Tutorial 6

1) The minimum spanning Tree is the one whose cumulative edge weights have the smallest value

Applications

i) Network design (i.e. telephone or cable networks)
ii) finding approximate Sol^n's for complex problems like Traveling salesman problem.
iii) cluster Analysis.

2) Prims algorithm has a time complexity of $O(V^2)$, V being the no. of vertices

→ We need an array to know if a node is in MST or not. Space $O(V)$

→ we need an array to maintain Min-Heap. Space $O(E)$

Total Space complexity $O(V+E)$

Kruskal's algorithm time complexity is $O(E \log V)$, V being no. of vertices

Since Disjoint Set Data structure take $O(|V|)$ Space to keep track of the roots of all vertices and another $O(|E|)$ Space to store all edges in sorted manner.

Total Space complexity :- $O(E+V)$.

Dijkstra algorithm time complexity is $O(V^2)$ when Graph is represent in adjacency matrix

there, time taken for selecting i with the smallest dist is $O(V)$.

for each neighbour of i, time taken for updating dist[j] is $O(1)$ and there will be min. V neighbours

time taken for each iteration of loop is $O(V)$ and one vertex is deleted from Q.

Space complexity is $O(V)$, we use an array to store the values of the shortest distance for every node in the graph. So space complexity is

$O(V) + O(V) = O(2V) \approx O(V)$.

T.C of Bellman ford algorithm is $O(|V||E|)$ where V is no. of vertices and $|E|$ is no. of edges. If graph is complete,

2

the values of $|E|$ becomes $O(V^2)$. So overall T.C becomes $O(V^3)$.

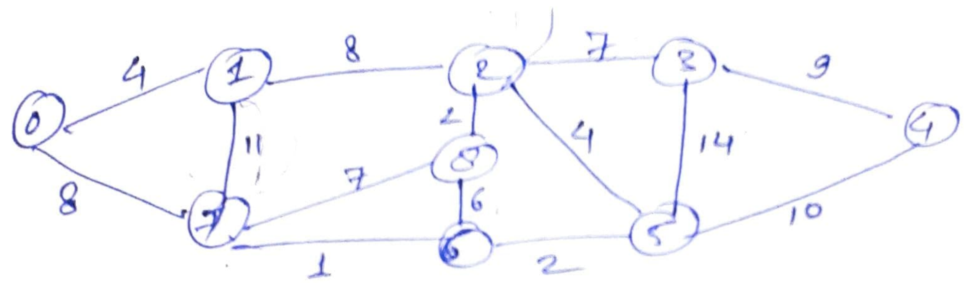
Space complexity in case of adjacency matrix is input + extra $(O(V^2)) + O(V)$ → using min heap $= O(V^2)$.

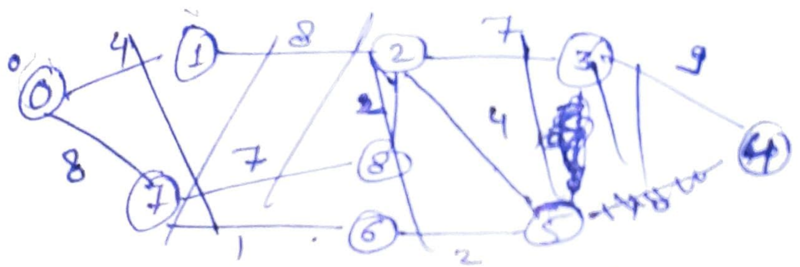
in case of adjacency list, space = input + extra

$E = O(V^2)$

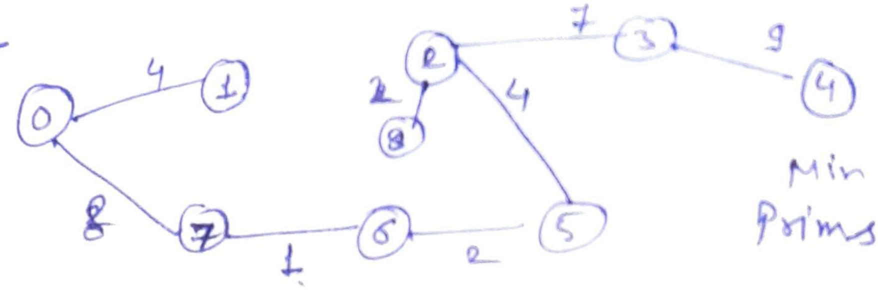
$O(V+E) + O(V)$ → min heap $= O(V^2)$.

3



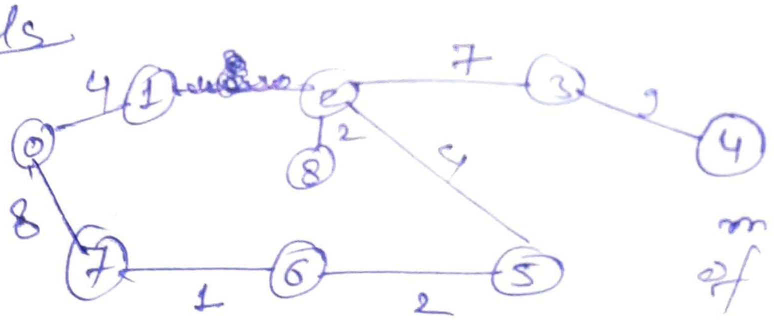


Prims



Min wt in case of Prims algo $= 4+8+1+2+4+2+7+9 = 37$.

Kruskals

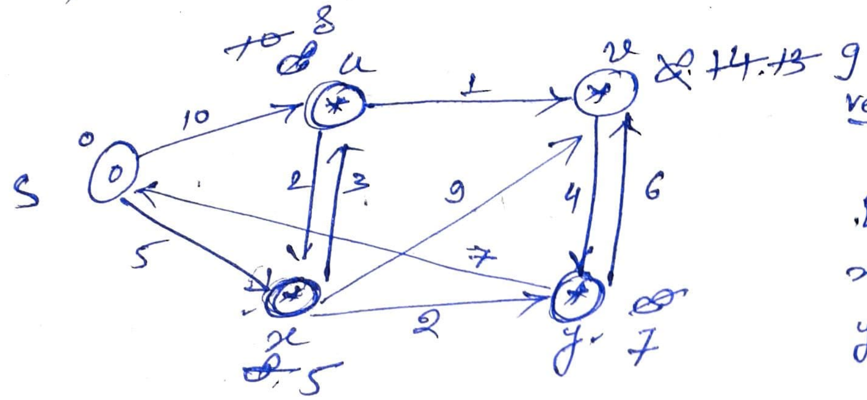


min. wt. in case of Kruskals algo $= 4+8+1+2+4+2+7+9 = 37$

Q ~~3~~ 4.

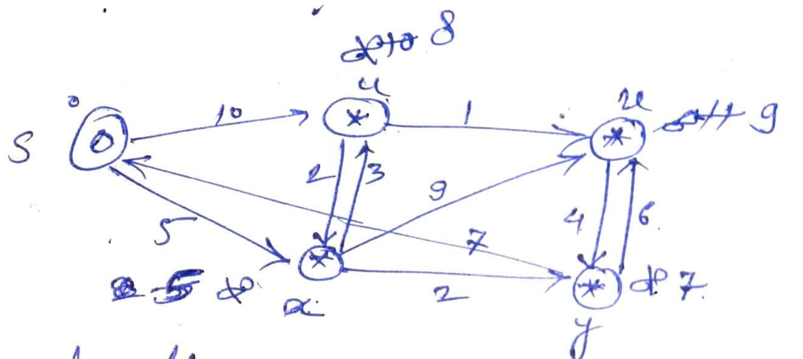
The shortest path may change. The reason is, there may be different no. of edges in different paths from 's' to 't'. for eg, let shortest path be of wt. 15 and 5 edges. Let there be another path with 2 edges and total wt 25. The wt. of the shortest path is increased by ~~2*10~~ 5*10 and becomes 15+50. Wt. of other path is increased by 2*10 and becomes 25+20. So, the shortest path changes to other path with wt. as 45.

5



Using dijkstra

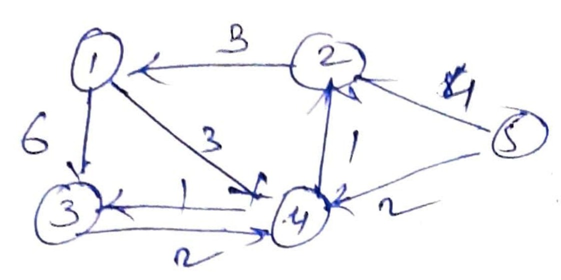
vertex	dist
u →	8
v →	9
x →	5
y →	7



edge list → (0,u), (0,x), (u,v), (u,x), (x,u), (x,y), (x,v), (v,y), (y,0), (y,v)

vertex	dist
u →	8
v →	9
x →	5
y →	7

(6)



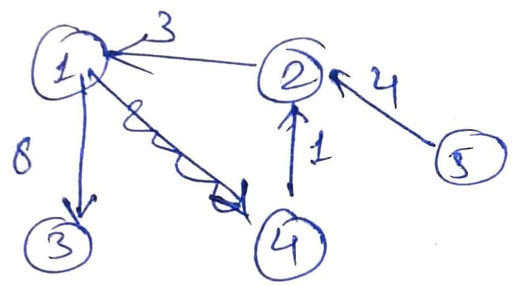
D

	1	2	3	4	5
1	0	~~∞~~ 4	~~∞~~ 4 6	3	∞
2	3	0	~~∞~~ 7 9	~~∞~~ 6	∞
3	~~∞~~ 6	~~∞~~ 3	0	2	∞
4	~~∞~~ 4	1	1	0	∞
5	~~∞~~ 6	~~4~~ 3	~~∞~~ 3	2	0

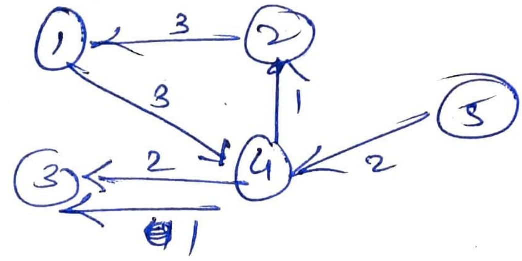
π (4)

	1	2	3	4	5
1	/	4	4	1	/
2	2	/	~~1~~ 4	~~1~~	/
3	~~2~~	4	/	3	/
4	~~2~~	4	4	/	/
5	~~2~~	~~5~~ 4	~~2~~ 4	5	/

i)



ii)



final shortest path for all pairs :-

D

	1	2	3	4	5
1	0	4	4	3	∞
2	3	0	7	6	∞
3	6	3	0	2	∞
4	4	1	1	0	∞
5	6	3	3	2	0

π

	1	2	3	4	5
1	/	4	4	1	/
2	2	/	4	1	/
3	2	4	/	3	/
4	2	4	4	/	/
5	2	4	4	5	/