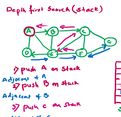
Depth first Search (dstack)

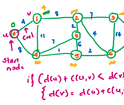


1) push A on stack
Adjacent of A push B on stack
Adjacent of B push D on stack
Adjacent of D push C on stack
Adjacent of C push E on stack
Adjacent of E unvisited push D on stack

1) find adjacent of D unvisited since A is pushed in stack
Backtracking - back to pop D
2) Find adjacent unvisited vertex if empty pop F on stack.
3) Adjacent of F is G push G on stack
Since no unvisited node (backtrack) pop G, pop F, pop E, pop C

ABCGDFG

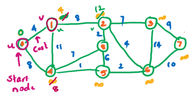
DFS o/p is spanning tree



- used to find shortest path between two nodes
- if any edge has -ve weight then it may or may not work
- But if all weights are +ve then this edge cannot be used
- It can be used for directed as well as undirected

if (d(u) + c(u,v) < d(v)) { d(v) = d(u) + c(u,v) }

d(u) + c(u,v) < d(v)
0 + 4 < ∞
d(v) = d(u) + c(u,v)
= 4



from 0 → 1, 4

0-1 d(u) + c(u,v) < d(v)
0 + 4 < ∞ d(v) = 4

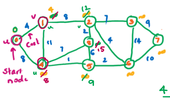
0-4 d(u) + c(u,v) < d(v)
0 + 8 < ∞
8 < 8 d(v) = 8

node 1 (A), 4 (8), other 2, 8, 5, 6, 3, 7 has ∞
Select node with shortest value i.e 1 (4)

from 1 we can visit 4, 2

1-2 d(v) + c(u,v) < d(v) 4 + 8 = 12 < ∞

1-4 d(v) + c(u,v) < d(v) 4 + 11 = 15, old weight was 8 which is less than 15 so do not update 4 let it be 8



2(12), 4(8), 8, 5, 6, 3, 7 ∞
Select node 4 since it's weight is less

from 4 we can visit 1, 8, 5
but 1 is already visited (8, 5)

4-8 d(u) + c(u,v) < d(v)
8 + 7 < ∞
15 < ∞ d(v) = 15

from 2(12), 8(15), 5(9) 3, 6, 7(∞)
Selecting node

4-5 d(u) + c(u,v) < d(v)
8 + 1 < ∞
9 < ∞ d(v) = 9

node selected was 5(9)
from 5 → 8, 6 both 8 & 6 are unvisited

5-8 d(u) + c(u,v) < d(v)
9 + 6 < 15
15 < 15 d(v) = 15

5-6 d(u) + c(u,v) < d(v)
9 + 2 < ∞
11 < ∞ d(v) = 11

8(15), 2(12), 6(11), 3, 7(∞)

from 6 → 2, 3, 7

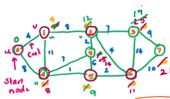
6-2 d(u) + c(u,v) < d(v)
11 + 9 < 12
15 < 12 donot update d(v) = 12

2(12) 8(15), 3(25), 7(21)

Selected node is 2(12)

6-3 d(u) + c(u,v) < d(v)
11 + 14 < ∞
25 < ∞ d(v) = 25

6-7 d(u) + c(u,v) < d(v)
11 + 10 < ∞ d(v) = 21
21 < ∞



from 2 we can visit 8, 6, 3
but 6 is already visited
remaining are 8, 3

2-3 d(u) + c(u,v) < d(v)
12 + 7 < 25 updated
19 < 25 d(v) = 19

2-8 d(u) + c(u,v) < d(v)
12 + 2 < 15 updated
14 < 15 d(v) = 14

8(14), 3(19), 7(21)

But from 8 we can visit 2, 5, 4 which are already visited no updation

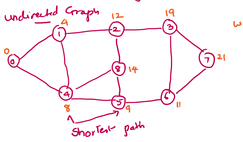
remaining are 3(19) 7(21)

3(19) is selected

3-6, 7 but 6 is already visited

3-7 d(u + c(u,v) < d(v)
19 + 9 < 21 donot update d(v) = 21
28 < 21

remaining node is 7, from 7 - 3, 6 already visited



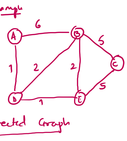
Undirected Graphs

We can find shortest path between two node

0 - 1 (4) 0 - 2 (12)
0 - 4 (8) 0 - 7 (21)
0 - 3 (19) 0 - 5 (9)
and so on

Shortest path

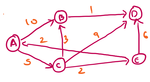
Example



Final shortest path

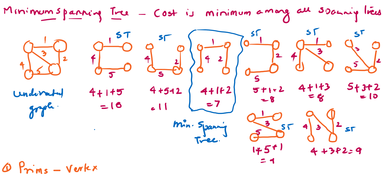
A - 0
B - 3
D - 1
C - 7
E - 2

Directed Graph



A - 0
D - 8
C - 5
D - 9
E - 7

Minimum spanning Tree - Cost is minimum among all spanning trees



Undirected graph

4+15 = 10

4+5+2 = 11

4+12 = 7

Min. Spanning Tree.

5+1+2 = 8

4+3+3 = 10

5+3+2 = 10

14+5+1 = 1

4+3+2 = 9

① Prims - Vertex

② KruskKals - edge