ શકતા નથી. પરંતુ, મારે તેમાંના એકને ત્યાં રહેવાની પરવાનગી આપવી જોઈએ, અન્યથા મેં કેવી રીતે સબ ઑપ્ટિમાઈઝ સોલ્યુશન બનાવ્યું. તેથી, તેમાંના એકને છોડી દેવું, તે નોકરીના શેડ્યૂલ કેસમાં જેવું છે, કારણ કે તમે કહો કે, જો આ કાર્ય બીજા કોઈ સાથે નથી, જો ત્યાં 0 છે, બી 0 નથી, બી 0 છે, તો ત્યાં છે 0 ત્યાં નથી, પણ તેનાથી તમે કહી શકતા નથી. તેથી, તેથી, હું આ બંને સમસ્યાઓ સિદ્ધાંતમાં હલ કરું છું અને તેમને મહત્તમ, તેમને વધુ સારી રીતે લઈશ. તેથી, આ ઈન્ટરેક્ટિવ માળખું છે, ક્યાંતો પહેલો અક્ષર મેચ કરે છે જેમાં હું શામેલ છું, તે મારા અનુક્રમ છે અને બાકીના ઉપપ્રોબ્લેમને હલ કરે છે. અથવા, પ્રથમ અક્ષર બે સબપ્રોબ્લેમ્સ જનરેટ કરે છે તે મેચમાં મેળ ખાતો નથી, એક શબ્દમાં દરેક અક્ષરને આગળ ધપાવીને અને હું મહત્તમને મહત્તમ લઈશ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 21:18)

તેથી, હંમેશની જેમ LCS હું કોમા જે કરીએ, આઈઆઈ અને બીજે થી શરૂ થતી સમસ્યાના એલસીએસ માટે ઊભા રહો. તેથી, જો Ai બરાબર બીજે છે, તો આપણે કહીએ છીએ કે, આઈજેની એલસીએસ 1 વત્તા 1 વત્તા 1 વત્તા 1 જા વત્તા 1 ની એલસીએસ છે, આ કહે છે, આપણે માનીશું કે એઆઈ સમાન બીજે સમાધાન નથી. અને પછી બાકીના ઈનપુટ સાથે આગળ વધો, જો તે ન હોય તો, પછી મારે તેમાંથી એક છોડવું પડશે. તેથી, કાં તો હું એલસીએસ આઈ પ્લસ 1 માટે LCS i plus 1 comma j જોઉં છું, હું બંનેને જોઉં છું અને પછી હું આમાંની મહત્તમતમ લે છે અને બીજી કોઈ વસ્તુ નથી, કારણ કે વર્તમાન મળતું નથી. તેથી, સૌથી લાંબી સામાન્ય સબવર્ડ સમસ્યા સાથે, અમે તે સ્થિતિને વિસ્તૃત કરીશું જે શબ્દને સમાપ્ત કરવા માટે શબ્દની બહાર છે. તેથી, આપણે 0 થી m વત્તા 1 અને 0 થી n વત્તા 1 થી જઈશું અને જ્યારે આપણી પાસે ઓછામાં ઓછું m વત્તા 1 અથવા n વત્તા 1 હશે, તો એલસીએસ સમસ્યા 0 આપશે, કારણ કે તે એક સામાન્ય અનુગામી હોઈ શકતી નથી, કારણ કે એક શબ્દો ખાલી રહેશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 22:19)

તેથી, એલસીએસમાં સબપ્રોબ્લેમ ડિપેન્સન્સી એલસીડબ્લ્યૂ કરતા થોડી વધારે જટીલ છે, એલસીડબ્લ્યૂ અમે ફક્ત આ નિર્ભરતા ધરાવતા હતા, તેવું આપણે કહ્યું છે કે, આઈજેએ મને વત્તા 1 જે વત્તા 1 પર આધાર રાખ્યો છે. પરંતુ, હવે અમે પણ વત્તા 1 જા અને ij પ્લસ 1 પર નિર્ભરતા ધરાવીએ છીએ. તેથી, અમારી પાસે નિર્ભરતા જમણી બાજુએ અને નીચલાથી પણ આવી રહી છે. તેથી, આપણી પાસે ત્રણ રીતની નિર્ભરતા છે કારણ કે આપણે જોયું છે કે આ બધા મૂલ્યો અહીં 0 હોવાના છે. તેથી, આ પ્રથમ નજીવી મૂલ્યવાન મૂલ્ય છે જે આપણે ગણતરી કરી શકીએ છીએ, કારણ કે તે બધા પાડોશીઓને જુએ છે તે ક્યાંય ક્યાંય નથી. તેથી, જો હું અહીં ઉદાહરણ તરીકે જોઉં છું, તો નીચેનાં પડોશીઓ નથી, જો હું અહીં પાડોશીઓને ન જોઉં તો. તેથી, એલસીએસ એમ કોમા એન, 5 અલ્પવિરામ 5 આ કિસ્સામાં, મૂલ્ય ગણતરી માટે ઉપલબ્ધ છે, કારણ કે તેની આજુબાજુની બધી બાબતો તેની આસપાસના ત્રણ અવલંબન ચોરસ બધા વસ્તીમાં છે. તેથી, હું તે કરી શકું છું અને પછી હું ફરીથી પંક્તિ દ્વારા પંક્તિ કરી શકું છું. એકવાર મને આ મળી જાય, હું આ કરી શકું, મારી પાસે ત્રણ વેલ્યુ હશે, એકવાર હું આ કરીશ, હું જઈ શકું અથવા હું ડાબી બાજુ જઈ શકું, હું આ કરી શકું છું અને તેથી, હું ત્રિપાત્રી કરી શકું છું, તેથી ચાલો તે કોલમ કરીએ. કૉલમ દ્વારા. તેથી, અમે બેઝ કેસથી પ્રારંભ કરીએ છીએ, જ્યાં LCS સીમા પર 0 છે, કારણ કે તમારી પાસે ખાલી શબ્દ સાથે સૌથી લાંબી સામાન્ય અનુગામી હોઈ શકતી નથી. પછી, અમે પ્રથમ કૉલમ ભરો અને અહીં આપણને 1 મળે છે, કારણ કે જ્યારે બે મેચ, આપણી પાસે 1 વત્તા સીએસઓ, હું પ્લસ 1, જે પ્લસ 1 છે. હવે, આપણો મતભેદ છે, તેથી જ્યારે તે મેળ ન આવે ત્યારે સી, તેથી જો હું c ને જોઉં અને તે મેચ ન થાય, તો હું શું કરું છું તેવું માનું છું, મને લાગે છે કે આમાંના મહત્તમમાં મહત્તમ લેવાનું છે. તો, આ મહત્તમ 2 છે, મને 1 મળે છે, હવે હું આમાં મહત્તમ 2 લે છે, મને 1 મળે છે, આમાં મહત્તમ 2, મને 1 મળે છે. તેથી, સૌથી લાંબી સામાન્ય સબવર્ડ સમસ્યા, તે કહે છે કે જો વર્તમાનને મેચ થતું નથી, તો મને 0 મળે છે, અહીં નથી, કારણ કે મને આ સેટ છોડવા અને માથા પર જવા દેવાની પરવાનગી છે. તેથી, તેથી, આ એક સાથે યોગ્ય રીતે મળે છે. એ જ રીતે, પહેલાની કોલમ સાથે આને મેળવવા માટેની એક મિલકત, કારણ કે કંઈપણ આર નથી. હવે, જ્યારે હું સી અને સી સુધી પહોંચું છું ત્યારે કેટલાક ઉમેરે છે, તે 1 વત્તા હું વત્તા 1 જે વત્તા 1 છે. તેથી, આપણે તે મેળવી શકીએ છીએ. તેવી જ રીતે, જ્યારે હું આગલા સ્તંભ પર જાઉં છું, ત્યારે હું ઈ અને ઇ સુધી પહોંચું છું, તે તે 1 વત્તા હું વત્તા 1 જે વત્તા 1 છે, તેથી મને 3 મળે છે. અને છેલ્લે, જ્યારે હું આ એસ અને એસ સુધી પહોંચું છું, મને આ 4 અને હવે મળે છે, આ 4 પ્રચાર કરે છે, કારણ કે અહીં હું આનો મહત્તમ ઉપયોગ કરું છું. તેથી, મને આ 4 માં મહત્તમ 2 મળે છે, મને 4 મળે છે. તેથી, આ ચોક્કસ કિસ્સામાં ખરેખર, 0

કામા 0 ની LCS એ મારો જવાબ છે, એલસીડબ્લ્યુમાં યાદ રાખો, શોધવા માટે સમગ્ર ગ્રિડની આસપાસ જુઓ, અમે મહત્તમ એલસીએસ, તમે તે કરવા નથી માંગતા. તમે 0 અલ્પવિરામ પ્રાપ્ત કરો છો તે મૂલ્ય એ તમે જે જવાબ શોધી રહ્યા છો તે કાર્ય કરે છે ...

(સ્લાઈડસમયનો સંદર્ભ લો: 25:07)

અને હવે તમે પાથને પાછળથી શોધી શકો તે પહેલાં, દરેક મૂલ્ય શા માટે ભરવામાં આવ્યું હતું, તે ભરવામાં આવ્યું હતું કારણ કે તે 1 વત્તા હું વત્તા 1, જે પ્લસ 1 અથવા તે ભરવા જાય છે, કારણ કે અન્ય બે નેટવર્ક્સ સાથે મહત્તમ, જો તેમ હોય તો તે મેચ હતી. તેથી, આ 4 ઘણું સ્પષ્ટ હતું, કારણ કે તે મહત્તમ હતું, કારણ કે એસ બી સમાન નથી. તેથી, તે નીચેથી આવ્યું છે અને આ 4 પણ એસ માંથી સમાન છે. તેથી, આ અહીંથી આવ્યું છે, તેથી તમે આ મૂલ્યને શોધી શકો છો અને જ્યાં પણ તમારી પાસે આ વિકર્ણ છે ત્યાં તે શોધી શકાય છે, કારણ કે કિંમત મેળ ખાતી છે. તેથી, વેલ્યુ 4 છે અને બરાબર ચાર ત્રાંસાત્મક પગલાં છે. અને પછી તળિયે એક, આ ચાર મેચો છે જે સૌથી લાંબી સામાન્ય અનુગામી રચના કરે છે અને તમે તે વસ્તુને વાંચી શકો છો અને આ અનુક્રમણિકા સંપ્રદાય બનાવે છે. તેથી, જેમ આપણે કહીએ તે પહેલા આપણે કહીએ કે તમે અંશને ગણતરી કરી શકો છો, તમે પાછા જઈ શકો છો અને તે ગણતરીને ફરીથી ચકાસી શકો છો અને સાક્ષીને શોધી શકો છો અને તેને ખરેખર કહેવામાં આવે છે અને ખરેખર જે શબ્દ ખરેખર આપે છે તેને મંજૂર કરવાની જરૂર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 26:07)

તેથી, એલસીડબ્લ્યુ માટે એલસીએસ માટેનો કોડ થોડો સરળ છે, કારણ કે આપણને આ મહત્તમ મૂલ્યનો ટ્રૅક રાખવાની જરૂર નથી અને વૈવિધ્યસભર થાય છે, કારણ કે આપણને માત્ર 0 પર મૂલ્ય શોધવાની જરૂર છે. સી. તેથી, પહેલા આપણે આર અને સી નો ઉપયોગ કરતા હતા કે જેથી થોડી સ્પષ્ટતા થાય કે પંક્તિઓ આ રીતે જાય છે અને કૉલમ આ રીતે જાય છે. તેથી, કૉલમ 0 થી n વત્તા 1 થી જાય છે અને પંક્તિઓ 0 થી એમ વત્તા 1 થી જાય છે. તેથી, અમે સીમાને 0 થી પ્રારંભ કરીએ છીએ, અને પછી આપણે કૉલમ પંક્તિ દ્વારા નીચેથી ઉપરથી પંક્તિ દ્વારા સ્તંભ કરીએ છીએ, જો બંને છે બરાબર, પછી તમે નીચે વત્તા 1 વત્તા ઉમેરો છો. જો બન્ને સમાન નથી, તો હું આ બે મૂલ્યોનો મહત્તમ ઉપયોગ કરું છું અને આખરે, જ્યારે આ ભરેલું છે, મેં કિંમત લખી છે, તે 0 અલ્પવિરામ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 26:53)

તેથી, આ છે એલસીડબ્લ્યુની જેમ, અમે મૂળરૂપે એમમાં ભરીએ છીએ, એન કદના કોષ્ટકમાં કોષ્ટકની દરેક એન્ટ્રી ગણતરી કરવી સહેલું છે, તે ત્રણ પાડોશીઓની દૃષ્ટિમાં ફક્ત સતત જ લે છે. અને તેથી, એકંદરે ગતિશીલ પ્રોગ્રામિંગનો ઉપયોગ કરીને, અમે ઓર્ડર એમ.એન. અલ્ગોરિધમનો પ્રદર્શન કર્યો છે, અમે સ્મૃતિના ખર્ચ પર મેમોઈઝેશનનો પણ ઉપયોગ કરી શકીએ છીએ.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 07
મૉડ્યૂલ - 05
લેક્ચર - 48
અંતર ફેરફાર કરો

સૌથી લાંબી સામાન્ય સબવર્ડ અને અનુગામી સમસ્યાને ધ્યાનમાં રાખીને, હવે આપણે અંતર ફેરફાર કરો નામની નજીકથી સંબંધિત સમસ્યા તરફ ધ્યાન આપીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:09)

તેથી, લક્ષ્ય એ કેવી રીતે બે સરખા ટેક્સચરને માપવા છે, તે કહેવાતું દસ્તાવેજ સમાનતા સમસ્યા છે. તેથી, ચાલો નીચે આપેલા બે વાક્યો જોઈએ; પ્રથમ વાક્યમાં જણાવાયું છે કે જે વિદ્યાર્થીઓને શ્રેષ્ઠ સબસ્ટ્રક્ચર પ્રોપર્ટી અને ડિઝાઈનિંગ એલ્ગોરિધમ્સમાં તેનો ઉપયોગ કદર કરવામાં સમર્થ છે. બીજા વાક્યમાં કહેવામાં આવ્યું છે કે વ્યાખ્યાનોએ વિદ્યાર્થીઓને પ્રશંસા કરી હતી કે અલ્ગોરિધમ્સના ડિઝાઈનમાં શ્રેષ્ઠ સબસ્ટ્રક્ચરના ખ્યાલનો ઉપયોગ કેવી રીતે થઈ શકે છે. તેથી, અહીં ત્રીજો વાક્ય સૂચવે છે કે કેવી રીતે આ બંનેમાંથી એકને બીજાથી મેળવી શકાય છે. જો તમે ચોક્કસ દસ્તાવેજ તૈયારી સિસ્ટમ્સનો ઉપયોગ કર્યો છે જે તમને ફેરફારોને ટ્રૅક કરવાની મંજૂરી આપે છે, તો તે સામાન્ય રીતે દસ્તાવેજના એક સંસ્કરણ અને આના જેવા અન્ય સંસ્કરણ વચ્ચેના ફેરફારોને સૂચવે છે. તેથી, અહીં લીલો અક્ષરો સૂચવે છે, તેથી જો આપણે આવૃત્તિ 1 અને આ સંસ્કરણ 2 તરીકે ઓળખાતા હોય, તો પછી આવૃત્તિ 1 થી આવૃત્તિ 2 માં જઈને, આપણે જે કર્યું છે તે છે કે તમે લીલોતરીમાં અક્ષરો રજૂ કર્યા છે. તેથી, તમે આ અક્ષરો શામેલ કર્યા છે, તમે તેમના દ્વારા રેખા સાથે લાલમાં ચિહ્નિત કરેલા અક્ષરો કાઢી નાખ્યા છે અને આ પીળામાં આપણે બદલાવી છે. તો, આપણે t ને v અને s દ્વારા n ને બદલીએ છીએ. તેથી, આપણી પાસે 28 અક્ષરો છે, આપણે ગણતરી કરી શકીએ કે 28 અક્ષરો છે જે જગ્યાઓ અને અન્ય સહિત શામેલ કરવામાં આવ્યા છે, 18 અક્ષરો કાઢી નાખવામાં આવ્યા છે અને 2 અક્ષરોને બદલવામાં આવ્યા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:41)

તો, હવે, આ એક માપ હોઈ શકે છે કે એકબીજાના નજીકના બે દસ્તાવેજો કેટલા નજીક છે. તેથી, એક દસ્તાવેજને બીજામાં પરિવર્તિત કરવા માટે ન્યૂનતમ ફેરફાર સંપાદન ક્રિયાઓ સાથે સંપાદન અંતર, તેથી અમારે અમારું નિર્ધારણ કરવું પડશે કે સંપાદન દ્વારા અમારું શું અર્થ છે. તેથી, ચાલો આપણે ફક્ત ખૂબ જ મૂળભૂત ઓપરેશનથી પ્રારંભ કરીએ, કાં તો તમે કોઈ અક્ષર શામેલ કરી શકો છો અથવા તમે એક અક્ષર કાઢી શકો છો અથવા તમે એક અક્ષરને એક પગલાથી બીજા સ્થાને બદલી શકો છો. તેથી, અલબત્ત, બદલવાનું વિચારીને અમને કાઢી નાખવા અને શામેલ કરવાનું એક વિચાર હોઈ શકે છે, તેથી તે બે પગલાની કામગીરી હોઈ શકે છે, હું પાત્રને કાઢી નાખું છું અને પછી હું જે પાત્રને ઈચ્છું છું તેમાં શામેલ છું. પરંતુ, હું એક અક્ષરને બી દ્વારા અથવા એક દ્વારા એક ઓપરેશન તરીકે બદલવાની પરવાનગી આપવા જઈ રહ્યો છું. તેથી, અમારા ઉદાહરણમાં અમે દાવો કર્યો છે કે 28 અક્ષરો શામેલ કરવામાં આવ્યા હતા, 18 કાઢી નાખવામાં આવ્યા હતા

અને 2 ને સ્થાનાંતરિત કરવામાં આવ્યા હતા. તેથી, આપણે કરેલા ફેરફારોની કુલ સંખ્યા 48 છે. જો સંપાદન અંતર ઓછામાં ઓછા ફેરફારોની સંખ્યા હોવાનું માનવામાં આવે છે, તો તે 48 કરતા વધુ હોઈ શકે નહીં, કારણ કે તેઓએ પહેલાથી બતાવ્યું છે કે તે 48 અક્ષરોમાં કરવાનું શક્ય છે, શક્ય તે કરવા માટે શક્ય છે. 48 કરતાં ઓછી હોંશિયાર રીતે. તેથી, અમે જે વાક્ય પહેલાં બતાવ્યું તે માટે મહત્તમ સંપાદન અંતર 48 જેટલું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:44)

તેથી, આ અંતરને લેવેનશિટેન(Levenshtein) અંતર પણ કહેવામાં આવે છે, કારણ કે સૌપ્રથમ સોવિયત દ્વારા સૂચિત કરવામાં આવ્યું હતું, હવે રશિયન વૈજ્ઞાનિક વ્લાદિમીર લેવેનશિટેન(Vladimir Levenshtein) કહેવાય છે અને આ સૌથી લાંબી સામાન્ય અનુગામી સમસ્યા છે, તે વ્યવહારમાં અત્યંત ઉપયોગી છે. . તેથી, પ્રથમ વસ્તુ જોડણી સુધારણા સૂચવવાનું છે. હવે, જો કોઈક ખોટું હોય તેવું કંઈક લખે છે, તો તે જોડણી સુધારક છે, તેને બદલવા માટે શબ્દકોશમાંથી યોગ્ય શબ્દ સૂચવવો પડશે. તેથી, કયા શબ્દ જોડણી સુધારક પસંદ કરવું જોઈએ? તેથી, પસંદ કરવા માટેનું એક માપદંડ શબ્દકોષમાંના બધા શબ્દોની વચ્ચે ઓળખવું છે જે શક્ય છે કે જે ટાઈપ કરવામાં આવેલ છે તે સૌથી નજીક છે. તેથી, આ સંપાદન અંતરની દ્રષ્ટિએ માપવામાં આવી શકે છે અને તે પછી તમે ઘણા નો અર્થ કરો છો.

((સમયનોસંદર્ભ લો: 03:29))

શું તમે ટાઈપિસ્ટનો અર્થ કરો છો. જ્યારે તમે ક્વેરીઝ અને શોધ એંજીન લખો ત્યારે પણ આ થાય છે. તેથી, જો તમે ગુગલ ને કંઈક લખો છો, તો ગુગલ કેટલીકવાર તમારી ક્વેરીને કોઈ શબ્દમાં બદલશે જે અર્થપૂર્ણ છે, કારણ કે તે ઓળખે છે કે તમે કોઈ નામ અથવા ખ્યાલ ખોટી રીતે લખ્યો છે. અમે એમ પણ કહ્યું હતું કે આપણે જે જોયું તે સૌથી લાંબી સામાન્ય અનુગામી સમસ્યા છે જેને આપણે અગાઉ જોયું છે, જૈનશાસ્ત્રમાં, બાયોઈન્ફોર્મેટિક્સમાં અને તે જ રીતે અંતરને પણ સંપાદિત કરો, જો તમે આનુવંશિક માહિતીની તુલના બે જુદી જુદી જાતિઓમાં કરી શકો છો, તો સરખામણી કરવી તે સ્વાભાવિક છે. તેમને ડીએનએ અને ડીએનએની સામગ્રીના સંદર્ભમાં માત્ર લાંબી લાકડીઓ હોય છે. તેથી, તમે શોધવા માંગો છો કે ડીએનએના એક ભાગને બીજામાં પરિવર્તન કરવું કેટલું સહેલું અથવા મુશ્કેલ છે અને તેના આધારે આપણે કહી શકીએ છીએ કે બે જાતિ એકબીજા માટે બંધ છે કે નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:15)

તેથી, જો તમે સૌથી લાંબી સામાન્ય અનુગામી સમસ્યા પર પાછા જાઓ છો, તો એક લાંબી સામાન્ય અનુગામી વિચારવાનો એક રીત એ છે કે મને લાગે છે કે હું ફક્ત તે બધા ભાગોને કાઢી નાખો જે એલસીએસનો ભાગ નથી, તેથી મારી પાસે દ્વિભાષી અને રહસ્ય જેવી વસ્તુઓ. તેથી, હવે, આમાં જો હું બીને કાઢી નાખીશ અને હું અહીંથી અને આર અને ઈને અહીંથી કાઢીશ, તો પછી હું સૌથી લાંબી સામાન્ય અનુગામી સાથે રહીશ. તેના વિશે વિચારવાની બીજી રીત એ ઓળખાણ છે, મેં એક શબ્દથી પ્રારંભ કર્યો છે અને હું તેને આ શબ્દમાં પરિવર્તિત કરી રહ્યો છું, તેથી હું પહેલા બી અને હું કાઢી નાખું છું અને પછી હું અહીં આર દાખલ કરું છું. તેથી, હું બંને શબ્દો પર કામ કરતો નથી અને બંનેમાંથી કાઢી નાખું છું, હું ફક્ત પ્રથમ શબ્દ પર જ કાર્ય કરે છે. તેથી, હું શબ્દોને ડીલીટ કરું છું, હું તે અક્ષરોને કાઢી નાખીશ જે મને જોઈતા નથી, કારણ કે મારી પાસે બીજું શબ્દ નથી અને હું અક્ષર શામેલ કરું છું, તેથી મને પહેલા જોઈએ છે કે બીજામાં પહેલામાં

નથી અને આ રીતે હું પરિવર્તન કરીશ. પ્રથમ શબ્દ બીજા શબ્દમાં છે અને આ બંનેમાંથી કાઢી નાખવા સમાન છે અને તેને સામાન્ય અનુગામીમાં આવે છે. તેથી, આ આપણને જણાવે છે કે જો LCS એ ફક્ત કાઢી નાંખો અને શામેલ કરવાની મંજૂરી આપી છે, તો સંપાદન અંતરની ગણતરી કરવાની સમકક્ષ છે. તેથી, સંપાદન અંતર વિશેની રસપ્રદ વાત એ છે કે તે એક અક્ષર અને અન્યના સ્થાનાંતરણને પણ મંજૂરી આપે છે. તેથી, તે લાંબી સામાન્ય અનુગામીથી અમને થોડું અલગ મેટ્રિક આપી શકે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:42)

તેથી, લાંબી સામાન્ય અનુગામી તરફ પાછા જવું, તે કહે છે કે જો કોઈ શબ્દની શરૂઆતમાં બે અક્ષરો મેળ ખાય છે, તો તે માનવું ઉપયોગી છે કે સામાન્ય અનુગામીમાં તે શામેલ છે. તેથી, મારી પાસે આ એક છે જે કહે છે કે Ai એ bj સમાન છે, તેથી હું આને મારા અનુગામીમાં શામેલ કરું છું અને પછી બાકીની સમસ્યાને ઉકેલું છું. નહિંતર, હું એઆઈ અથવા બી જે છોડીને મેળવેલી બે સબ સમસ્યાઓનો મહત્તમ ઉપયોગ કરું છું.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:13)

હવે, અંતરની સમસ્યામાં ફેરફાર કરવા માટે, આપણી પાસે સમાન માપદંડ છે, જો Ai એ બીજે સમાન છે, તો મારે કંઈ કરવાનું નથી. તેથી, i અને j થી શરૂ થતા શબ્દની સંપાદન અંતર એ પ્લસ 1 જે વત્તા 1 થી શરૂ થતી સંપાદન અંતર જેટલી જ છે. યાદ રાખો કે આપણે હવે લાંબી સામાન્ય અનુગામીની વિરુદ્ધ કરી રહ્યા છીએ, હવે આપણે સંખ્યા ઘટાડવાનો પ્રયાસ કરી રહ્યા છીએ મેચોની સંખ્યા વધારવા માટે ફેરફારો. તેથી, જો આપણે જોયું કે કંઈક પહેલાથી મેળ ખાતું હોય, તો કોઈ ફેરફારની જરૂર નથી, તેથી અમે આગળની સ્થાને આગળ વધીએ છીએ. હવે, જો તેઓ હવે મેળ ખાતા નથી તો ત્રણ વસ્તુઓ શક્ય છે, તેથી મારી પાસે ai, ai પ્લસ 1 અને so on છે અને મારી પાસે bj, bj પ્લસ 1 અને બીજું છે. તો, પ્રથમ વસ્તુમાં હું કરી શકું તે છે કે હું આને સીમાં સીધી બનાવી શકું છુંj, આ કિસ્સામાં મેં પહેલા અક્ષરોને બરાબર બરાબર સરખાવ્યા છે અને તેથી મને ફક્ત વત્તા 1 જે પ્લસ 1 જોવાની જરૂર છે. અન્ય વસ્તુ જે હું કરી શકું છું તે આ બધાને એકસાથે દૂર કરવું છે, હું તેને એક જ દૂર કરીશ. . તેથી, હવે, આ ખલેલ પડી ગઈ છે, પરંતુ હું હજી પણ બાકીની સમસ્યામાં જઈ રહ્યો છું, તેથી હવે, હવે જોવું પડશે કે હું પ્લસ 1 ને પછીથી જી જોડે છે. અને છેવટે, છેલ્લી વસ્તુ જે હું કરી શકું તે છે કે આ બીજે શરૂઆતમાં રજૂ કરવું, તેથી હવે, આનો અર્થ એ થાય કે આ બે વસ્તુઓ હવે મેચ થાય છે. તેથી, હવે, મને Ai આગળ અને બીજે પ્લસ 1 ને આગળ જોવું પડશે અને આમાંના ત્રણમાંથી કોઈપણને લઘુત્તમ કાર્યની જરૂર છે તે એક છે. તેથી, હું સોલ્યુશનમાં 1 વત્તા 1 વત્તા પ્લસ 1 અથવા 1 વત્તા 1 અથવા Ij પ્લસ 1 માટે 1 પ્લસ સોલ્યુશન i વત્તા 1 જા અથવા 1 વત્તા સોલ્યુશન અને આમાંના ત્રણનો ન્યૂનતમ લો.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:59)

તેથી, આ આપણને અંતિમ ઈન્ટરેક્ટિવ માળખું આપે છે જે આપણે જોઈએ છે. તેથી, i અને j પર શરૂ થતા શબ્દ માટે ED સંપાદન અંતર, જો Ai એ bj સમાન છે, તો તે i ના વત્તા 1 j વત્તા 1 છે. જો તે બરાબર નથી, તો તે સંપાદન અંતરની ઓછામાં ઓછી 1 વત્તા છે. આ ત્રણ જુદા પેટા સમસ્યાઓ અને સામાન્ય રીતે આપણે એમ વત્તા 1 એન વત્તા 1 સુધી મર્યાદિત છીએ, પરંતુ અર્થઘટન હવે અલગ છે. જો કોઈ એક શબ્દ ખાલી હોય તો, તમારે જે પરિવર્તન કરવાની જરૂર

છે તે સંખ્યા બીજું પરિવર્તન કરવાનો છે અથવા તમે જે શબ્દનો ઉપયોગ કરો છો તેનાથી બધું શામેલ કરો. તો, જો હું bj તરફ જોઉં છું, બીજે વત્તા 1 ને બીન અને તેની બાજુમાં કંઈ નથી, તો એડિટ અંતર શું છે. સારુ, મારે આ ઘણા બધા અક્ષરોને મૂળભૂત રીતે દાખલ કરવું પડશે. ત્યાં કેટલા અક્ષરો છે? એન n માઈનસ j પ્લસ 1, તેથી જ્યારે તમે ખાલી હોવ ત્યારે સંપાદન અંતર અને v પાસે z સ્થિતિ છે n માઈનસ j પ્લસ પ્લસ 1. તેવી જ રીતે, બી ખાલી છે અને તમે નથી, તે માઈનસ i પ્લસ 1 છે, આપણને આ બધા જેવા દાખલ કરો. તેથી, આ થોડો તફાવત છે, તે શૂન્ય નથી, પરંતુ તે અક્ષરોનો ઉપયોગ કરવાનો ખર્ચ છે, જે આ શબ્દમાં ખૂટે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:13)

તેથી, સંપાદન અંતરમાં ઈન્ડેક્ટીવ સબસ્ટ્રક્ચર બરાબર એ જ છે જે એલસીએસમાં છે. દરેક પોઝિશન હું વત્તા 1 જે વત્તા 1 પર આધાર રાખું છું અને જો કે હું વત્તા 1 જે અને આઈજે પ્લસ 1 છે. તેથી, અમારી પાસે ત્રણ સમાન પડોશીઓ સીને આધિન છે, જે આપણે પહેલા કર્યું હતું તેમ કર્યું છે અને અમારું હંમેશાં આપણે નીચે જમણા ખૂણે પ્રારંભ કરી શકીએ છીએ અને ખૂણા દ્વારા પંક્તિ અથવા ખૂણા દ્વારા પાછળની પંક્તિ પર કામ કરો. તેથી, આ કિસ્સામાં યાદ રાખો કે સીમાની સ્થિતિ શૂન્ય નથી, પરંતુ તે n માઈનસ j પ્લસ 1 છે, તેથી હું ઉપર જાઉં છું, સંખ્યા વધે છે. તેથી, આ મારી સીમા છે અને હવે હું ફક્ત મારા પુનરાવર્તિત વસ્તુને લાગુ કરી શકું છું જે ટી મેચો કહે છે, ટી પર અંતરની અંતર માટે શૂન્ય છે, કારણ કે મારી પાસે કશું કરવાનું નથી અને તેનાથી આગળ કંઈ નથી. બીજે ક્યાંય, કારણ કે હું મેચ નથી કરતો, સંપાદન અંતર ફક્ત બાકીની ત્રણની ન્યૂનત્તમ હશે. તેથી, બાકીના ત્રણ વત્તા 1 નો તે લઘુત્તમ છે, તેથી આ કિસ્સામાં જો હું ચૂંટો તો, ઉદાહરણ તરીકે આ જુઓ, 0, 1 અને 2 0 વત્તા 1 એ 1 છે. ન્યૂનતમ 4, 2 અને 3 એ છે. 2, પ્લસ 1 એ 3 અને તેથી વધુ છે, તેથી વર્તમાનમાં તે બનાવવા માટે તે ઓછામાં ઓછા તે વત્તા 1 છે. તેથી, હું આ કરવાનું ચાલુ રાખું છું અને હું ડાબી બાજુ જાઉં છું, આખરે હું શોધી શકું છું કે દ્વિભાષી અને રહસ્ય વચ્ચેની ન્યૂનતમ ફેરફાર અંતર ખરેખર 4 છે, મને પહેલા ચાર ફેરફારો કરવામાં આવ્યાં છે, ફક્ત આપણે અગાઉ કરેલી ઈન્ડિકિવિવ વ્યાખ્યાનો ઉપયોગ કરી રહ્યા છીએ. અને હવે પહેલાં પ્રશ્ન એ છે કે હું આમાંથી ઉકેલ કેવી રીતે પ્રાપ્ત કરી શકું છું, તેથી હું પાથને અનુસરીશ. તેથી, મને ત્યાં કેમ 4 મળ્યું, કારણ કે તે ન્યૂનતમ છે અથવા તે ત્રણ છે

((સમયનોસંદર્ભ લો: 10:53))

સમાન નથી. આ ત્રણમાંથી લઘુત્તમ શું છે? તે 3 છે, તેથી હું પાછો ગયો. એ જ રીતે, અહીં 3 શા માટે છે, કારણ કે આમાંનું આ લઘુત્તમ છે, તેથી હું નીચે ગયો? તેથી, દર વખતે જ્યારે હું નીચે જાઉં છું, ત્યારે તે કહે છે કે હું આઈજે થી હું વત્તા 1 જા પર જાઉં છું અને જો તમે અમારા કેસમાં યાદ રાખો છો, તો હું વત્તા 1 જ અને ii કાઢી નાખવાને અનુરૂપ છે. એ જ રીતે, હું આ સ્થળ પર જમણી બાજુ જાઉં છું. આનો અર્થ એ થાય કે હું આઈજે, આઈજે પ્લસ 1 થી જાઉં છું, આ શામેલ છે. તો, હું અહીંથી જે વાંચી શકું તે છે કે જ્યારે હું આ વી ને કાઢી નાખું છું, હું આને ડીલીટ કરું છું, તો હું અહીં આવીશ, હું આ આર દાખલ કરીશ અને હું આ દાખલ કરીશ. અને આ જ ચાર ફેરફારો છે જે મને ગુપ્તમાં દ્વિભાષી પરિવર્તન લાવવા માટે બનાવવાની જરૂર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:41)

તેથી, દરેક અંતર માટે સ્યુડો કોડ સૌથી લાંબી સામાન્ય અનુગામી સમાન છે. પ્રારંભમાં જ મોટો ફેરફાર કરવામાં આવ્યો છે, સીમા પરની કિંમતો શૂન્ય નથી, પરંતુ સંપાદન પૂર્ણ કરવા માટે જે પણ જરૂરી છે. પછી, હંમેશની જેમ હું નીચે જમણી બાજુએ જમણી તરફ શરૂ કરું છું અને હું જમણેથી ડાબે પંક્તિઓથી નીચેથી ટોચ પર કૉલમ કરું છું, જો વર્તમાન મૂલ્ય જો તમે જે મૂલ્યો હું આર અને v ની વિરુદ્ધ જોઈ રહ્યો છું તે સમાન છે, પછી મેં એડિટિંગ અંતરને આગળની વસ્તુ પર સ્થગિત કરી દીધું, મારે અહીં કરવાનું કંઈ નથી. નહિંતર, હું લઘુતમ સમસ્યાઓનો ઉપયોગ કરું છું અને વર્તમાન ફેરફારો માટે 1 ઉમેરીશ અને અંતે, મેં કિંમત 0 અને 0. પરત કરી છે

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:24)

જટિલતા ઓર્ડર m વખત n છે, તે બરાબર છે પાછલા એક જેટલા જ, કારણ કે કદના આ સમયના કોષ્ટકને ભર્યા પછી n અને તે ગણતરી કરવા સતત સમય લે છે. હવે, આ ત્રણેય સમસ્યાઓમાં એક બીજો મુદ્દો છે જેને આપણે સંબોધ્યો નથી. તેથી, ચાલો હવે તેની સાથે વ્યવહાર કરીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:41)

તેથી, આ એક જગ્યા જટીલતા છે. તો, આપણી પાસે આ કોષ્ટક છે જે આપણે ગણતરી કરવી પડશે જે આપણામાં ગ્રીન છે એમ વખત છે, પરંતુ નોંધ લો કે આપણે તેને ઉદાહરણ તરીકે કેવી રીતે ગણતરી કરી છે, આ કૉલમની ગણતરી કરવી, પછી આ કૉલમ, પછી આ કૉલમ અને બીજું ઘણું. તેથી, કોઈ પણ સમયે, જ્યારે હું આ બીજા કૉલમની ગણતરી કરી રહ્યો છું, ઉદાહરણ તરીકે, હું ફક્ત પ્રથમ કૉલમ વાંચું છું. જ્યારે મને ત્રીજી કૉલમની જરૂર હોય, ત્યારે હવે આ કૉલમની જરૂર નથી, કારણ કે ત્રીજી કૉલમ ફક્ત જમણી બાજુની બીજી કૉલમ પર આધારિત છે. તેથી, જો મારે કોલમ દ્વારા કોલમ ભરો તો મારે આગળના કૉલમ, મારા જમણા સ્તંભ અને વર્તમાન સ્તંભની જરૂર પડશે અથવા જો મને પંક્તિ દ્વારા પંક્તિ, વર્તમાન પંક્તિ અને વર્તમાન પંક્તિની જરૂર હોય. તેથી, બીજા શબ્દોમાં જ્યારે હું ફક્ત બે કૉલમ અથવા બે પંક્તિઓ રાખું છું, હું આ વસ્તુને 0 0 પર પૂર્ણપણે ગણતરી કરી શકું છું, તેથી, વાસ્તવમાં એમ ટાઈમ્સ n માપ ટેબલને સંગ્રહ તરીકે નિયમન કરવાની કોઈ જરૂર નથી. મારે હજુ પણ n વખત ગણતરી કરવી પડશે, પરંતુ સમય જટીલતા એ m વખત n રહે છે. તેથી, સમય એ એમ વખત છે કે આપણે શું કહી રહ્યા છીએ કે આ જગ્યા ઘટાડી શકાય છે, જો અમને એમ કહે છે કે n એ નાનું નાનું છે, તો આ જગ્યાને ફક્ત બે કૉલમ રાખીને ટોપર ને ઘટાડી શકાય છે. અને હવે તમે પૂછી શકો છો કે હું ઉકેલ કેવી રીતે પુનર્પ્રાપ્ત કરી શકું, કારણ કે સોલ્યુશનમાં હું આખી વસ્તુ પર પાથને પાછો શોધી શકું છું, તમે ખરેખર જેમ જ આગળ વધી રહ્યા છો તેમ ટ્રેકને વધારી શકો છો, તમે દરેક પ્રવેશને વધારીને સોલ્યુશન બનાવી શકો છો, તેથી પણ તે n જગ્યામાં કરી શકાય છે. તેથી, આપેલ આંકડાકીય ઉકેલ માટે સાક્ષીને ગણતરી કરવા માટે તમારે આ ટેબલની બિલકુલ જરૂર નથી. તેથી, આ બધી સમસ્યાઓમાં જ્યાં આપણી ગતિશીલ પ્રોગ્રામિંગમાં ખૂબ જ મર્યાદિત પડોશી અવલંબન છે, તમે ખરેખર ઘણી ઓછી જગ્યા કરી શકો છો, તે પછી તમને આપવામાં આવેલ વાસ્તવિક કોષ્ટકની દ્રષ્ટિએ તે આવશ્યક લાગે છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 07
મૉડ્યૂલ - 06
લેક્ચર - 49
મેટ્રિક્સ ગુણાકાર

આ અઠવાડિયામાં ડાયનામિક પ્રોગ્રામિંગના અમારા છેલ્લા ઉદાહરણ માટે, હવે આપણે મેટ્રિસના ક્રમને અસરકારક રીતે ગુણાકાર કરવાની સમસ્યા પર ધ્યાન આપીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:09)

તેથી, તમે કદાચ બે પરિમાણો A અને B ને ગુણાકાર કરવા માંગો છો, તેથી આપણને સુસંગત પરિમાણોની જરૂર છે, તેથી જો આપણી પાસે A અને B હોય, તો અમારે જરૂર છે કે A માં કૉલમ્સની સંખ્યા સંખ્યા સાથે મેચ કરવી આવશ્યક છે. બી માં પંક્તિઓ. તેથી, અમારી પાસે m વખત n મેટ્રિક્સ A અને n વખત p મેટ્રિક્સ બી છે અને અંતિમ મૅટ્રિક્સ તમારા સમય એમ ગુણ્યા પૃષ્ઠો બનશે, એ જ પંક્તિઓની સમાન સંખ્યામાં કૉલમ્સની સમાન સંખ્યા હોવી જોઈએ બી. અને તમે તે ઉત્પાદન AB ની ગણતરી કરો છો તે રીતે, તેથી જો હું એબીમાં આઈજે વી પ્રવેશ કરું, તો હું એમાં એ પંક્તિ હાંસલ કરીશ અને હું કેટલાકએ જે સ્તંભ બી. સાથે ગુણાકાર કર્યો છે તેથી, હું તે લઈશ અહીં પહેલી એન્ટ્રી, ત્યાં પહેલી એન્ટ્રી છે, તેથી હું એ 1 અને બી 1 જે લઈશ, તેમને એક એ 2 બી 2 જે વધારીશ, આખરે A i n b n j ગુણાકાર તેમને બધાને ઉમેરો. તેથી, આ ક્રમમાં n પગલાં લે છે, કારણ કે હું ક્રમમાં જોડી ઉત્પાદનો અને પછી તેમને બધા ઉમેરી રહ્યા છે. તેથી, તેથી, મારે છેલ્લે એમ ટાઈમ પી એન્ટ્રીઝની ગણતરી કરવી પડશે, દરેક પ્રવેશને ત્યાં રેખીય ઓર્ડર n ની જરૂર છે. તેથી, મૂળ અંકગણિત કામગીરીના સંદર્ભમાં બે મેટ્રિક્સનું ગુણાકાર કરવાની કુલ કિંમત એમ વખતના ગુણ n ગુણક્રમની છે. તેથી, આ મૂળભૂત તથ્ય છે કે આપણને સમસ્યાની જરૂર છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 01:34)

તેથી, હવે ચિંતા એ છે કે 3 અથવા વધુના ઉત્પાદનની ગણતરી કરીને બે પરિમાણોના ઉત્પાદનની ગણતરી કરી શકાતી નથી. તેથી, માની લો કે હું ત્રણ મેટ્રિસિસ એ, બી અને સીને ગુણાકાર કરવા માંગુ છું, એક સમયે હું ગુણાકાર પણ કરી શકું છું, તેથી મને 2 ની જરૂર છે. તો, મારે તેને એક વખત બી તરીકે ગુણાકાર કરવો જોઈએ, એક મધ્યવર્તી મેટ્રિક્સ એબીને દો અને પછી C દ્વારા ગુણાકાર કરો અથવા હું બી વખત C કરી શકું છું અને પછી બીમાંથી A દ્વારા ગુણાકાર કરીશ. તેથી, મેટ્રિક્સ ગુણાકાર એ બંધારણીય નથી, અમારે આ સામાન્ય રીતે નથી. તેથી, તે મહત્વપૂર્ણ છે કે ઓર્ડર સમાન છે, પરંતુ તે અનુક્રમમાં તે ક્રમમાં હું આ સરળીકરણ કરતો નથી. તેથી, તે એસોસિએટિવ છે, હું તેને એક કૌંસ બી દ્વારા અનુસરી શકું છું, C અથવા A વખત B પછી C દ્વારા અનુસરવામાં આવે છે, ક્યાં તો હું બે મેટ્રિક્સ ગુણાકાર કરી શકું છું અને કઈ સહયોગીતા એટલે કે જવાબ બદલાશે નહીં, અંતિમ ઉત્પાદન કરશે એ જ રહે છે. પરંતુ, આપણા માટે રસપ્રદ શું છે, હવે તે

છે કે જવાબની ગણતરી કરવાની જટિલતા, હું કયા ક્રમમાં તે કરું છું તેના આધારે બદલાઈ શકે છે, પછી ભલે હું એ વખત બી તરીકે અનુસરું કે બી દ્વારા અનુક્રમે બી અથવા પહેલા બી અને પછી ગુણાકાર કરી શકાય. એ.

(સ્લાઈડસમયનો સંદર્ભ લો: 02:47)

તેથી, ચાલો એક ખૂબ જ તુચ્છ ઉદાહરણ જોઈએ. તેથી, મારી પાસે આ પ્રકારની ત્રણ પરિમાણો છે, તેથી એ 1 છે અને આ મેટ્રિક્સ જે આની જેમ દેખાય છે, તેની પાસે 1 પંક્તિ અને 100 કૉલમ છે, બી એ એક મેટ્રિક્સ છે જે આ રીતે દેખાય છે, તેમાં 100 પંક્તિઓ અને 1 કૉલમ છે અને સી છે ફરીથી એક મેટ્રિક્સ, તેથી તે આના જેવું લાગે છે. તો, આ મારો એ છે, આ મારો બી છે અને આ મારો સી છે તેથી હવે, હું ધારું છું કે હું બી વખત સી કરીશ, જો હું બી વખત કરીશ તો શું થાય છે આ 100 માં હવે ફૂંકાય છે અને મને મળશે મોટા મેટ્રિક્સ જે 100 થી 100 છે. કારણ કે, મારી પાસે અહીં 100 પંક્તિઓ છે અને 100 સ્તંભ છે અને તેથી અંતિમ વસ્તુ બીમાં પંક્તિઓની સંખ્યા અને કોલમની સંખ્યા છે. તેથી, બી ટાઈમ્સ સી મેટ્રિક્સ ખરેખર 100 છે. 100 દ્વારા અને તે કેટલા પગલાં લેશે, તે m માં n માં લઈ જાય છે જે 100 માં 1 થી 100 માં 10,000 પગલા છે. હવે મારી પાસે એ 100 છે જે 100 છે અને બી 100 છે જે 100 થી 100 છે, તેથી હું ફરીથી કંઈક 100 બનાવું છું, 100 થી 100, જે તમને 100 માં 100 માં 100 માં લેવાનું છે, આપણે n ને n માં લઈએ છીએ. પી માં, બીજો શબ્દ

((સમયનોસંદર્ભ: 03:56)).

તેથી, એક સાથે આ ચોક્કસ અનુક્રમમાં 20,000 પગલાની જરૂર છે, બીજી બાજુ બીજી બાજુની આગાહી કરો, જો મેં પહેલા આ ઉત્પાદનની ગણતરી કરી હોય, તો પછી મેં તેને એક સરળ 1 થી 1 મેટ્રિક્સમાં પતન કર્યું છે. તો, અહીં પંક્તિઓની સંખ્યા 1 છે અને કોલમની સંખ્યા 1 છે, તેથી એબી 1 થી 1 મેટ્રિક્સ છે અને ગણતરી કરવા માટે તે ફક્ત 100 પગલાં લે છે. અને હવે, મારી પાસે 100 થી 1 મેટ્રિક્સની 1 થી 1 ગુણ છે, તેથી ફરીથી તે માત્ર 100 પગલાં લે છે, તેથી આ ફક્ત 200 પગલાં લે છે. તેથી, તમે જોઈ શકો છો કે, તમે જે કૌંસ પર છે તેના આધારે જટિલતામાં નાટકીય તફાવત હોઈ શકે છે, કૌંસ એ બી અથવા બી પછી અનુસરવામાં આવે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:33)

તેથી, સામાન્ય રીતે આપણી પાસે એમ જેમ કે મેટ્રિસનું અનુક્રમણિકા, તે બરાબર છે, તેથી એમ 1, એમ 2 અપ ટુ એમ એન અને તેમના પરિમાણો આર 1 સી 1, આર 2 સી 2 સુધી આરએન સી એન નંબરો પંક્તિઓ, કૉલમની સંખ્યા હશે. પરિમાણો આપણને આપવામાં આવશે, જેથી આપણે ગુણાકાર કરી શકીએ. ક્રમાંકની એન્ટ્રીઝને ધ્યાનમાં રાખવાની જરૂર છે તે સંખ્યા, સંવેદનાત્મક ગુણાકાર વધારાનાં ઓપરેશન્સ દ્વારા જોડાઈ શકે છે, જેને આપણે બે મેટ્રિસને ગુણાકાર કરવા માટે સક્ષમ થવાની જરૂર છે, તે પરિમાણો મેચ કરીશું. તેથી, ખાતરી કરો કે દરેક મેટ્રિક્સનું કૉલમ, કૉલમની સંખ્યા આગામી મેટ્રિક્સમાં પંક્તિઓની સંખ્યા જેટલી સમાન છે, જેથી તે ઉત્પાદન સારી રીતે વ્યાખ્યાયિત થયેલ છે. અને અમારું ધ્યેય આ ઉત્પાદનની ગણતરી કરવાની શ્રેષ્ઠ રીત શોધવાનું છે, બે મેટ્રિક્સનું ગુણાકાર કરવાના મૂળભૂત કાર્યક્રમોનો ક્રમ શું છે તે ઓછામાં ઓછા એકંદરે ખર્ચ મેળવવા માટે અમારે પ્રદર્શન કરવાની જરૂર છે. અને આ કૌંસનો મહત્તમ માર્ગ શોધવા સમાન છે. તેથી, યાદ રાખો કે મેં જ્યારે એ બી વખત બી સી કર્યું હતું, આ કૌંસને કેવી રીતે મૂકવું તે પસંદ કરવું, આને કૌંસ

કેવી રીતે મુકવું. સામાન્ય રીતે, જો હું તમને હવે એક વાર બી વખત સી વખત ડી આપીશ, તો તમે ઘણી બધી વસ્તુઓ કરી શકો છો, તમે એબી, સીડી કરી શકો છો અને પછી ગુણાકાર કરી શકો છો અથવા તમે સીડી કરી શકો છો, પછી બીસીડી, પછી એબીસીડી અથવા તમે એબી કરી શકો છો. એબીસી, પછી એબીસી ડી. તેથી, ત્યાં ઘણા જુદા જુદા માર્ગો છે જેમાં તમે આંશિક રીતે ઉત્પાદનના જોડી, જોડીના ઉત્પાદનો અને સમગ્ર વસ્તુનું નિર્માણ કરી શકો છો. તેથી, અમે આ કરવા માટેનો શ્રેષ્ઠ માર્ગ શોધવા માંગીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:58)

તેથી, ચાલો આપણે આ ભાગમાં ઈન્ડક્ટિવ માળખું અજમાવવા અને ઓળખીએ. તેથી, અમારું ધ્યેય આ લાંબી પ્રોડક્ટ એમ 1 થી એમ એનની ગણતરી કરવાનો છે, પરંતુ યાદ રાખો કે દરેક તબક્કે આપણે ફક્ત એક જ સમયે બે મેટ્રિક્સને વધારી શકીએ છીએ. તેથી, અંતિમ તબક્કે તમે બે સાદડી, કેટલાક બે જેવા મેટ્રિસને ગુણાકાર કરશો. હવે, આપણે કૌંસ દ્વારા ફરીથી જૂથબદ્ધ કરી શકીએ છીએ, પરંતુ આપણે ફરીથી ગોઠવી શકતા નથી. તેથી, જો આપણે અંતિમ તબક્કામાં કામ કર્યું હોત તો આપણી પાસે બે જૂથો હશે, તેથી તમે ડાબેથી મિડપોઈન્ટ અને કેટલાક ઉત્પાદનને મિડપોઈન્ટથી લઈને n સુધી કેટલાક ઉત્પાદનો કર્યા હોત. તેથી, કેટલાક કે માટે આપણે એમ એન થી એમ કે એમ અને એમ કે પ્લસ 1 થી એમ એન ની ગણતરી કરી હોત. આ બધા કર્યા પછી, પ્રથમ ભાગમાં, અમારી પાસે M1 ની જેમ ઘણી પંક્તિઓ હશે અને એમ કે તરીકે ઘણા બધા કૉલમ હશે, તેથી તે R 1 વખત C કે હશે. બીજું એક આર.કે. પ્લસ 1 થી સી એન હશે અને આપણે જાણીએ છીએ કે સી કે એ આરકે પ્લસ 1 ની બરાબર હશે, તેથી આ બંને એકબીજા સાથે ગુણાકાર થાય. તેથી, અંતિમ ખર્ચ m માં n માં આવશે, જે સી 1 માં સી કે માં n છે, તેથી આપણે જાણીએ છીએ કે અંતિમ પગલું કેટલું લે છે. માનવું છે કે તે એમ કે પર તૂટી ગયું હતું, છેલ્લા ગુણાકાર આને વધારે છે. હવે, આ ચોક્કસ પસંદગી સાથે તેને કરવાની કુલ કિંમત મેળવવા માટે, અમને એમ 1 અને એમ કે પ્લસ 1 થી એમ ના એમ 1 નું કમ્પ્યુટિંગ કરવાની કિંમત મળી રહેશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:30)

તેથી, આપણી પાસે આ અંતિમ પરિસ્થિતિ છે અને હવે આપણી પાસે આ બે પેટા સમસ્યાઓ છે, તેથી આપણી પાસે આ બે પેટા સમસ્યાઓ છે અને કુલ ખર્ચ પ્રથમ પેટા સમસ્યાના ખર્ચમાં છે. જો કે, એમ 1 થી એમ કે ગણતરી કરવા માટે ખૂબ જ લે છે; જોકે, એમ કે પ્લસ 1 થી એમ એન પ્લસની ગણતરી કરવા માટે આટલું જ લાગે છે, આ છેલ્લા ઉપાયની કિંમત જે આ બે પેટા સમસ્યાઓનું ગુણાકાર કરવા પછી થાય છે.ઈ મેટ્રિક્સ દરેકને તે બે મેટ્રિક્સને એક સાથે વધારવા માટે. પરંતુ, આપણે કહ્યું છે કે આ કે 1 અને એન વચ્ચે ક્યાંક હોઈ શકે છે, તેથી આપણે કેવા પસંદ કરીશું. તેથી, આ પ્રકારની સમસ્યાઓની ભાવના જે આપણે આ અનિશ્ચિત વસ્તુઓ જોયા છે તે એ છે કે આપણે પસંદગી કરવાનો પ્રયાસ કરતા નથી. અમે કહીએ છીએ કે અમને અગાઉથી કોઈ ખ્યાલ નથી કે કે સારો છે, તેથી અમે ફક્ત બધા સંભવિત કેસ અજમાવી જુઓ અને આમાં સૌથી શ્રેષ્ઠ મૂલ્ય લઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:21)

બીજા શબ્દોમાં કહીએ તો, ઓપરેશન્સની કુલ સંખ્યાના સંદર્ભમાં એમ 1 થી એમ ને ગુણાકાર કરવાની કિંમત એ 1 અને એન વચ્ચેના ન્યૂનતમ મૂલ્ય હોવા જોઈએ, તે એન કરતાં સખત રીતે ઓછી હોવી જોઈએ, કારણ કે બીજો ભાગ એનક

પ્લસ 1 હશે. તેથી, 1 અને એન વચ્ચે અથવા મેટ્રિક્સ 1 થી કે, ને મેટ્રિક્સ કે પ્લસ 1 થી n માં ગુણાકાર કરવો અને છેલ્લા ગુણાકારની કિંમત ઉમેરીને. તેથી, હવે બદલામાં જો હું તેને ઉદાહરણ તરીકે જોઉં છું, તો વિરામમાં, તે કેટલાક મધ્યવર્તી બિંદુ એમ તૂટી શકે છે. તેથી, હું કેટલાક એમ જે પ્લસ 1 ને એમ કેમાં ઉમેરીશ, તેથી મને મારી પેટા સમસ્યાઓના રૂપે 1 થી n સુધીના અવ્યવસ્થિત સેગમેન્ટ્સ મળશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:08)

તેથી, કુદરતી વાત એ છે કે એમ આઈ થી એમજેના અનિશ્ચિત સેગમેન્ટ પર ઈન્ડેક્ટિવ સ્ટ્રક્ચરને વ્યાખ્યાયિત કરવું, જ્યાં અલબત્ત, હું જે કરતા ઓછું છે. તેથી, અમે મધ્યમાંના કોઈપણ કે, લઘુતમ મૂલ્ય, આઈ અને જે વચ્ચે અથવા એમ આઈ માટે એમ કે અને ત્યારબાદ એમકે પ્લસ 1 થી એમજે અને કે જે છેલ્લા ગુણાકારની કિંમત છે તેના માટે ન્યૂનતમ મૂલ્ય જોઈએ છે જે સી 1 માં સી છે. કે જે સી માં. તેથી, અગાઉ આપણે ઈન્ડેક્સનો ઉપયોગ કરીશું, તેથી એમ 1 ની કિંમત લખવા, એમ 2 ની કિંમત લખવાને બદલે, જે પણ આપણે આઈજે લખીશું, તે માટે ij નો ખર્ચ એમ જેનો પૂર્ણાંક અનુક્રમે એમ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:47)

તેથી, ચાલો આપણે સમીકરણના અંતિમ ઈન્ડેક્સિવ સ્વરૂપને જોઈએ જે આપણે ગણતરી કરવાની જરૂર છે. તેથી, મૂળ કેસ તે છે જ્યારે આપણે લંબાઈ 1 નું અનુક્રમણિકા જોઈ રહ્યા છીએ, તેથી જો તમે ઉદાહરણ તરીકે ગણતરી કરી રહ્યા છો, એમ 1, એમ 2, એમ 3, એમ 4, તો એક શક્યતા છે જે એમ 1 વખત એમ તરીકે તોડી નાખે છે.એમ 2, એમ 3, એમ 4. હવે, જો હું આ વિરામ કરી રહ્યો છું તો એમ 1 કરવાની કિંમત કંઈ નથી, કારણ કે જો હું 1 થી 1 ની ગણતરી કરી રહ્યો છુ, તો હું ગુણાકાર કરું છું. તેથી, ii ની કિંમત 0 છે અને તે પછી હું ફરીથી પહેલાંની રીકર્સિવ ફોર્મ્યુલેશનનો ઉપયોગ કરું છું. અમે k નાં કક્ષાની ફોર્મ માટે લઘુત્તમ મૂલ્ય લઈએ છીએ, પરંતુ i કરતાં k ની કિંમત કરતાં કડક રીતે ઓછું નહીં, ખર્ચ કે પ્લસ 1 થી j અને ri સમય સી કે કે સી j. અને અલબત્ત, આપણે ફક્ત ત્યારે જ ગણતરી કરીશું જ્યારે હું j થી સમાન હોઉં, આપણે ક્યારેય સક્ષમ થઈ શકીશું નહીં, જ્યારે આપણે ઈન્ડેક્સ j એ કરતાં નાના હોય ત્યારે આપણે ક્યારેય ગણતરી કરવા માંગતા નથી. હવે, જ્યારે હું ખર્ચ ii વત્તા 1 જેવી વસ્તુ કરું ત્યારે શું થાય છે તે તપાસવાનું સૂચન આપવામાં આવે છે. તેથી, આ શું કહેશે કે હું એમ આઈ એમ હું પ્લસ 1 લેવા માંગું છું, હવે મારા માટે માત્ર એક જ સંભવિત મૂલ્ય છે, તેથી આ વચ્ચેનું હોવું જોઈએ હું અને હું પ્લસ 1 કરતાં સખત ઓછું છું. તેથી, તેનો મતલબ એ છે કે k માટે માત્ર એક જ સંભવિત મૂલ્ય છે, તેથી આ વસ્તુને M i થી M અને M i પ્લસ 1 થી M i પ્લસ પ્લસ 1 સુધી તોડી નાખશે અને પછી તે મને કિંમત 0 આપો અને પછી મને ફક્ત આ જોડીને વધારીને ખર્ચ થશે જે આ જોડીમાંથી બરાબર આવશે. તેથી, આ બેઝ કેસમાં કોઈ સમસ્યા નથી, જ્યારે બે સૂચકાંક બરાબર સમાન હોય ત્યારે આપણને માત્ર બેઝ કેસની જરૂર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:27)

તો, હવે, પહેલા, આપણી પાસે આ મેટ્રિક્સ ભરવા માટે હશે, કારણ કે આપણી પાસે આનો ઉપયોગ નથી, કારણ કે આપણને જરૂર છે, તેથી હું આ રીતે જઈ રહ્યો છું. તેથી, અમારી પાસે પહેલી અનુક્રમણિકા છે અને બીજી અનુક્રમણિકા તે રીતે જઈ રહી છે, તેથી મૂળભૂત રીતે પ્રારંભિક બિંદુ કરતા પહેલાંના મૂલ્યોને ધ્યાનમાં રાખવાની જરૂર નથી, તેથી અમારે

ફક્ત તે સમયની ટોચની જરૂર છે. અને હવે જો તમે પોઝિશન આઈજેની ગણતરી કરવાનો પ્રયાસ કરી રહ્યા છો, તો અમને કિંમત ik ની જરૂર પડશે, જે આ પંક્તિમાં છે, આપણને મૂલ્યની જરૂર પડશે એમ.જે. નાના મૂલ્યો માટે અને આપણને ફોર્મની કિંમતોની જરૂર પડશે જેકે, કેજે આ સ્તંભમાં અન્યમાં છે . તેથી, નીચે આપેલા અને ડાબી બાજુનાં મૂલ્યોની જરૂર પડશે જે વર્ણનાત્મક વિકર્ણ છે, તેથી આ કરવાનો એક રસ્તો આ રીતે ભરવાનો છે, જેથી આપણે જ્યાં પણ ભરીએ, ત્યાં આપણી પાસે પહેલાથી નીચેની કિંમતો હશે અને આપણી પાસે અન્ય મૂલ્યો પણ હશે અહીં બાકી. તેથી, આપણે આ મેટ્રિક્સને તળિયે ટોચથી ભરીશું, આપણે તેને ત્રિકોણથી પણ ભરી શકીએ છીએ, પરંતુ તે પ્રોગ્રામ કરવા માટે ખૂબ પીડાદાયક છે. તેથી, તે પંક્તિ અથવા કૉલમ દ્વારા ક્યાં તો પંક્તિ કરવી વધુ સારું છે, જેથી તમે આ રીતે પણ પ્રારંભ કરી શકો છો, તમે આ રીતે પણ કરી શકો છો. જ્યાંથી તમે શરૂ કરો છો ત્યાંથી, તમારી પાસે નીચે મૂલ્યો છે અને તમે તે મેળવી શકો છો, તેથી તમે ત્રિજ્યામાં અને બંનેને જમણે અથવા ઉપર શરૂ કરો. તેથી, ક્યાં તો તમે ઉપર ડાબેથી પ્રારંભ કરો અને કિનારી અને દરેક ત્રાંસા નીચે કાર્ય કરો, દરેક કૉલમ આપણે તળિયેથી ઉપર કરીએ છીએ. હવે, તમે તળિયે જમણે શરૂ કરો અને તમે ડાબેથી જમણી બાજુએ જમણી બાજુ અને દરેક પંક્તિને કાર્ય કરો, જેથી કાં તો આ થશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 13:01)

તેથી, અહીં તેના માટે સ્યુડો કોડ છે, તેથી આપણે પહેલા ત્રિકોણાકાર 0 થી પ્રારંભ કરીશું અને હવે આપણે જે કર્યું તે પહેલાં આપણે તે કરીશું, આપણે કહ્યું હતું કે આપણે બધા સ્તંભો શરૂ કરીએ છીએ

((સંદર્ભઆપો: 13 : 15)).

તેથી, આ ખરેખર એક કૉલમ માં હોવું જોઈએ. તો, કૉલમ 1 ને આપણે કરવાનું નથી કારણ કે તે ત્રિકોણ છે જેમાં આપણે કોલમ 2 ને આગળથી શરૂ કર્યું છે, તેથી 2 થી n. પછી પ્રત્યેક હાર માટે ત્રિજ્યાથી ટોચની હાર સુધી, આપણે હવે આ ન્યૂનતમ ગણતરી કરવાની જરૂર છે. તેથી, આપણે શરૂઆતમાં એવું માનીએ છીએ કે ફિલ્ટર કરવા માટેનું મૂલ્ય અનંત છે, હવે અનંત માત્ર તમામ પરિમાણો વત્તા 1 નું ઉત્પાદન કરવાની જરૂર પડી શકે છે, કારણ કે તમે જાણો છો કે

((સંદર્ભસમય: 13:42))

કિંમત વધી નથી તે તો, તમે મોટું મૂલ્ય પસંદ કરી શકો છો અને આપણે શું કરીએ છીએ તે હવે તમે દરેક મૂલ્ય માટે તપાસો છો, તમને તે આકર્ષક વસ્તુ મળે છે. તમે k થી k, k વત્તા 1 થી સી જુઓ અને પછી r વખત c k વખત સી સી જુઓ. તેથી, તમે આ વિશિષ્ટ મૂલ્યને જુઓ જે પ્રાસંગિક વસ્તુ છે અને જો તે અત્યાર સુધી તમે જોયેલી કિંમત કરતાં નાની છે, તેથી આ મૂળભૂત રીતે કમ્પ્યુટિંગ છે કે જે અનંતમાં સેટ કરીને અને પછી દર વખતે જોઈ અને અપડેટ કરીને k પર min. તેથી, આ માત્ર ઈન્ટરેક્ટિવ વસ્તુના પુનરાવર્તનનો સીધો અમલ છે અને તે માત્ર ઉપ-સમસ્યાઓનો સંદર્ભ આપે છે જે આધાર પર આધાર રાખે છે અથવા ઈનપુટ આપે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 14:29)

તેથી, એક રસપ્રદ વાત એ છે કે આપણે ઓર્ડર એન સ્ક્વેર ટેબલ ભરી રહ્યા છીએ, હવે જ્યારે આપણે સૌથી સામાન્ય અનુગામી અથવા અંતરને સંપાદિત કરીએ છીએ, ત્યારે અમે કહ્યું હતું કે સમયસર સમસ્યાની જટિલતા બરાબર ટેબલના કદની જટિલતા. અમારી પાસે એમ ટાઈમ્સ એમ ટેબલ હતી, તેને ભરવા માટે એમ ટાઈમ્સ એમનો સમય લાગ્યો હતો, કારણ કે તે બે સમસ્યાઓમાં અમે ત્રણ એન્ટ્રીઝને જોઈને દરેક એન્ટ્રી ભરી હતી. તેથી, અમારી પાસે એક સતત નજર હતી, હવે અહીં શું થાય છે તે છે કે જ્યારે હું આ એન્ટ્રીને ભરવા માટે કેટલાક અંતર પર ત્રિકોણને બંધ કરું છું, મને આ પંક્તિ અને આ કૉલમ્સને સ્કેન કરવાની જરૂર છે. તેથી, મારા મેટ્રિક્સમાં એક એન્ટ્રી ભરવા માટે જે સમય લાગે છે તે રેખીય હોઈ શકે છે. આત્યંતિક કિસ્સામાં જ્યારે હું ત્રિકોણમાં ખૂબ જ શક્તિશાળી છું, ત્યારે હું તે એન્ટ્રીને ભરવા માટે ઓર્ડર એન ટાઈમ કરી શકું છું. તેથી, મેટ્રિક્સના દરેક મધ્યવર્તી મૂલ્યને એન ઈન્ટરમિડિયેટ તરફ જોઈતી આવશ્યકતા હોઈ શકે છે, પ્રત્યેક મૂલ્યને પ્રત્યેક પોઝિશનને એન ઈન્ટરમિડિયેટ મૂલ્યની જરૂર પડી શકે છે. તેથી, જો આ કોષ્ટક કદ n ચોરસ છે, તો કોષ્ટક ભરવાની વાસ્તવિક જટિલતા કદ n ક્યુબ છે તેથી આ એક રસપ્રદ સમસ્યા છે, કારણ કે તે કહે છે કે તમે કદથી ડાયનેમિક પ્રોગ્રામિંગ એલ્ગોરિધમનો જટિલતા સીધો જ સમાપ્ત કરી શકતા નથી ટેબલની જે આપણે અપડેટ કરવાનો પ્રયાસ કરી રહ્યા છીએ. કારણ કે તે કોષ્ટકમાં પ્રત્યેક એન્ટ્રીને અપડેટ કરવા માટે તમારે કરેલા પ્રયત્નોની રકમ પર પણ આધાર રાખે છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 01
લેક્ચર - 50
લીનિયર પ્રોગ્રામિંગ

આ અભ્યાસક્રમના છેલ્લા અઠવાડિયામાં, આપણે કેટલાક અગ્રિમ વિષયો પર ધ્યાન આપ્યા. તો ચાલો આપણે લીનિયર પ્રોગ્રામિંગથી શરૂઆત કરીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:06)

તેથી, આપણે અત્યાર સુધી જે સમસ્યાઓનો સામનો કરવો જોઈએ તે મોટાભાગે ઑપ્ટિમાઈઝેશનનાં કાર્યો છે. તેથી, અમે ગ્રાફમાં ટૂંકા રસ્તાઓ શોધી રહ્યાં છીએ અથવા અમે ન્યૂનતમ ખર્ચ વિસ્તારવાના વૃક્ષને ઓળખવાનો પ્રયાસ કરી રહ્યાં છીએ અથવા અમે સૌથી લાંબી સામાન્ય સબ ક્રમ શોધી રહ્યા છીએ. તેથી, આપણે પાથની લંબાઈ અથવા વૃક્ષની કિંમત અથવા સબ ક્રમની લંબાઈને ઑપ્ટિમાઈઝ કરવાનો પ્રયાસ કરી રહ્યા છીએ અને પછી આ ઑપ્ટિમાઈઝેશન અવરોધના આધારે થાય છે. તેથી, જ્યારે આપણે પાથની શોધ કરીએ છીએ, ત્યારે આલેખ પોતે જ ગ્રાફ દ્વારા આપવામાં આવે છે, તેથી પાથને ગ્રાફમાં આપેલી ધારને અનુસરવું આવશ્યક છે. તેથી, જેમ ગ્રાફ એ જે પાથને ઓળખે છે તે બદલાશે, તે પણ બદલાશે. એ જ રીતે, આપણે આપેલ ગ્રાફના સંદર્ભમાં વૃક્ષો ફેલાવવાની માંગ કરી રહ્યા છીએ. તેથી, દરેક ગ્રાફમાં આપણે પહેલા અલગ અલગ વૃક્ષો ઓળખવા અને તેમની વચ્ચેની સૌથી નાની કિંમતની તપાસ કરવા માટે પૂછીએ છીએ. એ જ રીતે, બે સબ સિક્વન્સ સાથે અનુક્રમમાં વાસ્તવિક અક્ષરો નક્કી કરે છે કે તેમની વચ્ચે શું સામાન્ય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:02)

તેથી, આપણે રેખીયર પ્રોગ્રામિંગ કહેવાતા માળખામાં આવા અવરોધ ઑપ્ટિમાઈઝેશન સમસ્યાઓના નોડ સામાન્ય બનાવટને જોઈ શકીએ છીએ. તેથી, રેખીય પ્રોગ્રામિંગમાં આપણને કેટલીક વેરિયેબલ્સ આપવામાં આવે છે, કેટલીક માત્રા જે આપણે ગણતરી કરવા માંગીએ છીએ અને પછી કેટલાક રેખીય કાર્યો છે જે આ જથ્થાને અવરોધે છે. તેથી, રેખાંકિત ફંક્શન વેરિયેબલ x ની યાદ છે તે ફોર્મ x પ્લસ b ની કંઈક છે. તેથી, તેની પાસે x ચોરસ x ક્યુબ ટર્મ નથી, તે બધા રેખીય છે, તેથી આપણી પાસે x પ્લસ બી છે. તેથી, સામાન્ય રીતે જો આપણી પાસે બહુવિધ વેરિયેબલ xyz હોય, તો પછી ફોર્મ x પ્લસની અવરોધ હોઈ શકે છે અને વત્તા કંઈક સ્થિર અથવા x પ્લસ b વત્તા બરાબર કરતા ઓછા હોય છે અને વત્તા કંઈક સ્થિર કરતા સમાન હોય છે. અને પછી આ મૂલ્યો પરની અવરોધો છે જે ચલો લઈ શકે છે અને હવે અમારું લક્ષ્ય કેટલાક પ્રમાણને ઑપ્ટિમાઈઝ કરવું છે, આપણે કેટલાક કારણ અથવા કેટલાક વજન અથવા તેના જેવા કંઈકને મહત્તમ કરવા અથવા મહત્તમ કરવા માંગીએ છીએ અને તે જથ્થા દ્વારા ચલની દ્રષ્ટિએ વ્યક્ત કરવામાં આવે છે. હજુ સુધી એક અન્ય રેખીય કાર્ય. તો, આપણી પાસે કેટલાક ફંકશન છે જે હું ફરીથી x પ્લસ b y વડે લખું છું, પરંતુ તે એકબીજાના

અવરોધ માટે અને ઉદ્દેશ્ય કાર્ય માટે, અલબત્ત ગુણાંકનો એક અલગ સમૂહ છે. તો, આપણી પાસે એક અલગ રેખીય ફંકશન છે જે આપણને જણાવે છે કે આપણે તે ઑપ્ટિમાઈઝ કરવાનો પ્રયાસ કરી રહ્યા છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:17)

તેથી, રેખાંકન પ્રોગ્રામિંગને સમજવા માટેનો શ્રેષ્ઠ માર્ગ એ એક ઉદાહરણ જોવાનું છે. તેથી, ધારો કે અમે એક મીઠાઈની દુકાન ચલાવી રહ્યા છીએ જેને ગ્રાન્ડિઓઝ મીઠાઈ કહેવામાં આવે છે અને અમે બે પ્રકારની મીઠાઈઓ, બરફ્રિસ અને હલવા વેચીશું. હવે, આપણે જાણીએ છીએ કે બરફ્રિસના દરેક બૉક્સમાં 100 રૂપિયાનો નફો થાય છે અને હલવાના દરેક બૉક્સમાં 600 રૂપિયાનો નફો થાય છે. તેથી સ્પષ્ટપણે, બરફ્રિસ કરતાં વધુ હલવા બનાવવાનો અર્થ કરો. હવે, આપણે એ પણ જાણીએ છીએ કે આપેલ દિવસે 200 થી વધુ બરફી બૉક્સીસ વેચી શકતા નથી, અમે 300 થી વધુ હલવા બૉક્સ વેચી શકતા નથી અને એકસાથે સ્ટાફ ફક્ત 400 બોક્સ બનાવી શકે છે. તેથી, હવે જો તમે 400 બોક્સ બનાવો છો, તો આપણે ફક્ત 300 હલવા, 200 બરફ્રિસ વેચી શકીએ છીએ, તેથી અમારે બંનેને બનાવવાની જરૂર છે. તેથી, આ બાબતે આ અવરોધો આપવામાં આવે છે કે આપણે બરફ્રિસ અને હલવાના મિશ્રણને શું બનાવવું જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:13)

તો, ચાલો આપણે ચલોને ઓળખવા માટે પ્રયાસ કરી રહ્યા છે તે વેરિયેબલને ઓળખીએ. તેથી, હું આ મીઠાઈઓના દૈનિક ઉત્પાદન વિશે વાત કરી રહ્યો છું, તેથી અમે બે વેરિયેબલ્સ b અને h નો ઉપયોગ કરીશું જે દર્શાવે છે કે આપણે બરફ્રિસ અને હલવાનાં બોક્સની સંખ્યાને એક દિવસમાં રજૂ કરીએ છીએ. હવે, આ આપેલ છે કે આપણે જાણીએ છીએ કે આપણું નફો બરફ્રિસના 100 રૂપિયા અને હલવાના બોક્સ માટે 600 રૂપિયા છે. તેથી, તે 100 બી પ્લસ 600 એચ છે, આપણે કેટલી વેચી શકીએ તે વિશેની કેટલીક માહિતી પણ છે, તેથી તમે જાણો છો કે અમે એક દિવસ બરફીના 200 થી વધુ બોક્સ વેચી શકતા નથી. તેથી, b 200 થી ઓછા અથવા બરાબર હોવું જોઈએ અને આપણે એક દિવસ હલવાના 300 કરતા વધારે બોક્સ વેચી શકતા નથી, તેથી એચ 300 કરતા ઓછા અથવા બરાબર હોવું જોઈએ. અને અંતે, આપણે કહ્યું છે કે એકસાથે અમારા સ્ટાફ ફક્ત ઉત્પાદન કરવા સક્ષમ છે. 400 બૉક્સ બધા મળીને, બરફી અથવા હલવા કોઈપણ સંયોજનો અને અલબત્ત અલબત્ત, અમે માત્ર બરફ્રિનો બિન ઝીરો જથ્થો બનાવી શકીએ છીએ, તેથી અમે બરફ્રિસના ઓછા 7 બોક્સ બનાવી શકતા નથી. તેથી, આપણી પાસે આ અસ્પષ્ટ અવરોધ છે, આપણે કહીએ છીએ કે બ અને એચ બંને શૂન્ય કરતા વધારે હોવું આવશ્યક છે. તેથી, હવે આપણે આ બધાને અસમાનતાઓના સેટમાં મૂકી શકીએ છીએ અને પછી એક ઉદ્દેશ્ય કાર્ય કરી શકીએ છીએ અને તેને એક રેખીય પ્રોગ્રામ કહી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 04:29)

તેથી, અમારું ઉદ્દેશ્ય નફો વધારવાનો છે, તેથી આ નફો છે, તેથી આ તે છે જે આપણે મહત્તમ કરવા જઈ રહ્યા છીએ, જથ્થો 100 બી વત્તા 600 એચ અને આપણે સ્પષ્ટપણે પસંદ કરી શકતા નથી, જો બ અને એચ બદલાય છે, તો પછી આપણે તેમને જેટલું મોટું બનાવી શકીએ છીએ, પરંતુ આ અવરોધ છે. તેથી, b અને h અનુક્રમે 0 થી 200 અને 0 અને 300 વચ્ચે રહેવું જોઈએ અને સાથે મળીને તે 400 થી વધુ ન હોઈ શકે. તેથી, આ પાછલા પૃષ્ઠમાં આપણે લખેલી મર્યાદાઓ છે, હવે આપણે તેને નીચે લખી રહ્યા છીએ, સિસ્ટમ સમીકરણો અસમાનતા છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:04)

તેથી, આપણે આ કિસ્સામાં આને સહેલાઈથી કલ્પના કરી શકીએ છીએ, કારણ કે મૂળભૂત રીતે બી અને જથ્થાના બે જથ્થા છે જે આપણે ગણતરી કરવા માંગીએ છીએ અને આપણને તેમના વિશેની હકીકતોનો સમૂહ આપવામાં આવે છે. તેથી, આપણે જાણીએ છીએ કે b 0 થી 200 ની વચ્ચે છે, તેથી બી માટે શ્રેણી આ ક્ષેત્રમાં કંઈપણ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:24)

તે જ સમયે, આપણે પણ જાણીએ છીએ કે ધાર 0 થી 300 ની વચ્ચે છે, તેથી આ જગ્યાને અવરોધે છે જે આપણે આ જમણી તરફ જોઈ શકીએ છીએ

((સમય:05:30)).

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:33)

અને અંતે, આપણી પાસે અન્ય અવરોધ છે જે એક સાથે બી પ્લસ એચ 400 કરતા ઓછું હોવું જોઈએ, તેથી હવે તે આ રેખાઓની આ બાજુ છે. તેથી, હવે આપણે બી સીની આ બાજુમાં રહેવું પડશે.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:46)

તેથી, એકંદરે આ આપણને શક્ય ક્ષેત્ર કહી શકે છે, આ ક્ષેત્રમાં કોઈપણ બિંદુ b મૂલ્ય અને h મૂલ્યનું પ્રતિનિધિત્વ કરે છે જેનો અર્થ બધી અવરોધો છે. હવે, આપણે આપણા નફાકારક કાર્યનો પરિચય આપવો પડશે, તેથી અમારું ઉદ્દેશ 100 બી વત્તા 600 એચ છે અને આપણે જે મૂલ્યો B અને H ને પસંદ કરીએ તેના આધારે, આ સરવાળો સમાન હશે. તો, આપણે ચકાસી શકીએ છીએ કે આ જુદા જુદા મૂલ્યો માટે શું જુએ છે. તેથી, ઉદાહરણ તરીકે, જો તમે 60,000 હોવાનો અમારો નફો સેટ કર્યો છે, તો આ રેખા સાથેની કોઈપણ માત્રા, નારંગી સંભવિત ક્ષેત્રની અંદર બી અને એચ ની કોઈપણ કિંમત અમને 60,000 આપશે. આ બધા બિંદુઓ આપણને 100 બી વત્તા 600 એચ બરાબર 60,000 આપે છે. હવે, જો હું આ રેખાને આ ક્ષેત્રમાં ખસેડીશ, તો મૂલ્ય બદલાશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:34)

જો હું તેને આગળ ખસેડીશ અથવા હું 60,000 થી 1,20,000, 1 લાખ 20,000 સુધી જઈ શકું છું.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:40)

જો હું તેને હજી આગળ ખસેડીશ, તો મને 1 લાખ 50,000 મળશે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:46)

જ્યારે તે રેખા ખરેખર ક્ષેત્રને છોડે છે ત્યારે, તે ખૂણા પર, પીસીએમ વળાંક તરફ વળે છે કે નહીં તે આપણે શોધી કાઢીએ છીએ કે આપણે 1 લાખ 90,000 મેળવી શકીએ છીએ અને તે તારણ આપે છે કે આ ખરેખર શ્રેષ્ઠ બનશે મૂલ્ય હવે, આ શ્રેષ્ઠતમ મૂલ્ય આ પીસીએમના કોઈ ખૂણા પર થાય છે અને આ સામાન્ય રીતે કેસ છે, એક એવી દલીલ કરી શકે છે કે શ્રેષ્ઠ મૂલ્ય હંમેશાં એક શિખર પર થાય છે. કારણ કે, જેમ આપણે અહીં ફંક્શન જોયું છે તેમ આપણે લાઈનને ઑપ્ટિમાઈઝ કરવાનો પ્રયાસ કરી રહ્યા છીએ અને જેમ આપણે સંભવિત પ્રદેશમાં સફર કરીએ છીએ તેમ મૂલ્ય ક્યાં તો વધતો જાય છે અથવા ઘટાડો થાય છે અને જ્યાં સુધી તે પ્રદેશ છોડે ત્યાં સુધી તે થોડું કક્ષ હશે. હવે, તે આ ઈષ્ટતમ રેખાને સમજી શકે છે જે વાસ્તવમાં આ સમાન હોત, તો આ બંને શિરોબિંદુ અને આ કર્ણ બંને સંભવતઃ આ વાક્યની સાથે અને દરેક બિંદુ અનુકૂળ હશે. તેથી, એવું કહેવામાં આવતું નથી કે શિરોબિંદુઓથી શ્રેષ્ઠ મૂલ્યો દૂર નથી, પરંતુ ચોક્કસપણે ત્યાં અમુક શિરોબિંદુ હશે જેનો મહત્તમ મૂલ્ય છે. તેથી, અમારું ધ્યેય ફક્ત મહત્તમ ઉકેલ શોધવાનું છે, તે શિરોબિંદુ તરફ જોવું પૂરતું છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:44)

તેથી, હકીકતમાં, આ ખૂબ પ્રખ્યાત સરળ સિમ્પલેક્સ અલ્ગોરિધમ કેવી રીતે કાર્ય કરે છે. તેથી, તે શું કરે છે, તે આ સંભવિત ક્ષેત્રનું નિર્માણ કરે છે જે અવરોધથી ઘેરાયેલું છે અને પછી તે શિરચ્છેદ પસંદ કરે છે. તેથી, અમે તેને વિગતવાર વિગતવાર ચર્ચા કરીશું નહીં, પરંતુ તમારે એક ખૂણાના આ વિચારને સામાન્ય બનાવવું પડશે, બે પરિમાણોમાં શૂન્યતા શું છે તે કલ્પના કરવી સરળ છે. પરંતુ, અમારી પાસે ઘણીબધી અવરોધ છે જેમાં અમને બે કરતા વધુ પરિમાણોને ઑપ્ટિમાઈઝ કરવું પડશે, પછી તમારે યોગ્ય રીતે શિરોબિંદુને વ્યાખ્યાયિત કરવાની જરૂર છે, પરંતુ અંતર્દેશીય રૂપે આ સંભવિત ક્ષેત્રનો ખૂણો છે. તેથી, અમે આ શિર્ષકોમાંથી કોઈપણમાં પ્રારંભ કરીએ છીએ, ઉદ્દેશ્ય કાર્યનું મૂલ્યાંકન કરીએ છીએ, પછી આપણે પાડોશી શિરોબિંદુ તરફ ધ્યાન આપીએ છીએ. જો કોઈ નજીકના શિરચ્છેદ, કોઈ પાડોશીનું મૂલ્ય વધુ સારું હોય તો આપણે તે પાડોશી તરફ આગળ વધીએ, જો આપણા પડોશીઓમાંના કોઈ પણ આપણા કરતા વધુ સારા ન હોય, તો તમે ઉદ્દેશ્ય ચાલુ કરો છો, પછી આપણે બંધ કરીએ છીએ અને જાહેરાત કરીએ છીએ કે આ વાક્ય શ્રેષ્ઠ છે. તેથી, આપણે આ વ્યાખ્યાનમાં પુરાવા આપતા નથી કે, જો તે ખરેખર સાચું છે, પરંતુ તે દલીલ કરવી મુશ્કેલ નથી, પરંતુ અલબત્ત, ઘણી વિગતોસરળ રીતે અમલમાં મૂકવા માટે ich ને કામ કરવાની જરૂર છે. તેથી, સૈદ્ધાંતિક રીતે જે સિમ્પ્લેક્સ સૈદ્ધાંતિક છે તેમાંથી એક સમસ્યા ઘાતાંકીય હોઈ શકે છે, પરંતુ વાસ્તવમાં વ્યવહારમાં તે રેખીય પ્રોગ્રામ્સને હલ કરવા માટે એક ખૂબ જ અસરકારક અલ્ગોરિધમ તરીકે પરિણમે છે. હવે, સૈદ્ધાંતિક રીતે કાર્યક્ષમ અલ્ગોરિધમ્સ છે, તેથી રેખીય પ્રોગ્રામિંગ માટે પોલિનોમિયલ ટાઈમ એલ્ગોરિધમ છે, ત્યાં આંતરિક બિંદુ પદ્ધતિ કહેવામાં આવે છે અને નરેન્દ્ર કર્મકરને કારણે એક પદ્ધતિ છે જેને ઘણા લોકોએ સાંભળ્યું હશે. વર્ણન અને પ્રોગ્રામ્સ માટે આ વધુ જટીલ છે, સરળતા પ્રોગ્રામ માટે સંભવતઃ સરળ છે, પરંતુ જ્યાં સુધી આપણે ચિંતિત હોઈએ ત્યાં સુધી આપણે જે જાણવાની જરૂર છે તે છે કે એકવાર તમે અવરોધક સેટ અને એક રેખાત્મક અવરોધનો ઉપયોગ કરીને ઉદ્દેશ્ય કાર્ય સેટ કરી લો, તો તમે તેને હલ કરી શકો છો અસરકારક રીતે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:30)

તો, અલબત્ત, જ્યારે તમે સોલ્યુશન અસ્તિત્વમાં હો ત્યારે પણ તમે આવો છો. તેથી, આપણા કિસ્સામાં અમારી પાસે એક ઉકેલ હતો જ્યાં અમારી પાસે એક સંભવિત ક્ષેત્ર હતું જે આ પ્રકારના ટ્રેપેઝિયમ જેવું લાગે છે. તેથી, પહેલો મુદ્દો એ છે કે સંભવિત ક્ષેત્ર પહેલાથી જ સંક્ષિપ્ત હશે, ઉપગ્રહનો અર્થ એ થાય કે જો તમે આ પ્રદેશની અંદર કોઈ પણ બે બિંદુઓ લો અને તમે તેને લીટીથી કનેક્ટ કરી શકો છો, તો સમગ્ર રેખા પ્રદેશમાં રહે છે. તેથી, ઉદાહરણ તરીકે, જો મારી પાસે આ પ્રકારનો આકાર હોય તો, આ અભિવ્યક્તિ નથી, કારણ કે હું અંદર બે પોઈન્ટ પસંદ કરી શકું છું અને પછી એક લીટીથી જોડાયેલું છું અને આ ભાગમાં આ રેખા એ પ્રદેશને છોડી દે છે. તેથી, સંભવિત જગ્યા હંમેશા અભિવ્યક્ત થશે, પરંતુ તે ખાલી હોઈ શકે છે. ધારો કે, અમારી પાસે પહેલાની વસ્તુમાં વધારાની મર્યાદા હતી જે કહે છે કે બધું આ વાક્યની નીચે જ હોવું જોઈએ, હવે જ્યારે આ નારંગીની સાથે મને આ વાદળી અવરોધ હોય તો, ત્યાં કોઈ મુદ્દો નથી કે જે બધું જ સંતોષે. તેથી, અવરોધ અસંતોષકારક છે, સંભવિત ક્ષેત્ર ખાલી થઈ જાય છે અને ત્યાં કોઈ ઉકેલો નહીં હોય. બીજી શક્યતા એ છે કે આપણે તેને પૂરતી મર્યાદિત કરતા નથી, તેથી આપણે ફક્ત કહી શકીએ કે આપણે માત્ર શરૂ કર્યું છે, આપણે માત્ર એટલું જ કહીએ છીએ કે, ઉદાહરણ તરીકે કહીએ તો, એચ 300 કરતા ઓછું હોવું જોઈએ, પછી આ એક અવિભાજ્ય ક્ષેત્ર છે. તેથી, જેમ મેં મારી વાદળી રેખા દોરી છે તેમ મેં પહેલા કર્યું છે અને હું તેને ખસેડી રહ્યો છું, તે કહે છે કે નફો વધતો જાય છે, ત્યાં કોઈ ઉપલા પોઈન્ટ નથી. તેથી, આ પેથોલોજીકલ કેસો છે જ્યાં સંભવિત ક્ષેત્ર ખાલી છે અથવા તે અમર્યાદિત છે, જ્યાં તમને ઉકેલ મળી શકશે નહીં. પરંતુ, જો તે બંધાયેલું છે, તો પછી રેખીય પ્રોગ્રામિંગનો સિદ્ધાંત આપણને શું કહે છે તે છે કે ઉદ્દેશ્ય કાર્ય જૂઠું બોલશે, તે મહત્તમ અથવા ન્યૂનતમ સીમિત ક્ષેત્રની સીમા સાથે એક શિખર પર રહેશે અને તેથી તે ચકાસવા માટે તે પૂરતું છે. અને તે છે જે simplex કરે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:08)

તેથી, આના પર થોડો વધુ અભ્યાસ કરવા માટે, ચાલો આપણે ઉદાહરણને વિસ્તૃત કરીએ. તેથી, હવે, અમારી બરફી અને હલવા ઉપરાંત, જે બરફિસ માટે 100 રૂપિયાનો નફો પૂરો કરતાં સમાન વિગતો છે, 600 માટે હલવા માટે અમે બદામ રાસમાલાઈ ઉમેરી છે જે આપણને 1300 નું વધારે નફો આપે છે. વાસ્તવમાં લોકો બદામ રાસમાલાઈ અમર્યાદિત જથ્થામાં ખરીદી. તેથી, બરફિસ અને હલવા માટેની અગાઉની માંગ સમાન છે, 200 અને 300, પરંતુ બદામ રાસમાલાઈને અમર્યાદિત રકમમાં વેચી શકાય છે. દુર્ભાગ્યે, અમને કોઈ નવા સ્ટાફ મળ્યા નથી, તેથી અમે હજી પણ કુલ 400 બૉક્સીસ બનાવવા માટે પ્રતિબંધિત છીએ અને હવે અમારી પાસે એક વધારાનું અવરોધ છે જે આપણને મળતા દૂધ સાથે આપણે શું કરી શકીએ તેના સંદર્ભમાં છે. તેથી, દૂધ કે જે આપણને મળે છે તે સાથે આપણે એક દિવસમાં હલવાનાં 600 બૉક્સીસ બનાવી શકીએ છીએ અથવા રાસમાલાઈ અથવા તેના કોઈપણ મિશ્રણને જોઈ શકીએ છીએ. તેથી, રાસમાલાઈના અસરકારક રીતે બોક્સને હલવાના બોક્સની જેમ દૂધ કરતાં ત્રણ ગણી વધારે જરૂરી છે. તેથી, હવે ફરીથી, આ અવરોધો આપવામાં આવે છે, આપણે જાણવા માંગીએ છીએ કે સૌથી શ્રેષ્ઠ ઉત્પાદન શેડ્યૂલ કે જેને આપણે અમારી મીઠી દુકાન પર કંપોઝ કરીશું.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 12:17)

તેથી, જો આપણે સમીકરણને હવે જોવું જોઈએ, તો હવે ઉદ્દેશ્ય કાર્યમાં વધારાની માત્રા છે જે રાસમલાઈથી મળે છે. તો, આપણી પાસે અતિરિક્ત વેરિયેબલ છે, આપણી પાસે b અને h જે પહેલા બરફ્રિ અને હલવા અને rasmalai માટે r છે, તેથી આપણે બરફ્રી દીઠ બૉક્સ દીઠ 100 રૂપિયા, હલવાનાં બૉક્સ દીઠ 600 રૂપિયા અને રાસમાલાઈના 1300 બૉક્સ મેળવી શકીએ છીએ. બરફ્રી અને હલવા માંગ પરની મર્યાદાઓ અને રાસમાલાઈ માટે કોઈ અવરોધ નથી, તેથી અમે ત્યાં કંઈપણ ઉમેરી શકતા નથી. કુલ ઉત્પાદનમાં હવે રસમલાઈનો સમાવેશ થાય છે, તેથી આ ત્રણેય મળીને વાદળી હોવા જોઈએ અને આ દૂધની મર્યાદાને વ્યક્ત કરે છે. તેથી, આ એકદમ એકદમ છે, હું હલવાનાં 600 બોક્સ બનાવી શકું છું, બીજી બાજુ હું રાસમાલાઈના 200 બોક્સ બનાવી શકું છું, કારણ કે 200 વખત 600 રૂપિયા બદલે છે. પરંતુ, હું કોઈ પણ સંયોજન કરી શકું છું અને તે ઉત્પન્ન કરશે, તે જ દૂધ અથવા તેના જેટલું ઓછું જરૂરી છે. તેથી, એચ પ્લસ 3 આર 600 ની બરાબર કરતા ઓછું હોવું જોઈએ, તેથી આ દૂધની અવરોધ છે. અને આખરે, આ તમામ ત્રણ જથ્થાઓ 0 કરતા મોટી અથવા સમાન હોવી આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:20)

તેથી, આ કિસ્સામાં જો આપણે આ ચિત્ર દોરીએ, તો તે હવે એક ત્રિપરિમાણીય ભૂમિતિ બની જશે, કારણ કે આપણી પાસે ત્રણ અક્ષ બી, એચ અને હવે આર ઊભી દિશામાં આવી રહ્યા છે અને હવે રેખાઓના બદલે અવરોધો આવે છે, તે આ યોજના બની જાય છે. તેથી, બહુકોણને બદલે હવે બન્યા, અમારી પાસે પોલિટોપ કહેવામાં આવે છે, આપણી પાસે ત્રણ પરિમાણીય પદાર્થ છે. અને ફરીથી આપણે શોધીશું કે તમે કોઈ ખૂણામાં શ્રેષ્ઠ છો અને આ કિસ્સામાં શ્રેષ્ઠતમ બને છે કે તમે શૂન્ય બરફ્રિસ કરો છો, તમે પૂર્ણ રૂપે 300 હલવા કરો છો અને પછી તમે બાકીના રાસમાલાઈ માટે કરો છો અને પછી તમને નફો મળે છે 3 લાખ 10,000

(સ્લાઈડસમયનો સંદર્ભ લો: 14:01)

તો, તેથી આપણે કહી શકીએ કે આ ખરેખર કેસ છે? તો, શું આ ખરેખર અમારી શ્રેષ્ઠ વસ્તુ છે? તેથી, યાદ રાખો કે જો હું આ કરું, તો 300 માં 600 મળશે, તેથી મને અહીંથી 1 લાખ 80,000 મળે છે અને તમને અહીં 1 લાખ 30,000 મળે છે અને આ રીતે આપણને 3 લાખ 10,000 મળે છે. તેથી, હું કેવી રીતે જાણી શકું કે આ જવાબ ખરેખર શ્રેષ્ઠ છે? તેથી, અહીં એક રસપ્રદ નિરીક્ષણ છે જે આપણે કરી શકીએ છીએ. તેથી, આમાંની કેટલીક અવરોધો છે જે આપણે પહેલા કરી હતી, આપણે ફક્ત તે જ દૂર કર્યું હતું જે કહે છે કે ત્રણ મૂલ્યો B, H અને R શૂન્ય કરતા વધારે છે, કારણ કે તે ઉપયોગી નથી. હવે, આને ધ્યાનમાં રાખીને હું એ, બી અને સી ના લેબલની ત્રણ અવરોધો લે છે, સામાન્ય રીતે જો હું અવરોધ લઉં અને હું ગુણાકાર કરું, તો જો હું કહું કે 2 એચ 600 ની બરાબર કરતા ઓછી છે. તો, આપણે સાદા બીજગણિતમાંથી જાણીએ છીએ. કે જે એચ બરાબર 300 ની બરાબર છે તે જ છે. તો, મારે તેને વધારી શકો છો, તે એક સતત દ્વારા અવરોધ છે, હું બહાર આવીશ, તેથી તેને એક સાથે જોડો. હું A ના કેટલાક સંયોજનો લઈ શકું છું બી સાથે કેટલાક સંયોજનો અને વત્તા C ના કેટલાક સંયોજનો, તે આપણને કશું નવું કહેતું નથી, અમે તેને એક અલગ ફોર્મેટમાં જોડીએ છીએ. તેથી, મને લાગે છે કે હું 100 વાર આ લે છે અને હું ફરીથી 100 વખત બી લે છે અને હું 400 ગણી સી લે છે અને હું તેને ઉમેરીશ, તેથી મને 100 એચ પ્લસ 100 એચ પ્લસ 400 એચ મળશે જે 600 એચ છે. પછી મને 100 બી મળશે અને અંતે, મને 1200 વત્તા 100 મળશે, તે 1300 આર છે. તેથી, આ સંયોજન સાથે એ, બી અને સીને સંયોજિત કરો અથવા હું 100 દ્વારા A,

B થી 100, C 400 દ્વારા ગુણાકાર કરું છું, હું નવી અસમાનતા ઉત્પન્ન કરું છું. પરંતુ, આ નવી અસમાનતા અમને કંઈક રસપ્રદ જણાવે છે, કારણ કે હકીકતમાં ડાબી બાજુએ આપણી પાસે જે છે તે આપણું નફા છે. તેથી, નફો બરાબર 100 બી વત્તા 600 એચ, 1300 આર છે, આપણે આને મહત્તમ કરવાનો પ્રયાસ કરી રહ્યા છીએ અને આપણી મર્યાદાઓ આપણને શું કહે છે તે છે કે સંભવિત ક્ષેત્રમાં, આ મૂલ્ય ત્રાણથી વધુ હોઈ શકે

((સમયનોસંદર્ભ: 16:05)),

3 લાખ 10,000 હોવા જોઈએ. બીજા શબ્દોમાં કહીએ તો, આ ખરેખર શ્રેષ્ઠતમ નફો છે, કારણ કે આપણી અવરોધોથી આપણે આ હકીકત મેળવી શકીએ છીએ કે આપણે તેનાથી વધુ કંઈ મેળવી શકતા નથી, તેથી આ એક શ્રેષ્ઠ મૂલ્ય હોવું જોઈએ. તેથી, આ વિશિષ્ટ ઉદાહરણમાં એવું લાગે છે કે આપણે પોતાને સાબિત કરી શકીએ છીએ કે અમે તે મર્યાદાથી ચપળ કંઈક કરીને શ્રેષ્ઠતમ પ્રાપ્ત કર્યું છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 16:34)

તેથી, તે સદભાગ્યે બહાર આવ્યું છે કે આ એક સંયોગ નથી, તે તારણ આપે છે કે તમે હંમેશાં આવા સંયોજનો બનાવી શકો છો. તેથી, જો મારી પાસે સી -1, સી 2 ની અવરોધ હોય, તો આ મારા અવરોધ સમીકરણો છે, હું હંમેશાં કેટલાક સંયોજન શોધી શકું છું. તેથી, હું કેટલાક લેમ્બાડા 1 સી 1, લેમ્બેડા 2 સી 2 લઈ શકું છું, હું તેને ઉમેરી શકું છું અને પછી હું તે ઉપરના બાઉન્ડમાંથી મેળવી શકું છું અને તે ઉપલા બાઉન્ડ ખરેખર મને કહેશે કે, મને મળેલ સોલ્યુશન બરાબર નથી. તેથી, આને એલપીનું દ્વિઅવળું કહેવામાં આવે છે અને ફરી આપણે આ સિદ્ધાંતને ધ્યાનમાં લઈશું નહીં, પરંતુ તે જાણવું ઉપયોગી છે કે અમે ફોર્મ્યુલેશન લઈ શકીએ છીએ જે અમે એલ.પી. માટે શરૂ કરીએ છીએ અને પછી આપણે બીજા પ્રશ્ન પૂછી શકીએ છીએ. જે કેટલાક રેખીય મલ્ટિલાયર્સ સાથેના અવરોધોને સંયોજિત કરીને આપણે મેળવી શકીએ તે ન્યૂનતમ મૂલ્ય છે. તો, આપણે નવા વેરીએબલો મેળવીએ છીએ જે બરાબર આ ગુણાકાર છે. લેમ્બડા 1 થી લમ્બેડા કે જે મૂલ્યોને ઘટાડે છે તે લામ્બેડા 1 સી 1 વત્તા લેમ્બાડા 2 સી 2 વત્તા લેમ્બાડા કે સી કે શું છે? તો, આ મારો નવો પ્રશ્ન છે, હું આને ઘટાડવા માંગું છું અને પછી આ બીજી એલપી સમસ્યા બની જાય છે અને હું તેને ઉકેલું છું કે ડ્યુઅલ અને અસલની પાસે એક ઉકેલ છે, જો અને તે માત્ર ત્યારે જ જો તે ઉકેલ બંને માટે શ્રેષ્ઠ હોય. તેથી, આ અમને વાજબી ઠરાવવાનો એક રસ્તો આપે છે કે મેં રેખીય પ્રોગ્રામિંગ સમસ્યાને યોગ્ય રીતે ઉકેલી છે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 02
લેક્ચર - 51
એલપી મોડેલિંગ: પ્રોડક્શન પ્લાનિંગ

ચાલો આપણે ફરીથી ઉત્પાદન કરવા સાથે લીનિયર પ્રોગ્રામિંગનો ઉપયોગ કરીને સમસ્યાનું મોડેલિંગ કરવાના બીજા ઉદાહરણને જોઈએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:09)

તેથી, યાદ રાખો કે એક રેખીય પ્રોગ્રામ ઑપ્ટિમાઈઝેશન સમસ્યા છે, જ્યાં તમારી પાસે કેટલાક ચલો હોય છે જે તમે ગણતરી કરવા માંગો છો તેનું વર્ણન કરે છે. અને હવે તમારી પાસે આ ચલો પર લીનિયર અવરોધ છે, તેમજ એક રેખીય ફંક્શન છે જે તમે તે કરવા માંગો છો કે જેને તમે ઑપ્ટિમાઈઝ, મહત્તમ અથવા ઘટાડવા માંગો છો, જેને ઉદ્દેશ્ય કાર્ય કહેવામાં આવે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:28)

અને રેખીય પ્રોગ્રામને ઉકેલવાનો એક રસ્તો તે ભૂમિતિથી વિચારીને સિમ્પલેક્સ અલ્ગોરિધમનો ઉપયોગ કરે છે અને સિમ્પલેક્ષ અલ્ગોરિધમનો એ હકીકત છે કે રેખીય પ્રોગ્રામનો મહત્તમ મૂલ્ય હંમેશાં અસ્પષ્ટતાના શિખર પર છે. તેથી, તે કેટલાક શિરોબિંદુથી શરૂ થાય છે અને તે એક શિરચ્છેદથી પડોશી સુધી જાય છે, જ્યાં સુધી તે કોઈ શંકુ શોધી ન લે ત્યાં સુધી તેનું મૂલ્ય પડોશી છે અને દાવો એ છે કે કર્ણ ખરેખર સોલ્યુશનનું પ્રતિનિધિત્વ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:56)

અમે પણ કહ્યું હતું કે આપણે દલીલ કરી શકીએ છીએ કે આ ટ્વિઅલ બનાવીને આ શ્રેષ્ઠતમ મૂલ્ય છે, જે મિશ્રણને ઘટાડવા માટે અને મિશ્રણને ઘટાડવા માટે અને મૂળ અને ટ્વિ ખરેખર શ્રેષ્ઠતમ મૂલ્ય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:13)

તેથી, આગામી ઉદાહરણ આપણે કાર્પેટ મેન્યુફેક્ચરિંગ કંપની તરફ જોવું છે. તેથી, અમારી પાસે એક કંપની છે જે હાથથી બનાવવામાં આવતી કાર્પેટ બનાવે છે અને હાલમાં અમે કર્મચારી 30 કર્મચારીઓ છે, દરેક કર્મચારી એક મહિનામાં 20 કાર્પેટ બનાવે છે અને તેના પગાર 20,000 રૂપિયા છે. તેથી, જો આપણે ફક્ત કાર્પેટની કિંમત જોઈશું, તો પછી દરેક કાર્પેટ બનાવવા માટે અમે 1000 રૂપિયા ચૂકવીશું. હવે, તે તારણ આપે છે કે અમારી માસિક માંગ મોસમી છે, તેથી કેટલાક કાર્ય ડેટાને જોયા બાદ અમને વર્ષનાં દરેક મહિનામાં વેચાણ કરવાની અપેક્ષા રાખી કાર્પેટ્સની રકમનો અંદાજ બનાવવામાં આવે છે. તેથી, અમારી પાસે 440 થી 920 ની માંગની રેન્જ છે, તેથી યાદ રાખો કે તમારી પાસે 30 કર્મચારીઓ અને 20 કાર્પેટ છે, હું દર મહિને 600 કાર્પેટ્સ બનાવી શકું છું, આ હું વર્તમાન લિસ્સા સાથે કરી શકું છું. હવે, 440 જેટલી માંગ ઓછી છે. તેથી, જો હું 600 કાર્પેટ કરું છું, હવે જો મારી પાસે 30 કર્મચારીઓ હોય અને દરેક એક મહિનામાં 20 કાર્પેટ બનાવશે, તો હું તે પાછું મેળવીશ. તેથી, હું કાર્પેટને આદર્શ બનાવવા માટે ટકાવારી આપી રહ્યો છું, તેથી દરેક કર્મચારી એક મહિનામાં 20 કાર્પેટ બનાવશે. તેથી, હું હંમેશા 600 કાર્પેટ બનાવશે, પણ મારી પાસે 160 જોડી હશે જે હું વેચી શકતો નથી. પરંતુ, ત્યાં કેટલાક મહિના છે જ્યાં મને 320 ની માંગ હશે અને હવે મારી પાસે વેચવા માટે પૂરતી કાર્પેટ છે કે કેમ તેનો પ્રશ્ન છે. તેથી, હું જાન્યુઆરીથી ડિસેમ્બરની માંગને જાણું છું અને કહે છે કે આ જાન્યુઆરી માટે ડી 1, ફેબ્રુઆરી માટે ડી 2 અને ડિસેમ્બર માટે 12 સુધી સંગ્રહિત છે. તેથી, મારી પાસે આ 12 માત્રા છે જે ચોખ્ખા ગુણ તરીકે જાણીતી છે, આ કેટલી છે તે હું વેચી શકું છું અને આ દરેક ગુણ.

(સ્લાઈડસમયનો સંદર્ભ લો: 02:45)

તેથી, હવે, મને આ બદલાતી માંગમાં ઉલટા પડવાને કારણે, ઓછામાં ઓછી રકમ ગુમાવવાની ખાતરી કરવા માટે, આ અલગ માંગને પહોંચી વળવા માટે આ વ્યૂહરચના સાથે આવવું પડશે. તેથી, એક શક્યતા એ છે કે જ્યારે પણ મારી પાસે વધારાની માંગ હોય ત્યારે હું ઓવરટાઈમ લે, તેથી જો હું સમય જમા કરું તો હું કામદારોને વધુ કલાકો સુધી કામ કરું છું અને વધુ કાર્પેટ કરું છું. પરંતુ, આના માટે 2 ખર્ચ છે, ઓવર ટાઈમ માટે કલાક દીઠ કામ વેતન સામાન્ય ધાર કરતાં સામાન્ય રીતે વધારે છે, ચાલો ધારીએ કે તે 80 ટકા વધારે છે. તેથી, જો તમને મૂળ રૂપે યાદ છે કે અમે પ્રત્યેક કાર્પેટ દીઠ 1000 રૂપિયા આપણી શ્રમ કિંમત તરીકે ચુકવી દીધી છે, તેથી જો હું તેને ઓવરટાઈમમાં બનાવીશ, તો તેની કિંમત 1800 રૂપિયા પ્રતિ કાર્પેટ હશે. કારણ કે, તે સમાન સમય લેશે અને વ્યક્તિને 80 ટકા વધુ ચૂકવવા પડશે. તેથી, 1800 ની 80 ટકા, તેથી તેની કિંમત 1800 પ્રતિ કાર્પેટ છે જે ઓવરટાઈમ રેટ છે, બીજી બાબત એ છે કે તમે અલબત્ત, કોઈ દિવસની 24 કલાક કામ કરવાની અપેક્ષા રાખી શકતા નથી. તેથી, ત્યાં એક મર્યાદા છે જે કહે છે કે કોઈ કાર્યકર 30 ટકાથી વધુ ન કરી શકે, 30 ટકાથી વધારે સમય વિતાવી શકતો નથી. તેથી, જો તેઓ સામાન્ય રીતે દર મહિને 20 બનાવે છે, તો મોટાભાગે તેઓ 6 વત્તા જઈ શકે છે, એક કામદાર હવે મોટાભાગના સમયે કરી શકે છે, જેમાં ઓવરમાટ 26 પ્રતિ મહિનાનો સમાવેશ થાય

છે. મારા માટેનો બીજો વિકલ્પ કર્મચારીઓને ઉમેરવા અથવા ઘટાડવાનો છે, હું નવા કર્મચારીઓને ઉમેરવા માંગું છું, જો મારી પાસે એક મહિનાની ઊંચી માંગ હોય તો મારે 600 કર્મચારીની માંગણીમાં કેટલાક કર્મચારીઓને સમાપ્ત કરવાની જરૂર પડી શકે છે. પણ, આ પણ ખર્ચમાં આવો, તે મને થાય છે કે મારે કેટલાક કાગળનું કામ કરવું પડશે અને મારે કેટલાક વળતર પણ આપવું પડશે. તેથી, ચાલો ધારીએ કે અમુક ચોક્કસ ખર્ચ છે 3200 એ કાર્યકરને ભાડે રાખવામાં ખર્ચ સહયોગી છે અને 4000 રૂપિયા એક કાર્યકરને ફાયરિંગમાં ખર્ચ સહયોગી છે. અને અંતે, જ્યારે હું અસ્તિત્વમાં આવ્યો ત્યારે હું કાર્પેટ્સ સ્ટોર કરી શકું છું અને માંગ કરતી વખતે પછીથી તેને વેચી શકું છું, પરંતુ સ્ટોરેજ પણ કંઈક ખર્ચ કરે છે. તેથી, ચાલો ધારીએ કે કાર્પેટ ખર્ચ 80 પૈસા પ્રતિ મહિનામાં સંગ્રહિત કરે છે. કારણ કે, મને તેને ભેજથી થતા નુકસાનને ટાળીને કાળજીપૂર્વક રાખવું પડશે અને આથી, હું વિવિધ વિકલ્પો આપી શકું છું જે હું ઓવરટાઈમ ચૂકવી શકું છું. પરંતુ, મર્યાદા સુધી અને તે વ્યક્તિએ બનાવેલા કાર્પેટ માટે વધુ ખર્ચ કર્યો જેણે ઓવરટાઈમ ચૂક્યો. હું મારા વર્ક ફોર્સમાંથી ક્યાં ઉમેરી અથવા બાદ કરી શકું છું, પણ દર વખતે જ્યારે હું તે કરું છું ત્યારે તે પણ આવે છે. અને અંતે, મારી પાસે મારું ઉત્પાદન ચાલુ રાખવા માગે તો એક વધારાનું સ્ટોરેજ ખર્ચ છે, તેને એક મહિનાથી બીજા મહિના સુધી સોંપવું.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:17)

તેથી, આપણે આમાંથી રેખીય પ્રોગ્રામ બનાવવા માંગીએ છીએ, તેથી આપણને કેટલાક ચલોની જરૂર છે. તેથી, અમે આ 12 મહિના જાન્યુઆરી, ફેબ્રુઆરીથી ડિસેમ્બર સુધી કાર્યરત છીએ. તેથી, એક મહિનાના આ અવકાશમાં જે બન્યું તે બધું વિચારવું સ્વાભાવિક છે. તેથી, મહિના માટે હું જ્યાં 1 થી 12 સુધીનો છું, ચાલો ધારીએ કે અમારી પાસે વાઈ કામદારો છે, શરૂઆતમાં અમારી પાસે 30 કામદારો છે. ત્યારબાદ, આ 30 થી અમે કર્મચારીઓને ખાલી જગ્યામાં મેળવવા માટે ભાડે રાખીએ અથવા આગ લાવીશું, પછી ફેબ્રુઆરી મેળવવા માટે કેટલાકને ભાડે રાખીએ અથવા આગ લાવીશું. તેથી, દર મહિને સંભવિત રૂપે અમારી પાસે ઓછા અથવા ઓછા કામદારો હોય છે, પરંતુ અમે 30 થી પ્રારંભ કરીએ છીએ, હવે આ 30 કામદારો કેટલાય સંખ્યામાં કાર્પેટ બનાવશે. તેથી, ચાલો ધારીએ કે મહિનામાં બનેલી કાર્પેટની કુલ સંખ્યા એક્સ i અને O i છે જે કુલ રૂપે એક્સ i માં શામેલ છે. પરંતુ, ઓ, હું વિશેષરૂપે કાર્પેટ્સની સંખ્યા છે જે ઓવરટાઈમમાં બનાવવામાં આવે છે, કારણ કે આ માટે વધારાની કિંમત છે, તે અન્ય પગારમાં સમાપ્ત થઈ ગઈ છે જે અમે પહેલાથી જ કામદારને ચૂકવીએ છીએ. અને છેવટે, અમે કહ્યું કે અમારી પાસે આ ભાડે રાખવાની અને ફાયરિંગ ખર્ચ છે, તેથી ચાલો હું એક મહિનાની શરૂઆતમાં ભાડે રાખેલા કામદારોની સંખ્યા અને પ્રથમ ક્રમાંક આપું.એક મહિનાની શરૂઆતમાં એફ કાર્યકરો બરતરફ. અને એસને હું માલના અંતમાં સ્ટોકમાં રહેલી સરપ્લસ કાર્પેટ્સની સંખ્યા માટે કહું છું. તેથી, હું શરૂઆતમાં ધારે છે કે મારી પાસે ખાલી વેરિયેબલ છે, તેથી 0 ની 0 છે, પણ જાન્યુઆરી 1 ના અંતે કહે છે કે મારી વાર્તા પર કેટલી કાર્પેટ છે, ફેબ્રુઆરીના અંતમાં કહે છે કે મારી વાર્તા પર કેટલી કાર્પેટ છે અને તેથી. તેથી, ત્યાં 1, 2, 3, 4, 5, 6 જથ્થો છે, આમાંની દરેક માત્રા 12 મહિના માટે છે. તેથી, 72 વેરીએબલ્સ વત્તા મારી પાસે w0 અથવા s0 માટે આ મૂળભૂત પ્રારંભિક ચલો છે, તેથી 74 ચલો. તેથી, આ ઘણા બધા વસ્તુઓ છે જેનો હું અંદાજ કરું છું, પરંતુ તે વાસ્તવમાં રેખીય પ્રોગ્રામમાં પરિણમે છે, તે મોટી સંખ્યા નથી, તે સિમ્પલેક્સ સહિતની નવીનતમ વસ્તુઓ દ્વારા ખૂબ કાર્યક્ષમ રીતે હલ કરી શકાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:16)

તેથી, ચાલો હવે આપણે અવરોધની વસ્તુ કરીએ, તેથી સામાન્ય એકની પ્રથમ અવરોધ, જે આપણે જે દરેક જથ્થા વિશે વાત કરીએ છીએ તે બરાબર 0 કરતા વધારે છે. બીજી અવરોધો વાટાઘાટો વિશે, કાર્પેટ્સની સંખ્યા નિયમિત ઉત્પાદન અને

ઓવરટાઈમમાં કેવી રીતે તૂટી ગઈ છે. તેથી, જો હું એક મહિનામાં કારપેટ કરું છું, તો દરેક કાર્યકરને યાદ છે કે હું મહિનામાં મારા સ્થાપનામાં કામદારોની સંખ્યામાં મહિનો બનાવશે. તેથી, 20 વાર ડબલ્યુ હું મહિનામાં કયા કામદારોને સામાન્ય સમયની અંદર અને તે ઉપરાંત કે હું માત્ર બનાવવામાં કાર્પેટ સંખ્યા છે. તેથી, ઓ, હું ઓવરટાઈમ કલાકની સંખ્યા નથી, તેથી હું કાર્પેટ્સની સંખ્યા છે. તેથી, દર મહિને હું 20 વખત કર્મચારીઓ વત્તા O i કરું છું, હવે જો હું પ્રારંભ કરું તો મારે પહેલું મહિને ડબલ્યુ 1 ઓછા કામદારો હતા અને મેં થોડા ભાડે રાખ્યા અને થોડા બરતરફ કર્યા, તો હવે મારી પાસે કુલ સંખ્યા એ જૂની સંખ્યા છે વત્તા ભાડે રાખેલ વત્તા સંખ્યા ભાડે. તેથી, આ પ્રત્યેક ડબલ્યુ હું પાછલા ડબલ્યુ આઈ સાથે કેવી રીતે જોડાયેલું છે તેની આ અવરોધ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:28)

એ જ રીતે, સ્ટેક એ આ મહિને શરૂ થતાં પહેલાં મારી પાસે કેટલી હતી, મેં આ મહિને કેટલો ખર્ચ કર્યો અને કેટલો બધો વેચ્યો. તેથી, યાદ રાખો કે ડી 1 થી ડી 12 ની માંગ હતી, તેથી આ નિયત જથ્થો છે, ક્યાં તો ચલ નહીં તે જાણીતા મૂલ્યો છે. તેથી, આ મહિનામાં મેં ડી વેચી દીધી હોત, પરંતુ હું એક્સ પેદા કરું છું. તેથી, એક્સ આઈ માઈનસ ડી એ આ મહિને પેદા કરેલા એક્ઝિટ્સ છે, જે ખોટ છે, કારણ કે હું સરપ્લસ સી માઈનસ 1 થી શરૂ કરી શકું છું. તેથી, જો હું આ મહિનોનો સરપ્લસ ઉમેરું છું તો મને અવગણવામાં આવે છે. મહિનો. યાદ રાખો, મારી પાસે નકારાત્મક કાર્પેટ્સ નથી, સી તે બધી વસ્તુઓ હંમેશાં 0 કરતા મોટી હોવી આવશ્યક છે. અને છેલ્લે, જેમ જેમ આપણે કાર્યકર પહેલા નિયમિત મહિનામાં 20 કાર્પેટ્સ બનાવતા પહેલાં કહ્યું તેમ, જો હું હવે તે હકીકત ઉમેરીશ કે તમે 30 ટકા કરી શકો છો ઓવરટાઈમ, આ 20 વત્તા 6 છે. તેથી, ઓવરટાઈમમાં કરવામાં આવેલી કાર્પેટ્સની સંખ્યા પ્રતિ કર્મચારી દીઠ 6 જેટલી હોઈ શકે છે. તેથી, ઓ, હું 6 ગણી ડબલ્યુથી ઓછી અથવા બરાબર હોવું જોઈએ, તેથી આ બધી અવરોધ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:31)

તેથી, હવે આ અવરોધો આપીએ છીએ જે આપણે કરવા માંગીએ છીએ તે આપણે જે કિંમત ચૂકવવા જઈ રહ્યા છીએ તે કુલ રકમ ઘટાડે છે. તેથી, સૌ પ્રથમ નિયમિત વેતનનો ખર્ચ છે, નિયમિત વેતનનો ખર્ચ પ્રતિ કર્મચારીઓ દીઠ 20,000 છે અને અમારી પાસે મહિના 1, ડબલ્યુ 2 મહિના 2 અને ડબલ્યુ 2 મહિનામાં ડબલ્યુ 1 કામદારો છે, તેથી અમે કુલ કામદારોનો સમાવેશ કરીએ છીએ, દરેકમાંથી વાસણ માર્ગ 20,000 રૂપિયા. તેથી, આ વર્ષ પહેલાં કુલ પગાર છે, પછી દરેક મહિનામાં આપણે કેટલા લોકોને ભાડે આપીએ છીએ તેના આધારે, અમે દરેક ભાડે 3000 ચૂકવીએ છીએ, તેથી આ અમારી ભરતીની પસંદગી છે. એ જ રીતે, દરેક વ્યક્તિ કે જે અમે વ્યવસાય કરીએ છીએ તે માટે આપણે પહેલા 4000 ચૂકવવું પડશે. તેથી, આ ફાયરિંગ ચૂંટેલા છે, પછી દર મહિને ખોપરી કાર્પેટની સંખ્યા કે જે દર મહિને કાર્પેટના ખર્ચમાં સંગ્રહાય છે. તેથી, 80 દ્વારા કુલ સ્ટોરેજને ગુણાકાર કરવો પડશે અને છેવટે, જ્યારે હું ઓવરટાઈમ ચૂકવીશ અને અમારું નંબર પગાર ચૂકવીશ. તેથી, ઓવરાઈટમાં બનેલી પ્રત્યેક કાર્પેટ, 1800 રૂપિયાનો ખર્ચ યાદ રાખો, જે 1000 રૂપિયા છે, જે પહેલાથી પગાર માટે જવાબદાર છે, તેથી મારી ઓવરટાઈમ કિંમત છે. તેથી, આ કહે છે કે આપણે માંગ અને ભાડે આપતી અને પ્રેરણાદાયક અને ઉત્પાદકતા સાથે ઉત્પાદન વિશે આ એકદમ જટિલ જુદા જુદા પ્રશ્નને લઈ શકીએ છીએ અને તેને 72 વત્તા 274 ચલો અને જટિલ ખર્ચ કાર્ય સાથે સેટ કરી શકીએ છીએ અને તેને સરળ અને ફીડ આપી શકીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:01)

તેથી, આપણે સરળ કર્યું છે અને આપણે ઉકેલ શોધી કાઢ્યો છે, તેથી આપણે કર્યું છે, તેથી આપણે એવું અનુભવીએ છીએ કે આપણે એક જવાબ સાથે હોઈ શકીએ જે વાસ્તવમાં ઉપયોગમાં લઈ શકાય તેવું નથી. દાખલા તરીકે, આપણે તે જવાબ આપી શકીએ કે એચ 3 એ 10.6 બરાબર છે. તેથી, આ છે, અમે એક મહિનામાં લોકોની ભિન્ન સંખ્યા ભાડે રાખવી જ જોઈએ. હવે, દેખીતી રીતે, આપણે તે જ સમયે કરી શકતા નથી, અમારી પાસે કોઈ પણ અવરોધ નથી, જે દર્શાવે છે કે મૂલ્યો પૂર્ણાંક હોવું આવશ્યક છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:34)

તેથી, આ વિશે શું કરી શકાય છે, તેમજ આપણે આ 10.6 લઈ શકીએ છીએ અને આપણે પૂર્ણાંક 10 અથવા 11 માં ભૂલો શોધી શકીએ છીએ અને કિંમત પર શું થાય છે તેનું પુનઃમૂલ્યાંકન કરીએ છીએ, જો આપણે તેને પૂર્ણાંક પણ બનાવીએ . તેથી, આને પૂર્ણાંક રાઉન્ડિંગ કહેવામાં આવે છે, હવે જો મૂલ્ય મોટું હોય તો હવે 10, 11 બે અંક મૂલ્ય કહે છે, તો રાઉન્ડિંગ પ્રમાણમાં નાનું વિસ્થાપન છે અને મહત્તમ ગુણવત્તાની ગુણવત્તા બદલાશે નહીં. પરંતુ, જો સંખ્યા નાની હોય, તો હું કહીશ કે હું 1.52 અથવા 1 ની 0.51 કલાક 0 થી લઈ જઈશ તો રાઉન્ડિંગની અસર તદ્દન કાર્ય હોઈ શકે છે. તેથી, આ રેખીય પ્રોગ્રામિંગની સામાન્ય સમસ્યા છે જે રેખીય પ્રોગ્રામિંગ એ ગેરેંટી આપી શકે નહીં કે તમારી પાસે પૂર્ણાંક સોલ્યુશન્સ છે, જ્યારે તમે પૂર્ણાંક સોલ્યુશન ઈચ્છો ત્યારે તમે પૂર્ણાંક સોલ્યુશન પ્રાપ્ત કરવા માટે રાઉન્ડિંગનો ઉપયોગ કરી શકો છો, પરંતુ તમારે સાવચેત રહેવું જોઈએ કે ગોળીઓ ખરેખર આપે છે તમે એક શ્રેષ્ઠ હવે, શા માટે આગ્રહ કરશો કે તમે પૂર્ણાંક સોલ્યુશન જોઈએ છે. તેથી, તમે આ જ સમસ્યાને સેટ કરી શકતા નથી અને તેને હલ કરી શકો છો, પરંતુ સોલ્યુશન્સની પૂર્ણાંક હોવી જરૂરી છે, દુર્ભાગ્યે આ શરતો ખૂબ જ મુશ્કેલ તપાસ છે. તેથી, અમે શોધ્યું છે કે અમે દાવો કર્યો છે કે જ્યારે આપણે સામાન્ય રેખીય પ્રોગ્રામિંગ સમસ્યા સેટ કરીએ છીએ, ત્યારે અમે તેમને કાર્યક્ષમ રીતે હલ કરી શકીએ છીએ. તેથી, સિમ્પલેક્સ એ એક કાર્યક્ષમ ઉકેલ નથી, પરંતુ અસરકારક ઉકેલ છે, પરંતુ રેખીય પ્રોગ્રામિંગ માટે ઈન્ટિરિયર પોઈન્ટ પદ્ધતિઓ અને અન્ય પોલિનોમિયલ સમય અલ્ગોરિધમ્સ છે. પરંતુ, જો તમે આ રમતના નિયમોને બદલો છો અને કહે છે કે હું મનસ્વી ઉકેલ ન ઈચ્છું છું; તેનો અર્થ છે, અવરોધ, પરંતુ હું એક જોઈએ છે કે આપણે બધા મૂલ્યો છે જે મને પૂર્ણાંક મળે છે, પછી આપણી પાસે છે, તેથી પૂર્ણાંક રેખીય પ્રોગ્રામિંગને કૉલ કરો. અને પૂર્ણાંક રેખીય પ્રોગ્રામિંગ કમનસીબે અસરકારક રીતે સોલ્વબલ નથી અથવા તે અસરકારક રીતે વિભાજીત રીતે જાણીતી નથી. તેથી, પૂર્ણાંક રેખીય પ્રોગ્રામિંગ માટે આપણે જે શ્રેષ્ઠ આશા રાખી શકીએ તે વાસ્તવમાં તેને મનસ્વી રેખીય પ્રોગ્રામમાં ફેરવવાનો પ્રયાસ કરવો અને કેટલાક યોગ્ય ગોળીઓ કરીને જવાબો અથવા પૂર્ણાંકને અર્થઘટન કરવાનો પ્રયાસ કરવો.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 03
લેક્ચર - 52
એલપી મોડેલિંગ: બેન્ડવિડ્થ(Bandwidth) ફાળવણી

રેખીય પ્રોગ્રામિંગના અમારા આગલા ઉદાહરણ માટે, અમે નેટવર્ક બેન્ડવિડ્થ સાથે ગ્રાફ અને નેટવર્ક્સ સાથે સંકળાયેલી સમસ્યાને જોશે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:09)

તો, માની લો કે અમારી પાસે ત્રણ યુઝર્સ શામેલ એક નાનો સંચાર નેટવર્ક છે. તેથી, વપરાશકર્તાઓ મૂડી એ, મૂડી બી અને મૂડી સી હોય છે, અને તેમાંના દરેક કેટલાક સ્વીચો દ્વારા જોડાયેલ છે, જેને આપણે A, B અને C કહીએ છીએ. તેથી, અમારી પાસે એક નેટવર્ક છે જે ઈન્ટરનેટ નેટવર્કમાં છે જે આ ત્રણ વપરાશકર્તાઓને જોડે છે અને અમારી આવશ્યકતા એ છે કે વપરાશકર્તાઓના દરેક જોડી A થી B, B થી C અને A to C માં કનેક્ટિવિટીની સેકન્ડમાં ઓછામાં ઓછી બે મેગા બીટ્સ મળે. વપરાશકર્તાઓની જોડી વચ્ચે. તેથી, અમે આ ત્રણ વપરાશકર્તાઓ સાથે ઓછામાં ઓછા બે Mbps કનેક્ટિવિટી જોઈએ છે. હવે, નોંધ લો કે A થી B સુધી, હું પેકેટો મોકલી શકું તે બે માર્ગો છે, હું એ સી અને બી મારફતે સીધી મોકલી શકું અથવા હું તેમને અડકાર મોકલી શકું, મારી પાસે નાની માંગ છે. તેથી, તે બહાર આવે છે, તે ખરેખર દર્શકના મુદ્દાથી કોઈ બાબત નથી, અમે તેનો ઉપયોગ કરીએ છીએ કે આ 2 એમબીએસપી બેન્ડવિડ્થ ટૂંકા લાલ માર્ગ અને લાંબા ગ્રીન રૂટથી આવે છે. તેથી, અમારું લક્ષ્ય દરેક જોડી માટે લાલ અને લીલાની ક્ષમતાને એકીકૃત કરવાનું છે, તેથી સમાન હશે, ઉદાહરણ તરીકે, બી થી સી સુધીનો સીધો રસ્તો હશે અને ત્યાં આ પરોક્ષ માર્ગ હશે. તેથી, દરેક જોડી માટે, સીધો રસ્તો અને પરોક્ષ માર્ગ છે. તેથી, સીધા અને આડકતરી રૂટની ક્ષમતાના સંયોજનો ઓછામાં ઓછા 2 એમબીએસ સુધી ઉમેરે છે, અમારા ગ્રાહકો સંતુષ્ટ છે. તેથી, હવે આપણી પાસે જે મર્યાદા છે તે આ લિંક્સની ક્ષમતા ધરાવે છે. તેથી, જો હું ઉદાહરણ તરીકે બી અને સી વચ્ચેની લિંક જોઉં છું, તો તે માત્ર 13 મેગા બીટ્સ જ સેકન્ડ કુલ કુલ બધા જોડાણોમાં ટ્રાન્સમિટ કરી શકે છે જે તે એક ભાગ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 01:50)

હવે, બીજી બાજુ, અમે આ ગ્રાહક પાસેથી કેટલાક પૈસા કમાવીએ છીએ જે એકરૂપ નથી. તેથી, એ, બી લિંક માટે, અમને દર મહિને મેગા બીટ એમબીપીએસ માટે 300 રૂપિયા મળે છે, પરંતુ બી થી સી સુધી આપણને માત્ર 200 મળે છે, પરંતુ એ થી સી સુધી આપણને 400 મળે છે. તેથી, હવે, અમારે લઘુત્તમ 2 મેગા બીટ્સ. પરંતુ ગ્રાહકો તેટલું ઓછું લેશે જે અમે તેમને લઘુતમ વિષય પર આપી શકીએ છીએ અને અમે ક્ષમતાનો ઉપયોગ કેવી રીતે કરીએ છીએ તેના આધારે અમને અમુક ચોક્કસ આવક મળે છે. તેથી, અમારું ધ્યેય બેન્ડવિડ્થ ફાળવવાનું છે, જે ગ્રાહકોને 2 એમબીપીએસથી ઉપર કંઈપણ લેવા માટે તૈયાર હોય તેવું આવક વધારવા માટે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 02:26)

તેથી, આપણે જોયું છે કે તે એક રેખીય પ્રોગ્રામ તરીકે સેટ કરવાનો છે. તેથી, આ કિસ્સામાં, આપણે કયા વેરીએબલોનો ઉપયોગ કરવા જઈ રહ્યા છીએ? તેથી, યાદ રાખો કે અમે કહ્યું છે કે દરેક જોડાણમાં બે માર્ગ છે. તેથી, અમારી પાસે A થી B આવી રહ્યું છે જ્યાં ટૂંકા માર્ગનો અને અમારી પાસે A થી B ક્યાં સુધી લાંબો રસ્તો આવે છે. તો, આપણે જેનો ઉપયોગ કરીએ છીએ તે એ છે કે એ, થી એ બી સુધી, તે લાલ રુટ પર દર્શાવેલા જથ્થાને દર્શાવે છે અને સમાન રીતે, વાય થી એ બી સુધી જથ્થાને સૂચવવા માટે, તે લીલો માર્ગ પર વહે છે. તેથી, અમારી પાસે A થી B સેવા સાથે સંકળાયેલા બે વેરિયેબલ છે જે પ્રદાન થાય છે, સીધી એક્સ એબી કેટલી જાય છે, તે કેટલું પરોક્ષ રીતે વાય એ બી છે. એ જ રીતે, બી અને સી વચ્ચે એક્સ બીસી હશે જે ટૂંકા માર્ગ અને વાય બીસી સુધી જાય છે જે લાંબા માર્ગથી પસાર થાય છે અને એક્સ એસી અને વાય એ સી માટેનો કોઈ રસ્તો છે. તેથી, આપણાં કનેક્ટિંગ જોડીઓના જુદા જુદા રસ્તાઓ વર્ણવતા છ ચલો છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:27)

હવે, આ વેરિયેબલ્સ લિંક્સની ક્ષમતા દ્વારા અવરોધ છે. તેથી, અમને લાગે છે કે આપણે આ ચોક્કસ લિંકને જોઈએ છીએ, નાના બીથી કેપિટલ બીની લિંક, હવે તેની પાસે 10 ની ક્ષમતા છે. હવે તે કયા રૂટ પર છે, તેથી તે ચોક્કસ રૂપે એ ટૂ થી ટૂંકા રૂટ પર છે. તેથી, તે એક્સ એબી માર્ગ પર આવેલું છે. તેથી, જો એક્સ એબીને જે જથ્થો આપવામાં આવે છે તે આ 10 માં પહોંચશે, તે જ રીતે તે વાય એ બી રૂટ પર છે, તેથી તે પણ પ્રારંભિક રીતે મૂડી બીને નાના બી સુધી પહોંચે છે. તેથી, આમાં પણ તે મળશે, તે બી થી સી રૂટ પર પણ છે. તેથી, એક્સ બીસી અને છેવટે, બી થી સી રૂટ બીજા માર્ગ પર જઈ રહી છે, આ વાય બી સી છે તેથી, આ બધા ચાર માર્ગો એક સાથે મુકવામાં આવશે, આ લિંક દ્વારા જે પણ ક્ષમતાઓ વહે છે તે ઉમેરાશે અને આ લિંકની ક્ષમતા હશે 10, તેથી એક્સ એબી પ્લસ વાય એ બી પ્લસ x બીસી પ્લસ વાય બીસી મોટાભાગે 10 હોવું જોઈએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:32)

તો, જો હું આ લિંકને જોઉં છું, તો મારી પાસે તે જ છે આ વિવિધ વસ્તુઓ જે અહીં આવી રહી છે. તેથી, મારી પાસે ચાર જથ્થા છે, મારી પાસે A થી B ની સીધી લિંક છે, એ થી બી પરની આડકતરી લિંક, એ થી સીની સીધી લિંક અને એ થી સી પરની આડઅસર લિંક છે અને તે બધા વધુ સારી રીતે ઉમેરે છે. અને આ માટે ત્રીજા ભાગો, તેથી આપણે અહીં જોયેલી આ ત્રણ અવરોધો, તે પૂંછડી અને લિંક્સની ક્ષમતા માટે જવાબદાર છે. તેથી, અમે આ ક્ષમતા, આ ક્ષમતા અને આ ક્ષમતા માટે જવાબદાર છીએ, તેથી આ ત્રણ ક્ષમતાઓ અમારા પ્રવાહ સાથે જોડાયેલ છેઆ ત્રણ સમીકરણો દ્વારા. તેથી, આ હજી પણ અમને આ ત્રણ અવરોધો માટે જવાબદાર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:17)

તેથી, હવે, જો આપણે આ લિંકને જોઈશું, તો આ ત્રણ કનેક્શનનો ભાગ છે, તે A થી B ડાયરેક્ટ કનેક્શન છે, તે બી થી સી પરોક્ષ કનેક્શન પર છે અને અંતે તે A થી C પરોક્ષ જોડાણ પર છે, તેથી સીધો જોડાણ A થી B વત્તા આડકતરી જોડાણ

B થી C વત્તા આડકતરી કનેક્શન A થી C છે, બધા ભેગા મળીને 6 કરતાં વધુ હોઈ શકતા નથી. આ જ રીતે, આપણી પાસે સમાન છે આ લિંક માટે સમીકરણ. તેથી, તે A થી B પરના પરોક્ષ જોડાણ પર આવેલું છે, તે A થી B પરના પરોક્ષ જોડાણ પર આવેલું છે, તે બી થી સી સીધી કનેક્શન પર આવેલું છે અને તે અ સી થી સી પરના પરોક્ષ જોડાણ પર છે તેથી આ ત્રણ વસ્તુઓ 13 થી વધી શકતા નથી અને ત્રીજા જેટલા સંપૂર્ણ વર્ગમાં વિચારે છે 11. તેથી, 11 માટે, આપણી પાસે એ છે કે તે સી થી સી સીધી કનેક્શન પર અને આ બે પરોક્ષ જોડાણો પર, એ થી બી પરોક્ષ કનેક્શન અને બી થી સી પરોક્ષ છે. જોડાણ તેથી, આ રીતે હવે આપણે છ અવરોધ આવરી લીધાં છે, તેથી અમારી પાસે પૂંછડી અને લિંક્સને અનુરૂપ ત્રણ અવરોધો છે અને અમારી પાસે ત્રિકોણની લિંક્સને અનુરૂપ ત્રણ અવરોધો છે. તેથી, અમારી પાસે છ કુલ અવરોધ છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 06:43)

છેલ્લે, અમારી પાસે એ અને બી વચ્ચેની આ ન્યૂનતમ આવશ્યકતા છે, અમારે ઓછામાં ઓછા 2, બી અને સી વચ્ચે અરજી કરવી આવશ્યક છે, આપણે ઓછામાં ઓછા 2 અને એ અને સી વચ્ચે અરજી કરવી આવશ્યક છે, આપણે ઓછામાં ઓછા 2 અને આ પરોક્ષ અને સીધી રકમ છે, આપણે તેમની વચ્ચે તફાવત નથી. અને અલબત્ત, દરેક ક્ષમતા નકારાત્મક હોવી જ જોઈએ, તેથી આ આપણને બધી અવરોધો આપે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:03)

ઉદ્દેશ્ય શું છે? ઉદ્દેશ્ય એ આવક છે જેને આપણે સમજીએ છીએ. તેથી, એ થી બી કનેક્શન એ એક્સ એબી પ્લસ વાય એબી છે, આ કુલ વોલ્યુમ છે જે આપણને 300 આપે છે. એ જ રીતે, એક્સ બીસી અને વાય બીસી આપણને 200 આપે છે અને એક્સ એસી પ્લસ વાય એસી આપણને 400 આપે છે. તેથી, આપણે 300 માં ગુણાકાર કરીએ. એક્સ એબી પ્લસ વાય એબી 200 માં એક્સ બીસી પ્લસ વાય બીસી અને 400 એ એક્સ એસી પ્લસ વાય એસી અને તેને ઉમેરીને, આ અમારું કુલ આવક છે અને તમે આ આવક વધારવા માંગો છો.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 07:33)

તેથી, આ ચોક્કસ નંબરો માટે, આ એ જ જવાબ છે જે આપણને મળે છે, કે આપણી પાસે સી થી બી સુધી સીધી વહેતી કંઈ નથી, આપણી પાસે A થી B ની સીધી જ સીધી જ છે અને બીજું. તેથી, જો તમે આ લિંક પર ઉદાહરણ શોધી રહ્યાં છો. તેથી, આ લિંક A થી B ની સીધી રૂટ પર છે, તેથી તે 0 છે, તે A થી B ના પરોક્ષ રૂટ પર આવેલું છે, તે 7 છે, તે B થી C ના પરોક્ષ રૂટમાં આવેલું છે, તે 1.5 છે અને તે બી થી સી પરના પરોક્ષ માર્ગ પર આવેલું છે, તે અન્ય 1.5 છે. જો તમે જુઓ છો કે આ 10 છે, તો આ લિંક 10 ની કુલ ક્ષમતા અને તે તમામ સંયોજનોનો ઉપયોગ કરે છે, જે તે આપે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:21)

આ રીતે, તમે દરેક લિંક માટે તેનો પ્રયાસ કરી શકો છો અને તે શોધી શકો છો, 11 ની આ લિંક સિવાયની દરેક વસ્તુ વાસ્તવમાં આ પ્રવાહ દ્વારા સંતૃપ્ત થાય છે, તે પણ અહીં બદલાય છે કે તમે કેટલાકને જોઈ શકો છો જથ્થો કે જે

આપણને મળે છે અથવા અપૂર્ણાંક છે. પરંતુ અમે ઈન્ટરનેટ બેન્ડવિડ્થ સાથે કામ કરી રહ્યા હોવાથી, તે કોઈ પ્રકારનું નથી કારણ કે તે વ્યક્તિની ભરતી અથવા ફાયરિંગ અથવા કૉર્પોરેટ ઓફર કરતી નથી, અમે સહેલાઈથી કેટલીક બેન્ડવિડ્થને અમુક અપૂર્ણાંકમાં વિભાજીત કરી શકીએ છીએ, જેથી તે કોઈ સમસ્યા નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 08:52)

જો કે, બીજી સમસ્યા એ છે કે આપણે વાસ્તવમાં રેખીય પ્રોગ્રામ સેટ કર્યો છે. તેથી, અમે ટ્રાફિકની બહાર જવાના દરેક સંભવિત રીતને લેવાનું રેખાકીય પ્રોગ્રામ સેટ કર્યું છે. તો, આપણી પાસે A to B છે, આપણી પાસે એક લિંક છે, આપણી પાસે A to B આ છે, આપણી પાસે બીજા પાથો છે અને દરેક પાથ માટે, અહીં એક ચલ છે, x એબી, વાય એ બી અને બીજું. તેથી, દરેક પાથ તે પાથ દ્વારા વહેતી માત્રા દ્વારા રજૂ કરવામાં આવે છે. તેથી, સમસ્યા એ છે કે, ગ્રાફ દ્વારા વહેતી પાથોની સંખ્યા ઘાતાંકીય હશે. તેથી, આ મોડેલિંગ વ્યૂહરચના સારી નથી. તેથી, આપણે જે કરીએ છીએ તે છે, અમે નેટવર્ક બેન્ડવિડ્થ ફાળવણી મોડેલ લઈ રહ્યા છીએ અને અમે તેને લાગુ કરી રહ્યાં છીએ અથવા અમે રેખીય પ્રોગ્રામિંગનો ઉપયોગ કરીને તેનું વર્ણન કરી રહ્યાં છીએ. પરંતુ જો આપણે પ્રોગ્રામ, રેખીયર પ્રોગ્રામ સેટ કરીએ છીએ, તો તેમાં મોટી સંખ્યામાં ચલો છે, તો પછી તે ભાષાંતરમાં તે સંકલનમાં કેટલાક અર્થમાં સમસ્યા આવી રહી છે. તેથી, અમને આ નથી જોઈતું, આપણે કાર્યક્ષમ ભાષાંતર જોઈએ છે અને આ એક નથી, એવું બને છે કે ફક્ત ત્રણ ગ્રાહકોની જેમ નાની સમસ્યા માટે, તે સારું હતું, પરંતુ જેમ તમે મોટા અને મોટા થાઓ તેમ આ અનુવાદ વધશે નહીં . પરંતુ આપણે આ નેટવર્ક પ્રવાહને જોવાનું બીજી રીત જોશું અને કહેવાશે કે સામાન્ય નેટવર્ક પ્રવાહમાં સરળતાથી રજૂ થઈ શકે છે, તે રેખીય પ્રોગ્રામ્સની શરતો છે. પરંતુ તે નોંધવું મહત્વપૂર્ણ છે કે સામાન્ય રીતે, જ્યારે આપણે રેખીય પ્રોગ્રામમાં અનુવાદ કરીએ છીએ, ત્યારે અમને ઈનપુટ સમસ્યામાં નાના ચલણમાં મળતા ચલોની સંખ્યા જોઈએ છે. તેથી, ઈનપુટ સમસ્યામાં, જો અમારી પાસે ચોક્કસ કદ હોય, તો રેખીય પ્રોગ્રામ ઉડાવવો નહીં.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 04
લેક્ચર - 53
નેટવર્ક પ્રવાહ

બેન્ડવિડ્થ ફાળવણીની સમસ્યામાં, રેખા પ્રોગ્રામ તરીકે નેટવર્ક ફ્લો સમસ્યાને એન્કોડ કરવા માટે અમે એક ચલ દીઠ પાથનો ઉપયોગ કરીએ છીએ અને અમે દલીલ કરી હતી. તે કરવા માટે એક કાર્યક્ષમ માર્ગ નથી. તો, ચાલો રેખીય પ્રોગ્રામ્સના સંદર્ભમાં નેટવર્ક ફ્લોનું પ્રતિનિધિત્વ કરવા માટે એક મોડ સીધી રીતે જોઈએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 00:16)

તો, માની લો કે આપણી પાસે ઓઈલ નેટવર્ક છે જે આ નિર્દેશિત ગ્રાફમાં આપવામાં આવેલ છે. તેથી, અમારી પાસે સ્રોત વર્ટેક્સ છે અને અમારી પાસે લક્ષ્ય અથવા સિંક વર્ટેક્સ ટી છે અને અમારું લક્ષ્ય એ છે કે આપણે જે પાઈપ્સ આપીએ છીએ તેનાથી આપણે જેટલું તેલ મેળવી શકીએ છીએ. તેથી, અલબત્ત, પ્રવાહની એક મિલકત એ છે કે તે પ્રવાહ આવશ્યક છે, તેથી હું કોઈપણ મધ્યવર્તી નોડ પર કોઈપણ જથ્થો રાખી શકતો નથી. તેથી, જે કંઈપણ દાખલ કરે છે તે બી છોડી દેશે, જે કંઈપણ દાખલ થાય છે તે ડી છોડી જ જોઈએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:53)

તેથી, જો આપણે આ કરવાનો પ્રયાસ કરીએ, તો ઉદાહરણ તરીકે આ લીલો જથ્થો એક સંભવિત પ્રવાહને રજૂ કરે છે, હું s થી 1 થી 2 સુધી s થી b ને s થી c માં બે એકમો મોકલીશ અને તમે કરી શકો છો તપાસો કે આ 3, 3 અને 4 મોકલવા માટે ક્ષમતામાં સાથે સંપર્ક કરી શકે છે અને મેં 2, 1 અને 4 સેટ કર્યા છે અને હું ચાલુ રાખું છું. અને તેથી, હવે આ વિશિષ્ટ વસ્તુ પર મારી પાસે કુલ 7 એકમો છે જેમાંથી બહાર નીકળે છે અને 7 એકમો જે ટીમાં આવે છે. તેથી, હું આ નેટવર્કને એસથી ટીમાં 7 એકમોમાં પ્રવાહ આપી શકું છું. તેથી, આપણે ફક્ત આ સ્થાનિક રીતે ચકાસી શકીએ છીએ, પછી ડી પર ચેક ઈન્સ્ટન્સ દ્વારા આ પ્રવાહ છે, 2 વત્તા 1, 3 માં વહેતી કુલ જથ્થો જો કંઈપણ સંગ્રહિત નથી થતું કારણ કે બહાર નીકળતી કુલ જથ્થો ફરીથી 2 વત્તા 1, 3. છે. આપણે આ રીતે સ્થાનિક તપાસ કરી શકીએ છીએ, આપણે આપણી જાતને સંતોષી શકીએ કે આ એક માન્ય પ્રવાહ છે, કુલ રકમ 7 છે. પરંતુ, પ્રશ્ન એ છે કે, આ મહત્તમ છે? આપણે કેવી રીતે જાણી શકીએ કે અમે મહત્તમ પ્રાપ્ત કરીએ કે નહીં?

(સ્લાઈડટાઈમનો સંદર્ભ લો: 01:44)

તેથી, સમસ્યાને ફક્ત સામનો કરવા માટે સમસ્યા એ છે કે આપણે એક વિશિષ્ટ પ્રકારનો ગ્રાફ આપ્યો છે, તે એક નિર્દેશિત ગ્રાફ છે અને તેમાં બે વિશિષ્ટ ગાંઠો, સ્રોત અને સિંક છે. સ્રોતમાં કોઈ આવનારા કિનારીઓ નથી અને સિંક પાસે કોઈ

આઉટગોઈંગ ધાર નથી. દરેક ધારની ક્ષમતા હોય છે, જે ધાર સાથે સંકળાયેલ વજન છે અને અમારા ધ્યેય પ્રવાહ સાથે આવે છે, પ્રવાહ ફરીથી તે જથ્થો છે જે અમે દરેક ધારને સોંપીશું. અને હવે પ્રવાહ કેટલીક મૂળભૂત પરિસ્થિતિઓને સંતોષવા જ જોઈએ, પ્રવાહ હંમેશા ક્ષમતા કરતાં ઓછું હોવું જોઈએ, પછી અમારી પાસે સંગ્રહ નથી. તેથી, પ્રત્યેક આંતરિક ગાંઠ પર નોડમાં વહેતી કુલ રકમ, વહેતી કુલ રકમ જેટલી જ હોવી આવશ્યક છે. તેથી, આને પ્રવાહનું સંરક્ષણ કહેવામાં આવે છે, અમે મધ્યવર્તી સમસ્યાઓ પર કાંઈ પણ ગુમાવી શકતા નથી અથવા કંઈપણ ઉત્પન્ન કરી શકતા નથી, જે કંઈપણ આવે છે તે બહાર જવું જ જોઈએ. અને છેવટે, આપણે જે કરીએ છીએ તે છે કે આપણે પ્રવાહ પર કુલ જથ્થાને ઑપ્ટિમાઈઝ કરવા માંગીએ છીએ, પ્રવાહની કુલ માત્રા એ ફ્લોમાંથી બહાર આવતી ફ્લોની માત્રા છે જે અલબત્ત પણ છે, પ્રવાહમાં જે જથ્થો ચાલી રહ્યું છે ગુમાવી શકાય છે. તેથી, આપણે ક્યાં તો એનને જોઈ શકીએ છીએ, પરંતુ સ્રોતમાંથી બહાર જતાં પ્રવાહની કુલ માત્રા હોવા માટે, તેને ફક્ત વ્યાખ્યાયિત કરીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 03:01)

તેથી, હવે આપણી પાસે આ ફોર્મ્યુલેશન બાકી છે, તેથી આપણે હવે રેખીય પ્રોગ્રામ સેટ કરી શકીએ છીએ, આપણે શું જોડીએ છીએ તે આપણે જોડીએ છીએ અને દરેક ધાર માટે એક વેરિયેબલ કહ્યું છે. તેથી, ધાર માટેના ઉદાહરણો માટે આપણી પાસે એફએસએ છે અને પછી ઉદાહરણ તરીકે ધાર બી ડી માટે, અમારી પાસે એફ બી ડી છે અને પછી સી ઈ માટે, અમારી પાસે fce છે. તેથી, આપણી પાસે 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 11 કિનારીઓ ધારે છે અને તેથી, આ રેખીય પ્રોગ્રામમાં આપણી પાસે 11 ચલો છે. હવે, આપણે આ ચલોમાં શું કરી શકીએ, એક પહેલો આપણે કહી શકીએ કે દરેક વેરિયેબલ એજ ધારની ક્ષમતા દ્વારા મર્યાદિત છે. તેથી, એફ બી, તેથી બી એ હવે આ ધાર છે, તેની ક્ષમતા 10 છે. તેથી, હું આખરે જે પણ પ્રવાહમાંથી બી સુધી પહોંચું છું તે 10 કરતા ઓછું હોવું જોઈએ અને બીજું એવું લાગે કે આપણે કહી શકીએ છીએ કે આપણી પાસે દરેક આંતરિક નોડ પર પ્રવાહનું સંરક્ષણ. તેથી, ઉદાહરણ તરીકે જો હું આ નોડ ડીને જોઉં, તો ઈનકમિંગ ફ્લો એ એડ પ્લસ બી ડી છે, આઉટગોઈંગ ફ્લો એ ડી સી પ્લસ ડી પ્લસ ડી ટી છે અને આ બે જથ્થાઓ સમાન હોવી જ જોઈએ, ફ્રિડ પ્લસ એફબીડી એફડીસી પ્લસ એફડીડી પ્લસ એફડી સમાન હોવું જ જોઈએ. ટી. અને છેવટે, આ અવરોધ છે, તેથી અવરોધ કહે છે કે દરેક ધાર ફક્ત આ ક્ષમતામાં જ લઈ શકે છે અથવા પ્રવાહ કરી શકે છે અને દરેક આંતરિક નોડમાં પ્રવાહનું સંરક્ષણ થાય છે. અને આખરે, આ ત્રણ ધાર પર એફએસએસ પ્લસ એફએસબી પ્લસ એફએસ સી પર શું થાય છે તે મહત્તમ કરવાનો હેતુ છે, આ અમારું ઉદ્દેશ્ય કાર્ય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 04:45)

તો, પહેલાં, આપણે પહેલાં જ રેખીય પ્રોગ્રામિંગનો ઉકેલ લાવીશું જે આમાંથી કેટલાક સરળ સિમ્પ્લેક્સ દાખલા છે અને જવાબ મળે છે. પરંતુ, આપણે હવે શું કરીશું તે સમજવા એ છે કે આનો ખરેખર અર્થ શું છે, યાદ રાખો કે કેવી રીતે સરળ કાર્ય કરે છે. તેથી, સંભવિત ક્ષેત્રના શિરચ્છેદ સાથે સિમ્પલેક્સ શરુ થાય છે જે એક શિખરથી એક x સુધી જાય છે. તેથી, આમાં વધારો થતો પ્રવાહ વાસ્તવમાં હાલનો પ્રવાહ લઈ રહ્યું છે અને તેમાં કંઈક ઉમેરી રહ્યું છે અને આનો પ્રવાહ શોધ એલ્ગોરિધમનો સંદર્ભમાં સીધો અર્થઘટન કરી શકાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 05:19)

તેથી, આ ફોર્ડ-ફુલકર્સન(Ford-Fulkerson) એલ્ગોરિધમ તરીકે ઓળખાતું એલ્ગોરિધમ છે જે વાસ્તવમાં મહત્તમ ફ્લોનું નિર્માણ કરીને નેટવર્ક પ્રવાહ સમસ્યાને સીધો ઉકેલવાનો પ્રયાસ કરે છે. તેથી, એલ્ગોરિધમ ધારે છે કે તમે 0 પ્રવાહથી પ્રારંભ કરો છો અને પછી તમે કોઈ પાથ પસંદ કરો છો જેના પર જોડી ક્ષમતા છે અને પછી આ પાથ પર, તમે શક્ય તેટલો પ્રવાહ વધારો, જેથી પાથ સંતૃપ્ત થઈ જાય. તેથી, હવે જો તમે જમણી બાજુએ એલ્ગોરિધમ નેટવર્ક જુઓ છો, તો તે ખૂબ જ સ્પષ્ટ છે કે શક્ય એક પ્રવાહ છે, તમે તે રીતે એક એકમનો પ્રવાહ મોકલી શકો છો, તમે આ રીતે એક એકમનો પ્રવાહ મોકલી શકો છો. પરંતુ, ફોર્ડ-ફુલકર્સન(Ford-Fulkerson) એલ્ગોરિધમ કહે છે કે કોઈ પણ પાથ જે અસ્તિત્વ ધરાવે છે અને ત્યાં વહેતી વસ્તુઓ શરૂ કરે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:06)

તેથી, તમે આ પાથ સાથે ઉદાહરણો શરૂ કરી શકો છો, તે એક થી ડી સુધી અને ત્યારબાદ ડી ડાઉનથી ઈ અને પછી ટી પર શરૂ થાય છે. તેથી, એક એકમ વહે છે, પરંતુ હવે આ બિંદુ સંતૃપ્ત નથી અને આ ધાર સંતૃપ્ત નથી અને અલબત્ત, આ ધાર ખોટી દિશામાં છે. તેથી, હું પ્રવાહની બીજી એકમ ઉત્પન્ન કરવા માટે આ બે ક્ષમતાઓનો ઉપયોગ કરી શકતો નથી, તેથી લાગે છે કે ફોર્ડ-ફુલકર્સન(Ford-Fulkerson) એલ્ગોરિધમ એક સમસ્યા લે છે, જો તમે શરૂ થવાનો ખોટો રસ્તો પસંદ કરો છો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:34)

તેથી, ઉકેલને બચાવવા એ છે કે જો તમને પાથ પાછો લેવામાં આવે તો પણ, અમે વિચારીએ છીએ કે અમે જે નિર્ણય લીધો છે તે પાછો ફેરવી નાખે છે. તેથી, આપણે કહીએ છીએ કે જો આપણે આમાંથી પસાર થઈ રહ્યા છીએ, તો આપણે આ પ્રવાહ ઘટાડી શકીએ છીએ. તેથી, આપણે આ પ્રવાહને બીજી રીતે પાછો ફેરવી શકીએ છીએ, તેથી તે વર્ણન કરવા માટે થોડું જટિલ છે, પરંતુ તેને ઉકેલવાની એક રીત ખરેખર વાસ્તવમાં સેટઅપ અને અતિરિક્ત કિનારી છે જે અમને પાછું વિચારે છે. તેથી, આ આપણે શેષ ગ્રાફને કહીએ છીએ, તેથી બાકીના ગ્રાફમાં આપણે શું કરીશું તે આપણે ખરેખર તે પ્રવાહને લઈશું જેનો આપણે હમણાં જ કરાર કર્યો છે અને પછી અમે ક્ષમતાઓને બદલીશું. તેથી, ફોરવર્ડ ધાર ટુ ડી જે ક્ષમતા 1 અને પ્રવાહ 1 ધરાવે છે, હવે બાકીની ક્ષમતા 0 ધરાવે છે. તેથી, અમારી પાસે નિયમિત ધાર છે અને પછી અમે તેને સ્થાનાંતરિત વાસ્તવિક રકમ દ્વારા બદલીએ છીએ. તેથી, તે વર્તમાન પ્રવાહથી નીકળતી ક્ષમતા છે અને વધુમાં આપણે આ નવી ધારને પાછળથી બાજુએ ગોઠવીએ છીએ, જે આપણે કરેલા પ્રવાહને અનુરૂપ છે, પરંતુ આપણે ડેટાને બદલી શકો છો. તેથી, આપણે s થી d સુધી flow 1 મોકલી દીધું છે, પરંતુ હવે આપણે તે પ્રવાહને ડી થી એસ તરફ પાછા મોકલીને ઘટાડી શકીએ છીએ જે આ છે. તેથી, ઔપચારિક રીતે આ એક અવશેષો છે જે અવશેષ ગ્રાફ છે, તમે મૂળ ગ્રાફને તે પછીના ધારમાં સુયોજિત કરેલા કોઈપણ પ્રવાહને લઈ શકો છો, તમે તે ધાર દ્વારા તે ધારની ક્ષમતાને ઘટાડે છે અને તે પ્રવાહને અનુલક્ષીને સેટ કરો છો ધાર જે પછીથી આને પૂર્વવત્ કરી શકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:09)

તેથી, આ ઉદાહરણ પર પાછા જવું, તો આપણે શું કરીશું તે પહેલા આપણે આ ખોટા પ્રવાહથી પ્રારંભ કરીશું તો આપણે કહીશું, પરંતુ પછી આપણે અવશેષ ગ્રાફ કરીશું અને હવે આપણે અવલોકન કરીશું કે આ અવશેષમાં ગ્રાફ, ત્યાં જેવો માર્ગ છે. તેથી, અમે મૂળ ગ્રાફ વિશે વાત કરી રહ્યા નથી, અમે ફક્ત દરેક તબક્કે અવશેષ ગ્રાફ વિશે વાત કરીએ છીએ. તો, હવે આપણે આને રુટ કરીશું, આ નવા પ્રવાહમાં પરિણમશે. તો, અહીં 0 ને ઘટાડવાનું સમાપ્ત થશે અને નવી ધાર અહીં સમાન હશે 0 અહીં અને નવી ધાર અહીં પાછો આવશે અને આ હવે રદ થઈ જશે. કારણ કે હવે આ પ્રવાહ જેનો પ્રવાહ છે તે બતાવે છે અને આ ધાર તેને લાયક છે અને હવે આ નવા ગ્રાફમાં એવું લાગે છે કે ત્યાં કોઈ કિનારી બાકી નથી, કારણ કે મારી પાસે ક્ષમતા 0 થી 0 ની ક્ષમતા છે, જે 0 ની ક્ષમતા છે. તેથી, હું કોઈપણ પ્રવાહને ઝીણી ન કરી શકું, તેથી આ મારી શોધ થોડી છે, તેથી આ ફોર્ડ ફુલકર્સન એલ્ગોરિધમ છે, ચાલો આપણે ફરીથી તેને સમાન માળખામાં જોઈએ, પણ અમારી પાસે સંખ્યાઓનો થોડો અલગ સમૂહ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:07)

તો, આ એક ગ્રાફ છે જે એક જ વાર નથી, પરંતુ કેટલાક 20 અને 30 ના, તેથીઅહીં દાવો સાથે અંતઃકરણ એ છે કે તમામ 30 એકમો પ્રવાહ કરી શકે છે. પરંતુ, 30 યુનિટ બેંગના ધાર સાથે અગાઉના કેસથી વિપરીત પ્રવાહને વહન કરી શકતા નથી, કારણ કે જો હું 20 યુનિટ લઈશ અથવા તો તે મને 10 વત્તા 10 થી વિભાજીત કરશે. તેથી, મારે જાણવું જોઈએ કે 10 નીચે જવું જોઈએ અને 10 ત્યાં જવું જ જોઈએ , જો હું અહીં 10 મુકું તો આ 10 અને ત્યારબાદ આવનારા 10 આ 20 માંથી ભેગા થાય છે, તેથી આ ગ્રાફમાં મને 30 નો પ્રવાહ મળે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 09:44)

પરંતુ, જો હું ફોર્ડ-ફુલકર્સન(Ford-Fulkerson) એલ્ગોરિધમ શરૂ કરું, તો તે પાથને સંતૃપ્ત કરવાનો પ્રયાસ કરશે. તેથી, ધારો કે તે પાથ s થી d થી e થી t ની ઓળખે છે, પછી જો તે આ પાથને ઓળખે છે તો તે આ ધાર દ્વારા 20 પ્રવાહ મૂકે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:00)

તેથી, અમે આ પ્રવાહમાં પ્રારંભ કરીએ છીએ અને અમે અવશેષ ગ્રાફ તૈયાર કરીએ છીએ, બાકીના ગ્રાફ કહે છે કે આ 10 30 અવતરણ 20 છે. બાકીની ડી થી ઈ આ હવે 10 ની ક્ષમતા છે કારણ કે મારી પાસે 30 ની ક્ષમતા અને હું 23 થશે. અને વાદળી ધાર હવે છે, બાકીના ધાર જે મને 20 ની પાછળ ઘટાડવા દે છે, તેવી જ રીતે s થી ડી ની પહેલાની ક્ષમતા 20 ની 20 ની હતી. તેથી, તે ક્ષમતા હવે 0 ઘટાડે છે. પરંતુ, મારી પાસે પાછળની ધાર છે જે અમને આ પછીની જરૂરિયાતને પૂર્વવત્ કરવા અને સમાન પહોળાઈને ટીમાં લેવાની મંજૂરી આપે છે, e થી t ને 0 સુધી ઘટાડે છે, પરંતુ મારી પાસે પાછળનો ધાર છે. હવે, હું આ ગ્રાફમાં બીજા પાથની શોધ કરું છું, તેથી ઉદાહરણો માટે મને લાગે છે કે આ પાથ ત્યાં છે અને મારી પાસે s થી e થી d to t સુધીનો પાથ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:42)

અને

((સમયનોસંદર્ભ લો: 10:44))

અવરોધ 10 છે, કારણ કે મારી પાસે માત્ર 10 ની શરૂઆત થઈ છે, તેથી હું તે 10 લઈશ અને પછી હું અવશેષ ગ્રાફ બનાવીશ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:52)

તો, જ્યારે હું તે 10 ને લઈશ ત્યારે આ ધાર સાથે સંકળાયેલા આ જથ્થાને પાછળ પ્લગ ઈન સૂચિબદ્ધ કરો. કારણ કે ડીથી ઈ ની કુલ ફ્લો અગાઉના રાઉન્ડમાં 10 ની અંદર 20 હતી. બીજું રાઉન્ડ 10 છે, તેથી અવશેષો 30 ઓછા 10, તે 20 થશે અને બાકીનું બધું હવે લગભગ 0 થશે કારણ કે સંતૃપ્ત છે, હવે જો હું જોઉં તો બાકીના ગ્રાફમાં કોઈ આઉટગોઈંગ ફ્લો શક્ય નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:25)

તેથી, હું કહું છું કે ત્યાં કોઈ વધુ શક્ય માર્ગ નથી અને હું રોકાઉં છું અને બાકી રહેલા કિનારો એ છે કે જો હું તે પ્રવાહને ટ્રૅક રાખું છું તો તે મને કહેશે કે મેં 30 માં પ્રવાહ પ્રાપ્ત કર્યો છે આ ગ્રાફ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 11:35)

તેથી, ફરી જો તમે પ્રશ્ન પૂછવા માગતા હોવ તો આપેલ ફ્લો શ્રેષ્ઠ છે, તો અમે કેટલાક પ્રમાણ માટે કહી શકીએ જે શ્રેષ્ઠતા પ્રમાણપત્ર છે. તેથી, જો આપણે મૂળ ઓઈલ શિપિંગ પર પાછા જઈએ છીએ તો આપણે માનીએ છીએ કે આપણે 7 નું પ્રવાહ સેટ કરી શકીએ છીએ. તેથી, હાથ દ્વારા અમે ફ્લોર સપોર્ટ બનાવ્યું, હવે ચાલો આપણે આ ત્રણ ધાર, ધાર ધારથી ડી, ધાર તરફ જુઓ બી થી ડી, એજ સે થી સી, જો આપણે આ આકારોને કાપીને આ ગ્રાફને ડિસ્કનેક્ટ કરીએ, તો અન્ય જો આપણે આ ધાર કાપીશું તો અમે આ ગ્રાફને ડિસ્કાઉન્ટ કરીશું. તેથી, આ ત્રણ કિનારીઓને એક સમૂહ વચ્ચે કટ કહેવામાં આવે છે, હવે આમાં કુલ ક્ષમતા 4 વત્તા 1 વત્તા 2, 7 માત્ર કોઈપણ પ્રવાહ પર કાપી શકે છે; જોકે, તે વહે છે તે આ બાજુથી ડાબેથી જમણે આ બાજુથી પસાર થવું આવશ્યક છે. તેથી, તે કાપીના ભાગ સાથે ફક્ત કાંડાથી જમણે જમણી બાજુથી વહે છે અને કટ માત્ર 7 નું પ્રવાહ સપોર્ટ કરી શકે છે. તેથી, આ ઉદાહરણમાં મહત્તમ પ્રવાહ ચોક્કસપણે 7 કરતા વધી શકતું નથી, જે આપણે પહેલાથી પ્રાપ્ત કરી લીધું છે. . તેથી, આપણે જાણીએ છીએ કે આપણે 7 નું પ્રવાહ કરી શકીએ છીએ, પરંતુ 7 થી વધુ શક્ય નથી, કારણ કે આ કટ ડાબેથી જમણી તરફ વહેતા પ્રવાહથી 7 કરતા વધુ કોઈપણ વિચારોને અટકાવે છે. તેથી, વાસ્તવમાં આ કિસ્સામાં બતાવે છે કે 7 શ્રેષ્ઠ છે, તેથી સામાન્ય રીતે જો આવા વિવિધ કટ જોવા મળે છે. તેથી, કટ એ ડિસ્કનેક્ટ સાથે ટીના કોઈપણ સમૂહને કાપી નાખે છે અને તમે આ બધામાં ન્યૂનતમ કટ ગણતરી કરી શકો છો અને તે સ્પષ્ટ છે કે મહત્તમ પ્રવાહ ન્યૂનતમ કટ કરતા વધી શકતો નથી, કારણ કે તેની પાસે આ કટ ક્રોસ છે. તેથી, તમારે એક બાજુથી જવું પડશે, તે માત્ર તે જ ક્ષમતાઓને લઈ શકે છે, તેથી પ્રવાહ તે કરતા વધી શકતું નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 13:20)

તેથી, આશ્ચર્યજનક વાત એ છે કે વાસ્તવમાં તે આ ઉદાહરણમાં આપણે હંમેશાં સમાન હતા, પરંતુ તે હંમેશાં સમાન બનશે. તેથી, મહત્તમ પ્રવાહ મિનિ કટ પ્રમેય કહે છે, મહત્તમ પ્રવાહ વાસ્તવમાં ન્યૂનતમ કટની સમકક્ષ હંમેશા હોય છે. તેથી, અહીં સમજવાનો એક રસ્તો છે, તેથી જો આપણે જોઈએ છીએ કે lp સોલ્યુશન છે, જ્યારે આપણે મહત્તમ પ્રવાહ પ્રાપ્ત કરીએ છીએ, તો s એ t થી ડિસ્કનેક્ટ થઈ રહ્યું છે, જો હું વધારે વજન 0 અને હું જોઉં તો આગળ કોઈ માર્ગ નથી. તે ધાર દૂર કરો, પછી આગળ કોઈ માર્ગ નથી. તેથી, એસ ટીથી ડિસ્કનેક્ટ થઈ ગયું છે, તેથી ત્યાં એક કાટ છે જે અમુક ધાર છે જે ટીમાંથી ધારને ડિસ્કનેક્ટ કરે છે. હવે, ચાલો અવશેષ ગ્રાફમાં કોઈપણ ધારને જોઈએ જે ડાબેથી જમણી બાજુથી જમણી તરફ જાય છે. તેથી, બાકી બધું જ છે, તેથી આની અંદરના દરેક જગ્યાએ મારી પાસે બિનજરૂરી કિનારીવાળા રસ્તાઓ છે અને આની અંદરના દરેક જગ્યાએ મારી પાસે 50 નો માર્ગ ઝીરો નાહોય એવી ધાર સાથે છે. તેથી, હવે પણ હું વાદળી બાજુથી લીલી બાજુથી મેળવી શકતો નથી. તેથી; તેનો અર્થ છે કે, આગળની દિશામાં બધી કિનારીઓએ ક્ષમતાને સંતૃપ્ત કરી હોવી જોઈએ. તેથી, એલથી આર પ્રત્યે દરેક ધાર વાસ્તવમાં સંપૂર્ણ ક્ષમતા હોય છે, બાકીના ગ્રાફ એફમાં l થી l સુધીના ધાર વિશે, દાવા એ છે કે અહીં કેટલીક ક્ષમતાની ક્ષમતા છે, પછી તે એક વિરુદ્ધ બાજુ હશે જે આ રીતે જશે, જે બિનજરૂરી ક્ષમતા હશે. તેથી, ત્યાંથી કોઈ પાથ છે કે નહીં તે કોઈ પાથ હશે, તેથી આ 0 હોવું આવશ્યક છે. તેથી, વિરુદ્ધ ધાર 0 પર હોવું આવશ્યક છે, બધી આગળની ધાર સંપૂર્ણ ક્ષમતા હોવી આવશ્યક છે. તેથી, આ ખૂબ ચોક્કસ નથી, પરંતુ એક કારણ શા માટે છે, પરંતુ મહત્તમ પ્રવાહ વાસ્તવમાં આ કાપીને સંતૃપ્ત કરશે, પરંતુ આ કોઈપણ કટ હોઈ શકે છે. તેથી, તેથી લઘુત્તમ કટ ખાસ કરીને સંતૃપ્ત થવું જોઈએ અને તેથી, મહત્તમ પ્રવાહ હજી લઘુતમ કટ કરતા વધી શકતું નથી, તેથી મહત્તમએ ન્યૂનતમ કટ મૂલ્ય પ્રાપ્ત કરવું આવશ્યક છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:07)

તેથી, એક એવું લાગે છે કે ફોર્ડ-ફુલકર્સન(Ford-Fulkerson) એલ્ગોરિધમમાં વિશે સાવચેત રહેવું એ પાથને કેવી રીતે વધારવું તે પસંદ કરવું છે. તેથી, યાદ રાખો કે આપણા ઉદાહરણોમાં, અમે હીરાની આસપાસ જવાને બદલે તમે મધ્યમાં જાઓ છો. તેથી, આપણે એમ માનીએ છીએ કે મોડેલ થાય છે તે અહીં એક પુનરાવર્તન પછી છે, આપણે આ કેન્દ્રમાંથી પસાર થાય છે અને પછી આપણે 99 ને આ 0 થી 99 માં ઘટાડીએ છીએ અને આપણે 1 કદના વિપરીત કિનારીને સુયોજિત કરીશું. પછી આ આગળની પુનરાવર્તન આપણે કરીશું ઉલટા દિશામાં જાઓ અને અમે 99 ને આ 99 સુધી ઘટાડીશું અને પછી આને ફરીથી આ રીતે રીસેટ કરશે. તેથી, આ રીતે ઝિક-ઝૅકિંગ રાખી શકે છે. તેથી, હું 99 થી 98 સુધી જઈશ અને ડાઉન એલ્ગોરિધમ 99 થી 98 સુધી જઈશ, તેથી આ પાથ શોધવા માટે 200 પુનરાવર્તનો લેશે, આ પ્રવાહ બીજા 200 હાથ પર છે તે સ્પષ્ટ છે કે જો હું તેને બે પુનરાવર્તન કરી શકું તો cleaver કરવામાં આવી હતી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:59)

કારણ કે હું ફક્ત એટલું જ જાણું છું કે શરૂઆતમાં મને ખબર છે ત્યાં ત્યાં 200 નો માર્ગ અને સંતૃપ્ત છે. તેથી, તે બે ધાર છે અને ત્યાં 100 નો બીજો માર્ગ છે, અહીં 100 ના રસ્તામાં અહીં 100 ના રસ્તે છે. તેથી, બે પુનરાવર્તનમાં હું 200 પ્રાપ્ત કરી શકું છું, તેથી તે ખૂબ વિચિત્ર રીતે અને હું કેવી રીતે આગળ વધવા માટેનો માર્ગ પસંદ કરું છું, હું કેવી રીતે શક્ય માર્ગનો

ઉપયોગ કરી શકું છું જે બાકીના ગ્રાફમાં હજી પણ અસ્તિત્વ ધરાવે છે અને તેને કયું ઉમેરવા તે પસંદ કરે છે. તેથી, સામાન્ય રીતે આપણે કંઈક સારું કહી શકતા નથી અને ફોર્ડ-ફુલ્કરસન(Ford-Fulkerson) સમય લેશે જે ધારની ક્ષમતા માટે પ્રમાણસર છે જે એક મહાન વિચાર નથી. કારણ કે બધી ધાર મોટી હોઈ શકે છે, પરંતુ તમે કદાચ સીધું કહી શકો છો કે આ ધાર 100 લઈ શકે છે અને આગળનો ધાર 100 લઈ શકે છે. તેથી, એકવાર હું કહી શકું કે આખો રસ્તો 100 લાગી શકે છે, જો મારી પાસે એક સમયે હોય તો તે બને છે વધારાના પેરામીટર ગ્રાફનો વાસ્તવિક કદ નથી. પરંતુ, આપણે આ સંભવિત પાથ વ્યવસાય કેવી રીતે કરી શકીએ. તેથી, દર વખતે જ્યારે આપણે અવશેષ ગ્રાફ રચીએ છીએ, ત્યારે આપણે s થી t તરફનો પાથ શોધવો પડશે અને પછી વિસ્તૃત કરીશું. તેથી, આપણે આ કોર્સની શરૂઆતમાં જ જોયું હોત, આપણે ક્યાં તો શ્વાસ દ્વારા તેને પ્રથમ શોધ અથવા ઊંડાણમાં પ્રથમ શોધ કરીશું, એક નોડથી બીજા નોડ તરફનો પાથ શોધીશું તે સામાન્ય રીતે ગ્રાફનો નિકાસ છે, જો આપણે તેનો ઉપયોગ કરીએ શ્વાસ પ્રથમ શોધ, પછી આપણે જાણીએ છીએ કે આપણે કિનારીઓની સંખ્યાની દ્રષ્ટિએ સૌથી નાનો માર્ગ શોધીએ છીએ. તેથી, કોઈ એક સાબિત કરી શકે છે કે આમાં ફોર્ડ-ફુલકર્સન એલ્ગોરિધમ, જો આપણે શ્વાસનો ઉપયોગ પ્રથમ પુનરાવર્તનમાં દરેક શોધમાં કરીએ તો તે કયા માર્ગને વધારશે તે નક્કી કરવા માટે, પછી તમે કિનારીઓના સંદર્ભમાં હંમેશાં ટૂંકા માર્ગનો વિકાસ કરશો. તેથી, અહીં આ ઉદાહરણમાં ઉદાહરણો માટે, જો તમારી પાસે તે માર્ગ જે 100 જેટલો રસ્તો પસાર કરે અને જે માર્ગ જેવો હોય તે માર્ગની વચ્ચે પસંદગી હોય, તો શ્વાસની પ્રથમ શોધ એ કહેશે કે પહેલો લાલ માર્ગ મળ્યો છે. બે ધાર અને નારંગી પાથને ત્રણ કિનારીઓ મળી. તેથી, લાલ પાથ ટૂંકા છે, તેથી તે પહેલા પણ વધારો, તેથી જો આપણે શ્વાસનો ઉપયોગ પ્રથમ ફોર્ડ ફુલકર્સન એલ્ગોરિધમમાં શોધીએ છીએ, તો તે તારણ આપે છે કે તમને હંમેશાં કંઈક મળે છે જે કિનારીઓ પરના ઉત્પાદન પર પ્રસ્તાવના છે. તેથી, તે ક્ષમતાઓથી સ્વતંત્ર નેટવર્કના કદમાં બહુમતિરૂપ બનશે.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 05
લેક્ચર - 54
ઘટાડો (Reduction)

અમે જોયું છે કે આપણે અનેક સમસ્યાઓ ઉકેલવા માટે તકનીકી તરીકે રેખીય પ્રોગ્રામિંગનો ઉપયોગ કરી શકીએ છીએ અને ઔપચારિક રીતે આમાં સમાવેશ થાય છે જેને આપણે ઘટાડો કહીએ છીએ.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:12)

તેથી, રેખીય પ્રોગ્રામ્સને ઘટાડવાને બદલે, ચાલો જોઈએ કે આપણે વાસ્તવમાં નેટવર્ક પ્રવાહને કેવી રીતે ઘટાડી શકીએ છીએ જે આપણે પહેલાથી જોયેલી છે તે રેખીય પ્રોગ્રામ્સ ઘટાડી શકાય છે. તેથી, અહીં એક સમસ્યા છે જે આપણા પ્રવાહ સાથે કશું કરવાનું નથી. તેથી, અમારી પાસે શાળામાં કેટલાક કોર્સ ફાળવવાનું છે, અમારી પાસે શિક્ષકોનો સંગ્રહ છે અને અમારી પાસે અભ્યાસક્રમો અને દરેક શિક્ષકનો સંગ્રહ છે જે સૂચવે છે કે તે કયા અભ્યાસક્રમો શીખવવા માંગે છે. તેથી, અહીં આપણી પાસે 4 શિક્ષકો અબ્બાસ, ચિત્ર, મદન અને સુનિતા છે અને આપણી પાસે ચાર અભ્યાસક્રમો મઠ, ઈતિહાસ, જીવવિજ્ઞાન અને અર્થશાસ્ત્ર છે. તેથી, અબ્બાસ ઉદાહરણ માટે તે ઈતિહાસ અને જીવવિજ્ઞાન શીખવવા તૈયાર છે, તેથી આ બંને ધાર છે, તેથી આ સૂચવે છે કે અબ્બાસ આ અભ્યાસક્રમો શીખવવા સાથે છે. એ જ રીતે, જો આપણે મદનને જોઈ શકીએ તો, મદનને ઈતિહાસ અથવા અર્થશાસ્ત્ર શીખવવા માટે ખુશી છે. તેથી, દરેક શિક્ષકએ અભ્યાસક્રમોના સંગ્રહ તરીકે સૂચવ્યું છે કે તે અથવા તેણી શીખવવા તૈયાર છે. હવે, આપણે જે કરવું છે તે શિક્ષકો માટેના અભ્યાસક્રમો ફાળવવાનું છે. દેખીતી વાત એ છે કે, દરેક કોર્સ બરાબર એક શિક્ષક દ્વારા શીખવવામાં આવશે, પણ આપણે પણ ઇચ્છીએ છીએ કે દરેક શિક્ષક ફક્ત એક જ અભ્યાસ શીખવે અને તે કોર્સ કંઈક હોવું જોઈએ જે શિક્ષક શીખવવા તૈયાર હોય. તેથી, અમને ઉદાહરણ તરીકે ફાળવણી ગમશે નહીં, જે અબ્બાસને મઠ શીખવવા કહે છે, કારણ કે તે સૂચવે છે કે તે મઠ શીખવવા માટે આરામદાયક નથી. તેથી, અમે આવા ફાળવણી શોધવા માંગીએ છીએ કે દરેક કોર્સ એક પ્રશિક્ષક દ્વારા શીખવવામાં આવે છે, દરેક પ્રશિક્ષક ફક્ત એક જ અભ્યાસ શીખવે છે અને તે એક અભ્યાસક્રમ છે જે તે શીખવવા તૈયાર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:42)

તેથી, આને મેચિંગ સમસ્યા તરીકે ઓળખવામાં આવે છે, ખાસ કરીને તેને દ્વિપક્ષી મેચિંગ કહેવામાં આવે છે. તેથી, દ્વિપક્ષી ગ્રાફ શું છે? દ્વિપક્ષી ગ્રાફ એ છે કે જેમાં એક શિર્ષક બે જૂથોમાં આવે છે જેને આપણે વી 0 અને વી 1 તરીકે ઓળખીએ છીએ અને બધા ધાર વી 0 થી વિરુદ્ધ 1 થાય છે. તેથી, 0 ની અંદર કોઈ કિનારીઓ નથી, તેથી ત્યાં કોઈ કિનારીઓ નથી. 0. તેથી, અહીં આ એક ઉદાહરણ છે, તેથી શિક્ષકો માટે આ પ્રોગ્રામ v છે અને અભ્યાસક્રમો v1 છે, હવે અભ્યાસક્રમો વચ્ચે કોઈ કિનારીઓ નથી, શિક્ષકો વચ્ચે કોઈ કિનારીઓ નથી, બધા કિનારી શિક્ષકોમાંથી એક છે. કોર્સ હવે,

આપણે જે માંગ્યું છે તે ફાળવણી શોધવાનું છે. ફાળવણી શું કરશે? ફાળવણી એક ધાર પસંદ કરશે અને ઉદાહરણ તરીકે કહેશે કે અબ્બાસ ઇતિહાસ શીખવશે. અને એકવાર અબ્બાસને ઇતિહાસ સોંપવામાં આવે તે પછી, આનો અર્થ એ થયો કે આ ધાર હવે લઈ શકાશે નહીં, કારણ કે અબ્બાસને બીજું કંઈ પણ સોંપી શકાતું નથી, તેનો અર્થ એ પણ છે કે આ ધાર લેવામાં આવી શકતો નથી, બીજું કોઈ તેને સોંપશે નહીં, આ ધાર લઈ શકાતો નથી . તેથી, આપણે કિનારીઓ શોધવા માંગીએ છીએ, જેમ કે તેમાંથી કોઈ પણ એક અંત બિંદુ વહેંચશે નહીં. આ ઉદાહરણમાં, જો તેમાંના બે ડાબી બાજુએ એક અંતિમ બિંદુ વહેંચે છે, તો તેનો અર્થ એ કે એક જ શિક્ષકને બે અભ્યાસક્રમો શીખવવા માટે કહેવામાં આવે છે. જો તે બે ધાર છે, તો જમણી બાજુનો અંત બિંદુ વહેંચે છે, તેનો અર્થ એ છે કે તે જ અભ્યાસક્રમને બે ભિન્ન શિક્ષકોને ભણાવવામાં આવે છે, તેમાંથી એક તે છે જેને આપણે જોઈએ છે. તેથી, તમારે મેચિંગ જોઈએ છે, અમે મેળ ખાતા ઈચ્છીએ છીએ જે કિનારીઓનો સબસેટ છે, જેથી તેમાંના બેમાંથી કોઈ અંતિમ બિંદુ વહેંચે નહીં. અને ખાસ કરીને, જો દ્વિપક્ષી ગ્રાફમાં દરેક બાજુ પર સમાન સંખ્યામાં ગાંઠો હોય, તો આપણે અપેક્ષા રાખી શકીએ કે આશા રાખીએ કે ડાબી બાજુની દરેક વસ્તુ જમણી બાજુએ કંઈક સાથે મેળ ખાતી હોય. જો આપણી પાસે સમાન સંખ્યા નથી, તો કંઈક છોડી દેવામાં આવ્યું છે. પરંતુ દરેક ધાર બે લોકો સાથે મેળ ખાય છે, માની લો કે આપણી પાસે પાંચ શિક્ષકો અને ચાર અભ્યાસક્રમો છે, પછી એક શિક્ષક કોઈ કોર્સ શીખવશે નહીં અથવા આપણી પાસે ચાર શિક્ષકો અને પાંચ અભ્યાસક્રમો હશે, પછી ભલે શિક્ષક બે કોર્સ લે નહીં ત્યાં સુધી કેટલાક કોર્સ શીખવવામાં આવશે નહીં. પરંતુ તમારી પાસે ચાર શિક્ષકો અને ચાર અભ્યાસક્રમો છે, તમે તેમને પૂછી શકો છો કે શું તેમને મેચ કરવાની રીત છે કે જેથી દરેક શિક્ષક એક અભ્યાસક્રમ શીખવે અને શિક્ષક દ્વારા દરેક અભ્યાસક્રમ શીખવવામાં આવે. આ એક સંપૂર્ણ મેચ કહેવાય છે. તેથી, આ નેટવર્ક પ્રવાહ સાથે શું કરવાનું હતું?

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:48)

તો, અહીં આપણે નેટવર્ક પ્રવાહનો ઉપયોગ કરીને આ પ્રશ્નનો જવાબ કેવી રીતે આપી શકીએ તે અહીં છે. અમે શિક્ષકોમાં એક બનાવટી સ્ત્રોત નોડ ફીડિંગ ઉમેરીએ છીએ અને અમે એક ગુરુત્વ લક્ષ્ય નોડ, અભ્યાસક્રમોમાંથી બહાર આવતા સિંક નોડને ઉમેરીએ છીએ. તેથી, અસ્પષ્ટપણે બધા કિનારીઓ ડાબેથી જમણે જઈ રહ્યા છે, તેથી અમારી પાસે દિશા છે, તેથી અમે ડાબેથી જમણે કંઈક વહન કરવાનો પ્રયાસ કરી રહ્યા છીએ. હવે, આપણે જે જોઈએ છીએ તે દરેકને ક્ષમતા એક અસાઈન કરવાની છે, તેથી આમાંના પ્રત્યેક ધારની ક્ષમતા એક છે. તેથી, અમને લાગે છે કે અમે મહત્તમ પ્રવાહ શોધીએ છીએ, તેથી અમારું મહત્તમ પ્રવાહ આ શિક્ષકોના કેટલાક સબસેટને પસંદ કરશે, તે બધાને આશા છે. કારણ કે તે એક પ્રવાહ છે કે જે પ્રવાહ શિક્ષક અને શિક્ષકમાંથી જશે, તેથી હું આ ધારમાંથી એક પસંદ કરીશ, તેથી આ એક. અને તેથી, હવે તે અહીંથી બહાર આવશે, કારણ કે આઉટપુટ ફક્ત એક જ લઈ શકે છે, હું ફક્ત 1 ઇતિહાસમાંથી સિંક સુધી જ પ્રવાહ કરી શકું છું, તે અહીં બીજી વસ્તુને આવવા દેતી નથી. તેથી, જો મારી પાસે અહીં 1 નું પ્રવાહ હતું અને અહીં 1 નું પ્રવાહ છે, તો 2 ને હિસ્ટ્રી પર પ્રયાણ કરવું પડશે, પરંતુ મને તે મળી ગયું છે. તેથી, હું દરેક ધાર પર 1 ની ફ્લો ક્ષમતા ધરાવી રહ્યો છું અને મને ખાતરી છે કે મેચમાં એક જ શિક્ષક અને એક કોર્સ પસંદ કરી શકાય છે. અને હવે, જો તમે આ પ્રવાહને મહત્તમ કરો છો, તો તે શિક્ષકોની સંખ્યાને મહત્તમ કરે છે જે તેમના અભ્યાસક્રમો સાથે જોડાયેલા છે અને તેથી તે મેચિંગને મહત્તમ કરે છે. તેથી, અમે ગ્રાફમાં ખૂબ જ ભિન્ન ભિન્ન કંઈક શામેલ કરવામાં સમસ્યાને સંચાલિત કરી છે અને નેટવર્ક પ્રવાહનો ઉપયોગ કરીને તેને મોડ્યુલેટ કરીએ છીએ.

(સ્લાઈડસમયનો સંદર્ભ લો: 05:24)

તેથી, આ એક સામાન્ય છે આપણે જે ઘટાડો કહીએ છીએ તેનું ઉદાહરણ. તેથી, આપણે આપેલ સમસ્યા, સમસ્યા A ને ઉકેલવા માંગીએ છીએ, પરંતુ આપણે તેને કેવી રીતે ઉકેલવું તે વિશે જાણતા નથી, પરંતુ બી સમસ્યાને કેવી રીતે ઉકેલવું તે આપણે જાણીએ છીએ. તેથી, આપણા કિસ્સામાં આ બંધબેસે છે અને આ પ્રવાહ છે. તેથી, આપણે શું કરીએ છીએ તે છે કે આપણે તે સમસ્યાને મેચિંગથી પ્રવાહમાં રૂપાંતરિત કરીએ છીએ. તેથી, અમે અમારી મેળ ખાતી સમસ્યાને લઈએ છીએ અને પછી આપણે એક સ્રોત ઉમેરીએ છીએ, આપણે એક સિંક ઉમેરીએ છીએ, આપણે ક્ષમતાઓને ફ્લો સમસ્યામાં રૂપાંતરિત કરીએ છીએ. તેથી, તેને ફ્લો પ્રોબ્લેમમાં પરિવર્તિત કર્યા પછી, અમે તેને ઉકેલ્યું, તેથી નીચે બતાવેલ આકૃતિ છે, આપણી પાસે બી માટે એક અલ્ગોરિધમ છે. તો હવે, આપણી પાસે મેચિંગ ઈનપુટ છે, આપણે તેને પ્રવાહ ઈનપુટમાં રૂપાંતરિત કર્યું છે, આ એક પ્રવાહ છે. સોલ્યુશન જે બહાર આવે છે અને હવે આપણે ફ્લો સોલ્યુશન તરફ ધ્યાન આપીએ છીએ, જ્યાં આપણે એક મેચ જોઈ શકીએ છીએ જે આપણું મેળ ખાતું હોય, જ્યારે પણ આપણે શૂન્ય જોઈએ ત્યારે તે મેચિંગ નથી. તેથી, તેમાંથી આપણે મેળ ખાતા ઉકેલ મેળવીએ છીએ. તેથી, બાહ્ય સમસ્યાના ઉકેલ માટે અમે આંતરિક સમસ્યાનો ઉકેલ સમાવી શકીએ છીએ. તેથી, આ ઘટાડો છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:25)

તેથી, આપણે કહીએ છીએ કે બી ઘટાડે છે. એનો ઉકેલ લાવવા માટે, હું તેને B માં રૂપાંતરિત કરીશ અને રાજ્યમાં B ને હલ કરી શકું. અને તેથી, જો મારી પાસે બી માટે કાર્યક્ષમ સોલ્યુશન હોય અને જો આ રૂપાંતર પ્રક્રિયા કાર્યક્ષમ હોય, તો પ્રક્રિયા બી માટે સમસ્યાને સેટ કરી રહી છે અને જવાબનો અર્થ બંને કાર્યક્ષમ છે. તે પછી, પૂર્વ પ્રોસેસિંગની કાર્યક્ષમતા સાથે બી કંપોઝની એલ્ગોરિધમની કાર્યક્ષમતા છે અને પોસ્ટ પ્રોસેસિંગ પગલું એ અમને A માટે કાર્યક્ષમ અલ્ગોરિધમ આપશે. તેથી, સીને સીધી રીતે હુમલો કર્યા વિના, મેં પરોક્ષ રીતે શોષણ દ્વારા એ માટે કાર્યક્ષમ અલ્ગોરિધમ મેળવ્યું. બી માટે અલ્ગોરિધમનો અસ્તિત્વ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:03)

તેથી, જેમ આપણે આપણા ઉદાહરણ પહેલા કહ્યું હતું કે આપણે આ ભાષણમાં કર્યું છે તે કહે છે કે આ રીતે દ્વિપક્ષી મેચિંગ મેચ ફ્લોમાં ઘટાડો કરે છે. હવે, આપણે જે જોયું હતું, તે છેલ્લા સમયે મહત્તમ પ્રવાહ હતો તે રેખીય પ્રોગ્રામને ઘટાડે છે, પરંતુ કી એ છે કે પૂર્વ પ્રક્રિયા અને પોસ્ટ પ્રક્રિયા કાર્યક્ષમ હોવી આવશ્યક છે. જ્યારે અમે મહત્તમ પ્રવાહથી રેખીય પ્રોગ્રામિંગમાં ગયા, ત્યારે અમે શું કર્યું તે દરેક ધાર માટે અમે એક વેરિયેબલ ફી રજૂ કર્યું. તેથી, તે એક કાર્યક્ષમ અનુવાદ છે, કારણ કે તે રેખીય પ્રોગ્રામ બનાવે છે જેની કદ મહત્તમ પ્રવાહના ઈનપુટ ગ્રાફ સાથે તુલના કરી શકાય છે. અગાઉ, અમે નેટવર્ક બેન્ડવિડ્થનું એક ઉદાહરણ જોયું હતું, જ્યાં અમે રેખીય પ્રોગ્રામ પેદા કરીએ છીએ જેમાં નેટવર્કમાં એક વેરિયેબલ દીઠ પાથની આવશ્યકતા હોય છે, જે એક કાર્યક્ષમ ઘટાડો હોઈ શકતી નથી, કારણ કે તેને રેખાંકન પ્રોગ્રામ બનાવવા માટે કામના જથ્થાત્મક પ્રમાણની જરૂર પડશે અને કાર્યક્રમ વાંચવા માટે. તેથી, જો આપણે જોતા હોત તો, તમારે સમસ્યાનું સમાધાન કરવાનું ધ્યાન રાખવું એ એક સમસ્યા છે, પરંતુ જો આપણે કાર્યક્ષમતાના પાસાને શોષિત કરવા માંગીએ તો ધ્યાન રાખવું પડશે, પછી અનુવાદને બે અને આંતરિક સમસ્યામાંથી આવશ્યક છે. પણ કાર્યક્ષમ રહો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 08:13)

તેથી, આપણે જોશું કે ઘટાડાનું બીજું અર્થઘટન પણ આપણા માટે ઉપયોગી છે. તેથી, આપણે જે જોયું છે તે એ છે કે જો બીને ઘટાડે છે, તો બી કાર્યક્ષમ છે, આનો અર્થ એ છે કે એ પણ કાર્યક્ષમ છે, કારણ કે હું બી માટે સોલ્યુશનનો ઉકેલ પણ વાપરી શકું છું. પણ મને લાગે છે કે આ નથી કુશળ અથવા કાર્યક્ષમ હોવાનું જાણીતું નથી, જો મારી પાસે શંકા કરવાની કોઈ કારણ હોય તો એ કરવા માટે કોઈ કાર્યક્ષમ રસ્તો નથી, તો તે બતાવી શકે છે કે A ઘટાડે છે B સુધી, પછી આનો અર્થ એમ પણ થાય છે કે બી કાર્યક્ષમ નથી. કારણ કે તે કાર્યક્ષમ રીતે કરી શકાય છે, આપણે જાણીએ છીએ કે આપણે કાર્યક્ષમ રીતે કાર્ય કરી શકીએ છીએ, પરંતુ અમારી પાસે એ માનવા માટે એક કારણ છે કે A કાર્યક્ષમ રીતે કરી શકાતું નથી. તેથી, આપણે જે રીતે શોષણ કર્યું છે તેના આધારે ઘટાડેલી રીતનો ઉપયોગ B થી A ના હકારાત્મક પરિણામ સ્થાનાંતરિત કરવા અથવા A થી B ના નકારાત્મક પરિણામને સ્થાનાંતરિત કરવા માટે થાય છે, એ એ છે કે તે કાર્યક્ષમ હોવાનું જાણીતું નથી, બી પણ જાણીતું નથી. કાર્યક્ષમ

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 09:10)

તેથી, આ છેલ્લા કેટલાક પ્રવચનોમાં આપણે જે જોયું છે તે બે છે, હું મોટા હેમર્સને શું કહીશ. તેથી, અમારી પાસે રેખીયર પ્રોગ્રામિંગ અને નેટવર્ક પ્રવાહ છે. અમે રેખીય પ્રોગ્રામ રૂપે વાસ્તવમાં નેટવર્ક પ્રવાહ વ્યક્ત કરીએ છીએ, પરંતુ સ્વતંત્ર રૂપે નેટવર્ક પ્રવાહ ખૂબ શક્તિશાળી મોડેલિંગ ઔપચારિકરણ છે. તેથી, ઘણી એલ્ગોરિધમિક સમસ્યાઓ વાસ્તવમાં આમાંના એકમાંથી ઘટાડી શકાય છે, અથવા તો તમે રેખીય પ્રોગ્રામના સેટમાં ચલો ગોઠવી શકો છો અથવા તમે અસ્પષ્ટ સમસ્યાને ગ્રાફ લઈ શકો છો અને પ્રવાહનો ઉપયોગ કરીને તેને મોડ્યુલેટ કરી શકો છો. અને આ વિશેની સરસ વસ્તુ એ છે કે રેખીય પ્રોગ્રામિંગ અને નેટવર્ક પ્રવાહો બંને પ્રમાણભૂત સમસ્યાઓ છે જેના માટે લોકોને રસ છે, સામાન્ય હેતુ સાધનો લખવા માટે ઘણો સમય છે જે આ સમસ્યાને હલ કરે છે. તેથી, તમે મનસ્વી રેખીય પ્રોગ્રામ સેટ કરી શકો છો, મનસ્વી નેટવર્ક પ્રવાહ તેને પણ આપે છે, પછી સાધન તમને જવાબ આપશે. તેથી, તમારી પાસે સ્વ અમલીકરણની કાર્યક્ષમતા છે. તેથી, ઘણી વ્યવહારિક પરિસ્થિતિમાં સીધી રીતે ઍલ્ગોરિધમિક સમસ્યાને અજમાવવા અને હલ કરવાને બદલે, તેને લીનિયર પ્રોગ્રામ અથવા નેટવર્ક ફ્લો તરીકે મોડેલ કરી શકાય છે કે નહીં તે વિચારવામાં ઘણીવાર અર્થ થાય છે અને પછી તેને ઉકેલવા માટે ઑફ-ધ-શેલ્ફ પેકેજનો ઉપયોગ કરો. પરંતુ અલબત્ત, તમારે આમાંના કોઈ એક બાબતના સંદર્ભમાં સમસ્યા વ્યક્ત કરવા સક્ષમ હોવા જોઈએ. તેથી, તમે તે કરી શકો છો કે નહીં તે સમજવા માટે, તમારે રેખાંકન પ્રોગ્રામ્સને શું વ્યક્ત કરી શકે છે અને શું ન હોઈ શકે તે વિશ્લેષણ કરવામાં થોડો સમય પસાર કરવો પડશે. રેખીય પ્રોગ્રામમાં, નિર્ણાયક બાબત એ છે કે અવરોધો રેખીય કાર્યો હોવી આવશ્યક છે. કેટલીકવાર, તમારી મર્યાદાઓ જે ઉત્પાદનને બે ચલોમાં ઉલ્લેખિત કરે છે, તે રેખાકીય કાર્ય નથી. તેવી જ રીતે, નેટવર્ક પ્રવાહમાં તમારે ખાતરી કરવી પડશે કે તમે પ્રવાહને લક્ષ્ય બનાવવા માટે સ્રોત તરીકે ખરેખર મોડ્યુલેટ કરી શકો છો, તે નેટવર્કમાં આસપાસ અને આસપાસ કંઈક ચાલી રહ્યું નથી. તેથી, આ અવરોધો અંદર; જો કે, આ બંને અત્યંત ઉપયોગી સામાન્ય સમસ્યાઓ છે જેના માટે કાર્યક્ષમ સોલ્યુશન્સ અસ્તિત્વ ધરાવે છે અને જે આપવામાં આવેલી સમસ્યાને ઉકેલવા માટે મોટી સંખ્યામાં પરિસ્થિતિઓમાં શોષણ કરી શકાતું નથી.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 06
લેક્ચર - 55
અવ્યવહારક્ષમતા: એલ્ગોરિધમ્સ તપાસવું

આમાંનો મોટા ભાગનો અભ્યાસ સમસ્યાઓના કાર્યક્ષમ ઉકેલને ઓળખવા વિશે છે. પરંતુ એ પણ જાણવું અગત્યનું છે કે કેટલીક પરિસ્થિતિઓ છે, જ્યાં કોઈ જાણીતું કાર્યક્ષમ ઉપાય બહાર આવતું નથી, અને અમે તેને ઓળખી શકીએ છીએ, જેથી આપણે નિરાશાજનક રીતે નિરાકરણને શોધવાનો પ્રયાસ ન કરીએ, જ્યાં સમસ્યાને ઉકેલવા મુશ્કેલ હોઈ શકે છે. તેથી, ચાલો આપણે અવ્યવહારક્ષમતાને લગતી કેટલીક સમસ્યાઓ જોઈએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 00:24)

તેથી, આપણે જે ઘણી સમસ્યાઓ જોયેલી છે, અમે કેટલાક સંભવિત સંજોગોમાં શોધવાનો પ્રયાસ કરી રહ્યા છીએ જેથી કેટલાક પ્રકારના શ્રેષ્ઠ સંયોજન તરીકે પહોંચી શકાય. વાસ્તવિક શોધ જગ્યા ઘોષણાત્મક છે, જો આપણે ટૂંકા માર્ગો શોધી રહ્યા છીએ, પાથની ઘાતાંકીય સંખ્યા છે, જો આપણે લઘુત્તમ ખર્ચવાળા વૃક્ષની શોધ કરી રહ્યા છીએ, તો આ પ્રકારના વૃક્ષો શોધી કાઢવા માટે ઘાતાંકીય સંખ્યા છે. અમે મહત્તમ પ્રવાહ શોધી રહ્યા છીએ, અમારી પાસે કિનારીઓ સાથેના પ્રવાહને ઉમેરવા અને બાદ કરતાં ઘણાં વિવિધ માર્ગો છે. તેથી, જો તમે બધા સંભવિત પ્રવાહ જુઓ, બધા સંભવિત રસ્તાઓ, બધા સંભવિત વૃક્ષો અને પછી મહત્તમ પસંદ કરો, તો તે બ્રુટ ફોર્સ સોલ્યુશન હશે જે ઘાતાંકીય સમય લેશે. જ્યારે આપણી પાસે પોલિનોમિયલ ટાઈમ એલ્ગોરિધમ હોય છે, ત્યારે આપણે ખરેખર ઘાતાંકીય શબ્દો દૂવારા અને કેટલાક ખૂબ જ સખત માર્ગે કાપતા હોઈએ છીએ, અમારા શોધ સ્થાનને ઘાતાંકીયથી એક બહુમતિથી ઘટાડે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:15)

તેથી, તે માનવા માટે ટેન્નીંગ છે કે જો કોઈ ફક્ત લાંબા સમય સુધી પૂરતી અને સખત રીતે વિચારે છે, તો આવી કોઈ સમસ્યા છે, આપણે હંમેશાં આવા કાર્યક્ષમ ટૂંકા કટ શોધીશું, જ્યાં આપણે ઘાતાંકીય જગ્યામાંથી કાપ કરી શકીએ છીએ. શક્યતાઓ અને ઝડપી જગ્યાને એક બહુમતિપૂર્ણ સંખ્યામાં ઝડપથી સાંકળો, જેનાથી કાર્યક્ષમ ઉકેલ, આપણે જે વાસ્તવિક ઉકેલ જોઈએ છીએ તે ઉદ્ભવશે. હવે, દુર્ભાગ્યે આ આદર્શ વિશ્વ વાસ્તવમાં જગત નથી. તેથી, ઘણી સમસ્યાઓ છે, તેથી કયા કાર્યક્ષમ અલ્ગોરિધમ અસ્તિત્વમાં નથી અથવા અસ્તિત્વમાં નથી જાણતા અને કમનસીબે આમાંની ઘણી સમસ્યાઓ ખૂબ જ મહત્વપૂર્ણ વ્યવહારિક સમસ્યાઓ છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 01:56)

તેથી, આ ચર્ચામાં જવા માટે, ચાલો આપણે સમસ્યાનું નિર્માણ, ઉકેલ ઉત્પન્ન કરવા અને ઉકેલને ઉકેલવા વચ્ચેના તફાવત વિશે વાત કરીએ. તેથી, શાળાના મઠના શિક્ષકને નીચે મુજબનું હોમવર્ક સોંપેલું છે, મોટી સંખ્યામાં લો કે જે બે મુખ્ય નંબરોના ઉત્પાદન તરીકે જાણીતી છે અને આ બે મુખ્ય નંબરો શોધો. વિદ્યાર્થી દ્રષ્ટિકોણથી દેખીતી રીતે, સમસ્યાઓ ઉકેલ લાવવાની સમસ્યાઓ છે. તેથી, મોટી સંખ્યામાં એન આપવામાં આવે છે, વિદ્યાર્થીને બે મુખ્ય નંબરો પી અને ક્યૂ શોધવાનું અપેક્ષિત છે, જેમ કે પી ગુણ્યા ક્યૂ. Q બરાબર એન છે. હવે, વિદ્યાર્થી સબમિટ કરે છે અથવા વિદ્યાર્થીઓ મૂલ્યાંકન માટે શિક્ષકને તેમના ઉકેલો સબમિટ કરે છે. તેથી, શિક્ષક વધુ સારી રીતે સંબંધિત છે, શિક્ષકને પી અને ક્યૂ બનાવવાની જરૂર નથી, શિક્ષકને જવાબ જાણવાની જરૂર નથી. શિક્ષક ફક્ત વિદ્યાર્થી દ્વારા આપવામાં આવેલ જવાબ લઈ શકે છે, પી ગુણ્યા ગુણ q ને ગુણાકાર કરી શકે છે અને નક્કી કરી શકે છે કે પી ગુણ્યા X બરાબર એન છે કે નહીં. તેથી, જવાબને જાણ્યા વગર અથવા તો જવાબ ખોટો હોવા છતાં, શિક્ષક યોગ્ય જવાબ જાણતા નથી, શિક્ષક નક્કી કરે છે કે જે વિદ્યાર્થી યોગ્ય જવાબ આપે છે કે નહીં. તેથી, શિક્ષક શું કરી રહ્યું છે તે તે સોલ્યુશન ચકાસી રહ્યું છે કે જે વિદ્યાર્થી સોલ્યુશન કરવા અથવા બનાવવાની કોશિશ કરી રહ્યો છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 03:12)

તેથી, આ અમને એક ચકાસણી અલ્ગોરિધમનો વિચાર આપે છે. તેથી, જો તમને કોઈ સમસ્યા હોય તો, હું કહી શકું છું કે મારી પાસે એક ચકાસણી અલ્ગોરિધમનો છે, જો દરેક ઉદાહરણ માટે હું તે ઉદાહરણનો ઉકેલ લાવી શકું છું અને ઝડપથી તે માન્ય છે કે નહીં તે સચોટ છે. તેથી, એલ્ગોરિધમ તપાસવાનું સમસ્યા માટે ઈનપુટ લે છે, એક ઉકેલ કે જે માત્ર ઉકેલ જ નથી, પરંતુ સોલ્યુશન ઉપરાંત કેટલીક વધારાની માહિતી, અને પછી તે નક્કી કરે છે કે નહીં તે એક માન્ય ઉકેલો છે, જો તેમ હોય તો તે સોલ્યુશન કહે છે સાચી છે અને એસ આઉટપુટ છે, અન્યથા તે કહે છે. તેથી, પરિબળકરણ ઉદાહરણ પહેલાં આપણાં ઉદાહરણમાં, ઈનપુટ ઘટક એ પરિબળ બનવાની સંખ્યા N છે. સમસ્યાનો ઉકેલ લાવવા માટેના ઉમેદવાર તરીકે આપણે જે ઉકેલ મેળવીએ છીએ તે છે primes p અને q ની જોડી જે વિદ્યાર્થીની ગણના થાય છે અને ચકાસણી અલ્ગોરિધમનો સમાવેશ થાય છે તે ચકાસવું શામેલ છે કે પી વખત q એ વાસ્તવમાં એન છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:12)

તેથી, આ સંદર્ભમાં, ચાલો આપણે ખૂબ જ કેનોનિકલ સમસ્યા જોઈ શકીએ જેમાં બુલિયન(Boolean) સંતોષકારક ચકાસણી અલ્ગોરિધમનો સમાવેશ થાય છે. તેથી, આપણી પાસે કેટલાક બુલિયન(Boolean) વેરિયેબલ x, y અને z છે. તેથી, બુલિયન(Boolean) વેરિયેબલ મૂલ્ય, સાચું અથવા ખોટું લઈ શકે છે અને બુલિયન(Boolean) વેરિયેબલ્સ પર અમારી પાસે પ્રમાણભૂત કામગીરી છે, નકારાત્મક સત્યનું મૂલ્ય લે છે અને ખોટા અને શુધ્ધ કલમોને ટ્રાંઝિટ કરે છે. તેથી, અમે કેટલાક પ્રોગ્રામિંગ ભાષા પરિભાષાઓનો ઉપયોગ કરીને આ ઉદ્ગારવાચક ચિહ્ન સાથે લખીએ છીએ, તેથી એક્સ નથી, તે એક્સક્લેમેશન ચિહ્ન x નો ઉપયોગ કરે છે. પછી, આપણી પાસે x અથવા y છે જે સાચું છે, તેમાંથી ઓછામાં ઓછું એક તે સાચું છે, તેથી જો x એ સાચું છે અથવા વાય સાચું છે અથવા બંને સાચું છે અથવા બંને સાચું છે, તો x અથવા y true છે, આપણે તેને આ વર્ટિકલ બાર અથવા જોડી સાથે લખીએ છીએ. વર્ટિકલ બાર અને એમ્પર્સાન્ડ નો ઉપયોગ કરો AND, x અને y એ ફક્ત ત્યારે જ સાચું છે જો બંને સાચું હોય. તેથી, x એ સાચું હોવું જોઈએ અને વાય true હોવું જોઈએ, કાં તો એક ખોટું છે, x અને y ખોટું છે. તેથી, આ પ્રમાણભૂત બુલિયન(Boolean) ઓપરેશન્સ છે જે

આપણે બુલિયન(Boolean) ચલો વિશે જાણીએ છીએ. હવે, આપણે ખૂબ જ ખાસ સ્વરૂપમાં બુલિયન(Boolean) ફોર્મ્યુલા સેટ કર્યા છે, અમે કહ્યું છે કે આપણે જે પહેલું કહ્યું છે તે ક્લોઝ છે. તેથી, કલમ એ એક મોટો ભાગ છે, તે x છે અથવા વાય અથવા z અને કંઈક નથી. તેથી, જોડાણની અંદર અથવા તો ચલો અથવા તેમની નકારાત્મકતાઓની અંદર શું છે. તેથી, આને શાબ્દિક કહેવામાં આવે છે, તેથી શાબ્દિક એક વેરિયેબલ અથવા નકારાત્મક વેરિયેબલ છે. તેથી, ક્લોઝ એક્સક્લોઝ એક્સ અથવા વાય અથવા ઝેડ ડોટ, ડોટ, ડોટ અથવા ડબલ્યુ તરીકે વાંચવામાં આવે છે. તેથી, આ એક કલમ છે અને છેવટે, ફોર્મ્યુલા એ AND દ્વારા જોડાયેલા આવા ક્લોઝનો સંયોજન બનશે. તેથી, સી કેટલાક એક્સ અથવા વાય અથવા કંઈક હશે, ડી કેટલાક ઝેડ હશે અથવા વાય નહીં અથવા કંઈક બીજું હશે અને તેથી દરેક કલમમાં શાબ્દિકના મોટા જોડાણનો આ માળખું હશે અને પછી તે ઍમ્પર્સંડ દ્વારા જોડાયેલા છે, તેથી તે છે બધા એકસાથે નિયંત્રિત. તેથી, મને ક્લોઝ સી સાચી બનાવવાની જરૂર છે અને મને ક્લોઝ ડી સાચી બનાવવાની જરૂર છે અને મને ક્લોઝ ઈ સાચી બનાવવાની જરૂર છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 06:05)

તેથી, અમારું લક્ષ્ય એ છે કે x, y અને z પર યોગ્ય મૂલ્યોને અસાઈન કરીને આ આપવામાં આવેલ ફોર્મ્યુલા સાચું કરી શકાય છે કે કેમ; સૂત્રમાંના બધા ચલો. તેથી, આ મૂલ્યાંકન કહેવાય છે. વેલ્યુએશન એ એક ફંકશન છે જે x ને સાચું કહે છે, વાય ખોટું છે, z true છે અને તેથી, તેથી તે દરેક બુલિયન(Boolean) વેરિયેબલનું મૂલ્ય સુધારે છે. હવે, ઠીક છે કે હું મૂલ્યાંકન કરી શકું છું, તેથી ઉદાહરણ તરીકે, જો હું x ને સાચું હોઉં, તો y true અને z ને false હોવા જોઈએ, તો પછી આ કલમમાં થી, x એ સાચું છે તેથી આને પૂર્ણ કરવા માટે પૂરતું છે. કલમ સાચું, વાય પણ સાચું છે, આ કલમમાં ઉદાહરણ તરીકે x એ સાચું છે, તેથી આ ભાગ સાચું છે, x એ સાચું છે. અહીં, તે કહે છે કે વાય સાચું છે, ઝેડ ખોટું છે, તેથી આ બંને વાસ્તવમાં સાચું છે અને હવે, અહીં તે કહે છે કે x એ ખોટું નથી, પણ x એ સાચું છે, તેથી x એ ખોટું નથી, y એ અમારા મૂલ્યાંકન દ્વારા સાચું નથી. ખોટું છે, પરંતુ સદભાગ્યે z એ ખોટું છે, તેથી z એ સાચું નથી. તેથી, આ દરેક કલમોમાં ઓછામાં ઓછા એક શાબ્દિક મૂલ્યાંકન આ મૂલ્યાંકન હેઠળ સાચું બને છે, તેથી આ સંપૂર્ણ ફોર્મ્યુલા ખરેખર સંતુષ્ટ છે. હવે, જો મારી પાસે આ ફોર્મ્યુલામાં આ વધારાની કલમ છે, તો હવે તે શબ્દ છે, એક્સ, વાય, ઝેડ બનાવવું તે સાચું ખોટું સોંપી દેવાનો કોઈ રસ્તો શોધવાનો કોઈ રસ્તો નથી. તમે ચકાસી શકો છો કે આ વિશિષ્ટ મૂલ્યાંકન કામ કરતું નથી, કારણ કે જો હું આને જોઉં છું કે x એ સાચું છે, તો આ કામ કરતું નથી અને તે કહે છે કે z એ ખોટું છે, તેથી તે કાર્ય કરતું નથી. તેથી, ઝેડ ખોટું છે, તેથી ઝેડ સાચું નથી, x એ સાચું છે, તેથી x ખોટું નથી, તેથી આ ખોટું અથવા ખોટું છે. તેથી, આ કલમ ખરેખર ખોટી કિંમત છે. તેથી, આ વિશિષ્ટ મૂલ્યાંકન જે પ્રથમ ફોર્મ્યુલાને સાચું બનાવે છે, તે બીજા ફોર્મ્યુલાને સાચું બનાવતું નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 07:50)

અને ખાસ કરીને કોઈ વેલ્યુએશન વાસ્તવમાં તે સાચું બનાવશે નહીં અને જો તમે આ ત્રણ કલમો જોશો તો વાસ્તવમાં તપાસ કરી શકાય છે, આ કહેવામાં આવે છે કે વાય y સૂચવે છે, આનો અર્થ એ છે કે જો તમે સાચું છે, તો x એ સાચું હોવું જ જોઈએ. અને આ કહે છે કે જો z true છે, તો y true હોવું જોઈએ અને આ કહે છે કે જો x true છે, તો z એ true હોવું જ જોઈએ, તેથી આ વધુ અથવા ઓછું કહે છે કે x, y અને z એ બધા સમાન હોવું જોઈએ. પરંતુ જો એક્સ, વાય

અને ઝેડ બધા સમાન હોય, તો પછી છેલ્લામાંના એકમાંથી એક ખોટું જતું રહ્યું હતું, ખરેખર ત્યાં કોઈ સંતોષકારક સોંપણી નથી. તેથી, અમારું લક્ષ્ય એ શોધવાનું છે કે ત્યાં સંતોષકારક સોંપણી છે કે નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:28)

તેથી, ઉકેલ લાવવા માટે, અલબત્ત બ્રુટ ફોર્સ (Brute force)અભિગમ એ x, y અને z દરેકને સોંપવાનો પ્રયાસ કરવાનો છે; સાચું અને ખોટું અને ચાલુ કરો, અને પછી એક વાર અમે એક્સ, વાય અને ઝેડને સાચા ઠરાવવા, ખોટા; અમે મૂલ્યાંકન કરી શકીએ છીએ અને તપાસ કરી શકીએ છીએ કે ફોર્મ્યુલા સાચું છે કે કેમ અને અમે આ શક્ય તેટલી બધી સોંપણી માટે આનો પ્રયાસ કરીએ છીએ. જો એન વેરીએબલ્સ હોય, તો તેમાંની દરેકમાં બે શક્યતાઓ હોય, સ્પષ્ટપણે ત્યાં 2 હોય છે, તેથી સંભવિત અસાઈનમેન્ટની ઘાતાંકીય સંખ્યા. હવે, આશ્ચર્યજનક વાત એ છે કે સામાન્ય રીતે આ સમસ્યા માટે કોઈ સારી અલ્ગોરિધમ જાણીતી નથી, ચળવળની જેમ વસ્તુઓ સ્થાયી થાય છે, આપણે આ ફોર્મ પર ફોર્મ્યુલા લેવાની કાર્યક્ષમ રીત જાણતા નથી અને તે શોધી કાઢે છે કે, તે સંતોષકારક સોંપણી ધરાવે છે કે નહીં. જો કે, તે તપાસવું સરળ છે કે તેમાં એક એલ્ગોરિધમ છે જે ઉકેલને ચકાસી શકે છે. તેથી, જો હું તમને ફોર્મ્યુલા આપું છું અને હું દાવો કરું છું કે આપેલ વેલ્યુએશન મૂલ્ય ખરેખર સંતોષકારક સોંપણી છે, મને ફક્ત તે જ કરવું પડશે, હું તે સોંપણીને પ્લગ કરું છું, જેમ કે અમે પહેલાના કેસ માટે કર્યું હતું. હું વિચારું છું

((સમયનોસંદર્ભ લો: 09:25))

વી x એ સાચું છે, મને સી એક્સ તરીકે દરેક જગ્યાએ સાચું દો, તમે કહી શકો કે વાય વાય ખોટું છે, હું સી વાય પર દરેક જગ્યાએ ખોટું મુકું, અને પછી સૂત્રનું મૂલ્યાંકન કરું, શોધવા શું એન્ડ્સ અને OR ની સાચી જવાબ છે કે નહીં. તેથી, તે જોવાનું સરળ છે કે તેમાં ચકાસણી એલ્ગોરિધમનો છે, પરંતુ તેમાં જનરેટિંગ એલ્ગોરિધમ નથી, ઓછામાં ઓછું આપણે પાથને જાણતા નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 09:45)

તેથી, આ કિસ્સામાં, કૃપા કરીને ચાલુ કરો અને અન્ય કેસ પણ છે, કેટલીક વખત તમારી સમસ્યાનું પ્રસ્તુતિ મહત્વપૂર્ણ છે. તેથી, અમે કહ્યું હતું કે એક કલમ શાબ્દિકનો એકરૂપ હતો, અને પછી સૂત્ર ક્લોઝનું એક જોડાણ હતું, તેનાથી શું બદલાવ થશે તે કહેશે, કલમો શાબ્દિક સાથે જોડાય છે. તેથી, એક વિભાગમાં, હું બધું સાથે કનેક્ટ કરું છું અને, અને પછી હું કલમોને સાથે જોડું છું અથવા, આનો અર્થ એ છે કે હું સી સાચું અથવા ડી સાચું અથવા ઈ સાચું બનાવું છું. પહેલાં, હું સી સાચું અને ડી સાચું અને ઈ સાચું કરું છું, હવે જો હું કલમની અંદર જોઉં છું, તો હું આ રીતે સાબિત કેવી રીતે કરી શકું છું, તમે લાલ જુઓ છો, હું આ પ્રત્યેક સત્ય બનાવું છું અને બીજું બધું હું સાચી બનાવું છું, આ સંપૂર્ણ અને નિષ્ફળ જશે. તેથી, જો હું કલમ લઈશ, તો કલમ ચાર સ્નીઝની અંદરના ચલોની સેટિંગ્સ, તે કહે છે કે x એ સાચું હોવું જોઈએ, y એ હોવું જ જોઈએ નહીં, y બરાબર હોવું જ જોઈએ, તેથી y ખોટું હોવું જોઈએ અને z એ true હોવું જોઈએ અને તેથી, તેથી મારી પાસે વધુ પસંદગી નથી. હવે, એક કલમની અંદર, હું y ને જોઈ શકું છું અને પછી ક્યાંક હું y જોઈ શકું છું. તો, તેથી, આવી વસ્તુ કહેશે કે મને સાચું અને y ખોટું કહેવા માટે કહેવામાં આવ્યું છે, તેથી આ કલમ સાચું ન હોઈ શકે, તો પછી હું આગળ વધું છું. તેથી, હું દરેક કલમ પર ધ્યાન આપું છું, હું ચકાસી શકું છું કે અજોડ વેલ્યુએશન એ શક્ય અથવા

સંભવિત છે, જો તે નલ હોય તો, કારણ કે મને ફક્ત એક જ કલમને સંતોષવાની જરૂર છે, બીજા માર્ગો પર જવા માટેના અન્ય માર્ગો. તેથી, ડાબેથી જમણે એક રેખીય સ્કેન માં, જો હું આ ફોર્મમાં આપું છું, તો મૂળભૂત રીતે હું આ સમસ્યાને હલ કરી શકું છું, જ્યારે પહેલાના સ્વરૂપમાં મને કોઈ કાર્યક્ષમ અલ્ગોરિધમ નથી. તેથી, પ્રસ્તુતિ સાથે સાફ થયેલ સમસ્યા સમસ્યા છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 11:13)

તેથી, ચાલો એક સંપૂર્ણપણે અલગ સમસ્યાની તપાસ કરીએ. તેથી, અમારી પાસે આ જાણીતી મુસાફરી કરનાર સેલ્સમેન સમસ્યા છે. તેથી, સેલ્સમેનને શહેરોના નેટવર્કની મુલાકાત લેવાની અને શહેરોના દરેક જોડી વચ્ચે જવાની ધારણા છે, અમારી પાસે એક અંતર છે. તેથી, આપણે વિચારી શકીએ છીએ કે આ એક સંપૂર્ણ ગ્રાફ છે, દરેક શહેર સાથે જોડાયેલા દરેક શહેર અને બે શહેરો વચ્ચેની દરેક ધાર, ત્યાં એક કારણ છે જે કોઈ કારણ અથવા સમય અથવા અમુક જથ્થાના અંતરને સૂચવે છે.એક શહેરથી બીજા શહેરમાં મુસાફરી કરવા માટે વેચાણ માણસે ઉપયોગ કરવો પડશે. તેથી, આ નકશા પર દરેક શહેરની મુલાકાત લેવાનું સેલ્સમેનનું લક્ષ્ય છે. તેથી, સેલ્સમેન ટૂંકા ટૂર શોધવા માંગે છે જે પ્રત્યેક શહેરને એક જ વાર મુલાકાત લે છે. પછી, ગ્રાફ થિયેટ્રિક્સનો અર્થ એ છે કે તેનો અર્થ એ છે કે તે સરળ ચક્ર હતો, સાદા ચક્રનો અર્થ એ થયો કે કોઈ નોડ પુનરાવર્તિત થતો નથી, હું આ જ શહેરની શરૂઆત અને અંત સાથે બે વાર સમાન વર્ટેક્સની મુલાકાત લેતો નથી, તેથી જ એક ચક્ર શરૂ થાય છે અને તે જ શહેરના અંત, સમાન શિરચ્છેદ, દરેક શિર્ષકની વચ્ચે અને લઘુત્તમ ખર્ચની મુલાકાત લો.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 12:19)

તેથી, એક સેકંડ, જનરેટિવ એલ્ગોરિધમનો લખવાનો કોઈ સરળ રસ્તો નથી જે વાસ્તવમાં વિશ્લેષણ કરશે અને સારો ઉકેલ શોધશે. તેથી, હવે, અમારો પ્રશ્ન એ છે કે ત્યાં એક ચકાસણી સોલ્યુશન છે, ત્યાં એક ચકાસણી એલ્ગોરિધમ્સ છે. તેથી, યાદ રાખો કે, ચેકિંગ એલ્ગોરિધમ શું કરે છે તે ઈનપુટ તરીકે લે છે, તે ઈનપુટ ઉદાહરણ લે છે અને તે એક ઉકેલ લે છે, અને પછી તે હા અથવા ના કહે છે, આ ઉકેલ કામ કરે છે આ ઉકેલ કામ કરતું નથી. તેથી, હવે, અમારી પાસે ગ્રાફ છે અને કોઈક આપણને એક ચક્ર આપે છે, અમે તે ચક્ર છે તે ચકાસી શકીએ છીએ. તેથી, તે કણો, આપણે ચક્રની કિંમત પણ ગણતરી કરી શકીએ છીએ જે પણ સરળ છે, પરંતુ આપણે કેવી રીતે જાણી શકીએ કે બધા તફાવત ચક્ર વચ્ચે, આ ઓછામાં ઓછી કિંમત છે. તેથી, એ સ્પષ્ટ નથી કે ત્યાં અલ્ગોરિધમનો તપાસ થઈ રહ્યો છે, કારણ કે અંતે, આપણે આ ઉકેલના ભાગને ચકાસી શકીએ છીએ, તે એક ચક્ર છે, આ બધા શહેરો છે અને આપણે જાણીએ છીએ કે તે ખર્ચ છે, અમારી પાસે કોઈ રસ્તો નથી સમસ્યા હલ કર્યા વગર. એલ્ગોરિધમનો તપાસ કરતાં ધ્યેયને યાદ રાખવું એ સમસ્યાનું સમાધાન નથી, તે આપેલ છે કે આ આપેલા ઉકેલ માટે આ આપવામાં આવેલું સોલ્યુશન સાચું છે કે નહીં. તેથી, સમસ્યાને કેવી રીતે ઉકેલવી તે આપણે જાણી શકીએ નહીં, જેમ કે શિક્ષક કે જેમણે પરિબળનું ઘરકામ સોંપ્યું છે, શિક્ષકને કેવી રીતે પરિબળિત કરવું તે જરૂરી નથી, આ પરિબળોને જાણવાની પણ જરૂર નથી, શિક્ષકને માત્ર જાણવાની જરૂર છે, કેવી રીતે સંભવિત પરિબળોમાં ગુણાકાર કરવા અને તપાસ કરવી કે જવાબ મોટી સંખ્યા જેટલો જ છે. એ જ રીતે, અહીં આપણી તપાસ કરવાની જરૂર છે, આપેલ નોડ્સ ચક્ર બનાવે છે કે નહીં, પરંતુ જ્યાં સુધી આપણે સમસ્યાને કેવી રીતે ઉકેલવું તે જાણતા નથી, ત્યાં સુધી ચેકિંગ એલ્ગોરિધમ ઓછામાં ઓછું ખર્ચ ચક્ર છે કે નહીં તે જાણી શકતું નથી.

(સ્લાઈડસમયનો સંદર્ભ લો: 13:52)

તેથી, આપણે એક રાઉન્ડ કેવી રીતે મેળવી શકીએ? તેથી, સોલ્યુશન એ આવી ઑપ્ટિમાઈઝેશન સમસ્યાઓ છે જે તેમને ચકાસણી અલ્ગોરિધમ્સમાં કન્વર્ટ કરવા માટે છે જેથી સમસ્યાને બંધબેસશે, જેથી હું બાહ્ય બાઉન્ડ અથવા નીચલા બાઉન્ડને બહારના આધારે આપી શકું. તેથી, આ કિસ્સામાં, અમે પૂછીશું કે પ્રવાસ ક્યાં છે કે કેમ તે પૂછશે નહીં, જો ત્યાં મુસાફરીના વેચાણના પુરુષોની સૌથી ઓછી કિંમતની મુસાફરી હોય તો, અમે કહીએ છીએ કે મોટા ભાગની કિંમતના ખર્ચ સાથે પ્રવાસ છે. તેથી, અમે ફક્ત કિંમત પર બંધાયેલા છીએ, અમે શ્રેષ્ઠ પ્રવાસ માટે પૂછતા નથી, અમે માત્ર પ્રવાસ માટે જ પૂછતા સાથે કે કે કરતાં વધુ કિંમત નથી. હવે, અમે એક ઉકેલ આપ્યો છે, કારણ કે આપણે તેને ચકાસી શકીએ છીએ. આપણે તપાસ કરી શકીએ કે તે એક ચક્ર છે, અને પછી આપણે બધા કિનારીઓ ઉમેરી શકીએ છીએ જે પ્રવાસનો ભાગ બને છે અને તે શોધી કાઢે છે કે, તે કે કે એસને અજાણ કરે છે, તો પછી અમને કેવી રીતે મદદ કરવી, કારણ કે અમારો ધ્યેય હતો ટૂંકા પ્રવાસોને શોધો, હવે આપણે જે કહ્યું છે તે છે કે જો હું તમને અપર બાઉન્ડ કદની ટૂર આપીશ, તો હું હા અથવા ના માટે ચકાસણી એલ્ગોરિધમનો ઉપયોગ કરી શકું છું, પરંતુ હવે અમે અલગ કેસ અજમાવી શકીએ છીએ. તેથી, આપણી પાસે ન્યૂનતમ મૂલ્ય K ની સ્પષ્ટ રૂપે 0 છે અને મહત્તમ મૂલ્ય અમુક ઉચ્ચ બાઉન્ડ છે. સારી ઉપલા બાઉન્ડ શું છે? ઠીક છે, આપણે જાણીએ છીએ કે એક ટૂર ટોચની કિનારીઓ લેશે, જો હું ગ્રાફમાં બધી ધાર લઈશ અને બધી કિંમત ઉમેરીશ, તો કુલ પ્રવાસ તે કરતા વધુ ન હોઈ શકે. તેથી, ગ્રાફમાં બધી ધાર વજનની રકમ કહેવા માટે 0 થી ટૂરના ખર્ચ માટે મારી પાસે મૂલ્યની શ્રેણી શક્ય છે. તમે તેના કરતાં વધુ સારી રીતે બંધાયેલા પણ હોઈ શકો છો, કારણ કે દેખીતી રીતે, આપણે બધા ધાર વજનનો ઉપયોગ કરી શકતા નથી, અમે કદ n ના ચક્રને પૂર્ણ કરવા માટે ફક્ત એનનો ઉપયોગ કરી શકીએ છીએ. પરંતુ, આ ખૂબ સંરક્ષિત છે, હવે આપણે શું કરીએ છીએ, આપણે આ શ્રેણીમાં કરીએ છીએ, આપણે બાઈનરી શોધ કરીએ છીએ. તેથી, અમે સૌ પ્રથમ તપાસ કરીએ છીએ કે ત્યાં કોની કિંમત મધ્ય માર્ગ છે, જો ત્યાં હોય તો હું હવે નીચે તપાસ કરીશ કે તેમાં અડધો પ્રવાસ ખર્ચ છે. તેથી, દ્વિસંગી શોધ કરીને, અમે વસ્તુને સાંકડી કરી શકીએ છીએ અને છેલ્લે, તે શોધીશું કે આ એક સ્તર છે, જ્યાં અમારે પ્રવાસ છે અને તે વિશે, અમારી પાસે પ્રવાસ છે અને નીચે છે, અમારી પાસે પ્રવાસ નથી. તેથી, આ લઘુત્તમ ખર્ચ પ્રવાસ છે. તેથી, દ્વિસંગી શોધ અને ઉપલા બાઉન્ડનો ઉપયોગ કરીને, અમે ઑપ્ટિમાઈઝેશન સમસ્યાને લઈ શકીએ છીએ જે અમે એક શ્રેષ્ઠ જવાબ શોધી રહ્યા છે અને તેને સમસ્યાઓના નિરીક્ષણના ક્રમમાં રૂપાંતરિત કરી શકીએ છીએ, જે સીમાની આ જગ્યા દ્વારા લોગરિધર્મિક શોધ પછી મને વાસ્તવિક ઉકેલ આપશે અથવા નહીં.

(સ્લાઈડસમયનો સંદર્ભ લો: 16:05)

તો અહીં બીજી સમસ્યા છે. તેથી, આને સ્વતંત્ર સેટ બાર કહેવામાં આવે છે. તેથી, અમે કહ્યું કે બે શિરોબિંદુઓ સ્વતંત્ર છે; જો તેઓ તેની સાથે જોડાયેલ ન હોય તો તે ધાર છે. તેથી, ઉદાહરણ તરીકે, અહીં તમે 1 અને 7 કહેવું સારું લાગે છે, પછી ત્યાં કોઈ જોડાણ સાથે સ્વતંત્ર છે. તેથી, 6 અને 5 સ્વતંત્ર હોઈ શકે છે, તેથી આપણે કહીએ છીએ કે જો કોઈ ધાર હોય તો બે શિરોબિંદુઓ સ્વતંત્ર છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 16:32)

હવે, જો હું આ સેટને 1 અને 7 લઈ લઈશ, તો તે એક સ્વતંત્ર છે, પરંતુ જો હું તેમાં 5 ઉમેરીશ તો ઉદાહરણ તરીકે, હવે 5 અને 7 ધાર સાથે જોડાયેલા છે, તેથી આ સ્વતંત્ર નથી. તેથી, એક સ્વતંત્ર સમૂહ એ છે જેમાં દરેક જોડી સ્વતંત્ર હોય છે. તેથી, 1 અને 7 સ્વતંત્ર સમૂહ છે, પરંતુ 1, 5, 7 સ્વતંત્ર સમૂહ નથી, કારણ કે 5, 7 એક તટસ્થ છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:55)

તેથી, અહીં ઉદાહરણ તરીકે, તમે સ્વતંત્ર સેટમાં 3, 4, 5 સ્વરૂપો જોઈ શકો છો, કારણ કે 3 અને 4 ની વચ્ચે નથી, 4 અને 5 વચ્ચે કોઈ ધાર નથી, ત્યાં છે 3 અને તેથી કોઈ વચ્ચે કોઈ ધાર નથી, ઉદાહરણ તરીકે, જો તમે કહેતા પ્રયાસ કરો છો કે તમે આ નોડ્સને લોકો તરીકે અર્થઘટન કરો છો અને તમે એકબીજાને જાણીને ધારને નોડ કરો છો. તેથી, હવે તમે કલ્પના કરો છો કે તમે કોઈ પ્રકારની સમિતિની રચના કરવા માંગો છો, જ્યાં તમે હંમેશાં રહેવા માંગો છો કે બધી સમિતિના સભ્યોની સ્વતંત્ર અભિપ્રાય છે તે હકીકત દ્વારા પ્રભાવિત થતી હકીકતોથી પ્રભાવિત થતી નથી. પછી તમે એક સ્વતંત્ર સમૂહ પસંદ કરી શકો છો, એક સ્વતંત્ર સેટ પેદા લોકો, જે એક બીજાને પારખતા નથી. તેથી, તેથી, જ્યારે તેઓ પહેલી વખત મળે છે, તો આશા રાખીએ કે, જો તેઓ એકબીજા વિશે અને સમસ્યાની બાબતે તટસ્થ મંતવ્યો ધરાવતા હોય તો ... તેથી, આ એક ઉદાહરણ હોઈ શકે છે કે તમે સ્વતંત્રતા સમૂહ કેમ પસંદ કરો છો. તેથી, મોટાભાગના સ્વતંત્રતા સમૂહ કરતાં એલ્ગોરિધમિક સમસ્યા શોધવાનું છે. તેથી, અહીં આપણે કદ 3 ની સ્વતંત્રતાના સમૂહમાં શોધી કાઢ્યું છે, હું વધુ સારી રીતે કરી શકું છું અને મને કદ 4 માંનું એક, કદાચ, કદાચ નહીં મળે. તેથી, આ મારી અલ્ગોરિધમિક સમસ્યા છે, આલેખને સૌથી મોટો સ્વતંત્રતા સમૂહ શું છે તે હું શોધી શકું છું. જેમ આપણે પહેલા જોયું તેમ, આ એક સમસ્યા છે જ્યાં અમને જવાબ મળવો પડશે. તેથી, જો કોઈ મને કહે છે કે 3, 4, 5 એ મહત્તમ સ્વતંત્રતા સેટ છે, તો હું સ્વતંત્રતા 3, 4, 5 કારણોસર સરળતાથી ચકાસી શકું છું, પરંતુ જો કોઈ મોટો સેટ ન હોય તો હું તેને ચકાસી શકતો નથી. આ મુસાફરીના કોશિકાઓની જેમ એક સમસ્યા છે, જ્યારે હું ચકાસી શકું છું કે મને આપવામાં આવેલો પ્રવાસ એ એક સરળ ચક્ર છે, પરંતુ હું એ ચકાસવામાં સમર્થ ન હોઈ શકું કે આ પ્રવાસનો ખર્ચ આમાં શ્રેષ્ઠ છે. તો, ફરી એકવાર, જો તમે સમસ્યાનું પરીક્ષણ કરતી આવૃત્તિને સેટ કરવા માંગો છો, તો અમે સેટ કરીશું, કહેશે કે કદના સ્વતંત્ર સમૂહ છે. તેથી, અમે સૌથી મોટો એક શોધવાનો પ્રયાસ કરી રહ્યા છીએ, તેથી તમે કહો ઓછામાં ઓછું કદ કે. તેથી, જો હું ઓછામાં ઓછું કદ 3 અને આનો ઉપજ, તો હું ઓછામાં ઓછું કદ 3 ચકાસો. અને ઓછામાં ઓછું કદ 4 અને આ મારો સાક્ષી છે અને આ મારો ઉકેલ છે, હું કહું છું. તેથી, મુસાફરી કરનારા સેલ્સમેન અને બુલિયન(Boolean) જેવી ક્ષમતા ફરીથી સેટ કરવાની સ્વતંત્રતા, અમે એક સારા જનરેટિવ ઉકેલ વિશે જાણતા નથી. પરંતુ આ બાઉન્ડ્રી સંસ્કરણનો ઉપયોગ પ્રદાતાના નંબર જથ્થા દ્વારા કરવામાં આવ્યો હતો, અમે અનુમાન કરવાનો પ્રયાસ કરી રહ્યા છીએ કે અમે ચકાસણી સંસ્કરણો બનાવી શકીએ છીએ.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 19:04)

તેથી, સંબંધિત જોઈતી સમસ્યા, આને સંબંધિત સમસ્યાને વર્ટેક્સ કવર કહેવામાં આવે છે, અમે કહીએ છીએ કે એક નોડ જે તમે દરેક ધારને આવરી લે છે. તેથી, ઉદાહરણ તરીકે, જો હું આ નોડ લઈશ, તો તે આ ત્રણ, આ ચાર ધારને આવરી લેશે, કારણ કે આ બધી ધાર શરૂ થઈ હતી. તેથી, હવે, કર્ણક આવરણ કંઈક છે જે બધી ધારને આવરી લે છે. તેથી, ઉદાહરણ તરીકે, તે ધારને 2 આવરી લે છે, હવે આ ધાર ત્યાં નથી, કદાચ હું તેને આવરી લેવા માટે 3 લઈ શકું. તેથી, હું આ કવરમાં 3 લઈ શકું છું, આ ધારમાં હજી પણ કેટલાક ખૂટે ધાર છે, તેથી હું 7 પસંદ કરી શકું છું, જે 7 આ ત્રણ ધારને આવરી લે છે.

તેથી, મને 1, 2 3 શિરોલંબનો સંગ્રહ મળ્યો છે જે આ તારાના બધા ધારને આવરી લે છે. તેથી, હવે, આપણે શું કરવા માંગીએ છીએ તે આપેલ ગ્રાફમાં આ સૌથી નાના કર્ણને આવરી લે છે, કેમ કે આપણે તે જાણતા નથી કે તે કેવી રીતે સૌથી નાનું છે, આપણે કહીશું કે, આપણે કોઈપણ કક્ષાનો કવર લઈશું જે સૌથી વધુ કહે છે કે, જો દૂર સૌથી નાનો હોય. તેથી, તે મુસાફરી કરતા સેલ્સમેન જેવા છે, અમે ટૂંકા ટૂરમાં જઈશું, તેથી તે કહે છે કે મારા ખર્ચની કિંમત કરતાં ઓછા અથવા બરાબર એક પ્રવાસ છે, તેથી ત્યાં ખર્ચમાં સરખુ જ મૂલ્ય ઓછું હોય છે. તેથી, આ બે સમસ્યાઓ જોડાયેલ દેખાય છે અને પાછળથી તે છે. તેથી, અહીં ઉદાહરણ તરીકે, આપણી પાસે કદ 4 નું શિરોબિંદુ છે, જે 3 માં સૂચવેલું છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 20:40)

તેથી, જો કનેક્શન એ યુ કદ કદના સ્વતંત્ર સેટ હોય તો, જો તે માત્ર પ્રશંસા હોય તો તે કદ N માઈનસ કે. નું વર્ટેક્સ કવર છે. તેથી, આપણા કિસ્સામાં યુ 3, 4, 5 કદ 3 ની એક કક્ષાની સ્વતંત્રતા હતી, જ્યારે તેની પ્રશંસા કરવામાં આવે ત્યારે તેનું માપ N માઈનસ K ના કદનું આવરણ હોવું આવશ્યક છે. તેથી, જો હું હલ કરી શકું ચોક્કસ કદ માટે વર્ટેક્સ કવર સમસ્યા, હું એન માઈનસ કે. માટે સ્વતંત્રતા સેટ સમસ્યાને હલ કરી શકું છું. તેથી, હું એકને બીજામાં રૂપાંતરિત કરી શકું છું અને આનો પુરાવો ખૂબ જ સરળ છે. તેથી, મને લાગે છે કે સ્વતંત્ર સેટનો ઉપયોગ કરીને, સ્વતંત્ર સમૂહમાં ઉપયોગ કરો, પછી ફક્ત યુ સાથે નહીં. તેથી, કોઈપણ ધાર યુવી લઈને, પછી જો કોઈ યુની અંદર હોય, તો બીજો યુ બહારની અંદર અથવા બંને N પોઈન્ટ બહાર હોય. તો, તેથી, જો હું તેમાં ઓછામાં ઓછું એક ધાર લઉં તો અંત પોઈન્ટ વી માઈનસ યુ માં રહે છે. તેથી, V ઓછા યુ બધા ધારને આવરી લે છે, તેથી, V ઓછા યુ કેટલાક શિરોબિંદુઓ. અને અલબત્ત, આને K શિરોબિંદુ તરીકે આપવામાં આવે છે, તો બીજામાં એન માઈનસ કે શિરોલંબ હોવું જોઈએ, તેનાથી વિપરીત હું ધારું છું કે વી માઈનસ યુ એ વર્ટેક્સ કવર છે, પછી દરેક ધાર ત્યાંથી શરૂ થાય છે. જો દરેક ધાર ત્યાંથી શરૂ થાય છે, તો તેમાંનો એક છે અંત પોઈન્ટ છે. તેથી, મારી પાસે અંત નથી જે પૂર્ણપણે પૂરકની અંદર છે, કારણ કે જો તેની પાસે ધાર હોય તો, તે ધાર છે જે શામેલ કવર દ્વારા ઢંકાયેલું નથી. તેથી, તેઓ યુમાંની સાથે કોઈ ધાર હોઈ શકતા નથી, તેથી, યુ એ સ્વાતંત્ર્ય સમૂહ છે. અને ફરી એકવાર, કદની ખાતરી આપવામાં આવે છે, કારણ કે તેમની પૂરક શિરોબિંદુ શોધે છે. તેથી, આ કહે છે કે સ્વતંત્રતા સમૂહ પૂરક કદ સાથે શિરોબિંદુઓને આવરી લે છે, આ બંધાયેલા શિરચ્છેદ સંસ્કરણ અને શ્વેત કવર સ્વતંત્રતા સમૂહમાં ઘટાડે છે. તેથી, એકબીજાને ઘટાડે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 22:25)

તેથી, જ્યારે આપણે લાઈનર પ્રોગ્રામિંગ અને નેટવર્ક ફ્લોરની સ્પર્ધામાં ઘટાડાની રજૂઆત કરીએ છીએ ત્યારે અમે કહ્યું હતું કે એક હેતુ બી થી એમાં કાર્યક્ષમ ઉકેલ સ્થાનાંતરિત કરવાનો છે તેથી, જ્યારે હું A થી B ઘટાડે, B એક કાર્યક્ષમ છે, તો એ કાર્યક્ષમ છે. પરંતુ આ હરીફાઈમાં, આપણે જે કહેવાનો પ્રયાસ કરી રહ્યા છીએ તે છે કે આ એક કાર્યક્ષમ હોવાનું જાણીતું નથી, તો તે પણ કાર્યક્ષમ હોવાનું જાણીતું નથી. તેથી, સ્વતંત્ર રીતે આપણે માનીએ છીએ કે તે સ્વતંત્ર છે તે પછી કાર્યક્ષમ નથી, કારણ કે શંકુનું આવરણ ઘટાડે છે, વર્ટેક્સ કવર એક કાર્યક્ષમ અને યોગ્ય ગ્રોતો હોવાનું પણ સંભવ છે. તેથી, તે શરતો છે કે ચેકલિબલ સમસ્યાઓના ઘણા જોડીઓ અથવા આવી અંતર્ગત ઘટાડેલી સમસ્યા, સીધી જ તમને કોઈ ચક્ર અથવા વસ્તુઓ મળી શકે છે, બીને ઘટાડે છે, બીને ઘટાડે છે અને સીને પાછળથી ઘટાડે છે. તેથી, તે અર્થમાં, તે બધા સમાન સમાન અથવા સમાનરૂપે સખત હોય છે અને ત્યારથી, આપણે માનતા હોઈએ છીએ કે કોઈએ હજી સુધી કોઈપણને હલ

કરી લીધા નથી, જો તેમને સખત મહેનત કરવી જોઈએ, તો આ માનવામાં આવે છે કે સમસ્યાઓના મોટા જૂથ છે જે વાસ્તવમાં તેમની ઉપયોગીતાની દ્રષ્ટિએ વ્યવહારુ છે. પરંતુ તેમને અસરકારક રીતે ઉકેલી સૂચના તરીકે ઓળખવામાં આવતી નથી અને તેમની દ્રષ્ટિએ અલ્ગોરિધમિક કાર્યક્ષમતાના બધા વિચારો. તેથી, આપણે આ છેલ્લી ભાષણમાં આ થોડું વધારે વિગતવાર જોશું.

ડિઝાઈન અને એનાલિસિસ ઓફ એલ્ગોરિધમ્સ (Design and Analysis of Algorithms)
પ્રોફેસર માધવન મુકુંદ (Prof. Madhavan Mukund)
ચેન્નઈ મેથેમેટિકલ ઈન્સ્ટિટ્યુટ (Chennai Mathematical Institute)
વીક - 08
મોડ્યુલ - 07
લેક્ચર - 56
અવ્યવહારક્ષમતા: પી અને એનપી

આ અભ્યાસક્રમના છેલ્લા ભાષણમાં, આપણે પી અને એનપી વિશેના આ પ્રખ્યાત પ્રશ્નને જોઈશું, જેનો ઉલ્લેખ ઘણી વાર કરવામાં આવે છે. કમ્પ્યુટર સાયન્સની છેલ્લી અને સૌથી મહત્વપૂર્ણ ખુલ્લી સમસ્યાઓ પૈકીની એક તરીકે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:14)

તેથી, આપણે જોયું છે કે ચકાસણી એલ્ગોરિધમ કંઈક છે જે આપેલ સમસ્યાઓ ઉકેલ માન્ય છે કે નહીં તે ચકાસવા માટે અમને પરવાનગી આપે છે. તેથી, એક ચકાસણી એલ્ગોરિધમ સમસ્યાને જલ્દીથી હલ કરી શકતું નથી, પરંતુ તે શું કરી શકે છે તે સમસ્યાનો ઈનપુટ લે છે, તે સમસ્યાના સંભવિત ઉકેલને અને તે સોલ્યુશન ખરેખર છે કે કેમ તે માન્ય કરે છે. તેથી, સમસ્યાને હલ કર્યા વિના, તે ફક્ત ત્યારે જ ચકાસી શકે છે કે કોઈ સોલ્યુશન એક માન્ય ઉકેલ છે કે નહીં.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 00:44)

તેથી, ક્લાસ એનપી એ એવી સમસ્યાઓનો વર્ગ છે કે જેના માટે ચેકિંગ એલ્ગોરિધમ વધારાની મર્યાદા સાથે અસ્તિત્વ ધરાવે છે કે ચેક ઈનપુટના કદમાં પોલિનોમિયલ ટાઈમમાં ચાલે છે. તેથી, તપાસ યોગ્ય હોવી જોઈએ, તેથી બધી સમસ્યાઓમાં, આપણે છેલ્લા ભાષણમાં જોયું છે, આ હકીકતમાં સાચું છે. ફેક્ટોરાઈઝેશન એ એનપી છે, કારણ કે બે પરિબળો આપ્યા છે, જો આપણે ઈચ્છો અને તપાસ કરીશું તો જવાબ મેળવવા માટે આપણે તેમને અસરકારક રીતે ગુણાકાર કરી શકીએ છીએ. સંતોષકારકતા, ફરી, કારણ કે આપણે માત્ર સચોટ અસાઈનમેન્ટ લેવાનું છે, ફોર્મ્યુલા ચલોને ખોટા છે અને ચકાસો છે કે પછી ફોર્મ્યુલા તેના પછી સાચું મૂલ્યાંકન કરે છે. ફરીથી, સીમિતમેનને સીમિત વર્ઝન સાથે મુસાફરી કરીને, અમે આપેલ પાથનો સેટ લઈ શકીએ છીએ, તપાસો તે એક સરળ ચક્ર છે, વજન વધારવાનું અને બાઉન્ડ્રીને સંતોષવું, તેવી જ રીતે વર્ટેક્સ કવર અને સ્વતંત્ર સમૂહની ચકાસણી કરવી. તેથી, આ બધા એનપી છે, કેમ કે ઈનપુટ કદના સંદર્ભમાં ઉકેલને કાર્યક્ષમ રીતે માન્ય કરી શકાય છે. એક વસ્તુ ધ્યાનમાં લેવાની વાત છે કે મુસાફરી કરનાર સેલ્સમેન, વર્ટેક્સ કવર અને સ્વતંત્ર સેટ જેવી સમસ્યાઓમાં, અમે ઑપ્ટિમાઈઝેશન સમસ્યાને લીધે અને તેને બાઉન્ડ્રી તપાસ સમસ્યામાં રૂપાંતરિત કરી. પરંતુ અમે એમ પણ કહીએ છીએ કે વાસ્તવિક જવાબ શોધવા માટે બાઈનરી શોધનો ઉપયોગ કરીને બાઉન્ડનો ઉપયોગ કરી શકાય છે, તેથી આ દ્વિસંગી શોધમાં લઘુગણક પરિબળ છે. તેથી, કોઈ અર્થમાં એકંદરે સામાન્યતાનો કોઈ નુકસાન નથી, કારણ કે તમે બહુવચન સમય પગલું લે છે, અને પછી તમે લઘુગણક વસ્તુ દ્વારા ગુણાકાર કરો છો, તેથી બધું હજી પણ સમાન છે. તેથી, શુદ્ધ ચકાસણી સમસ્યાઓ જેવી કે સંતોષ અને સુધારેલી ચેકિંગ સમસ્યાઓ વચ્ચે કોઈ તફાવત નથી, જેમાં બાઉન્ડ, મુસાફરી કરનાર સેલ્સમેન અથવા વર્ટેક્સ કવરનો સમાવેશ થાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 02:24)

તો, શા માટે આ વર્ગને એનપી કહેવામાં આવે છે, તેથી એનપી બિન-નિર્ણાયક પોલિનોમિયલ સમયથી આવે છે, તેથી આ એક ઐતિહાસિક સંદર્ભ છે. તેથી, તે હકીકતનો ઉલ્લેખ કરે છે કે આ સમસ્યાઓ સાથે અનુરૂપ છે જ્યાં તમે ઉકેલને અનુમાનપૂર્વક અનુમાન કરી શકતા નથી, તમે જાદુઈ રીતે સોલ્યુશન ઉત્પન્ન કરી શકો છો અને પછી પોલિનોમિયલ સમયમાં, તમે ચકાસી શકો કે સોલ્યુશન સાચું છે કે નહીં. તેથી, આ શબ્દનો વાસ્તવિક મૂળ ગણતરીત્મક સિદ્ધાંતથી આવ્યો છે, બિન-નિર્ધારિત ટ્યુરિંગ મશીનોના સિદ્ધાંતથી. તેથી, જો આપણે તેનો અભ્યાસ કર્યો છે, તો તમે જાણી શકશો, તે એન.પી. કેમ કહેવાય છે, પરંતુ જો તમે તેનો અભ્યાસ ન કર્યો હોય, તો એ સમજવું એટલું મહત્વપૂર્ણ નથી કે, એલ્ગોરિધમ્સ માટે તે શું સૂચવે છે. તેથી, એનપી બિન-નિર્ણાયક પોલિનોમિયલ સમયથી આવે છે, જે શબ્દોમાં ટ્યુરિંગ મશીનોથી આવે છે.

(સ્લાઈડસમયનો સંદર્ભ લો: 03:13)

તેથી, બે વર્ગો જે લોકો પી અને એનપી વિશે વાત કરે છે. તેથી, આપણે હમણાં જ એન.પી. શું જોયું છે, તો પી શું છે? તેથી, પી એ માત્ર એવી સમસ્યાઓનો વર્ગ છે જે આપણે આ કોર્સમાં અન્વેષણ કરવાનો પ્રયાસ કરી રહ્યા છીએ. આ તે સમસ્યાઓ છે જેના માટે તમારી પાસે ખરાબ કેસ એલ્ગોરિધમનો છે જે બહુભાષી સમયમાં ચાલે છે. તેથી, યાદ રાખો કે આપણે શરૂઆતમાં જ કહ્યું હતું કે અમે એન સ્ક્વેર્ડ ફોર્મની સમસ્યાઓ જોઈ રહ્યા છીએ, એન પછી તે બધા, તે પછી આપણે સંભવિત વસ્તુઓ અથવા એન લોગ એન જેવી બાબતોમાં પણ ઓછું જોવું જોઈએ, પરંતુ આપણે ચોક્કસપણે એન અને એન ફેક્ટોરિયલ બનવું નથી. તેથી, પી સૌથી ખરાબ કિસ્સામાં નિયમિત પોલિનોમિયલ ટાઈમ જટિલતા સાથે બધી સમસ્યાઓનો વર્ગ છે. હવે, તમામ વ્યાખ્યા દ્વારા, પીમાંની દરેક વસ્તુ પણ એનપીમાં છે, કારણ કે તેને તપાસવા માટેનું એક ઉકેલ મેળવવાની જરૂર નથી, કારણ કે તે પીમાં છે, આપણે ખરેખર સ્વયંને ઉકેલ બનાવી શકીએ છીએ. તેથી, એલ્ગોરિધમ તપાસવું કોઈકને અમને ઇનપુટ આપવા અને કોઈ ઉકેલ પ્રદાન કરવા પૂછે છે અને પછી તેને માન્ય કરે છે. પીમાં સમસ્યા, આપણે ખરેખર ઉકેલો પેદા કરી શકીએ છીએ, તેથી અમે ફરીથી માન્ય કરી શકીએ છીએ અને તેથી, પીમાંની દરેક સમસ્યા ખરેખર એનપીમાં છે. તેથી, લોકો જે પ્રશ્ન જાણવા માંગે છે તે છે કે પી એ એનપી સમાન છે કે કેમ. તેથી, તે એક વિપરીત જૂથ છે, તે સંભવ છે કે જ્યારે પણ મારી પાસે કાર્યક્ષમ ચકાસણી એલ્ગોરિધમ હોય, ત્યારે મારી પાસે સોલ્યુશન જનરેટ કરવા માટે કાર્યક્ષમ અલ્ગોરિધમનો પણ સમાવેશ થાય છે. તેથી, આત્મવિશ્વાસથી આ બાબત જોવા મળતી નથી, કારણ કે આપણે જોયું છે કે શિક્ષક જે પરિબળનું હોમવર્ક તપાસે છે, તેને કેવી રીતે પરિબળ બનાવવું તે પણ જાણવાની જરૂર નથી, શિક્ષકને માત્ર જાણવાની જરૂર છે, ગુણાકાર કેવી રીતે કરવો, તેથી તે વધુ સરળ રીતે સરળ લાગે છે ઉકેલ ચકાસવા માટે, પછી ખરેખર એક બનાવવું. તેથી, ચાલો જોઈએ કે આ સૂચના ઔપચારિક બનાવી શકાય છે કે નહીં.

(સ્લાઈડસમયનો સંદર્ભ લો: 04:51)

તેથી, લોકો કેમ માને છે કે પી એનપી સમાન નથી? ઠીક છે, કારણોમાંની એક માત્ર અનુભવથી છે અથવા તે અનુભૂતિશીલ છે. તેથી, અમે ઘણી કુદરતી સમસ્યાઓ, પરિબળ, સંતોષ, મુસાફરી કરનાર સેલ્સમેન, વર્ટેક્સ કવર, સ્વતંત્ર સેટ જોયું છે, આ બધા એનપીમાં છે. આપણે જોયું છે કે કંકણ આવરણ અને સ્વતંત્ર સમૂહ એકબીજાને ઘટાડી શકાય છે. આજે, આપણે

જોશું કે કેટલીક અન્ય એનપી સમસ્યાઓ પણ એકબીજામાં ઘટાડી શકાય છે. તેથી, હકીકતમાં, તમે શોધી શકો છો કે આ બધી સમસ્યાઓ વાસ્તવમાં અંતર ઘટાડે છે, આનો અર્થ એ થાય કે અસરકારક રીતે સમાન સ્તરનું સ્તર છે. કારણ કે, જ્યારે વસ્તુઓ ઓછી થાય છે, એક અને બીજા વચ્ચેનો, એક તો તમે કાર્યક્ષમ ઉકેલો પરિવહન કરી શકો છો અથવા તમે અયોગ્ય લોકોનો દાવો કરી શકો છો. પરંતુ અસરકારક રીતે ત્યાં બધા જ પ્રકારના, તમે એકને હલ કરી શકો છો, તમે લગભગ એક જ સમયમાં અન્યને હલ કરી શકો છો. તેથી, જે પણ લે છે તેમાંથી કોઈપણ આને હલ કરે છે અને આ બધી સમસ્યાઓ પછી એક કાર્યક્ષમ ઉકેલ હશે. પરંતુ ઘણા લોકો આ સમસ્યાઓનો ઘણા વર્ષોથી જુદા જુદા ખૂણાઓથી અજમાવી રહ્યા છે, હવે અને નૌકાઓએ એક શોધી કાઢ્યું છે. તેથી, ત્યાં સારા પ્રાયોગિક પુરાવા છે કે આ માટે કોઈ સારું ઉપાય નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 05:56)

તેથી, ચાલો આ આંતર-ઘટાડવાની ક્ષમતાને વધુ નજીકથી જોઈએ. તેથી, અમે બુલિયન સંતોષ સાથે પ્રારંભ કરીએ છીએ, યાદ છે કે બુલિયન સંતોષમાં, અમારી પાસે કલમો છે, કલમો શાબ્દિકના જોડાણ દ્વારા કંપોઝ કરવામાં આવે છે, તેથી આપણે x ને અથવા y અથવા z લેતા નથી. તેથી, અમારી પાસે આ તમામ કલમો એક કલમ બનાવવા માટે એકસાથે છે, આ કલમોને ક્રિયા સાથે એકસાથે મૂકો. તેથી, સામાન્ય રીતે, અમે દલીલ કરી હતી કે સંતોષકારક સોંપણી કેવી રીતે મેળવવી તે વિચારવું મુશ્કેલ છે, પરંતુ જો અમને વેલ્યુએશનની સોંપણી મળે છે, તો વેરિયેબલ માટેનાં મૂલ્યો, આપણે તે ચકાસી શકીએ કે તે સાચી છે કે નહીં. તેથી, સંતોષના પ્રતિબંધિત ફોર્મને 3-સેટ કહેવામાં આવે છે. 3-એસએટીમાં, આપણી પાસે માત્ર એક્સ જેવા કે ક્લોઝ વાય અથવા ઝેડ છે અથવા આપણી પાસે પણ બે, એક્સ અથવા વાય અને બીજું હોઈ શકે છે. તેથી, આપણે ઝેડ જેવી નાની બાબતો પણ હોઈ શકીએ, પરંતુ યાદ રાખીએ કે, જો અમારી પાસે એક કલમ છે, તો તે કહે છે કે, તેને બનાવવાનો એકમાત્ર રસ્તો છે, આ મધ્યને ઝેડ સાચી બનાવવાની મધ્યમ બનાવે છે, કારણ કે ત્યાં કોઈ વિકલ્પ નથી. જો હું એક્સ અથવા વાય જોડું છું, તો હું એક્સ સાચી અથવા વાય સાચી બનાવી શકું છું, તેથી આનો અમારો કલમ વધુ સમસ્યાને ફાળો આપતું નથી, કારણ કે તે કોઈ ચોક્કસ વેરિયેબલ માટે વેલ્યુએશનને દબાણ કરે છે, અને પછી હું સમસ્યાને એકમાં ઘટાડી શકું છું જે 2 ધરાવે છે અથવા 3. તેથી, 3-એસએટીમાં સામાન્ય રીતે બે વેરિયેબલ્સ અથવા ત્રણ વેરિયેબલ છે, પરંતુ ત્રણથી વધુ નથી, તેમાં ચાર અથવા પાંચ વેરિયેબલ નથી.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 07:14)

તેથી, પ્રથમ દાવો એ છે કે 3-એસએટી સામાન્ય રીતે એસએટી છે. તો, આપણે આ એક સરળ ઉદાહરણ દ્વારા બતાવ્યું છે. તેથી, માનીએ કે મારી પાસે કોઈ ક્લોઝ છે, જેમાં આ જેવા પાંચ ચલ છે. તો, હું શું કરવા માંગુ છું, મારે માત્ર ત્રણ વેરીએબલો જોઈએ છે, તેથી હું શું કરીશ, આ ભાગની સ્લાઈસ. તેથી, હું આ ભાગની પ્રથમ બે અને હું સ્લાઈસ રાખું છું, તેથી હું એક નવી વેરિયેબલ દાખલ કરું છું અને હું કહું છું કે તે V છે અથવા W અથવા A નથી અને હું બાકીની સાથે મુકું નથી. તેથી, હવે આ મને શું કહે છે, તેથી આ મને કહે છે કે આ કિસ્સામાં માનવું, કે આ બંને ખોટા છે. પછી, આ સાચું બનાવવા અને સાચું બનાવવા માટે, પરંતુ સાચું છે, તો તે બીજી બાજુ ખોટું છે, એટલે તેનો અર્થ એ છે કે આ કલમો સાચા છે, તેમાંની એક સાચું છે. તેથી, બીજા શબ્દોમાં કહીએ તો, એ અને ડબ્લ્યુ એમ બંને કહે છે, તે ખોટા નથી, તો પછી અન્ય ત્રણ ચલો સાચું હોવા જોઈએ, જે મૂળ સમસ્યા છે તે જ છે; આ પાંચમાંથી એક સાચું હોવું આવશ્યક છે. બીજી બાજુ, જો

વી અથવા નહીં w true છે, તો હું ખોટો સેટ કરવાનો પ્રયાસ કરી શકું છું, જો ખોટી ન હોય તો પણ સાચી થઈ જાય, પછી હું બાકી ચલોની કોઈ એપ્લિકેશન નથી.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 08:35)

તેથી, દાવા એ છે કે આ વિસ્તરણ દ્વારા આને દૂર કરીને, પ્રથમ બે વેરિયેબલ્સને વિભાજીત કરીને મધ્યવર્તી વસ્તુમાં તેને ઉમેરીને બે કલમોને જોડીને બે કલમો પેદા કરે છે, જે સમાન સંતોષ ધરાવે છે મૂળ કલમ. હવે ફરીથી, મારી પાસે કમનસીબે ઘણા બધા વેરિયેબલ છે, તેથી હું શું કરીશ, હવે હું ફરીથી બે રાખીશ અને જવાબને વિભાજીત કરીશ. તો, હું a x a નવો વરિયેબલ b રાખીશ નહીં, અને પછી હું b લેશે નહીં અને બાકીના b લેશે, આ બિંદુએ, સદભાગ્યે હું હવે ત્રણ વેરિયેબલમાં બધું ઘટાડીશ, જો હું ચાર વેરિયેબલ હોય તો હું તેને રોકી શકું છું. થોડું સી કરો અને સી ન કરો અને બીજું પણ કરો. તો, આ રીતે, હું કોઈપણ કલમ લઈ શકું છું જેમાં ત્રણ કરતા વધુ શાબ્દિક હોય છે અને ત્રણ શાબ્દિક કલમોના અનુક્રમમાં નવા શાબ્દિકને ઉમેરીને તેને વ્યવસ્થિત રીતે વિખેરી નાખે છે. તેથી, હું કોઈપણ કલમને કન્વર્ટ કરી શકું છું જે 3-એસએટી ક્લોઝસના અનુક્રમમાં 3-સેટમાં નથી અને અનુક્રમ કેટલું મોટું છે, તે પણ કલમમાં શાબ્દિક સંખ્યાઓનો ભાગ બનશે. તેથી, મારા ફોર્મ્યુલામાં ફટકો મૂળ સૂત્રમાં રેખીય બનશે, તે મહત્વપૂર્ણ છે, કારણ કે અમે કહ્યું છે કે ઘટાડા થશે, અમે ઈચ્છું છું કે ઘટાડો ભાગ એ કહેવું કાર્યક્ષમ બનશે કે, અમે એટલો સમય વિતાવી રહ્યા નથી ઘટાડાનો, અન્ય અમે તે દાવાઓ પર અસરકારક રીતે સ્થાનાંતરિત કરી શકતા નથી. તેથી, આમાં આપણને શું કહે છે, જો સીએટી સખત હોય તો આપણે માનીએ છીએ, તો 3-એસએટી છે, કારણ કે હું SAT ને 3-SAT થી ઘટાડે છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 10:01)

તેથી, અત્યાર સુધીમાં આપણે એવા ઘટાડા જોયા છે જે કદાચ આશ્ચર્યજનક નથી, આપણે સ્વતંત્ર સેટ અને વર્ટેક્સ કવર જોયું છે, પરંતુ આસમાન સમસ્યાઓ છે. એ જ રીતે, આપણે SAT અને 3-SAT જોયું છે, આ સમાન સમસ્યાઓ છે. હવે, આપણે જે કરવા જઈ રહ્યા છીએ તે છે, આપણે આ બે જુદી જુદી સમસ્યાઓ વચ્ચેનો તફાવત ભરવા જઈ રહ્યા છીએ. અમે 3-એસએટી પર ધ્યાન આપીએ છીએ અને કહીએ છીએ કે 3-એસએટી ખરેખર તમે સ્વતંત્ર સેટમાં ઘટાડો કરી શકો છો, જે થોડી વધુ આશ્ચર્યજનક છે, કારણ કે તે જુદા જુદા ડોમેન્સમાંથી આવે છે. તેથી, અહીં એક લાક્ષણિક 3-SAT સૂત્ર છે. તેથી, આપણી પાસે 1, 2, 3, 4 કલમો છે અને તેમાંથી પ્રત્યેક ત્રણ શાબ્દિક છે, તેથી આમાં બે શાબ્દિક છે, તેથી બીજું એક છે ... તેથી, કારણ કે આપણી પાસે 3-SAT છે, તો પછી આપણી પાસે એક માળખું છે ત્રણ દ્વારા બંધાયેલ છે. તેથી, આપણે આ ત્રિકોણ દ્વારા દરેક કલમનું પ્રતિનિધિત્વ કરીશું અને ત્રિકોણમાં એક લેબલ મુકશે જે શાબ્દિક સૂચવે છે. તેથી, x એ x દ્વારા નથી બદલાઈ રહ્યું છે, આ વસ્તુમાં છે, વાય અને તેથી આ નથી x અથવા y અથવા z z એ આ ત્રિકોણ નથી. એ જ રીતે, આ એક્સ નથી અથવા વાય અથવા ઝેડ નથી, તેથી આ રીતે દરેક કલમ ગ્રાફ માટે ત્રિકોણમાં ફાળો આપે છે. તેથી, આ ત્રિકોણ શું સૂચવે છે, આ ત્રિકોણ સૂચવે છે કે જો સ્વતંત્ર સેટ આ વસ્તુઓમાંથી એક પસંદ કરી શકે છે, તો y કહો, પછી તે બીજા બેને પસંદ કરી શકશે નહીં. તેથી, આ જે કહેવામાં આવે છે તે આમાંની એક વસ્તુ બરાબર બનાવશે, તે મારા માટે ઓળખશે જે આ ફોર્મ્યુલાને વાસ્તવમાં કાર્યવાહી કરવા માટે શાબ્દિક સાચા છે. તેથી, દરેક કલમમાંથી મને સાક્ષી આપવી એ એક પ્રકારનું છે. તેથી, અહીં માન્ય સાક્ષી હોવાનું સારું નથી 2, પછી હું આગામી કલાકે સાક્ષી બનવા માટે વધારે પસંદ કરી શકતો નથી, કારણ કે વાય અને વાય એકબીજાને ફાળો આપતા નથી. તેથી, ત્રિકોણ

ઉપરાંત, મારી પાસે આ લીલી ધાર છે કે ચલણ કહે છે અને તે ખુશામતને લીલી ધારથી જોડશે. તેથી, આ વાય જોડીને કદાચ વધુ લીલી ધારો હોવી જોઈએ કે જે વાય નથી, તેથી ત્યાં ક્લોઝ હોવું જોઈએ, તેથી આ y ને તે નથી. તેથી, દરેક વેરિયેબલ જોડાયેલ છે તે સૂચવવા માટે ખુશામત છે કે જો ત્યાં એક છે, તો બીજું ત્યાં હોઈ શકતું નથી. તેથી, આપણે સ્વતંત્ર સેટ શું કહીએ છીએ કે આ એક ઉકેલ પસંદ કરવાનો સમય છે, દરેક કલમ જોઈને એક ઉકેલ પસંદ કરવાનો પ્રયાસ કરી રહ્યો છે. અને એમ કહીને કે હું અહીં y પસંદ કરી શકું છું, હું અહીં y પસંદ કરું છું, પછી હું આ વસ્તુઓને શાસન કરવા જાઉં છું અને હું કદાચ હવે હું ત્યાં યુ પસંદ કરીશ, પછી હું અહીં કંઈક પસંદ કરીશ. તેથી, કદાચ હું અહીં x પસંદ કરું છું, જો હું અહીં x પસંદ કરું, તો મેં નકારી કાઢ્યું કે x નથી, તમારે આને x નાપસંદ કરવો પડશે. હવે, મને તકલીફ છે, કારણ કે મારી પાસે કંઈપણ નથી, તેથી કદાચ હું તે પસંદ કરી શકું નહીં. તો, તેથી, હું પાછો જાઉં છું અને કહું છું કે, આને x ને પસંદ કરવું જોઈએ નહીં, આ x એ આ એક્સને નકારી કાઢશે નહીં, આ એક્સને નકારી કાઢશે, આ ઝેડ રાખવા માટે બળ મળે છે. હવે, જો મારી પાસે કદ 4 નું સ્વતંત્ર સેટ હોય, તો હું પૂર્ણ કરીશ, તેથી હું કહી શકું છું કે હવે હું જે ઉકેલ માંગું છું તે x છે ખોટા છે, y true પર જાય છે અને z true થાય છે. તેથી, કદ 4 ની આ વિશેષ વસ્તુ પર સ્વતંત્ર સમૂહ અસ્તિત્વ, જે 1 દીઠ ખર્ચ કરે છે. તો, નોંધ લો કે એક કલમની અંદર, હું માત્ર 1 પસંદ કરી શકું છું. તેથી, જો મારી પાસે ચાર કલમો હોય અને જો હું કદ પર આગ્રહ રાખું તો સ્વતંત્ર પ્રક્રિયા એ પ્રક્રિયાની સંખ્યા છે, તો પછી મને ચોક્કસપણે એક શાબ્દિક દીઠ એક સાક્ષી મળશે. અને તે સાક્ષીઓ આપણે પરસ્પર જોડાયેલા છીએ, કારણ કે હું ઝેડને જોડું છું અને ઝેક્સ નથી અને દરેક જગ્યાએ x ને નથી જે કિનારીઓ દ્વારા સમાન વેરિયેબલ ફ્લેબને રદ કરે છે, એક વખતમાં સ્થાનાંતરિત રીસેટ દ્વારા એક વખત ખોટો છે. તેથી, આ સ્વતંત્ર સમૂહ હવે તે શાબ્દિકને પસંદ કરે છે જે ચાર, જે મારું ફોર્મ્યુલા સાચું બનાવે છે. તેથી, જો કદ 4 નો સ્વતંત્ર સમૂહ હોય, તો દરેક કલમ સાચી થઈ શકે છે અને આ ચાર ક્લોઝનો સમૂહ બની શકે છે. અને તેથી, મારી પાસે 3-SAT નો ઉકેલ છે અને જો કદ 4 નો કોઈ સ્વતંત્ર સમૂહ નથી, તો ત્યાં કોઈ ઉકેલ નથી. તેથી, 3-એસએટી સ્વતંત્ર સેટ ઘટાડે છે જે થોડી વધારે આશ્ચર્યજનક છે, જ્યારે આપણે એસએટીથી 3-એસએટી સુધીના ઘટાડામાં ઘટાડો કરીએ છીએ, જે બંને સમાન પ્રકારની સમસ્યાઓ છે. સ્વતંત્ર સેટથી વર્ટેક્સ કવર અને પછાતથી ઉપર જે સમાન પ્રકારના ગ્રાફમાં સીધું માળખાગત સંબંધિત સમસ્યા છે. તેથી, આ આશ્ચર્યજનક વસ્તુ છે બુલિયન સંતુષ્ટતા અને શંકુ મુદ્દાઓ, હું સ્વતંત્ર શિરોબિંદુ છું અને ગ્રાફ જોડાય છે.

(સ્લાઈડટાઈમ નો સંદર્ભ લો: 14:26)

તેથી, આપણે બતાવ્યું છે કે 3-SAT 3-SAT ઘટાડે છે, 3-SAT સ્વતંત્ર સેટ ઘટાડે છે, સ્વતંત્ર સેટ અને વર્ટેક્સ કવર પરસ્પર સક્ષમ થાય છે. તેથી, હમણાં પૂરતું કારણ કે હવે હું આ ઘટાડાને કંપોઝ કરી શકું છું, એસએટી હવે કર્કશ ઘટાડે છે અને તમે સાહિત્ય, પૂર્ણાંક, રેખીય પ્રોગ્રામિંગ વગેરે જેવી મુસાફરી કરતી બીજી વિવિધ સમસ્યાઓ માટે આવા ઘટાડાને શોધી શકો છો. હવે, આ બધા મહત્વપૂર્ણ છે યાદ રાખો કે આપણે આ સંદર્ભને ઘટાડવા માટે જોઈ રહ્યા છીએ, જ્યાં પહેલા પ્રોસેસિંગ અને પોસ્ટ પ્રોસેસિંગ પગલાં બહુવર્ષીય સમયે છે. કારણ કે, આપણે કહીએ છીએ કે ઘટાડામાં કોઈ સમય ગુમાવવો નથી, તેથી, બધી જટિલતા સોલ્યુશનમાં છે, જો મારી પાસે બી માટેનો સરળ ઉકેલ હોય, તો મારા માટે એક સરળ ઉકેલ છે, જ્યારે હું b ઘટાડે છે. જો મારી પાસે સંપૂર્ણ ઉકેલ છે, તો હું કહી શકતો નથી કે સોલ્યુશન રિડક્શન એ જટિલ સેટ કરી શકે છે, તે કારણ છે કે ઘટાડો સરળ છે, તે ફક્ત તે જ હકીકત છે જે બે સમસ્યાઓ સમાન છે કે તે ખરેખર સમાન જટિલતા ધરાવે છે. તેથી, આ બધી સમસ્યાઓ જ્યારે અંતરાય ઘટાડે છે, ત્યારે તેનો અર્થ એ છે કે તે બધા સમાન છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 15:26)

હવે, કૂક અને લેવિન દ્વારા પ્રસિદ્ધ પ્રમેય કહે છે કે દરેક સમસ્યા એનપીને સ્ટેકમાં ઘટાડી શકાય છે. હવે, આ એક આશ્ચર્યજનક પ્રમેય છે, કારણ કે આ બધું જ એન.પી. લે છે, ત્યાં કંઈપણ હું સીએટી ઘટાડી શકું છું. તેથી, સમસ્યાને તાત્કાલિક લઈને થોડો ઘટાડો થાય છે, હું એસએટીના આગામી મહિને રૂપાંતરિત થઈ શકું છું, તેથી તે બધા બેઠેલા શામેલ છે અને તે પાછું જાય છે. તેથી, આપણે અહીં આ પ્રમેયને અજમાવવા અને સાબિત કરવા માટે ચોક્કસપણે જતા નથી, તે જાણવું પૂરતું છે કે આ પુરાવો ગણતરીના યોગ્ય સામાન્ય મોડ્યુલને એન્કોડિંગ કરીને છે. તેથી, મશીનો દરમિયાન કૂકનો મૂળ પુરાવો, તમે બુલિયન વર્તુળો રજિસ્ટર મશીનો લઈ શકો છો, જે કંઈક સાર્વત્રિક છે, કંઈપણ જે દરેક સમસ્યાને એનપીની ગણતરી કરી શકે છે. અને પછી દલીલ કરે છે કે તે સમસ્યાની ગણતરી, તેથી જો એનપી જેવી કંઈક હોય, તો એનપી ગણતરી હોય, ત્યાં તપાસ કરતી સમસ્યા છે, એલ્ગોરિધમ તપાસો. તેથી, તમે SAT માં તે સમસ્યા માટે તે ચકાસણી અલ્ગોરિધમનો વર્તણૂક એન્કોડ કરી શકો છો અને તેથી, તે તારણ આપે છે કે SAT બરાબર છે તે બધું SAT ને ઘટાડે છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 16:34)

તેથી, આ અમને પૂર્ણતાની કલ્પના આપે છે, અમે કહીએ છીએ કે એસએટી સંપૂર્ણ છે, કારણ કે તે એન.પી.ની છે અને એન.પી.માં દરેક સમસ્યા તેને ઘટાડે છે, આ એનપી પૂર્ણતાની વ્યાખ્યા છે. સમસ્યા એ એનપી સંપૂર્ણ છે, જો તેની પાસે પહેલીવાર ચકાસણી એલ્ગોરિધમનો એનપી, કાર્યક્ષમ ચકાસણી એલ્ગોરિધમનો છે. અને વધુમાં, એનપીમાં દરેક સમસ્યા એક બહુમતિ સમય ઘટાડા દ્વારા તેને ઘટાડે છે. તેથી, કૂક લેવિન થિયરેમ દ્વારા વ્યાખ્યા દ્વારા એસએટી એનપી પૂર્ણ છે, કારણ કે તે એન.પી.ની છે, આપણે જાણીએ છીએ. અને એન.પી.માં દરેક સમસ્યા પર કૂક લેવિન પ્રમેય સ્થાપિત કરે છે, પરંતુ હવે આપણે જોયું છે કે બેઠા પોતે 3-SAT જેટલું ઘટાડે છે. તેથી, પ્રત્યેક સમસ્યાને SAT માં ઘટાડી શકાય છે અને પછી એક વધુ વ્યુત્પન્ન પગલામાં 3-SAT સુધી ઘટાડે છે. તેથી, દરેક સમસ્યા હવે 3-સેટમાં ઘટાડે છે, ત્રણ બેઠકો પણ એનપી છે, તેથી 3-એસએટી પણ એનપી પૂર્ણ છે. તેથી, આ હવે સામાન્ય તકનીક છે, તેથી તમારી પાસે કોઈ સમસ્યા છે જે એનપી પૂર્ણ છે. અને પછી હું તેને મારી સમસ્યા અહીં ઘટાડે છે જે બીને પણ એનપી પૂર્ણ કરે છે, કારણ કે બી એ એનપી છે, જો તે એનપી નથી, તો તે એનપી હાર્ડ કહેવાય છે તે કંઈક નબળું છે, તે ઓછામાં ઓછું મુશ્કેલ છે એનપીની દરેક સમસ્યા, પરંતુ એનપી પોતે નહીં. તેથી, એનપી હાર્ડ કહે છે કે, તે દરેક સમસ્યાને ઘટાડે છે, પરંતુ તમે ખાતરી કરો કે એનપી સાથે સંકળાયેલ નથી, તમારી પાસે તેની ચકાસણી અલ્ગોરિધમનો નથી. તેથી, આવી કેટલીક સમસ્યાઓ છે જેના માટે ચકાસણી એલ્ગોરિધમનો વર્ણન કરવું સરળ નથી, પરંતુ તમે ચોક્કસપણે ઘટાડો બતાવી શકતા નથી. પરંતુ અન્યથા જો બંને કરી શકે છે, તો તમે એક ચકાસણી એલ્ગોરિધમ અને દરેક સમસ્યા શોધી શકો છો, તમને કેટલીક સમસ્યા મળી છે જે એનપીમાં એનપી પૂર્ણ સમસ્યા છે જે તેને ઘટાડે છે, પછી તે એનપી પૂર્ણ થાય છે.

(સ્લાઈડટાઈમનો સંદર્ભ લો: 18:20)

તેથી, પી એ પી.પી. સમાન એનપી સમાન છે કે કેમ તેના પ્રશ્નનો જવાબ આપે છે. તેથી, ઘણી ઉપયોગી સમસ્યાઓ જે રોજિંદા જીવન શેડ્યૂલિંગ, બિન પેકિંગ માટે અત્યંત મહત્વપૂર્ણ છે. તેથી, બૅન પેકિંગ મૂળભૂત રીતે હવે છે જો હું વિવિધ ઑબ્જેક્ટોને કન્ટેનરમાં એક શ્રેષ્ઠ માર્ગ અથવા મુસાફરીના પ્રવાસોમાં મુસાફરી કરવા માંગુ છું, તો અમે શ્રેષ્ઠ પ્રવાસ શોધવા

માંગીએ છીએ, આ બધી જ મહત્વપૂર્ણ જીવન સમસ્યાઓ છે, જે લોકો જીવનને અસરકારક રીતે હલ કરવા માંગે છે. એનપી સંપૂર્ણતાની થિયરી અમને જણાવે છે કે આ બધી સમસ્યાઓ ખરેખર અસમર્થ છે, અને એકબીજાને સમકક્ષ છે, હું એકને હલ કરી શકું છું, હું તેને હલ કરી શકું છું. હવે, અમારી પાસે આ પ્રાયોગિક પુરાવા છે કે કેમ કે આ મહત્વપૂર્ણ સમસ્યા છે કારણ કે મોટી સંખ્યામાં સ્માર્ટ લોકો આ સમસ્યાઓને કમ્પ્યુટર વિજ્ઞાનની શોધમાં આવ્યાં તે પહેલાંથી જોઈ રહ્યા છે. લોકો 100 થી વધુ વર્ષ માટે સુનિશ્ચિત સમસ્યા અને બૅન પેકિંગ સમસ્યાઓ જોઈ રહ્યા છે. તેથી, આ સમસ્યાઓ, જો તેઓએ હજી સુધી તે અને કાર્યક્ષમ ઉકેલ પ્રાપ્ત કર્યું નથી, તો તે સંભવિત લાગે છે કે ત્યાં કોઈ કાર્યક્ષમ ઉકેલ નથી. તેથી, સંપૂર્ણ પ્રાયોગિક પૂરાવાથી પ્રયોગમૂલક દૃષ્ટિકોણથી, એવું સૂચન કરવામાં આવે છે કે એનપી પીથી જુદું છે. પરંતુ કમનસીબે આ એક પુરાવા નથી, ગણિતમાં એક એવી દલીલ કરવી જોઈએ કે એનપી પુરાવા આપીને પીથી અલગ છે કહેવું એ જ કારણ છે કે કેટલીક એન.પી.ની સમસ્યાને લીધે કેટલીક સમસ્યાઓને સમસ્યાઓ શા માટે પોલિનોમિયલ સમયનો ઉપયોગ કરીને હલ કરી શકાય નહીં. આને નીચલા બાઉન્ડ તરીકે ઓળખવામાં આવે છે, તેથી અમારે નિમ્ન બાઉન્ડ સ્થાપિત કરવાની જરૂર છે. કેટલાક એનપી સંપૂર્ણ સમસ્યા માટે અને આ ખૂબ જ મુશ્કેલ છે, આ પડકારરૂપ સાબિત થયું છે, એનપી પૂર્ણતાની ઓળખ 1970 ની શરૂઆતમાં કૂક અને લેવિન દ્વારા કરવામાં આવી હતી. તેથી, હવે 40 વર્ષથી વધુ છે, કારણ કે એન.પી.ની સિદ્ધાંતની શોધ થઈ ગઈ છે અને ઘણા લોકો મુશ્કેલી અનુભવે છે અને કોઈએ આ કરવા માટે સીધી રીત શોધી નથી. તેથી, ઔપચારિક સાબિતી પ્રપંચી છે અને મિલિયન ડોલરના મૂલ્ય છે, કેમ કે ક્લે મેથ ઈન્સ્ટિટ્યુટ દ્વારા આપવામાં આવેલ ઈનામ છે. તેથી, ગણિતમાં આ એક ખુલ્લી ખુલ્લી સમસ્યાઓ છે. તેથી, માટી સંસ્થાએ એક મોટો ઈનામ આપ્યો છે. તેથી, કોઈક સાબિત કરે છે કે આ ખરેખર કેસ નથી, પછી ત્યાં મોટો ઈનામ છે. તેથી, આ સાથે આ કોર્સને સમાપ્ત કરવા માટે તે યોગ્ય સ્થાન લાગે છે. તેથી, આ સમયે હું તમારો આભાર માનું છું.