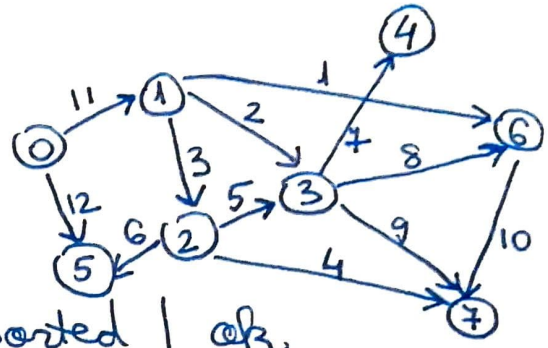
(1) Topological sorting using DFS algorithm

File : exec-topo1.txt



8 vertices
12 edges

calls	x, y	inProcess	fullyProcessed	sorted	ok
TopoSortDFS(0,[],{},{})	x=0	{} {0} {}	{} {0}	[] [0]	true
TopoSortDFS(1,[0],{0},{})	x=1 y=0	{} {1} {}	{0} {0,1}	[0] [0,1]	true
TopoSortDFS(2,[0,1],{0,1},{})	x=2 y=1	{} {2} {}	{0,1} {0,1,2}	[0,1] [0,1,2]	true
TopoSortDFS(3,[0,1,2],{0,1,2},{})	x=3 y=1 y=2	{} {3} {}	{0,1,2} {0,1,2,3}	[0,1,2] [0,1,2,3]	true
TopoSortDFS(4,[0,1,2,3],{0,1,2,3},{})	x=4 y=3	{} {4} {}	{0,1,2,3} {0,1,2,3,4}	[0,1,2,3] [0,1,2,3,4]	true
TopoSortDFS(5,[0,1,2,3,4],{0,1,2,3,4},{})	x=5 y=0 y=2	{} {5} {}	{0,1,2,3,4} {0,1,2,3,4,5}	[0,1,2,3,4] [0,1,2,3,4,5]	true
TopoSortDFS(6,[0,1,2,3,4,5],{0,1,2,3,4,5},{})	x=6 y=1 y=3	{} {6} {}	{0,1,2,3,4,5} {0,1,2,3,4,5,6}	[0,1,2,3,4,5] [0,1,2,3,4,5,6]	true
TopoSortDFS(7,[0,1,2,3,4,5,6],{0,1,2,3,4,5,6},{})	x=7 y=2 y=3 y=6	{} {7} {}	{0,1,2,3,4,5,6} {0,1,2,3,4,5,6,7}	[0,1,2,3,4,5,6] [0,1,2,3,4,5,6,7]	true → topological order

Highest cost path

→ between 1 and 4 : 15

dist = [-∞, -∞, -∞, -∞, -∞, -∞, -∞, -∞]

dist [1] = 0 ⇒ dist = [-∞, 0, -∞, -∞, -∞, -∞, -∞, -∞]

sorted = [0, 1, 2, 3, 4, 5, 6, 7]

u = 0

inbound edges : —

outbound edges : 1 → dist [1] < (dist [0] + 11) ? ⇔ 0 < -∞ NO

5 → dist [5] < (dist [0] + 12) ? ⇔ -∞ < -∞ NO

u = 1

inbound edges : 0 → dist [1] < (dist [0] + 11) ? ⇔ 0 < -∞ NO

outbound edges : 2 → dist [2] < (dist [1] + 3) ? ⇔ -∞ < 3 YES ⇒ dist [2] = 3

3 → dist [3] < (dist [1] + 3) ? ⇔ -∞ < 3 YES ⇒ dist [3] = 3

6 → dist [6] < (dist [1] + 3) ? ⇔ -∞ < 3 YES ⇒ dist [6] = 3

u = 2

inbound edges : 1 → dist [2] < (dist [1] + 3) ? ⇔ 3 < 3 NO

outbound edges : 3 → dist [3] < (dist [2] + 5) ? ⇔ 3 < 8 YES ⇒ dist [3] = 8

5 → dist [5] < (dist [2] + 6) ? ⇔ 3 < 9 YES ⇒ dist [5] = 9

7 → dist [7] < (dist [2] + 4) ? ⇔ 3 < 7 YES ⇒ dist [7] = 7

u = 3

inbound edges : 1 → dist [3] < (dist [1] + 2) ? ⇔ 8 < 2 NO

2 → dist [3] < (dist [2] + 5) ? ⇔ 8 < 8 NO

outbound edges : 4 → dist [4] < (dist [3] + 7) ? ⇔ -∞ < 15 YES ⇒ dist [4] = 15

6 → dist [6] < (dist [3] + 8) ? ⇔ 3 < 16 YES ⇒ dist [6] = 16

7 → dist [7] < (dist [3] + 9) ? ⇔ 7 < 17 YES ⇒ dist [7] = 17

u = 4

↳ inbound edges : 3 → dist[4] < (dist[3] + 7) ? <=> 15 < 15 NO

↳ break the loop

=> the highest cost path is dist[4] = 15 .