Date.
Page No.

Tutorial - 6

1) Minimum Spanning Tree

→ It is a special kind of tree that minimizes the length or weights of edges of the tree.

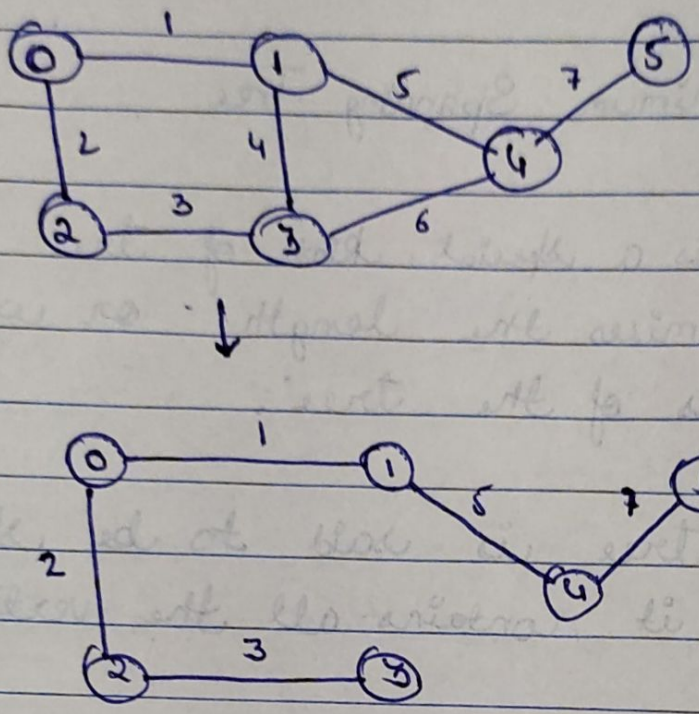
→ A tree is said to be spanning tree:
- If it contains all the vertices.
- Spans all the vertices with n-1 edges
- Is acyclic

→ A tree is minimum spanning tree if it spans the minimum weight while spanning all the vertices.

Applications

→ Telephone
→ TV Cable
→ Computer Networks
→ Constructing roads while spanning several areas / cities.

Date.
Page No.



2) Time Complexity

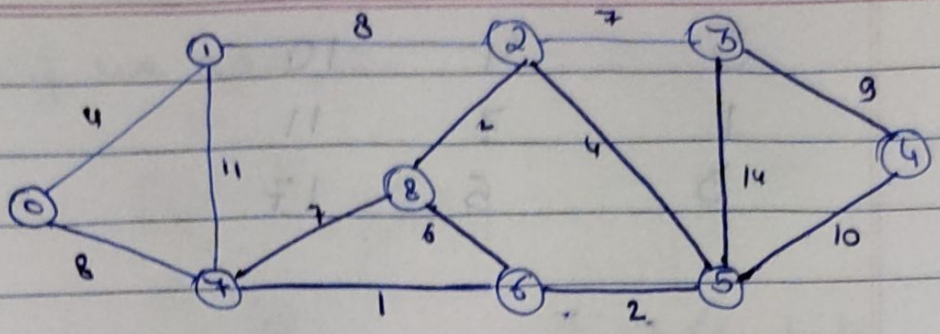
Kruskal's algorithm: $O(E \log V)$

Dijkstra's algorithm: Matrix - $O(n^2)$
Heap - $O(E \log V)$

Prim's algorithm: Matrix - $O(n^2)$
Heap - $O(E \log V)$

Date.
Page No.

3)



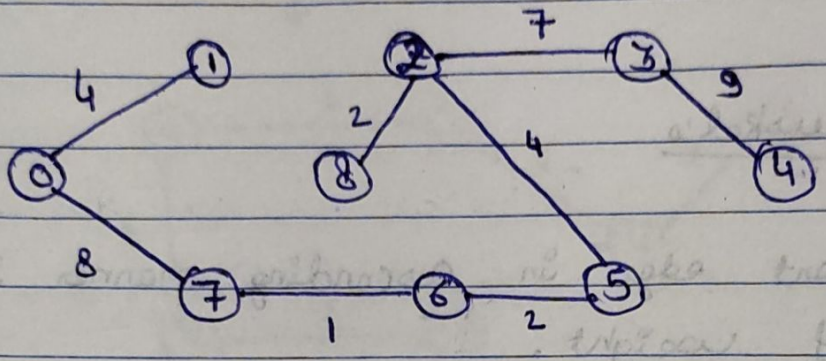
Kruskal's

- Sort edges in ascending manner in terms of weight.
- Pick an edge with min weight and push it to result.
- Continue this for V-1 edges until cycle does not come.

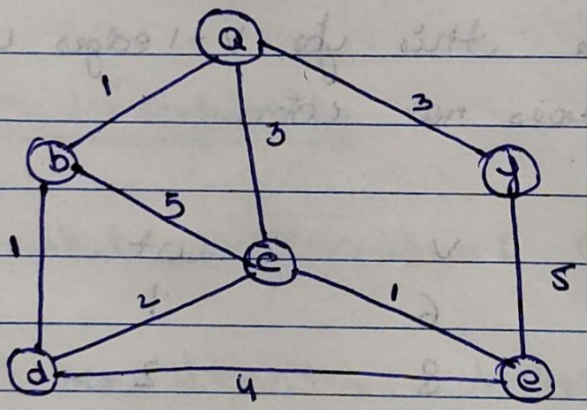
u	v	wt
7	6	1
2	8	2
6	5	2
0	1	4
2	5	4
8	6	6
7	8	7
2	3	7
0	7	8
1	2	8
3	4	9

Date.
Page No.

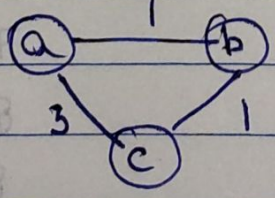
u	v	wt
5	4	10
1	7	11
3	5	17



4)



- If we add the weights of the graph by 10, Yes the shortest path can change Consider

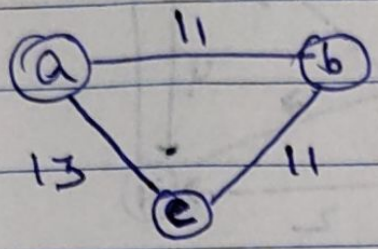


Shortest path

a → b → c

Date.
Page No.

if we add 10



Shortest path
$a \rightarrow c$

- There is no change in the shortest path if we multiply all the weights of edges by 10.

If we multiply

let $10N < 10M$

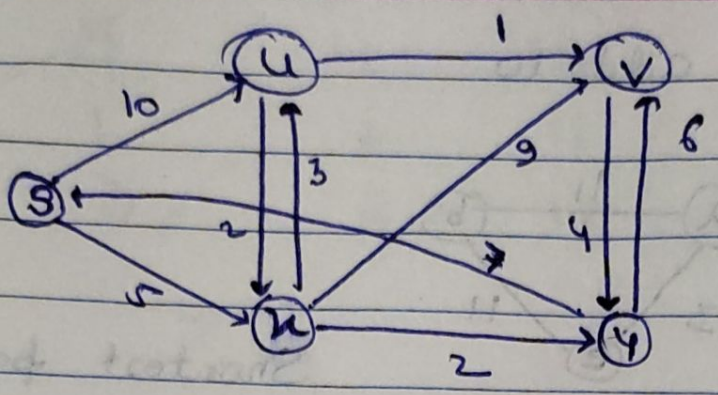
if $N < M$

$\Rightarrow 10N < 10M$

$\therefore$ no change

Date.
Page No.

5)



Dijkstra's Algorithm

- Create sptset which keeps track of vertices
- We assign all the vertices with distance infinite. Then we assign distance of source node to 0.
- while sptset does not include all the vertices.

(i) Pick a vertex which is not sptset and has min distance.
(ii) Include it in sptset.
(iii) Update distance value of all the adjacent vertices of the above vertex using condition.

```
if (dist[v] > dist[u] + graph[u][v])
        dist[v] = dist[u] + graph[u][v]
```

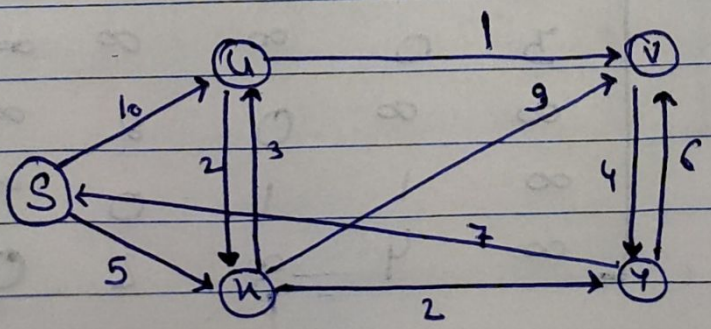
Date.
Page No.

Node	Shortest dist. from source
s	0
u	8
x	5
v	9
y	7

Bellman's Ford's Algorithm

- Initialise distances of all vertices to ∞
  assign dist[0] = 0.
- Repeat this for v-1 times to calculate shortest distance for each edge u-v.
  if (dist[v] > dist[u] + weight[u][v])
  dist[v] = dist[u] + weight[u][v]

- This will report if there is a negative weight.
  do this for each edge u-v.
  if (dist[v] > dist[u] + weight[v][u])

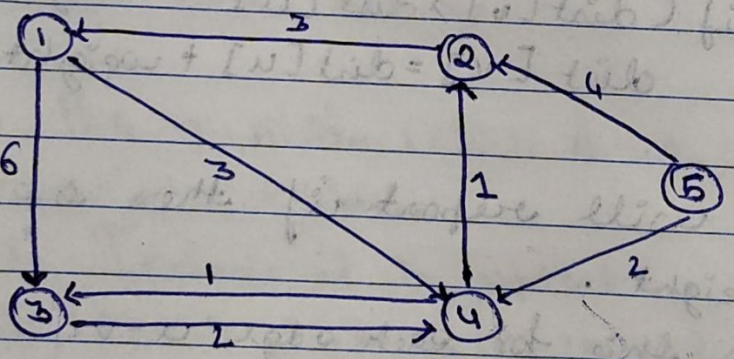


Date.
Page No.

(s,u), (s,x) (u,x) (u,v) (v,y) (y,v) (y,z) (x,u)
(x,v) (x,y)

Node	Shortest dist
s	0
u	8
x	5
y	7
v	9

6) Floyd Warshall:-



$$G = \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & 2 & \infty \\ \infty & 1 & 1 & 0 & \infty \\ \infty & 4 & \infty & 2 & 0 \end{bmatrix}$$

Date.
Page No.

$$G_1 = \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & 9 & 6 & \infty \\ \infty & \infty & 0 & 2 & \infty \\ \infty & 1 & 1 & 0 & \infty \\ \infty & 4 & & 2 & 0 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & 9 & 6 & \infty \\ \infty & \infty & 0 & 2 & \infty \\ 4 & 1 & 1 & 0 & \infty \\ 7 & 4 & 13 & 2 & 0 \end{bmatrix}$$

$$G_3 = \begin{bmatrix} 0 & \infty & 6 & 3 & \infty \\ 3 & 0 & 9 & 6 & \infty \\ \infty & \infty & 0 & 2 & \infty \\ 4 & 1 & 1 & 0 & \infty \\ 7 & 4 & 13 & 2 & 0 \end{bmatrix}$$

$$G_4 = \begin{bmatrix} 0 & 4 & 4 & 3 & \infty \\ 3 & 0 & 7 & 6 & \infty \\ 6 & 3 & 0 & 2 & \infty \\ 4 & 1 & 1 & 0 & \infty \\ 6 & 3 & 3 & 2 & 0 \end{bmatrix}$$

Date.
Page No.

$$G_5 = \begin{bmatrix} 0 & 4 & 4 & 3 & \infty \\ 3 & 0 & 7 & 6 & \infty \\ 6 & 3 & 0 & 2 & \infty \\ 4 & 1 & 1 & 0 & \infty \\ 6 & 3 & 2 & 2 & 0 \end{bmatrix}$$