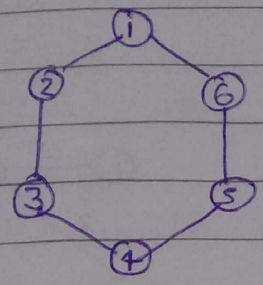
unit 4

Graph Algorithm : Elementary Graph Algorithm.

Page No.
Date

- **minimum cost spanning tree :**



$G = (V, E)$

$|V| = \{1, 2, 3, 4, 5, 6\} = 6$

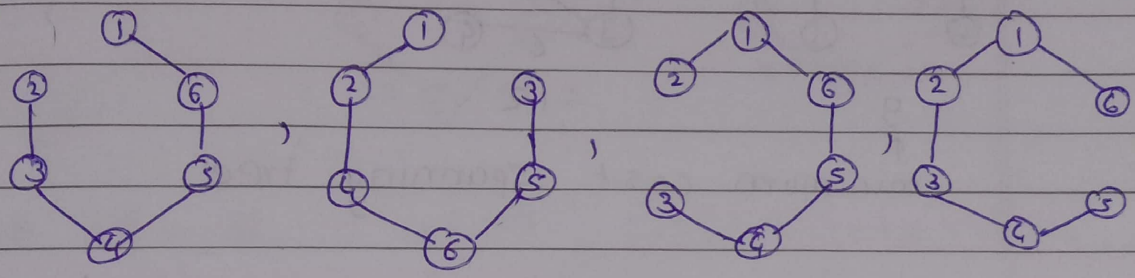
$|E| = \{(1,2), (2,3), (3,4), (4,5), (5,6), (6,1)\} = 6$

spanning tree - subgraph of given graph

$|V|_C \Rightarrow {}^6C_5 \Rightarrow 6$ possibilities of spanning tree.

$E = |V| - 1$

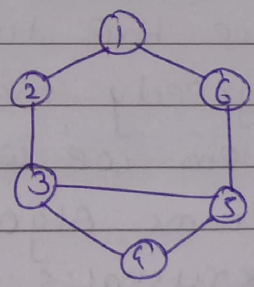
e.g.



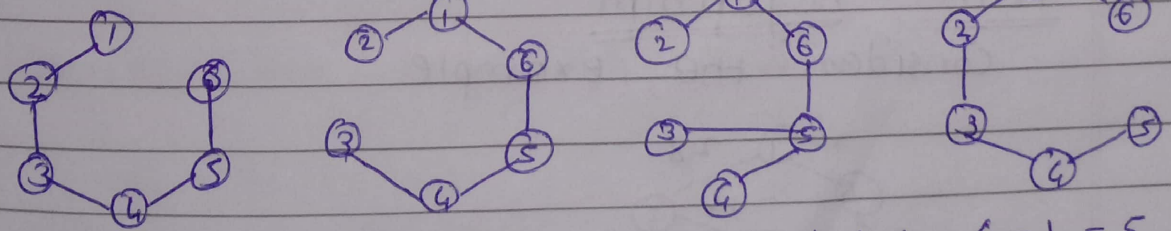
$G = (V', E')$

$V' = |V| \qquad E' = |V| - 1 = 6 - 1 = 5$

e.g. consider the graph



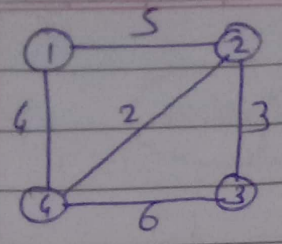
Spanning tree will be



here $|V| = 6$, $|E| = 7$ $\therefore |E| = |V| - 1 = 6 - 1 = 5$

modified formula : $|V|_C$ - no. of cycles $= {}^6C_5 - 2 = 6 - 2 = 4$

$|V| - 1$

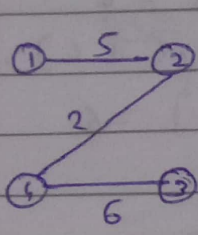


$|V| = 4$, $|E| = 5$, $|E| = |V| - 1$

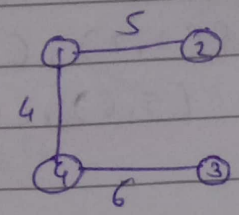
$= 4 - 1$

$= 3$

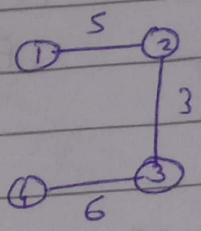
$^4C_3 = 4$



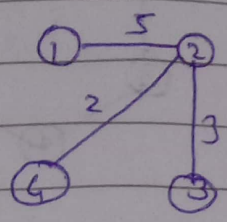
$= 15$



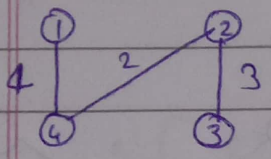
$= 15$



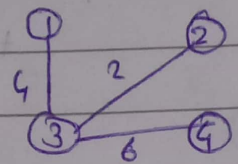
$= 14$



$= 10$



$= 9$



$= 12$

↑ minimum cost spanning tree

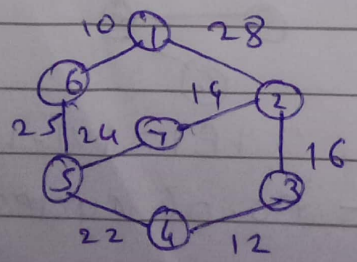
Against the given graph, considering a graph and determining all possible spanning trees and cost of each is time consuming process, so to reduce the time we preferably use greedy method. In greedy method to solve such a kind of problem we will have to use two methods

① Prims Algorithm

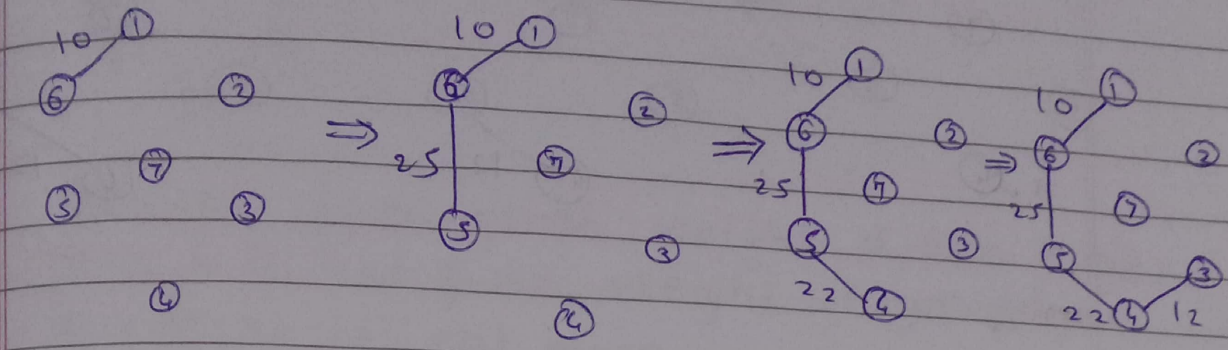
② Kruskal's Algorithm.

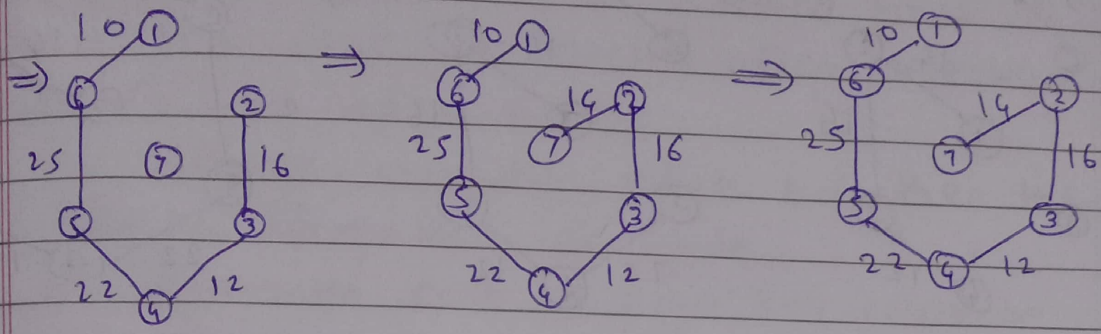
- **Prims Algorithm**

Consider the example.



choose the minimum cost edge and then find adjacent minimum cost edge.

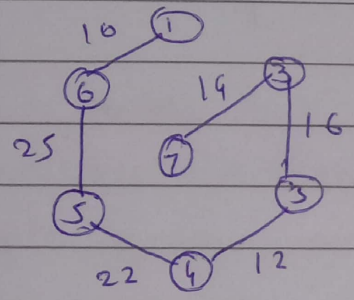




here $|E| = 8$ $|V| = 7$ $\therefore |E| = |V| - 1 = 7 - 1 = 6$

we get minimum cost spanning tree of 6 edges.

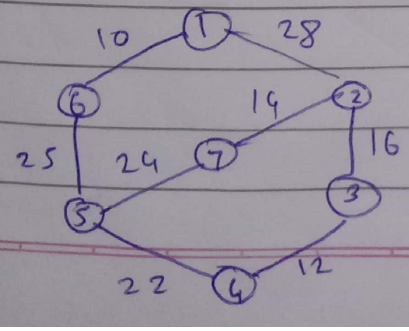
i.e.

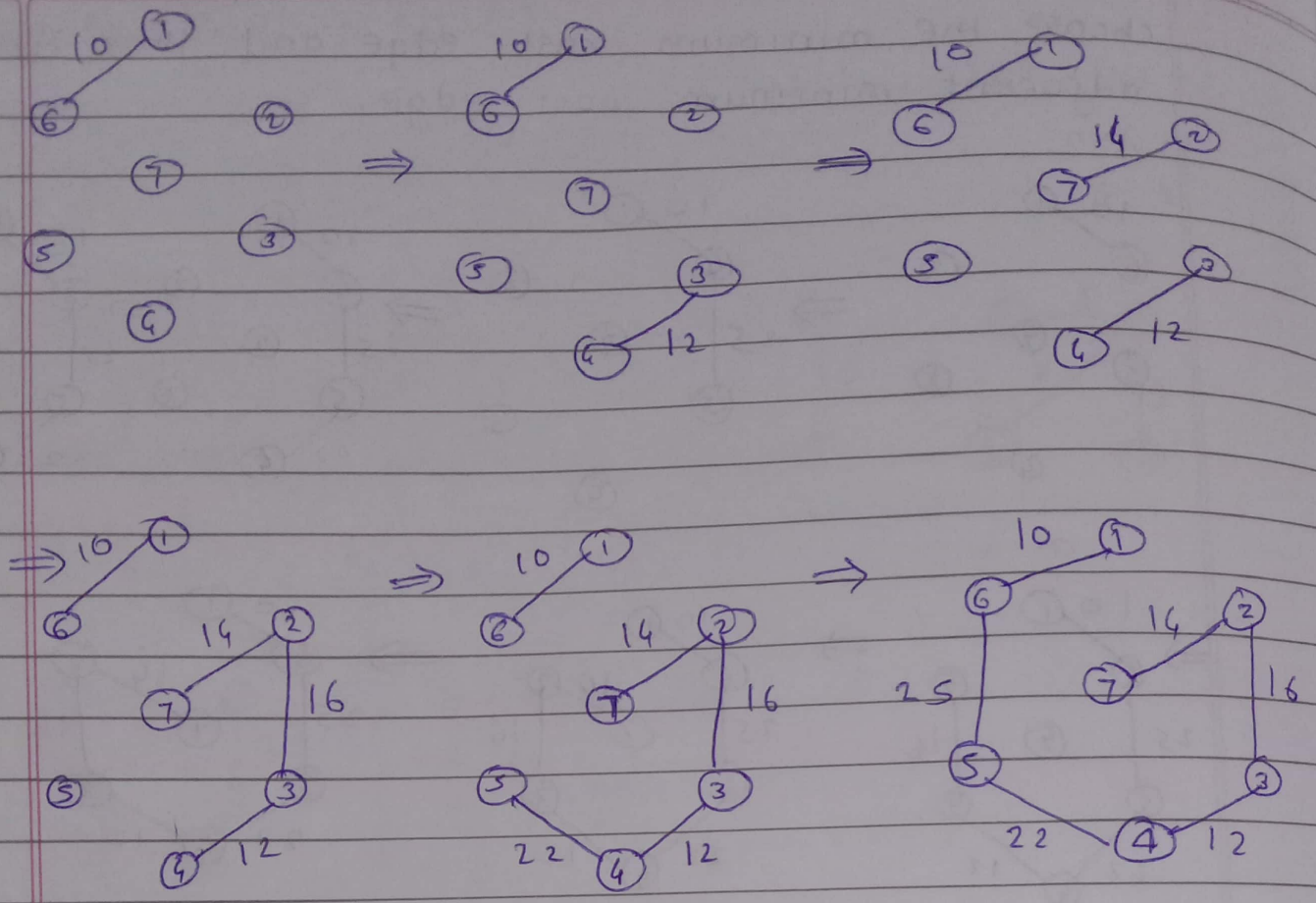


$\Rightarrow$ 99 is minimum cost.

- **Kruskals Algorithm** -

consider the same example here





we get the minimum cost = 99.

worst case time complexity.

$O(|V| \cdot |E|)$

$O(n \cdot (n-1))$

$O(n^2 - n)$

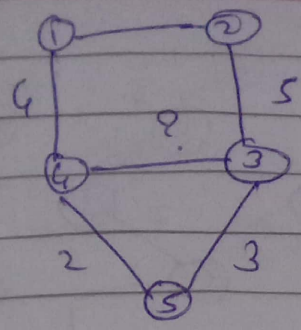
$O(n^2)$

Best case time complexity (min heap).

Suppose $f = \log n$

$O(|V| \cdot |E|)$

$O(n \log n)$



In the given graph weight of two edges (1-2, 4-3) not given, weight of two edges (1-2, 3-4) is not given.

Edge (3-4) is not selected first. it means that this edge is selected after selecting edge (4-5) and edge 3-5.

If we consider edge 3-4 then there could be formation of cycle.

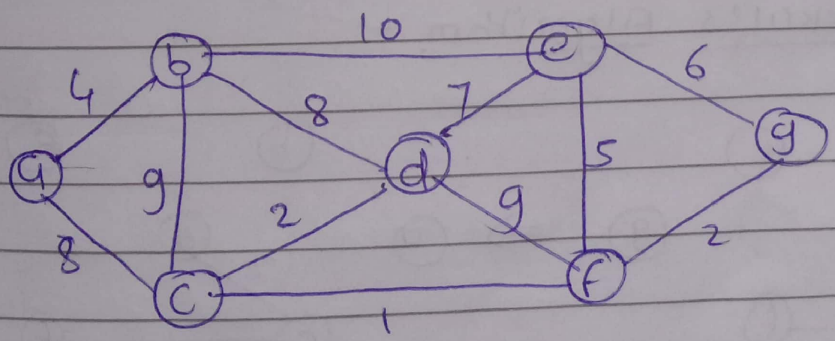
so minimum cost of edge 3-4 could be greater than 3.

so minimum cost of edge 3-4 = 4

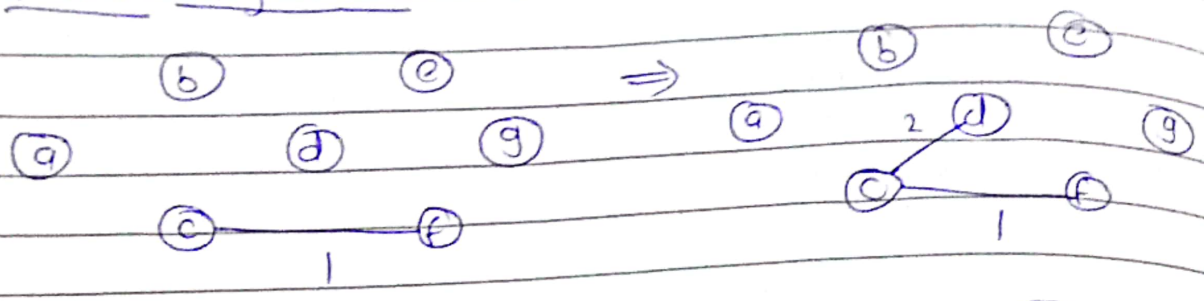
similarly cost of edges 1-2 = 6.

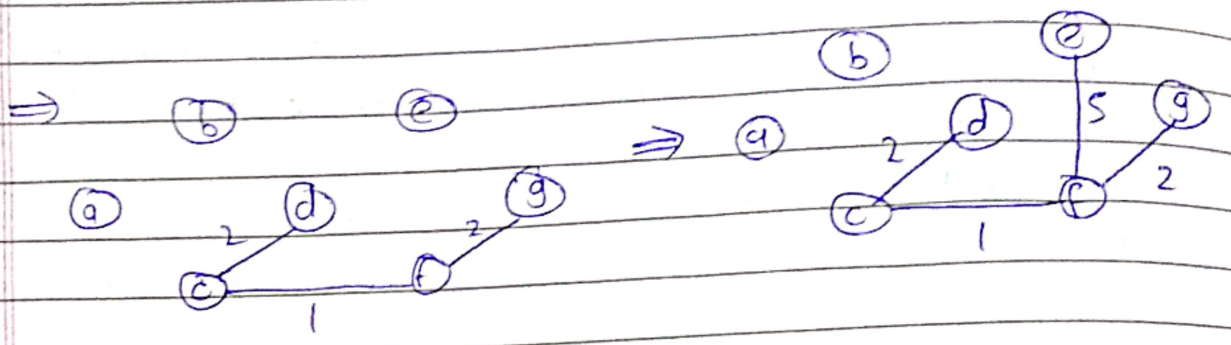
- Note -

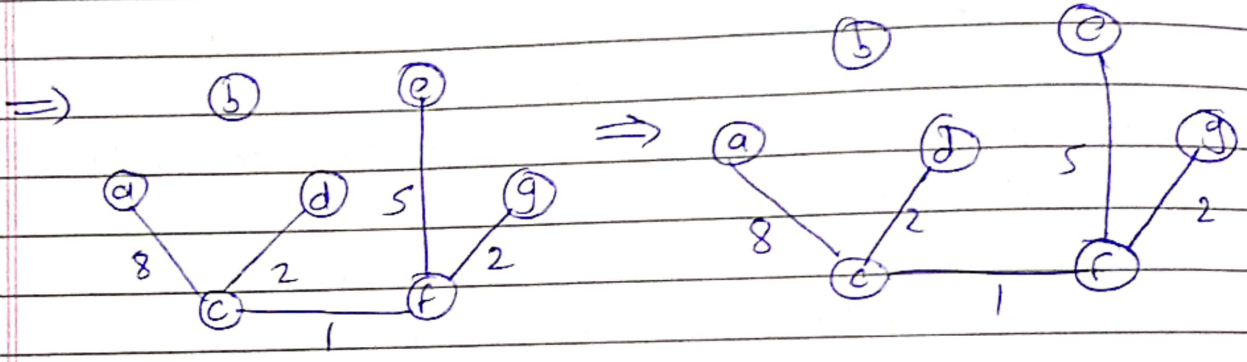
① Prims does not works on disconnected graph but, kruskal may work on disconnected graph.

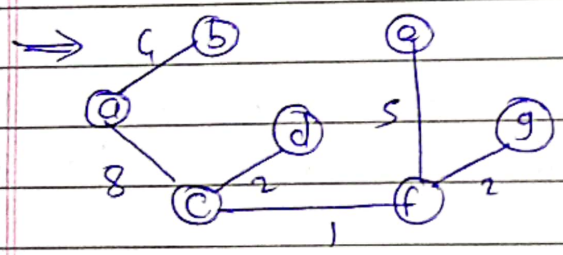


Prims algorithm -







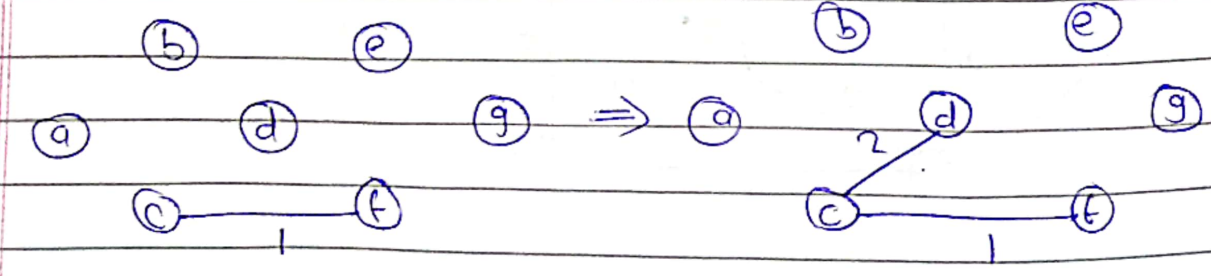


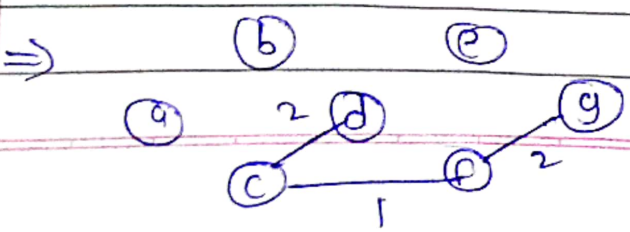
$|V| = 7$

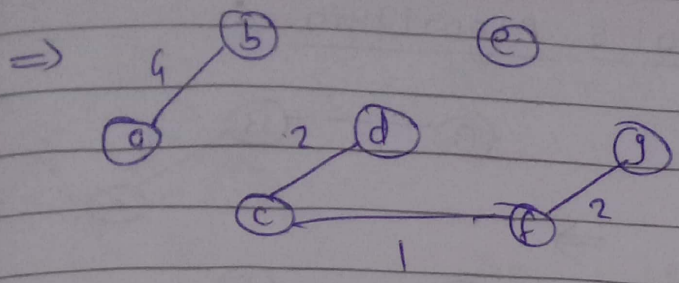
$|E| = |V| - 1 = 7 - 1 = 6$

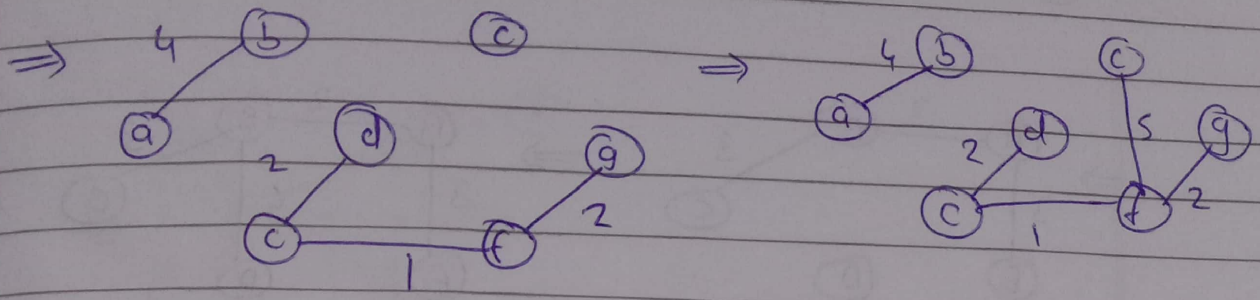
$\Rightarrow$ minimum cost = 22

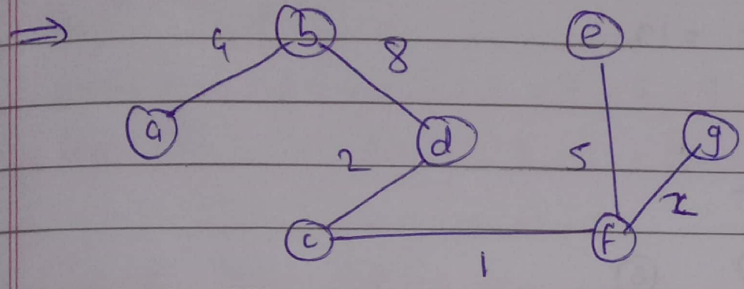
Apply Cruskal's Algorithm -

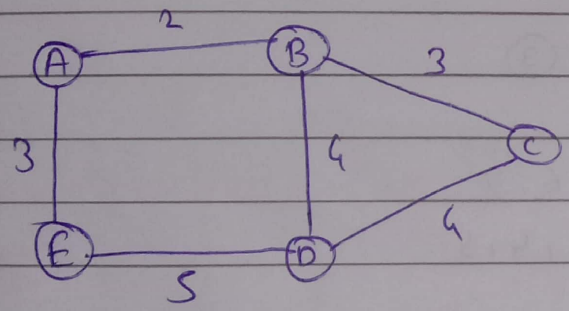




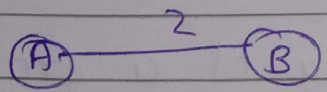


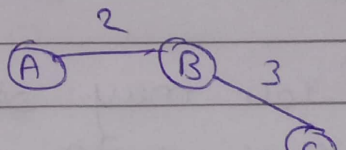


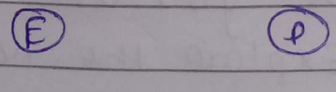


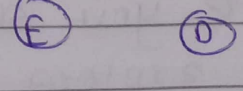
- 

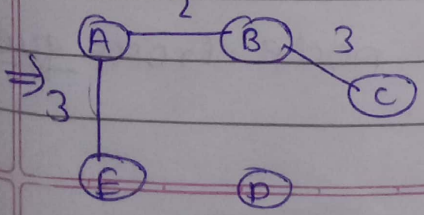
Applying prims algorithm -

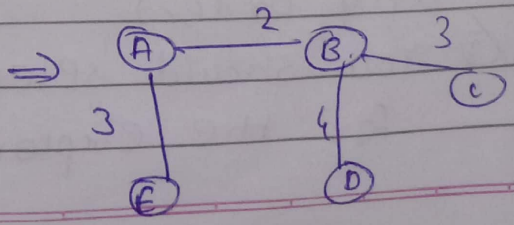




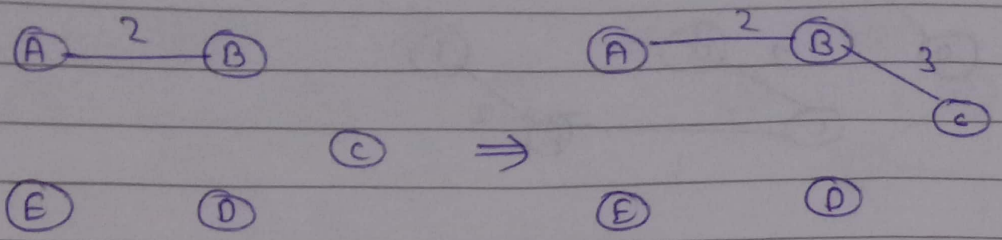


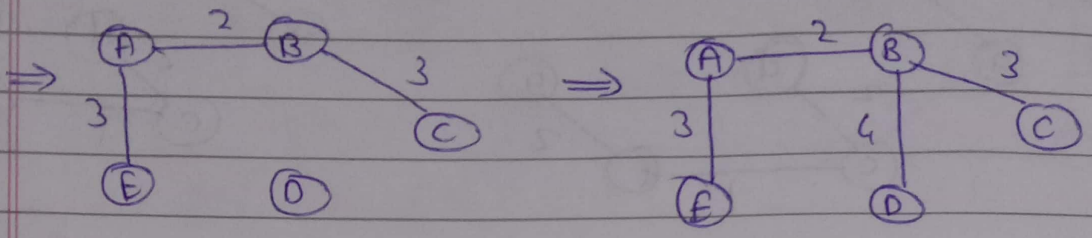






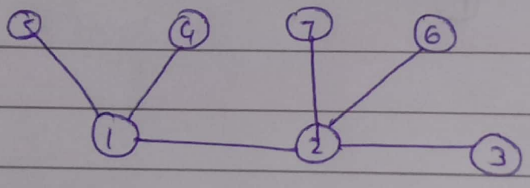
- Applying kruskal's Algorithm :-





minimum cost = 12.

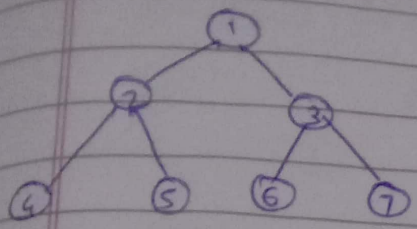
- BFS and DFS :-



BFS - 1, 2, 4, 5, 3, 6, 7
DFS - 1, 2, 3, 6, 7, 4, 5.

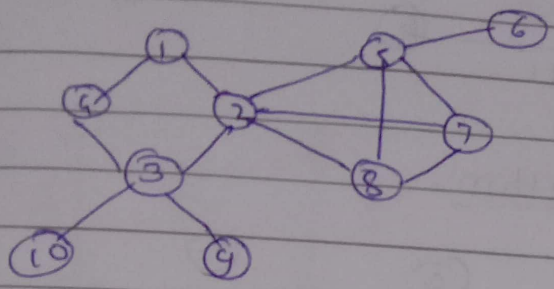
Rules for BFS -

① You may start traversing of a graph from any node.
② Once you select the node you are going to explore the node (Explore the node in any order).
③ You should select next node from queue for the exploration.



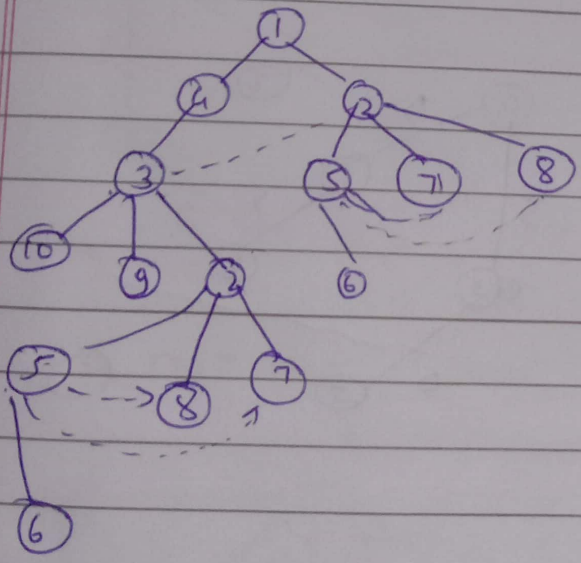
BFS - 1, 2, 3, 4, 5, 6, 7
DFS - 1, 2, 4, 5, 3, 6, 7.

BFS - level ordering.
DFS - preorder.

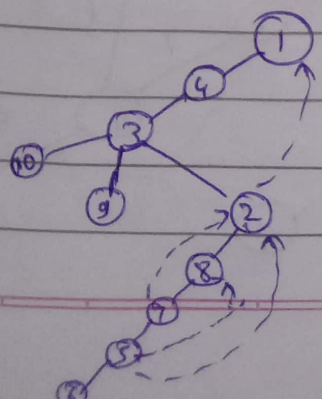


BFS - 1, 2, 4, 3, 5, 7, 8, 9, 10, 6
DFS - 1, 4, 3, 10, 9, 2, 5, 6, 7, 8

Tree for BFS

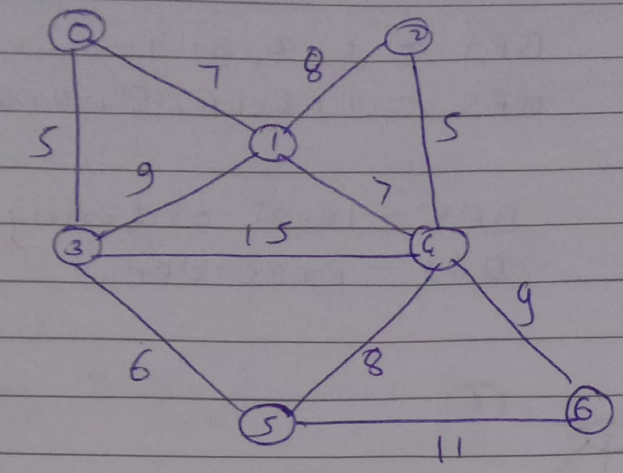


using DFS -



1, 4, 3, 10, 9, 2, 8, 7, 5, 6.

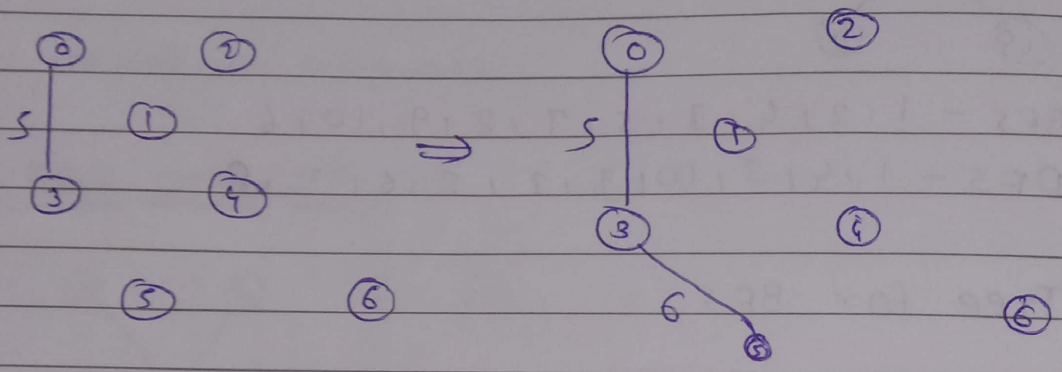
eg.

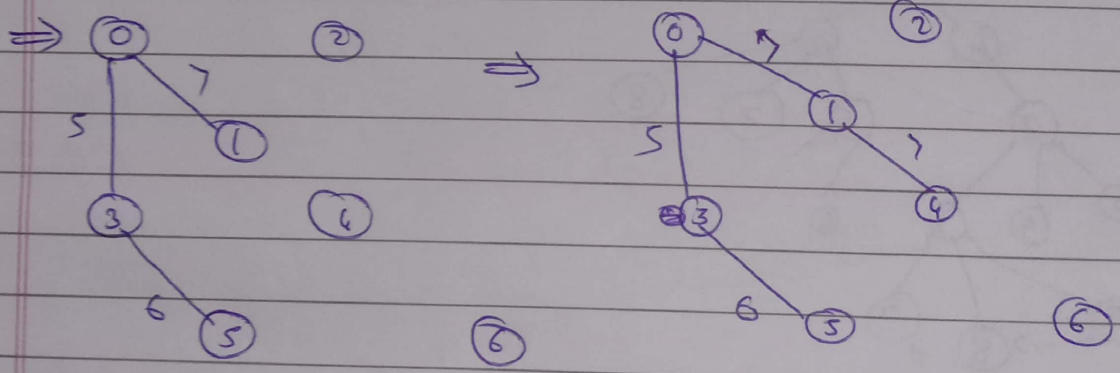


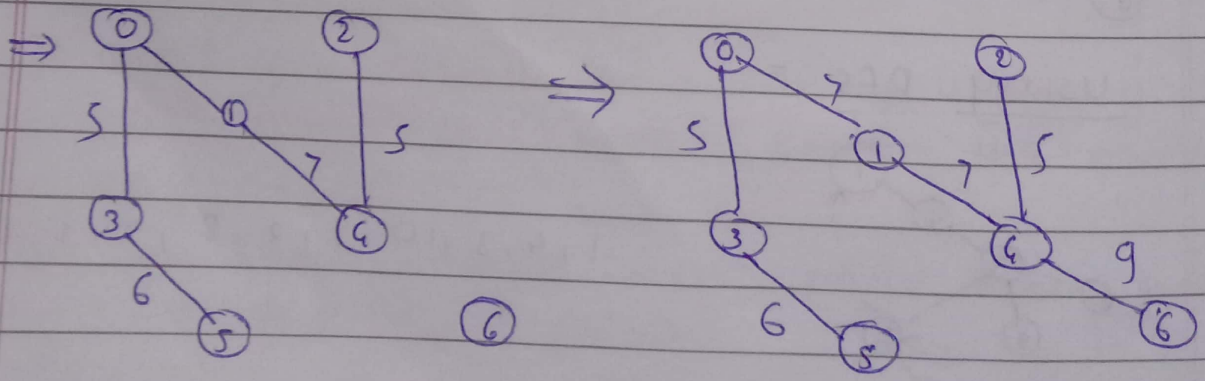
$|V| = 7$

$|E| = 6$

Applying Prims algorithm.

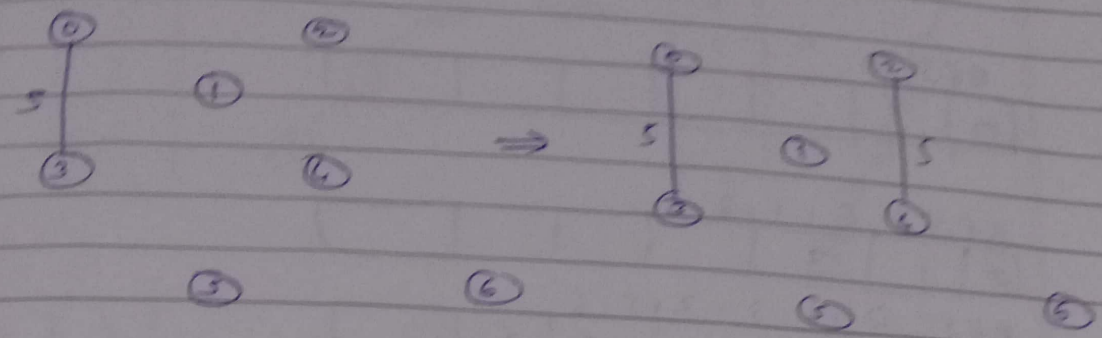


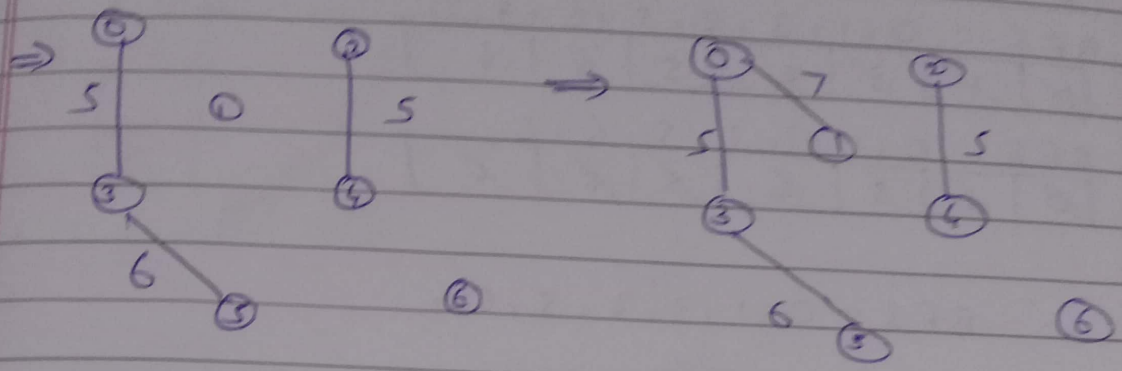


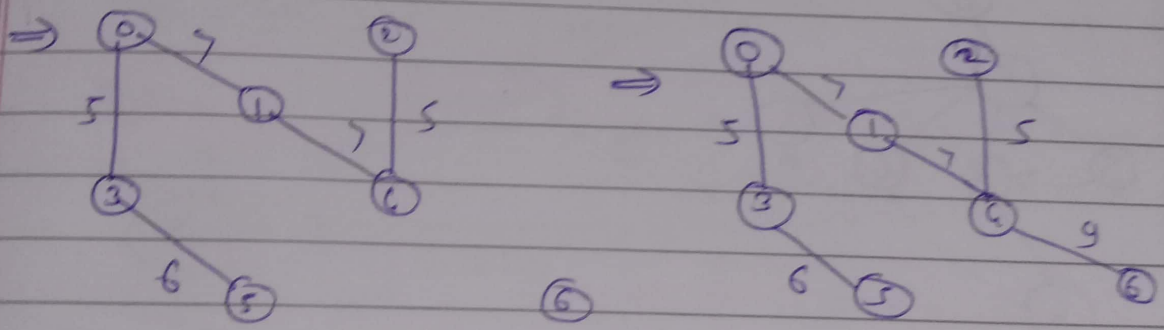


$\Rightarrow 39$

Applying Kruskal's Algorithm -

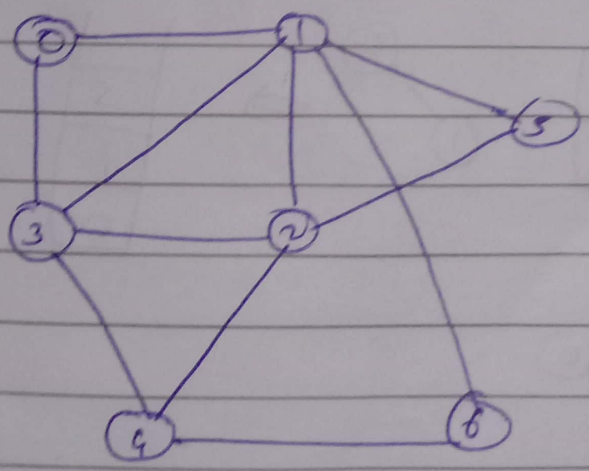






minimum cost ⇒ 39.

eg. •



0, 1, 3, 2, 5, 6, 4

BFS 0:

0						

BFS : 0, 1, 3

∅	1	3				

BFS : 0, 1, 3, 2, 5, 6

∅	X	3	2	5	6	

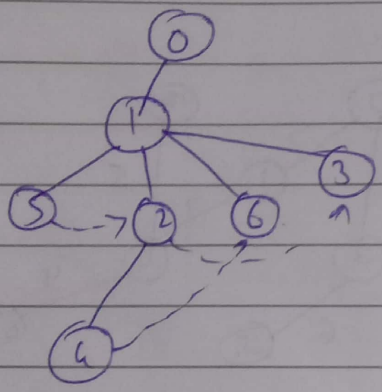
BFS : 0, 1, 3, 2, 5, 6, 4

∅	X	X	2	5	6	4

BFS : 0, 1, 3, 2, 5, 6, 4

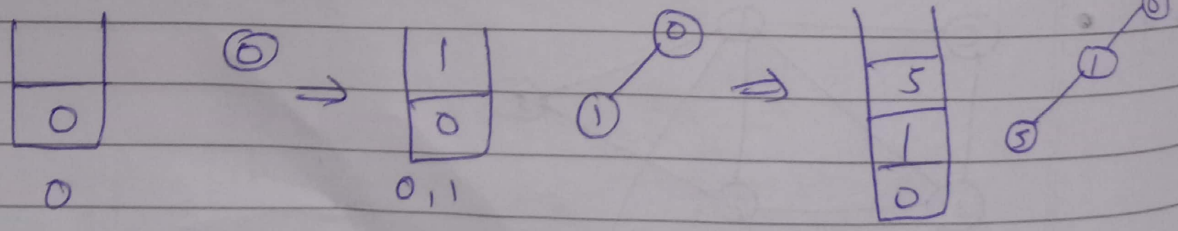
∅	X	X	X	X	X	X

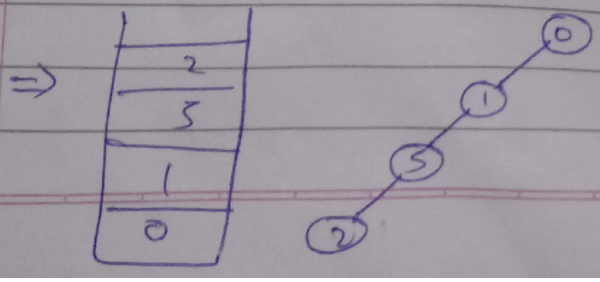
∴ BFS = 0, 1, 3, 2, 5, 6, 4

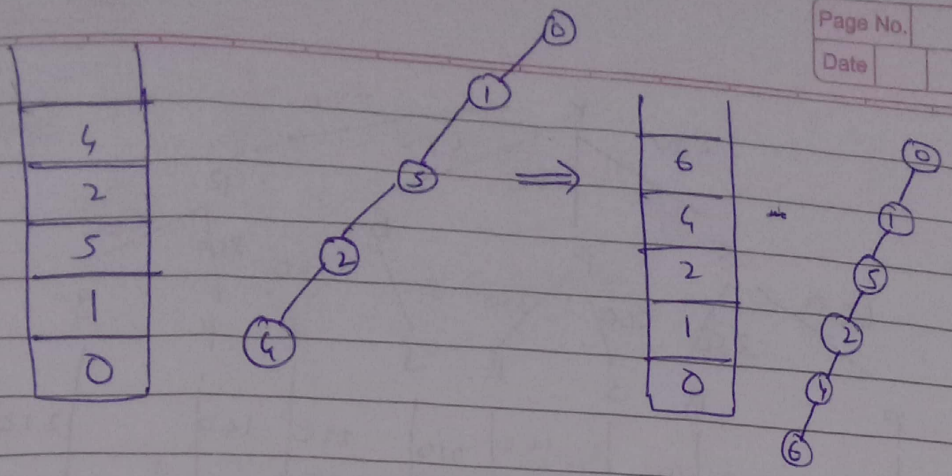


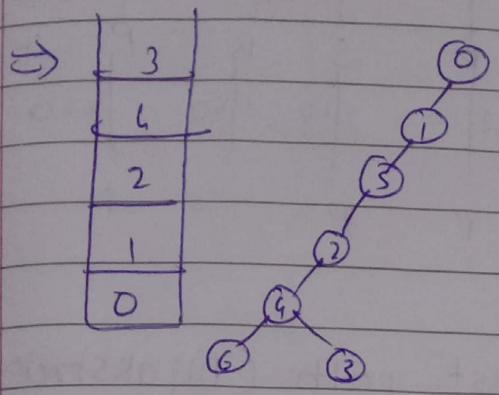
DFS :

0, 1, 5, 2, 4, 6, 3

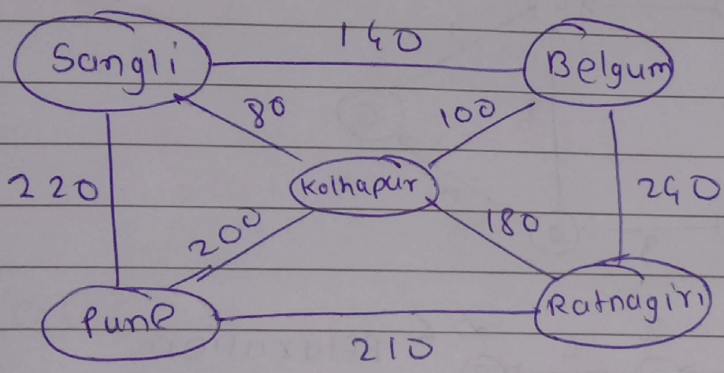




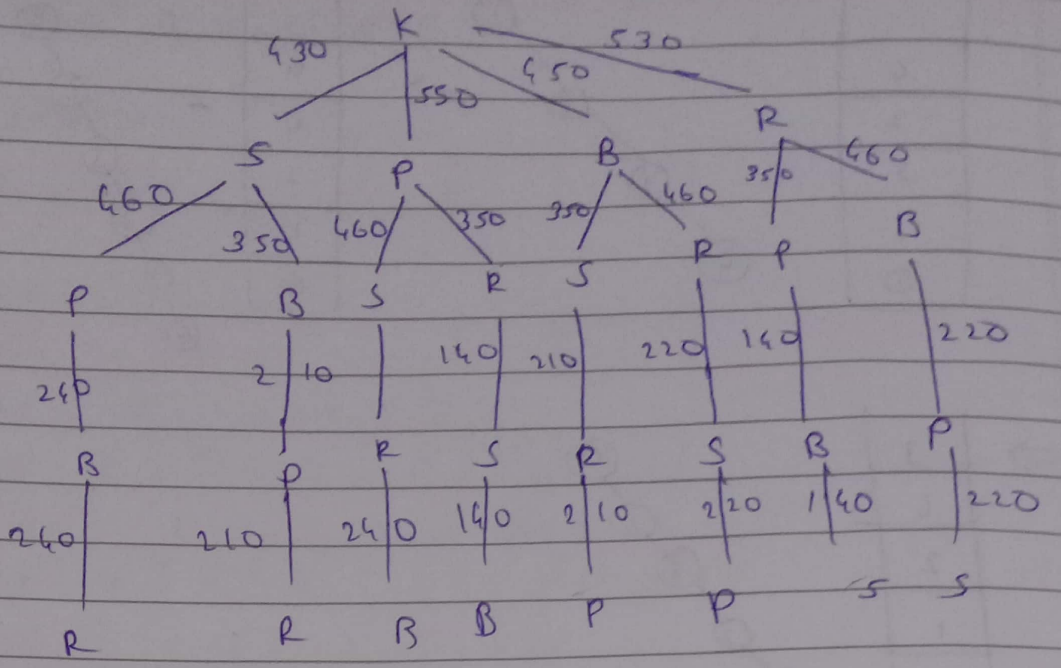




Eg.

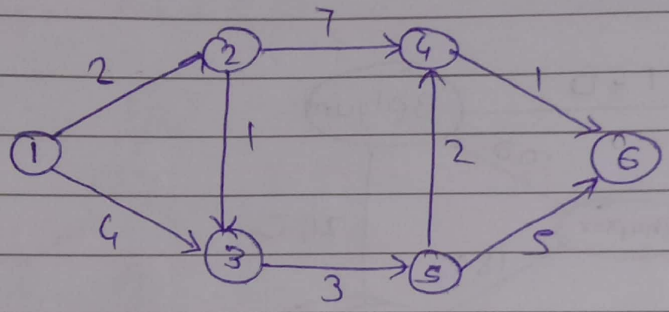


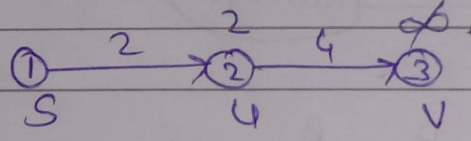
	S	P	K	B	R
S	0	220	80	140	0
P	220	0	200	0	210
K	80	200	0	100	180
B	140	0	100	0	240
R	0	210	180	240	0



- Single source shortest path (Dijakstra's Algo)

eg. ①





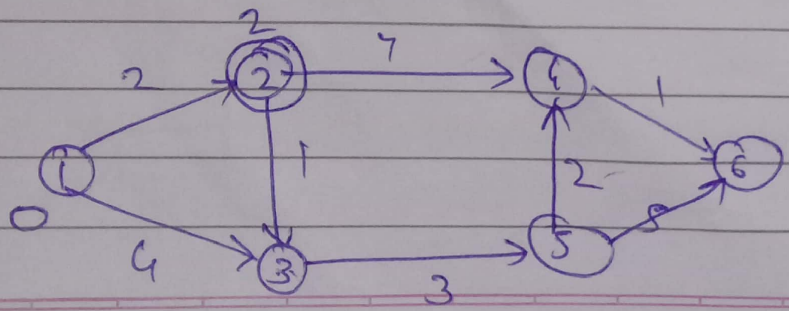
Relaxation

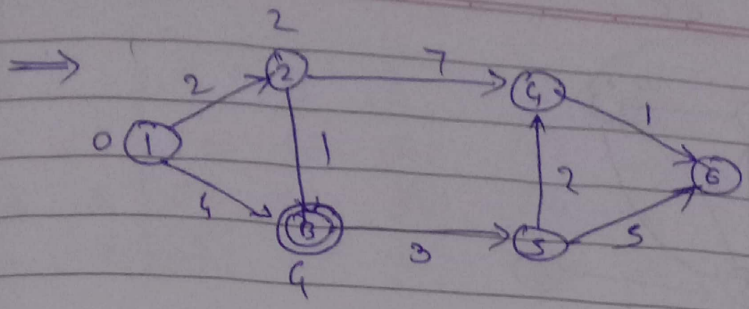
if (d[u] + c[u, v] ≤ d[v])

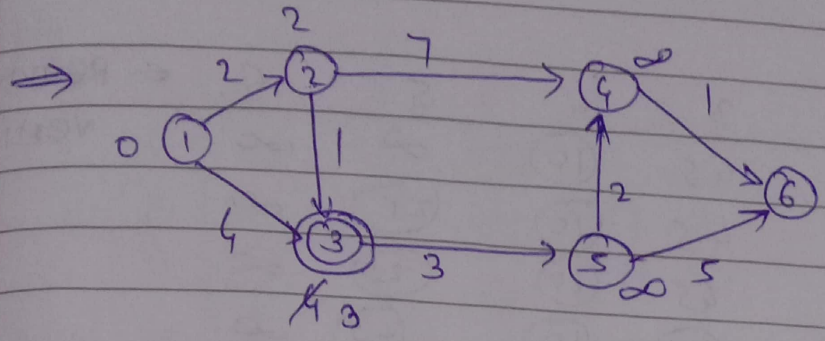
2 + 4 = 6 ≤ ∞

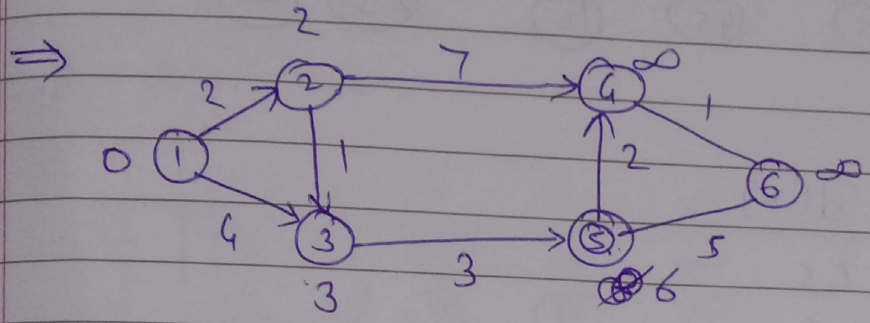
d[v] = d(u) + c[u, v].

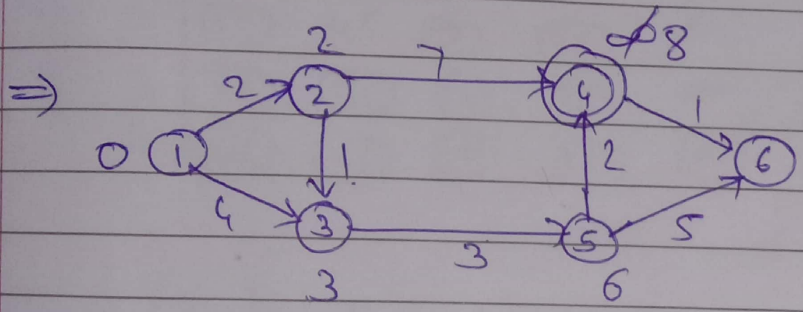
→

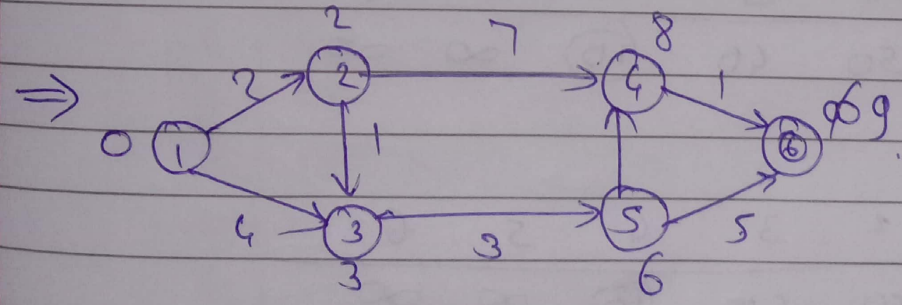






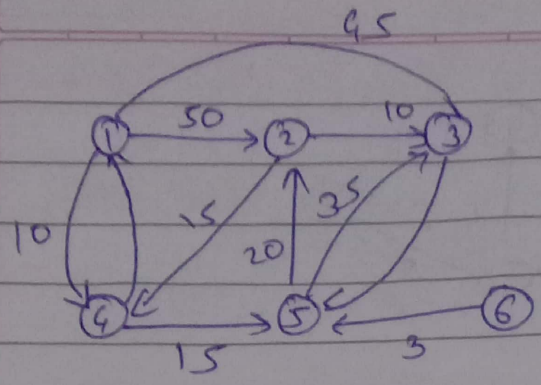






v	d[v]
2	2
3	3
4	8
5	6
6	9

eg. ②



→

	2	3	4	5	6	← Remaining vertices
4	50	45	(10)	∞	∞	
selected vertices 5	50	45	(10)	(25)	∞	
2	(45)	45	(10)	(25)	∞	
3	(45)	(45)	(10)	(25)	∞	
6	(45)	(45)	(10)	(25)	(∞)	

V	d[V]
2	45
3	45
4	10
5	25
6	∞

step ①

	2	3	4	5	6
4	50	40	(10)	∞	∞

step ②

	2	3	4	5	6
4	50	45	(10)	∞	∞
5	50	45	(10)	(25)	∞

step ①

	2	3	4	5	6
4	50	45	(10)	∞	∞
5	50	45	(10)	(25)	∞
2	(45)	45	(10)	(25)	∞

step ④

	2	3	4	5	6
4	50	45	(10)	∞	∞
5	50	45	(10)	(25)	∞
2	(45)	45	(10)	(25)	∞
3	(45)	(45)	(10)	(25)	∞

step ⑤

	2	3	4	5	6
4	50	45	(10)	∞	∞
5	50	45	(10)	(25)	∞
2	(45)	45	(10)	(25)	∞
3	(45)	(45)	(10)	(25)	∞
6	(45)	(45)	(10)	(25)	(∞)

V	d[V]
2	45
3	45
4	10
5	25
6	∞

- Time complexity of sortest path -

With this concept you will determine shortest path to the all the vertices. consider V=n now you need to determine shortest path using relaxation.

So total how many vertices going to be relax it depends on the graph.

Atmost how many vertices going to be relax is equal to all m edges.

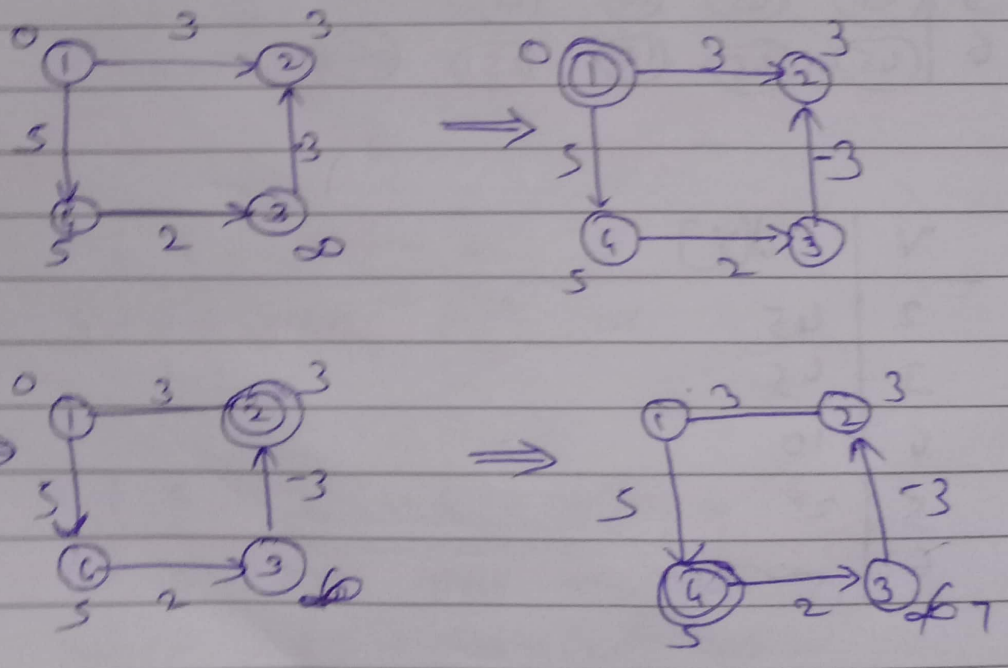
There is edge present betⁿ all pair of vertices.

so time complexity = $O(V \cdot E)$

$O(n \cdot n)$

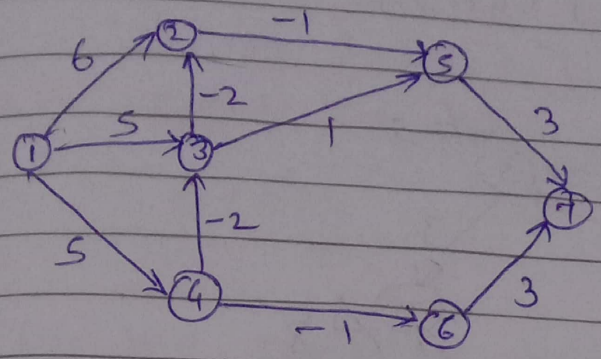
$O(n^2)$

- limitations on single ~~step~~ ~~sh~~ source shortest path -

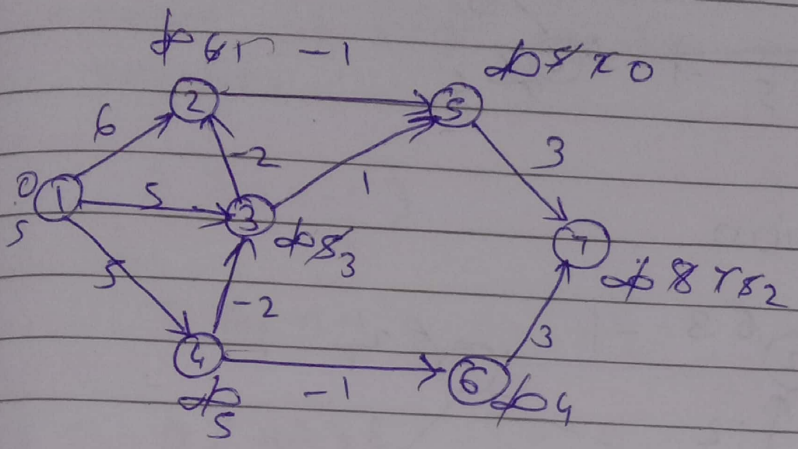


- Dijakstra's algo sometime work on negative weight & sometime it may not work.

Bellman Ford Algorithm :-



→



if [(d[u] + c[u,v]) ≤ d[v])

d[v] = d[u] + c[u,v]

|V| = 7

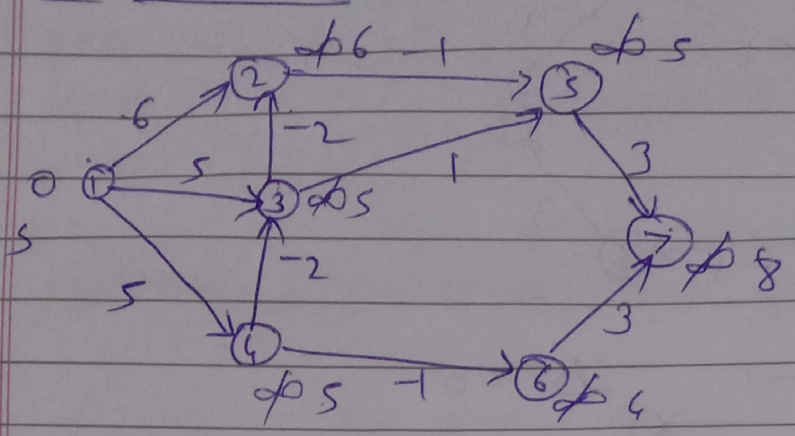
No. of relaxation on edges = |V| - 1 = 6.

E = {(1,2) (1,3) (1,4) (2,5) (3,2) (3,5) (4,3) (4,6) (5,7) (6,7)}

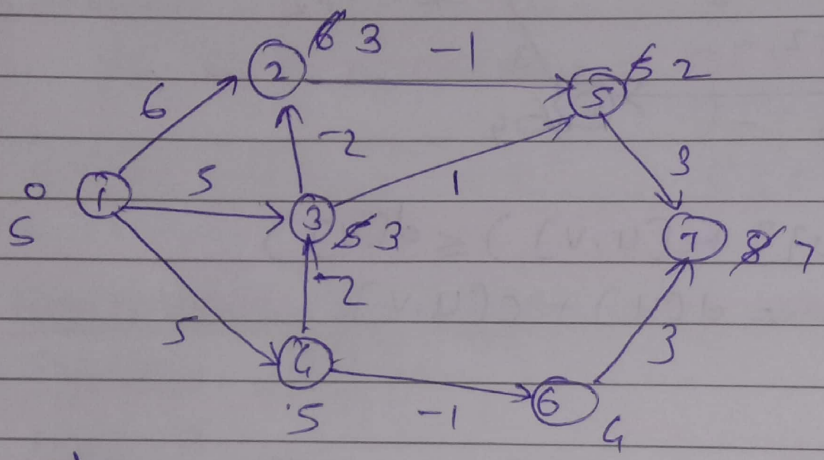
V	d[V]
2	1
3	3
4	5
5	0
6	4
7	3

⇒ E = {(1,2), (1,3) (1,4) (2,5) (3,2) (3,5) (4,3) (4,6) (5,7) (6,7)}

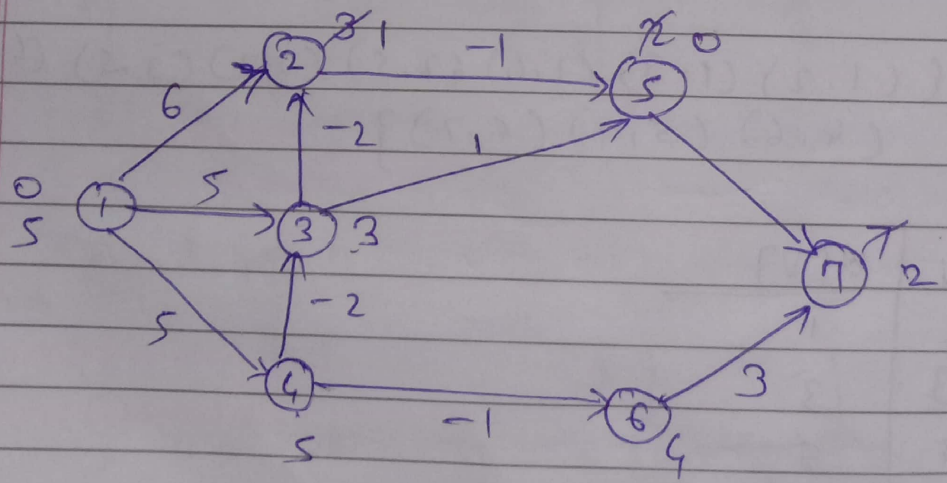
1st iteration



2nd iteration



3rd iteration.



after 3rd iteration no change occurs so it is final graph.

- Time complexity -

$O(|E| \cdot (|V|-1))$

$O(n \cdot (n-1))$

$O(n^2)$

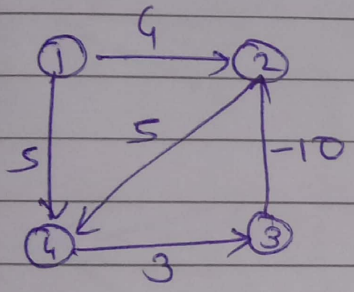
- for the complete graph.
  - There is edge present ~~at~~ betn every vertices.

$$E = \frac{n(n-1)}{2}$$

$O\left(n\left(\frac{n-1}{2}\right) \cdot (n-1)\right)$

$O(n^3)$

- Drawbacks of Ballmanford algorithm -



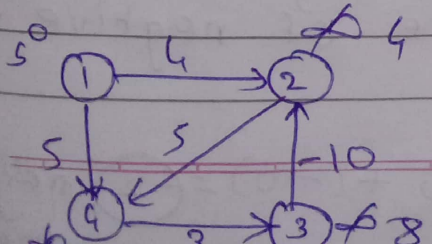
$E = \{(3,2)(4,3)(1,4)(1,2)(2,4)\}$

no. of relaxation on the edges $= |V| - 1$

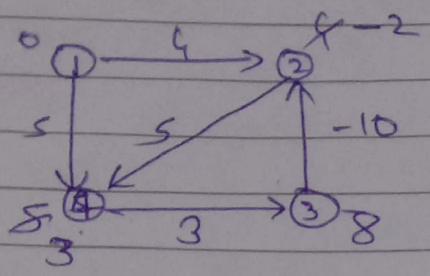
$= 4 - 1$

$= 3$

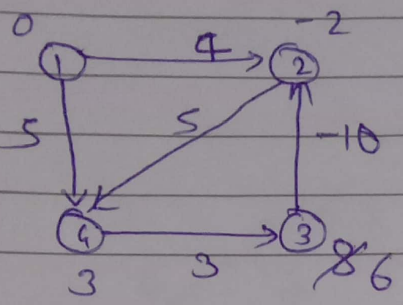
1st iteration.



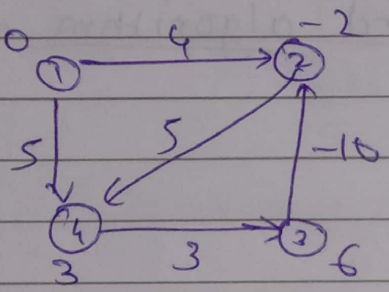
2nd iteration.



3rd iteration.

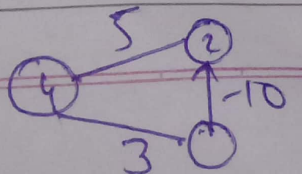


4th iteration



- Modified in 4th iteration is not valid in Bellman ford algo (occurs due to -ve wts) can't handle -ve wt cycle.

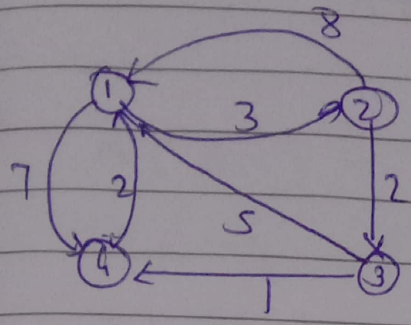
- For this given example in 4th iteration values are again relaxing this should not happen this is because of negtive weight cycle.



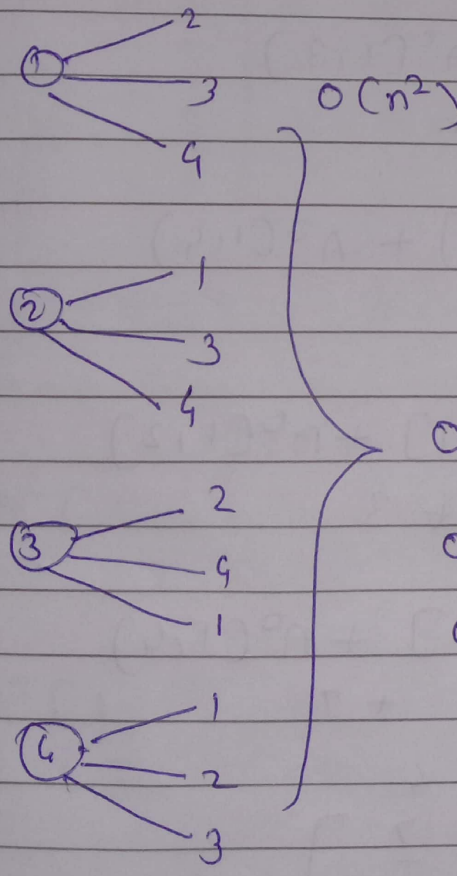
5+3+(-10) = -2 negative weight.

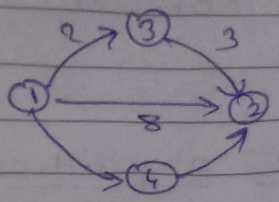
-ve weight cycles does not handle by Bellman ford algorithm.

- **All pair shortest path algorithm (floyd warshall Algo) :-**

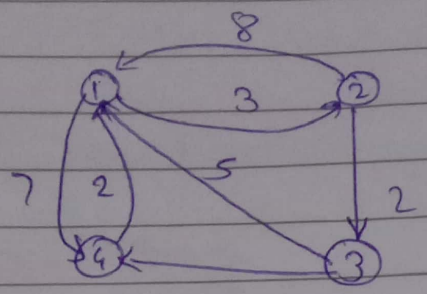


	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0





via vertex 3 will get shortest path.



$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 7 \\ 2 & 5 & \infty & 0 \end{bmatrix} \end{array}$$

$A^0[2,3] = A^0[2,1] + A^0[1,3]$

$2 \quad < \quad 8 \quad + \infty$

$A^0[2,4] = A^0[2,1] + A^0[1,4]$

$\infty \quad > \quad 8 + 7$

$A^0[3,2] = A^0[3,1] + A^0[1,2]$

$\infty \quad > \quad 5 + 3$

$A^0[3,4] = A^0[3,1] + A^0[1,4]$

$1 \quad < \quad 5 \quad + 7$

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$A^1[1,3] \quad A^1[1,2] + A^1[2,3]$
$\infty \quad > \quad 3 + 2$

$A^1[1,4] \quad A^1[1,2] + A^1[2,4]$
$7 \quad < \quad 3 + 15$

$A^1[3,1] = A^1[3,2] + A^1[2,1]$
$5 \quad < \quad 8 \quad + 8$

$A^1[4,1] = A^1[4,2] + A^1[2,1]$
$2 \quad < \quad 5 \quad + \quad 8$

$A^1[4,3] = A^1[4,2] + A^1[2,3]$
$\infty \quad > \quad 5 \quad + \quad 2$

$$A^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 0 & 2 & 3 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$A^2[1,2] = A^2[1,3] + A^2[3,2]$
$3 \quad < \quad 5 \quad + \quad 8$

$A^2[1,4] = A^2[1,3] + A^2[3,4]$
$7 \quad \geq \quad 5 \quad + 1$

$A^2[2,1] = A^2[2,3] + A^2[3,1]$
$8 \quad \geq \quad 2 \quad + 5$

$A^2[4,1] = A^2[4,3] + A^2[3,1]$
$2 \quad < \quad 7 \quad + 5$

$$A^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$A^3[1,2] = A^3[1,4] + A^3[4,2]$

$3 < 5 + 8$

$A^3[1,3] = A^3[1,4] + A^3[4,3]$

$5 = 3 + 2$

$A^3[2,1] = A^3[2,4] + A^3[4,1]$

$7 = 3 + 52$

$A^3[2,3] = A^3[2,4] + A^3[4,3]$

$5 > 1 + 2$

```
for (k=1 ; k=n ; k++)
{
    for (i=1 ; i<=n ; i++)
    {
        for (j=1 ; j<=n ; j++)
        {
            A[i,j] = Min{A^(k-1)[i,j], A^(k-1)[i,k] + A^(k-1)[k,j]}
        }
    }
}
```

No. of matrix = No. of vertices.

To Generate matrix