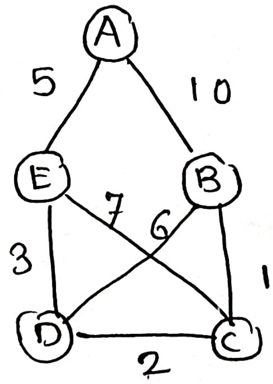
* Minimum Spanning Tree (MST)

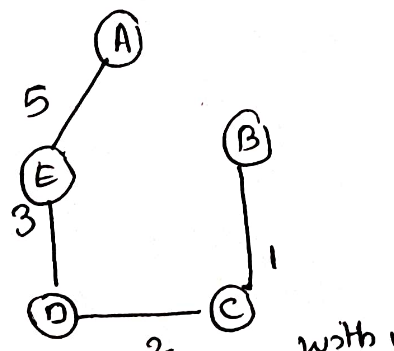
Spanning Tree :- Contains all the vertices of Graph 'G' but no. circuit (no cycle) Which does not consist of loop.

Total no. of MST = (n^{n-2}) where n = no. of nodes

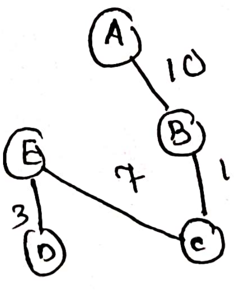
MST — Minimum spanning tree of a weighted connected Graph 'G' is a spanning Tree with minimum or smallest weight.

Example





Spanning Tree — (11) with weight



Spanning Tree → 21

Application of Spanning Tree

1. Designing efficient routing Algorithm
2. N/W Design, N/W Topology

Topic Unit No.

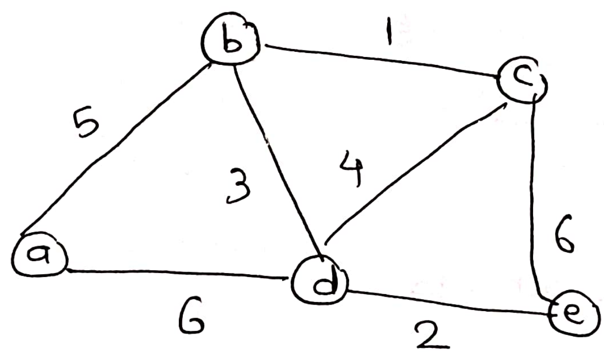
* MST :- Prim's Algorithm

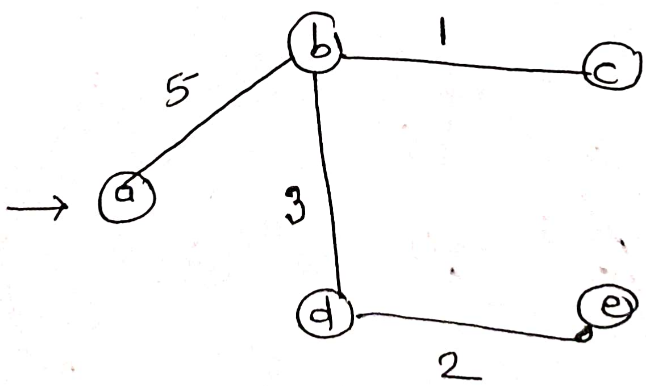
① The Tree starts from an orbitory root vertex 'r' and grows until the tree spans all the vertex in V.

② Select an edge with minimum weight

③ Keep selecting adjacent edges with minimum weight.

Example :-

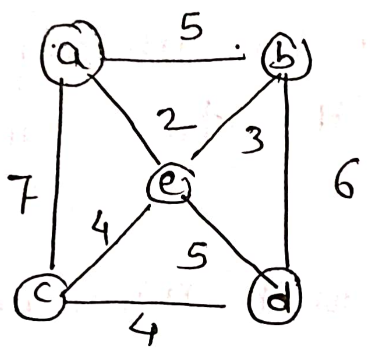




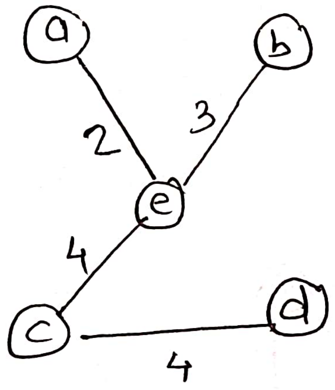
MSP : weight = 11

Ex - 2

Ex

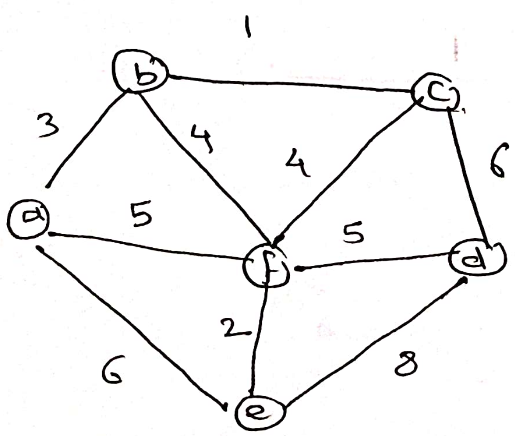


⇒

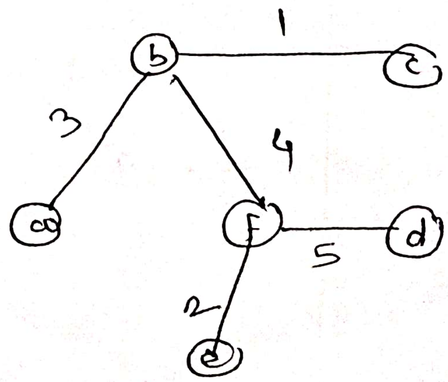


MST weight = 13

Ex - 3



⇒



weight = 15

Sanjivani Rural Education Society's

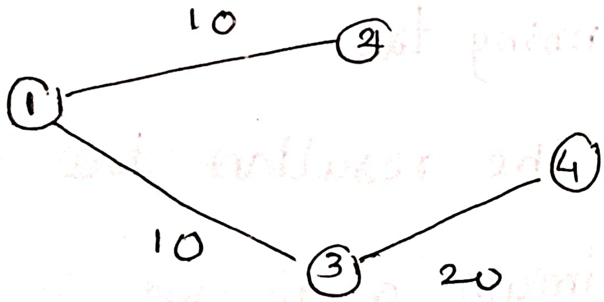
SANJIVANI COLLEGE OF ENGINEERING, KOPARGAON

DEPARTMENT OF COMPUTER ENGINEERING

Topic **Unit No.**

Ex-4) Find weight of the MST for the following Matrix

$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & 10 & 10 & 50 \\
2 & 10 & 0 & 40 & 30 \\
3 & 10 & 40 & 0 & 20 \\
4 & 50 & 30 & 20 & 0
\end{array}
$$



MST weight = 40

Prims Algo	Kruskal's Algo
1) Select Adjacent vertices of the Already Selected Vertices	2) Not necessary to select the adjacent vertices of already vertices.

Kruskal's Algorithm

This method used to convert a network ~~G~~ 'G' of n vertices to a MST 'T'. It is also known as greedy algorithm.

The constraints on using this algorithm

1. Every node in the network must be included in the spanning tree.
2. Cost of the resultant tree must be as much minimum as possible.

Algorithm

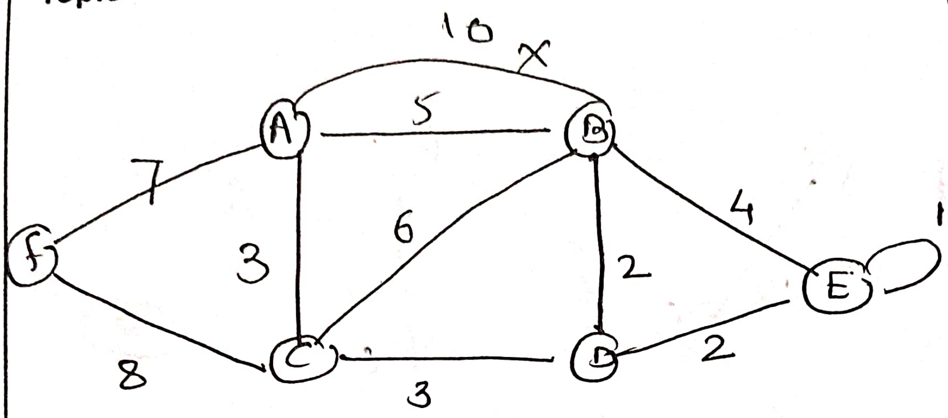
1. Remove all the loops & parallel ~~A~~ edges ~~ei~~ + from the graph.
2.

Sanjivani Rural Education Society's

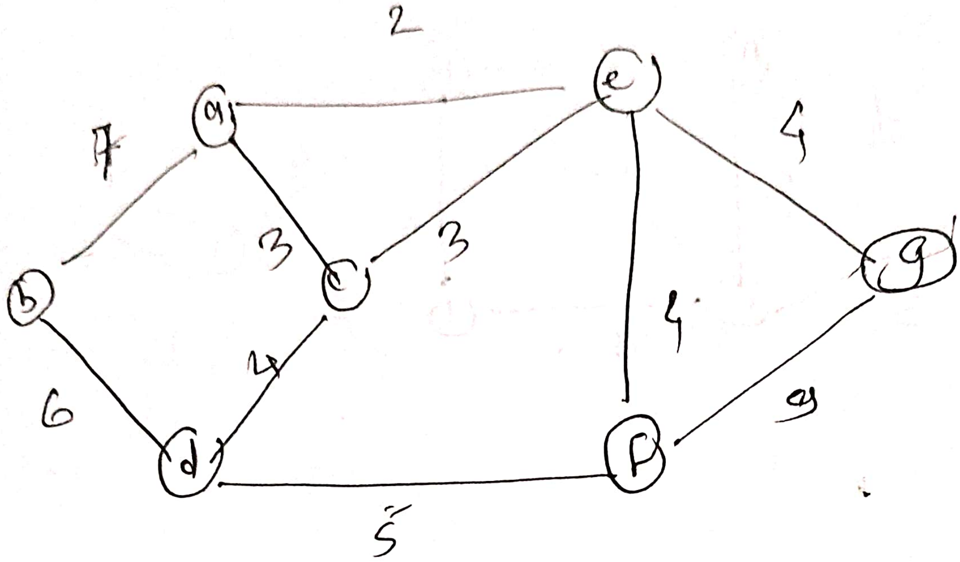
NJIVANI COLLEGE OF ENGINEERING, KOPARGAON

DEPARTMENT OF COMPUTER ENGINEERING

Topic **Unit No.**



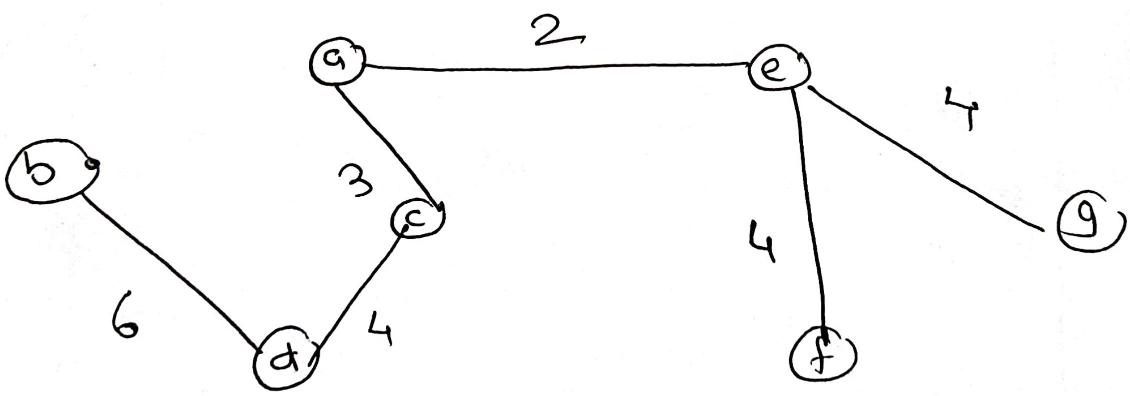
Q.



⇒ ① Arrange the edges in ascending order according to their weight

ae = 2, ac = 3, ce = 3, cd = 4, ef = 4, eg = 4, df = 5, ba = 7, fg = 9

② Select the edge which having minimum cost, here ae = 2



③ select next edge ~~is~~ which having minimum weight but does not from the cycle..

④ continue step ③ & ④ until all nodes are visited.

Dijkstra's Algorithm

- Developed by Edsger Dijkstra in 1959
- Generalization of BFS Algorithm and used to Find shortest path tree.
- Find shortest path from source node to all the vertices in the graph
- Single source ~~shortest~~ shortest path is find. [To find a shortest path from a given source vertex $s \in V$ to every vertex $v \in V$]
- It uses greedy approach.
- Does not work with -ve weights (Bellman Ford Algorithm will work for -ve weight)
- It works for Directed & undirected graph.

Algorithm

① Initialize weights, source Node = 0 & other nodes = ∞ (or -1)

② For current node, consider all unvisited nodes and calculate the distance. and Finally assigned smallest value.

> if dist [u] + cost [u,v] < dist [v] then
>
> dist [v] = dist [u] + cost [u,v]

u = 1, v = 2, cost [u,v] = 2

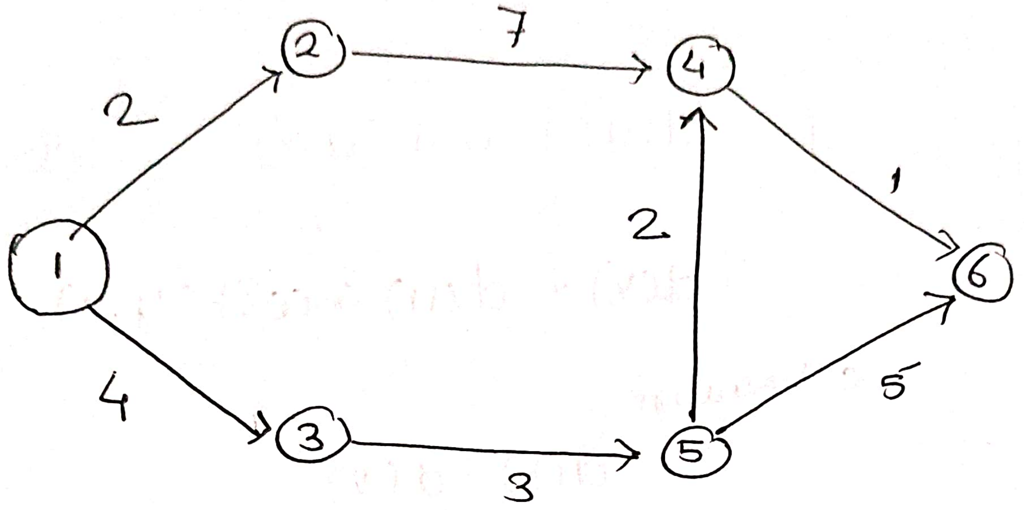
dist [u] = 0, dist [v] = ∞

if 0 + 2 < ∞ then

$\boxed{dist[v] = 0 + 2 = 2}$

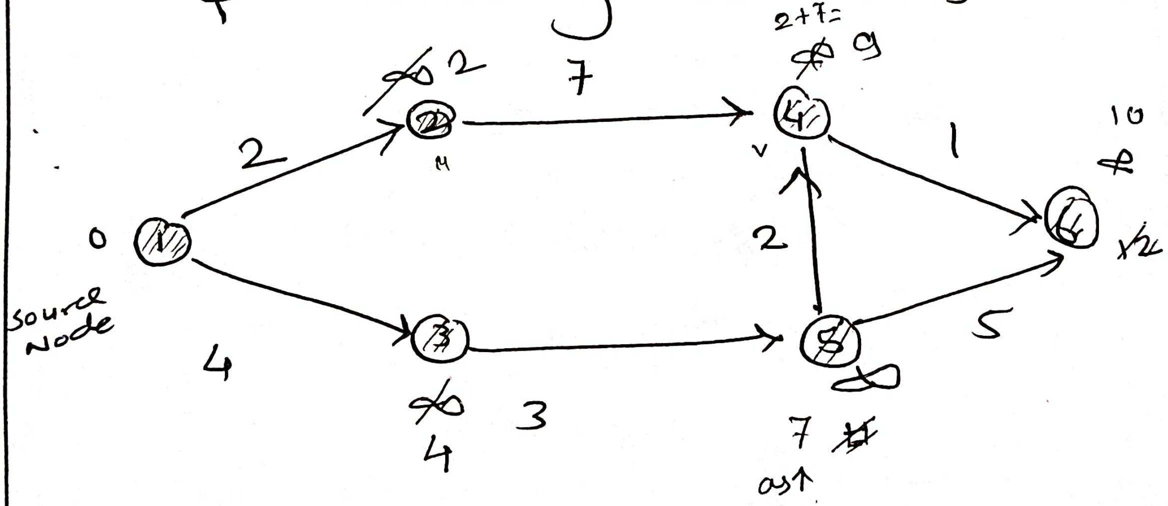
③ When all neighbors are considered current node is marked visited and never explored again.

④ stop when all nodes are visited



Step 1 initialized ~~10 = 0~~ Source Node = 0

& remaining all nodes = ∞



Step 2 :- For Current node, mark it visited & consider all unvisited node [adjacent] from that Vertex, & calculate distance consider minimum distance..

For Node 2 $d(u) = 0$, $d\,c(u+v) = 2$, $d(v) = \infty$

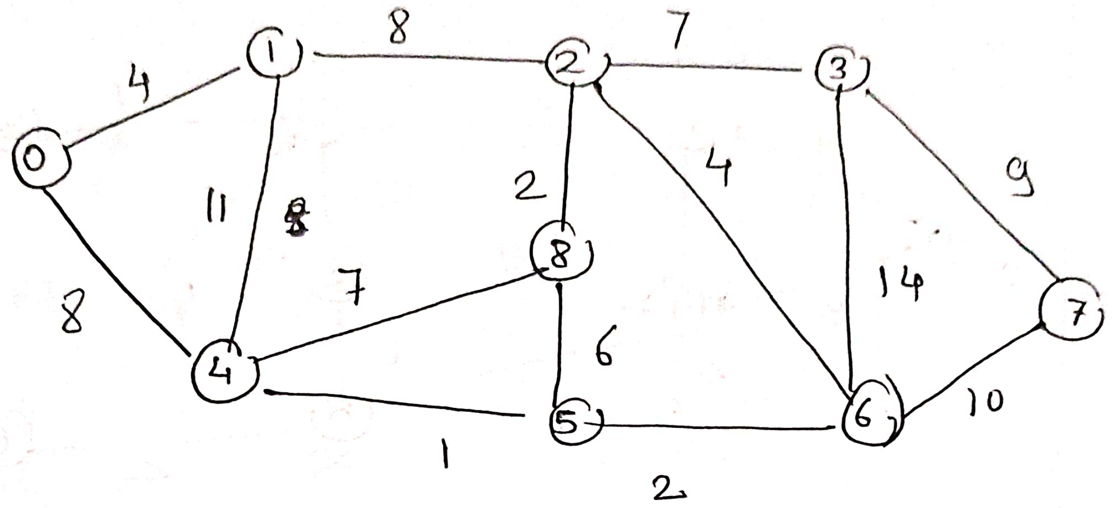
if $d(u) + cost(u,v) < d(v)$ then

$$d(v) = d(u) + cost(u,v)$$

otherwise

$$d(v) = d(v).$$

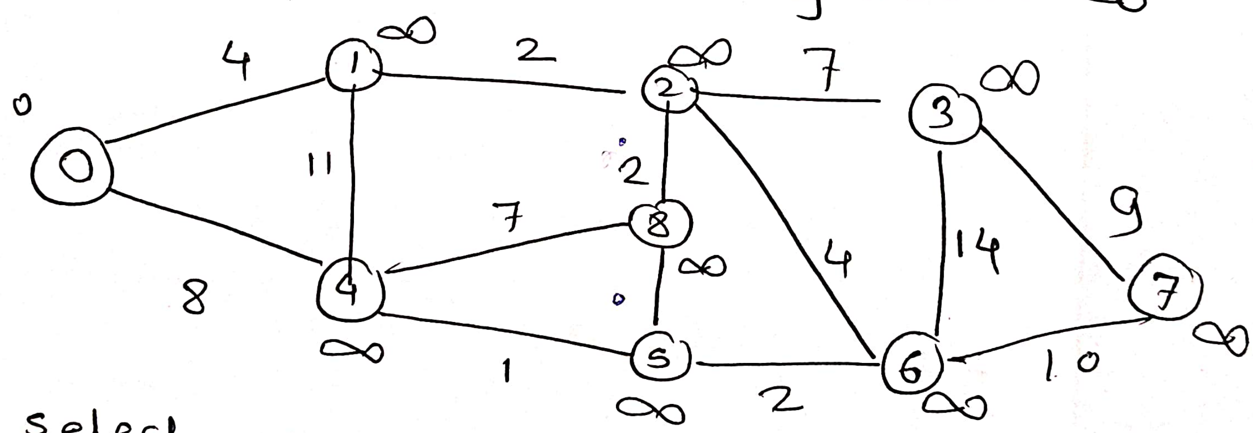
Ex - 2



⇒ step - 1

Assign source Node = 0 &

All other Remaining Node = ∞



select source vertex as '0'

∴ $u = 0, \quad v = 1$

∴ $dist(u) = 0, \quad cost(u, v) = 4, \quad dist(v) = \infty$

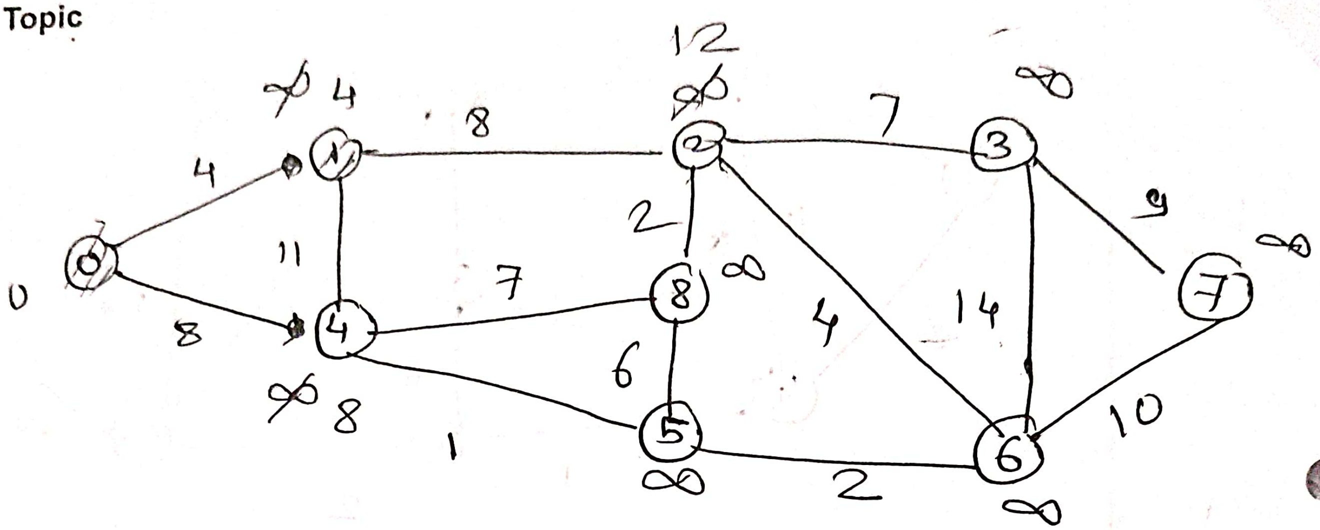
∴ if $dist(u) + cost(u, v) < dist(v)$ then

$$\boxed{dist(v) = dist(u) + cost(u, v)}$$

Sanjivani Rural Education Society's
SANJIVANI COLLEGE OF ENGINEERING, KOPARGA
DEPARTMENT OF COMPUTER ENGINEERING

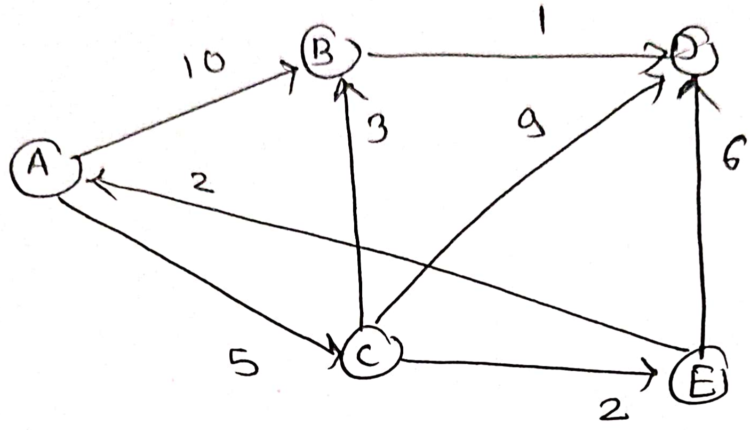
Unit No.

Topic



select Minimum dist among all vertex

Ex ②



(as source vertex is A (written first))

	A	B	C	D	E
A	0	∞	∞	∞	∞

(select minimum dist)

Now By Applying formula

if $d(u) + cost(u,v) < d(v)$ then

$$d(v) = d(u) + cost(u,v)$$

Now ~~By~~, As there is directed edge to B & C from A,

	A	B	C	D	E
A	0	∞	∞	∞	∞
		10	$\boxed{5}$	∞	∞

~~As~~

Now select minimum distance vertex i.e. C

	A	B	C	D	E
A	$\boxed{0}$ ~~∞~~	∞	∞	∞	∞
C		10	$\boxed{5}$	∞	∞

Now, from C, edge CB, CD, CE, by using formula, update weight (cost)

	A	B	C	D	E
A	$\boxed{0}$	∞	∞	∞	∞
C		8	$\boxed{5}$	14	7
E		8		14	$\boxed{7}$

	A	B	C	D	E
A	0	∞	∞	∞	∞
C		8	$\boxed{5}$	14	7
E		$\boxed{8}$		13	

	A	B	C	D	E	
A	$\boxed{0}$	∞	∞	∞	∞	
C		10	$\boxed{5}$	∞	∞	← C
E		8		14	$\boxed{7}$	← E as dst no changed
B		$\boxed{8}$		13		← B
D				$\boxed{9}$		← D

Find Path to A to D

~~A~~ D, B, C, A

$\boxed{\text{D B C A}}$

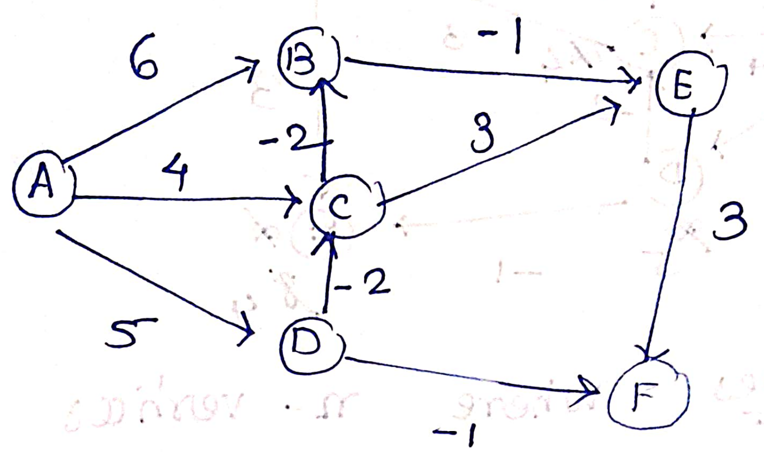
Shortest path is A C B D and distance is 9.

BCA
ACD

To Find shortest path select Node let us say D, check it's column value if it is changed, then select selected vertex of row, & so backwards continue the same procedure until we reached to source vertex.

Bellman Ford Algorithm -

(Single Source shortest Path)



step 1

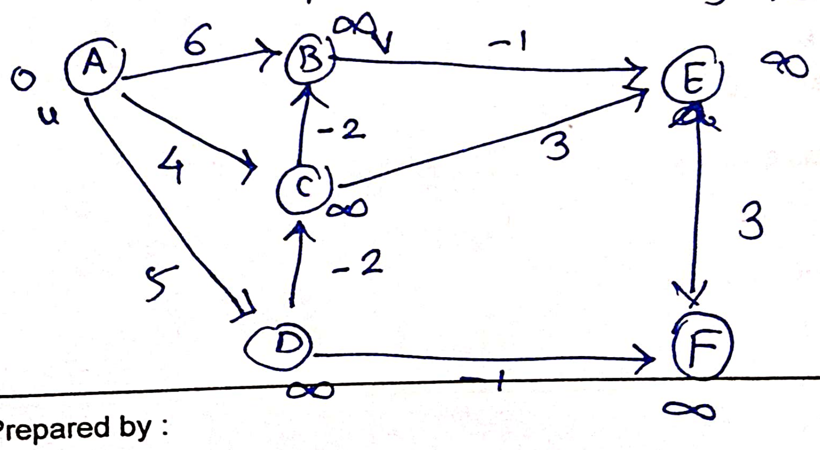
Write all edges

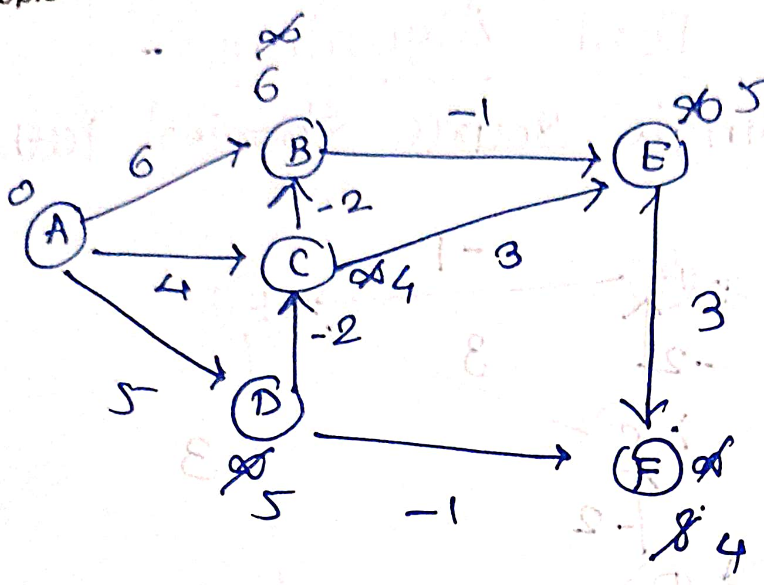
(A,B) (A,C) (A,D) (C,B), (D,C),

(C,E), (B,E) (E,F) (D,F)

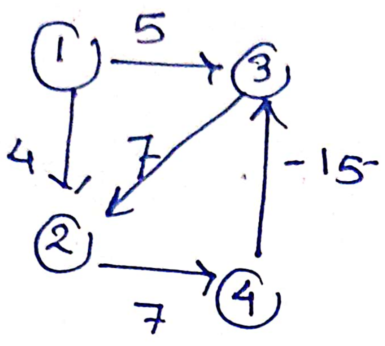
step 2 initilize source vertex dist = 0

& all remaining vertex dist = ∞



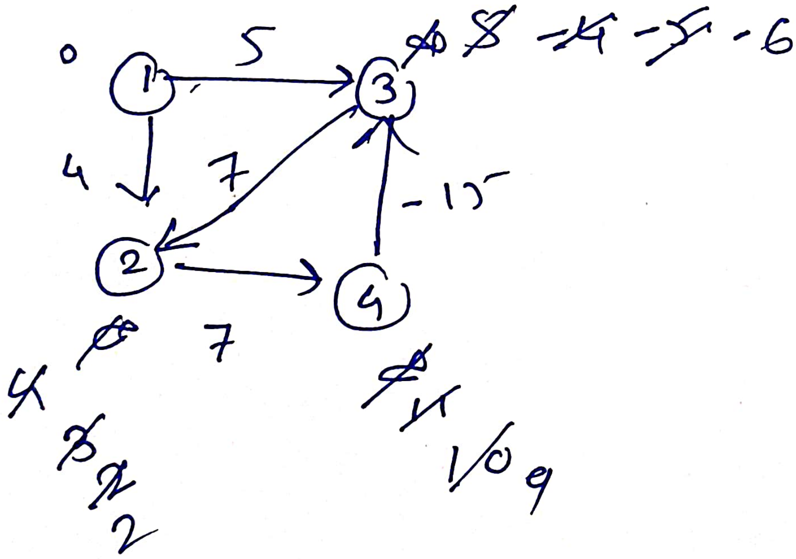


(n-1) times where n - vertices
many times we need to update the distance



Cycle sum is is -ve then Bellman ford will not give Correct destance or shortest path.

(1,3) (1,2) (2,4) (4,3) (3,2)



where n is vertices

After Applying (n-1) iteration still weight is changing means Graph Consist of negative cycle