Unit - IV

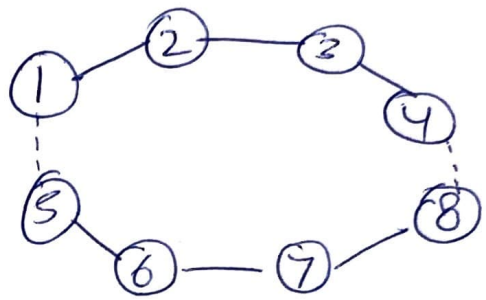
Disjoint Sets Representation -

↳ **Disjoint sets & operation** are useful to find cycle in undirected graph. (Two sets not having the common element)

e.g. $S_1 = \{1, 2, 3, 4, \}$

$S_2 = \{5, 6, 7, 8\}$

$S_1 \cap S_2 = \emptyset$



(4, 8) ✓✓

(1, 5)

1- **Find** - Find out which vertex belongs to which set.

e.g. (7) belongs to S_2

2- Union = $S_1 \cup S_2 = \{1, 2, 3, 4, 5, 6, 7, 8$

$S_3 =$

e.g. 2

$U = \{1, 2, 3, 4, 5, 6, 7, 8\}$

(1, 2)

$S_1 = \{1, 2\}$

$S_2 = \{3, 4\}$

$S_3 = \{5, 6\}$

$S_4 = \{7, 8\}$

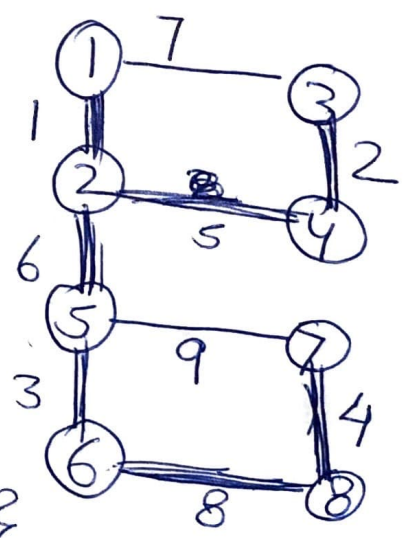
$\{2, 4\}$

$S_1 \cup S_2 = \{1, 2, 3, 4\}$

$\{6, 8\}$

$S_3 \cup S_4$

$= \{5, 6, 7, 8\}$



(2, 5)

$S_5 = \{1, 2, 3, 4, 5, 6, 7, 8\}$

(1, 3) = cycle don't include

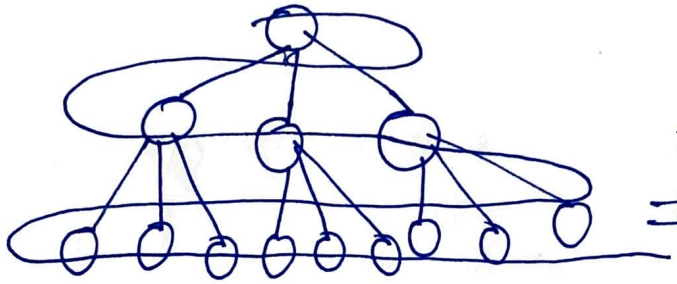
Introduction to Graph Traversal—

The process of visiting (vertex) and exploring a graph for processing is called graph traversal.

BFS
DFS

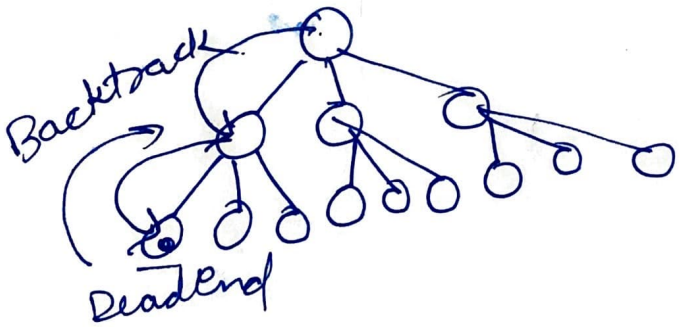
Every graph is a tree but every tree may or may not be a graph

G(V, E)

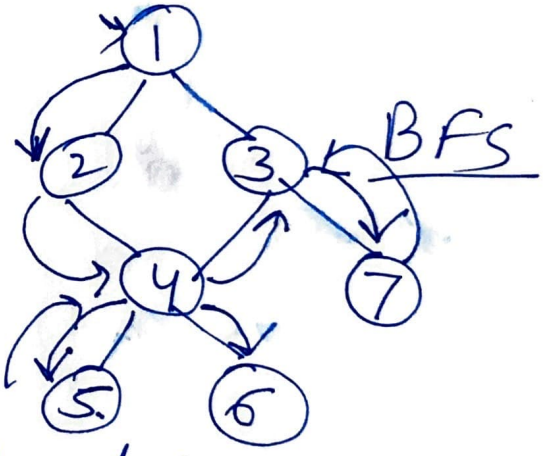


BFS (level ordering)
completes every level first
No backtrack.

DFS

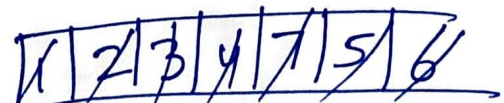


e.g.



we use queue data structure.

FIFO



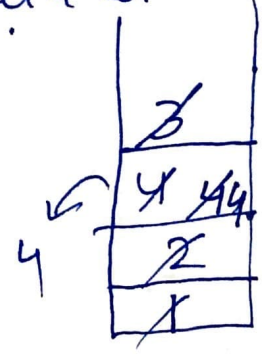
1 2 3 4 7 5 6

(sequence may vary but traverses by level)

In DFS we use stack data structures

1, 2, 4, 5 6 3 7

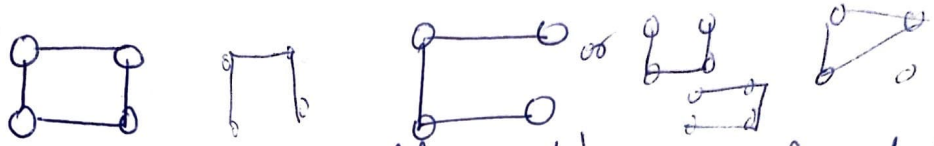
O(V+E)



Sequence

Spanning Tree: A connected subgraph 'S' of graph G(V, E) is said to be spanning if -

i) 'S' should contain all vertices of G.
ii) 'S' should contain (|V| - 1) edges

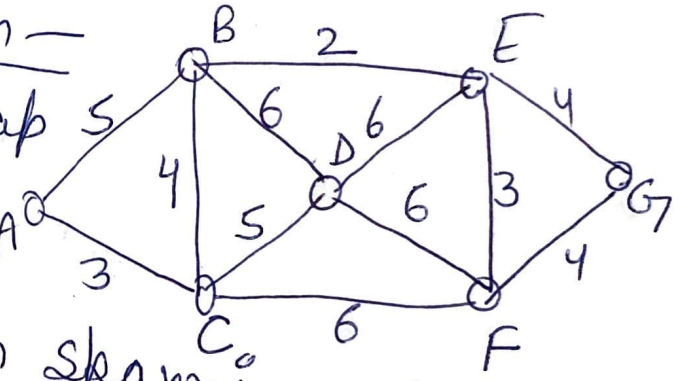
e.g.: 

All vertices are reachable.

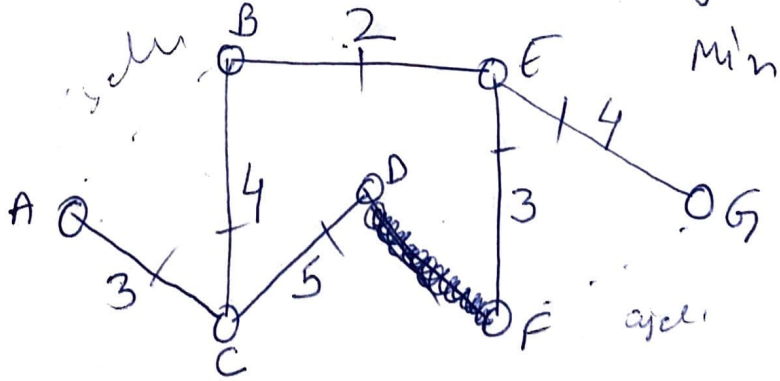
Tree is acyclic.

spanning tree = n^{n-2} (no of vertices)

Kruskal Algorithm -



i) Construct Min heap with e edges. O(e)
ii) Take one by one edge and add in spanning tree (cycle should not be created)
Best case (n-1) edges
worst case 'e' edges



Min edge

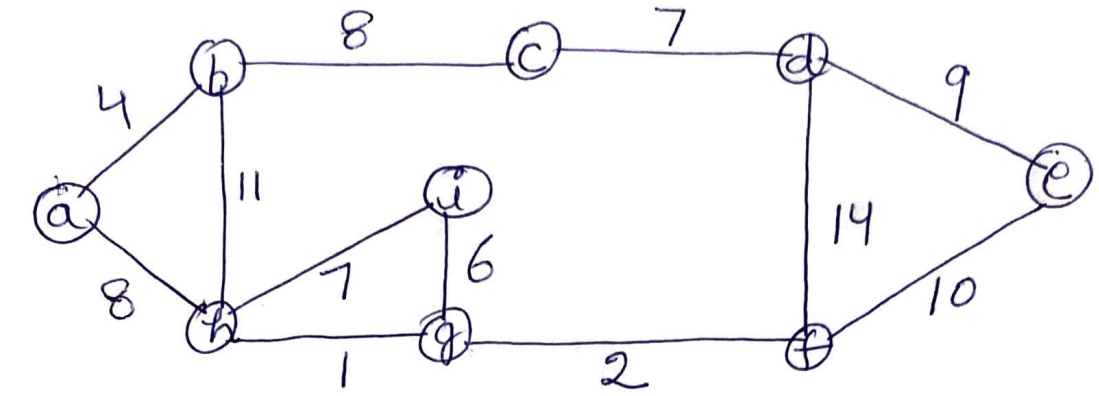
edges = 7-1
= 6 edges
no loop
cost low
connected graph.

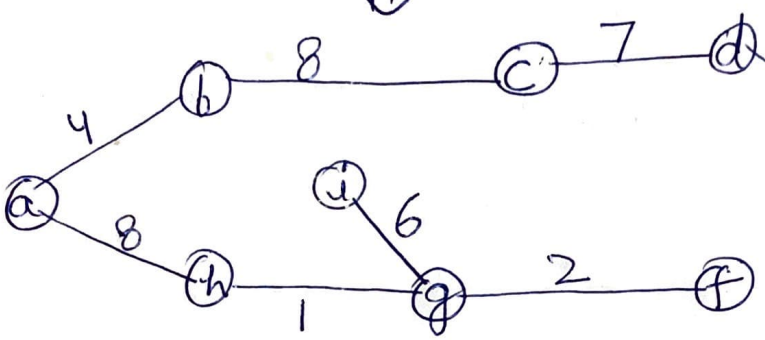
Intermediate is disconnected.

3+4+2+5+3+4

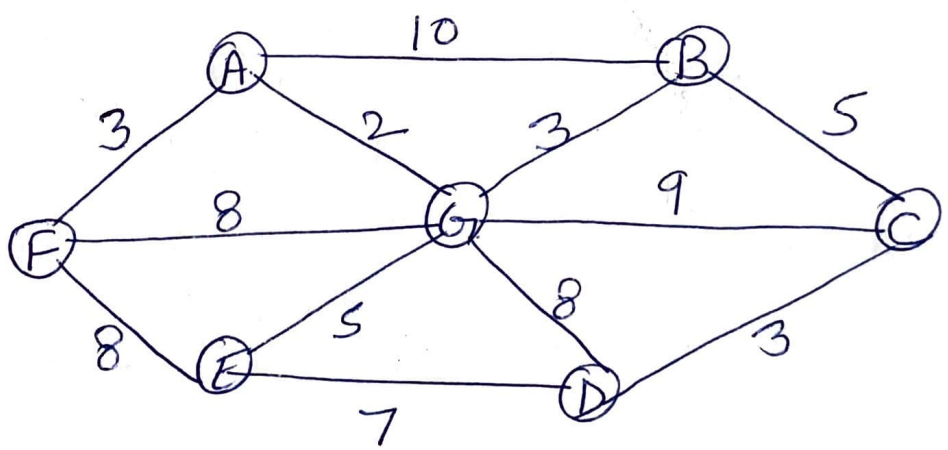
= 21 Min cost spanning tree

Q1> Find the minimum spanning tree of the following graph using Kruskal's algo.



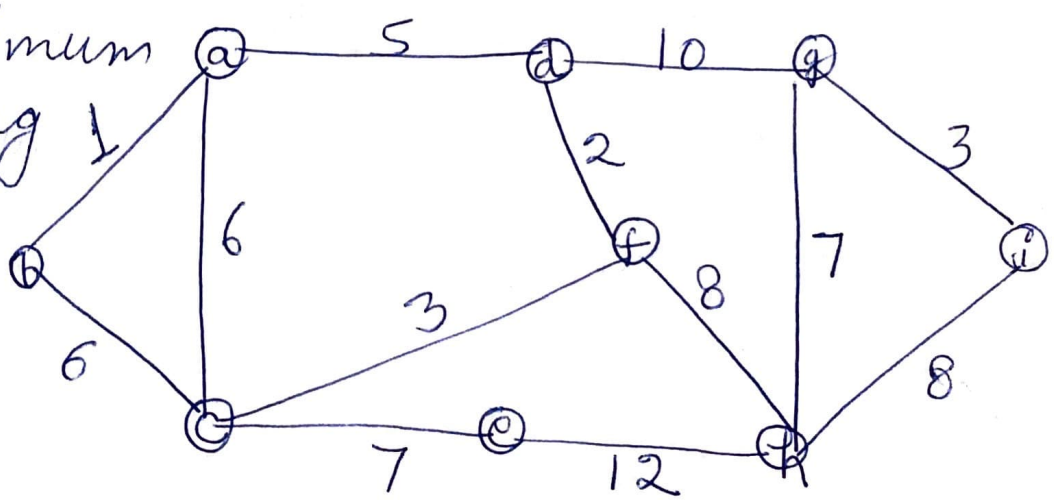


Q2-



Prim's Algorithm:-

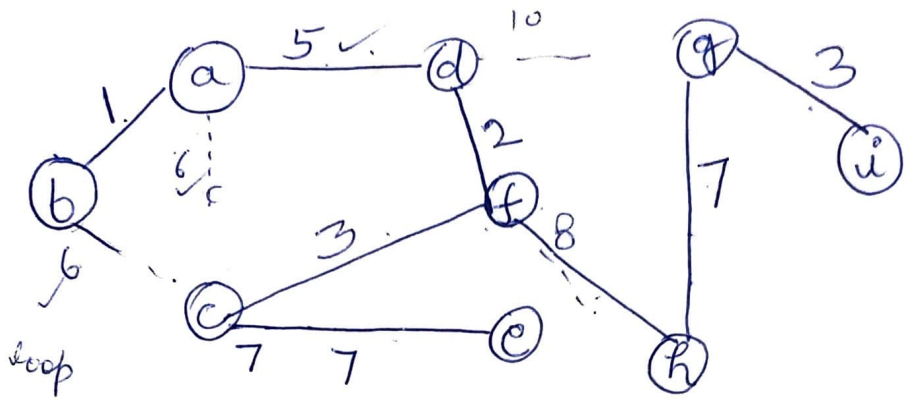
To find minimum cost spanning tree.



edges = n-1

You can start from any vertex.

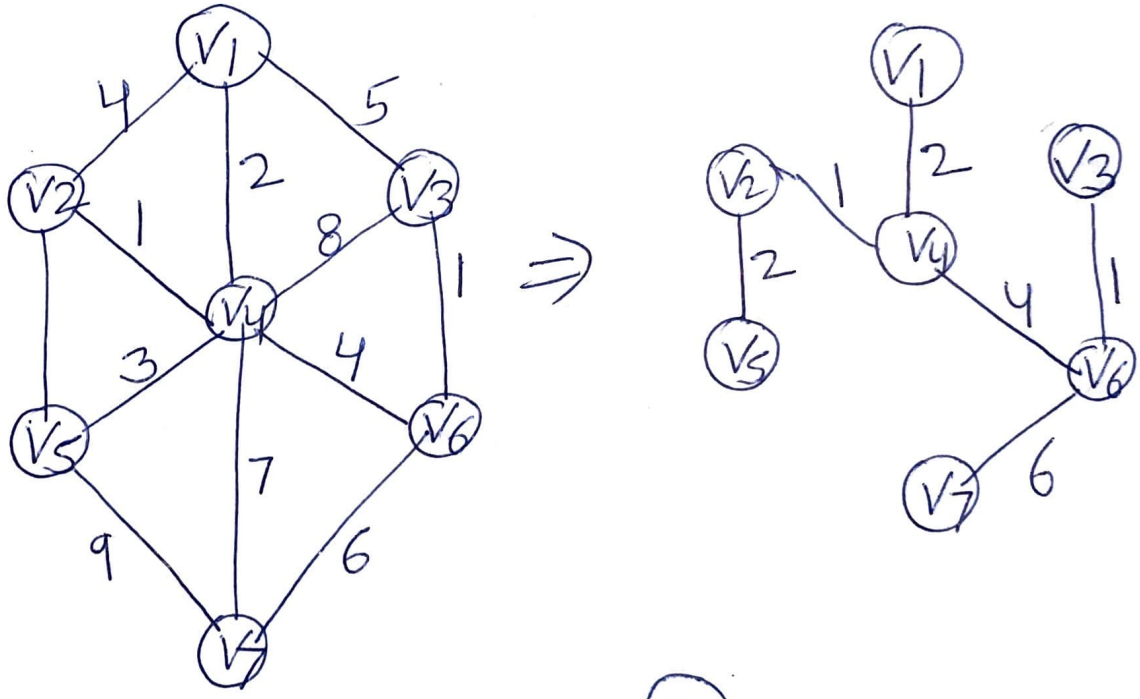
connected in intermediate way.



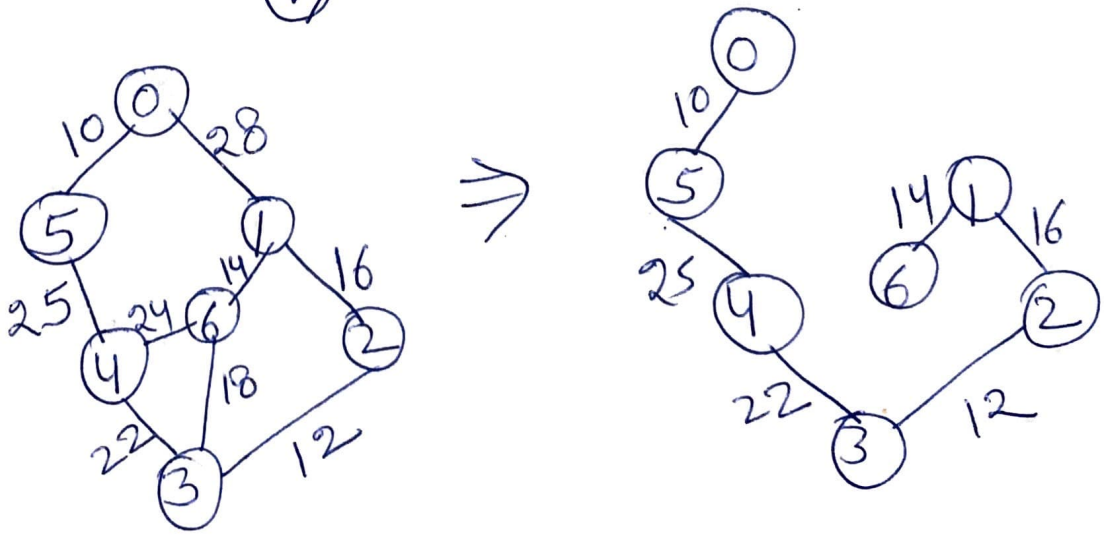
$1+5+3+7+8+2+7+3$

$=$

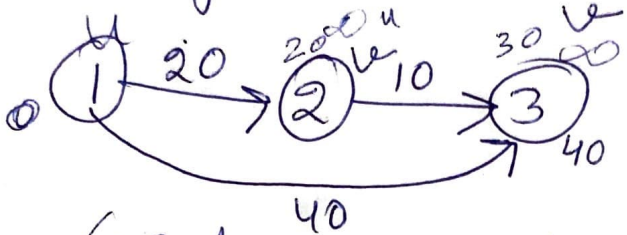
Q1-



Q2-



Shortest Path Algorithm – (Dijkstra's (Single Source Shortest Path) – Algorithm)



Relaxation
if $d(u)+c(u,v)<d(v)$
$d(v)=d(u)+c(u,v)$

Greedy approach optimal soln –

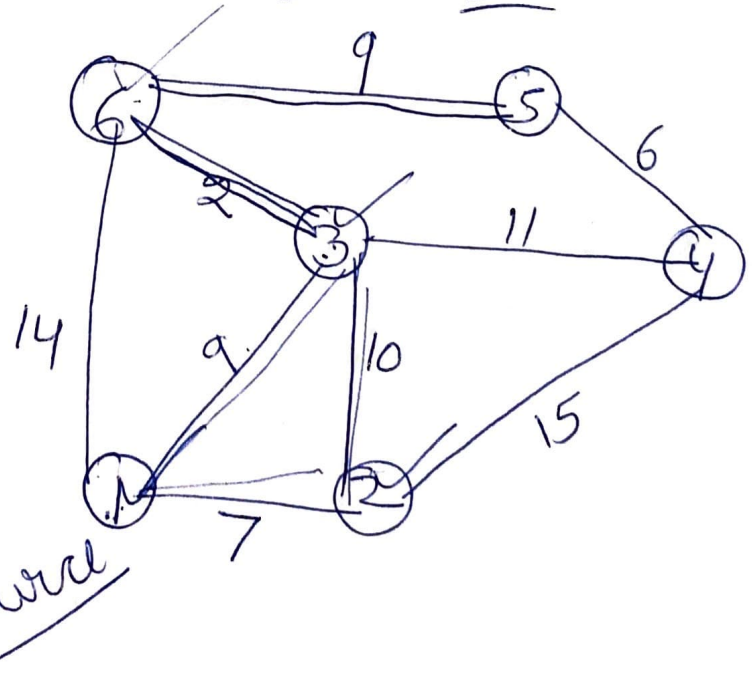
step 1 – for ①

$d(u)+c(u,v)<d(v)$
$0+(20+0)<\infty$
$d(v)=\underline{20}$

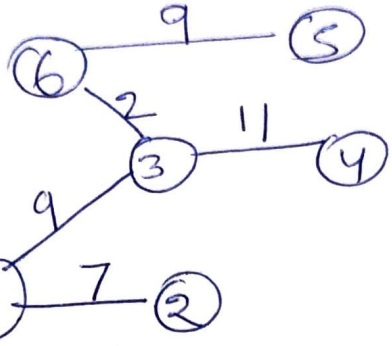
$d(u)+c(u,v)<d(v)$
$0+(0+40)<\infty$
$d(v)=\underline{40}$

$d(u)+c(u,v)<d(v)$
$20+10<40$
$30<40$
$d(v)=\underline{30}$

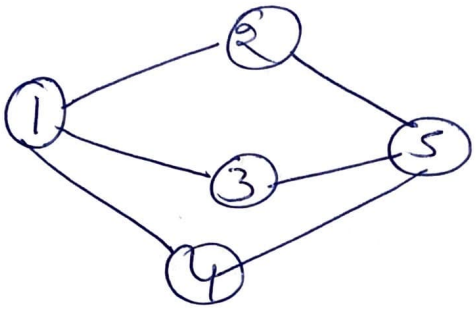
Q1–



Source	Destination 2	3	4	5	6
	∞	∞	∞	∞	∞
1,2	(7)	9	∞	∞	14
1,2,3	(7)	(9)	22	∞	14
1,2,3,6	7	9	20	∞	(11)
1,2,3,6,4	7,	9,	(20)	20	11
1,2,3,6,4,5	7,	9,	20,	20,	11

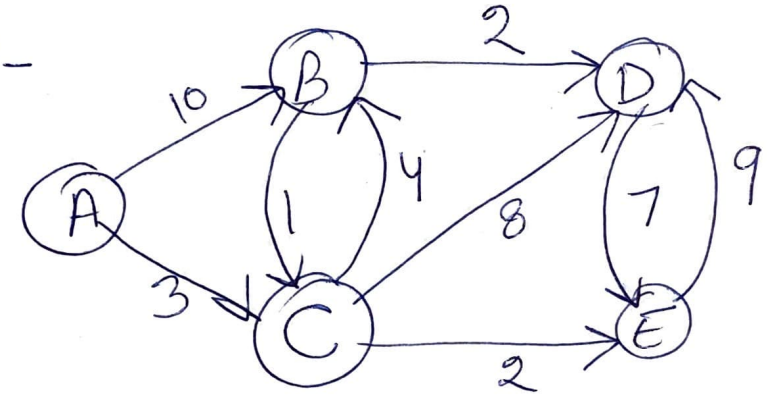


Dijkstra(Graph, Source) Algo –



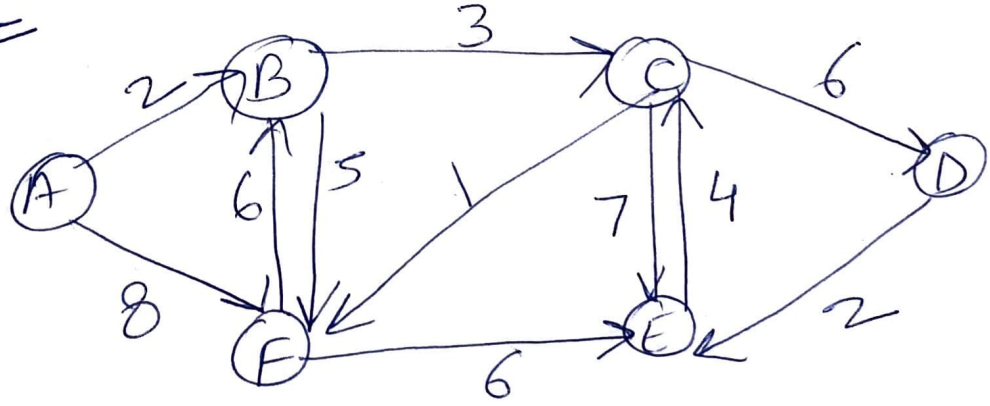
Q1-

A Source



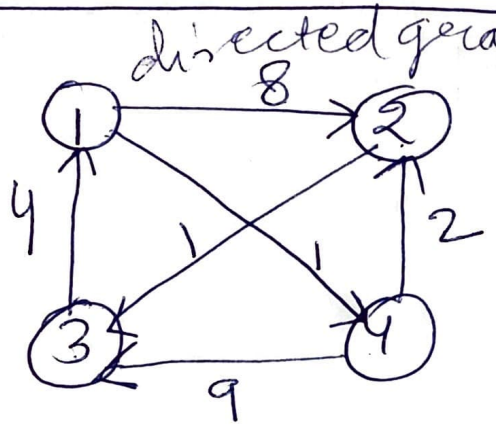
A Source

Q2-



All Pair Shortest Path (Floyd Warshall Algo)

Eg:



$D^0 =$ (Direct without including any node)

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	∞	0	∞
4	∞	2	9	0

$D^1 =$ via node 1

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	9	0

$D^2 =$ via 1 and 2

	1	2	3	4
1	0	8	(9)	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	(3)	0

$D^3 =$ via 1, 2, 3

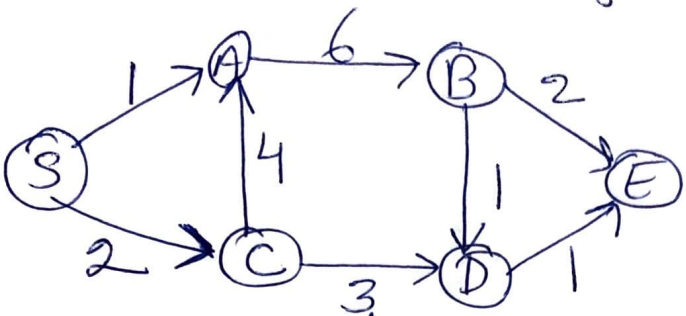
	1	2	3	4
1	0	8	9	1
2	(5)	0	1	(6)
3	4	12	0	5
4	7	2	3	0

$D^4 =$

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

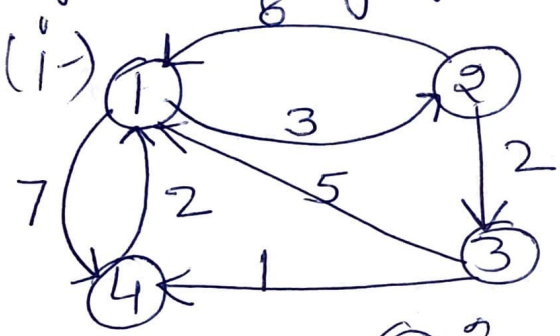
Final

Q1- Considering 'S' as source vertex. Apply the Dijkstra's shortest path algorithm on the following graph.

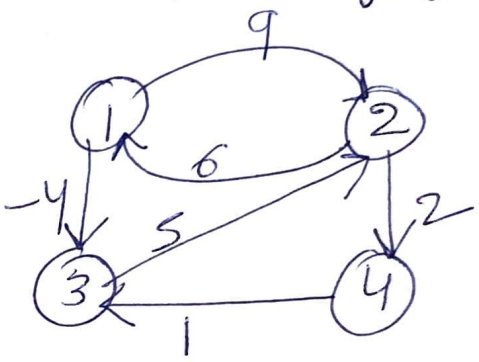


Q2- Find all pair shortest path for the following graph.

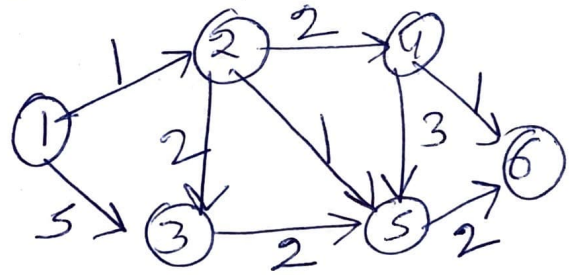
(i-)



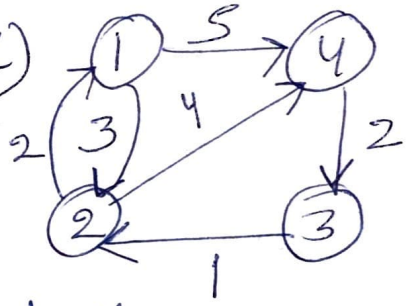
(ii-)



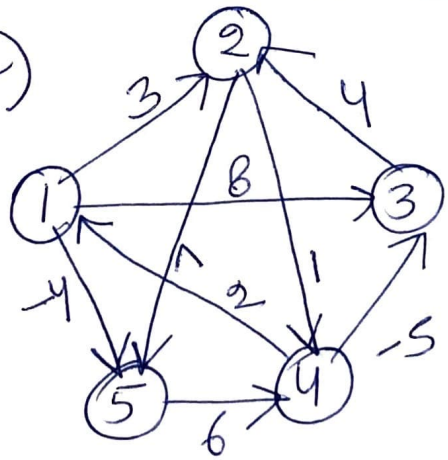
(iii-)



(iv-)



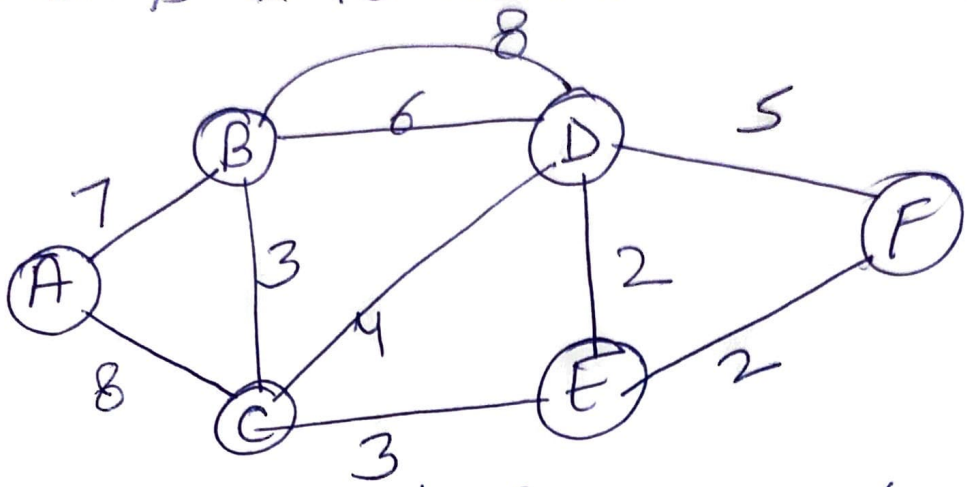
(v-)



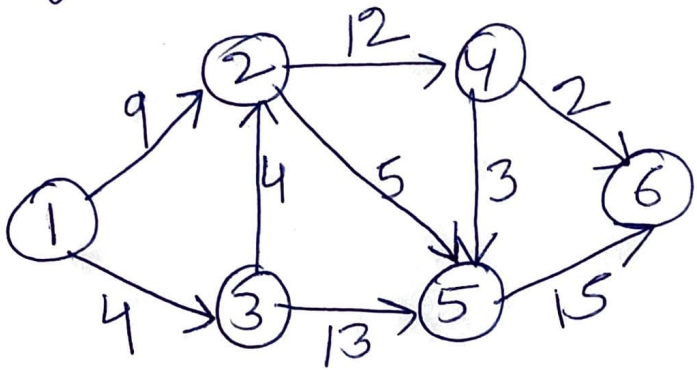
Floyd_Warshall(w)
1- n ← rows[w]
2- D⁰ ← w
3- for K ← 1 to n
4- do for i ← 1 to n
5- do for j ← 1 to n
6- do $d_{ij}^{(K)} \leftarrow \min(d_{ij}^{(K-1)}, d_{iK}^{(K-1)} + d_{Kj}^{(K-1)})$

Prim's & Kruskal's examples –

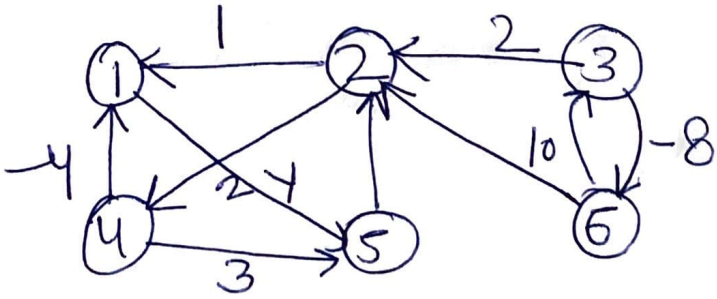
1)



Q- By considering vertex '1' as source vertex. Find the shortest paths to all other vertices in using dijkstra's algorithms.



Q→ Floyd warshall's algo -



Graph Traversal - Graph traversal is a technique used for a searching vertex in a graph. The graph traversal is also used to decide the order of vertices is visited in the search process. A graph traversal finds the edges to be used in the search process without creating loops. That means using graph traversal we visit all the vertices of the graph without getting into loop path.

DFS - Algorithm -

1- Defines a stack of size total number of vertices in the graph.

2- Select any vertex as starting point for traversal. Visit that vertex and push it on to the stack.

3- Visit any one of the non visited adjacent vertices of a vertex which is at the top of stack and push it on to the stack.

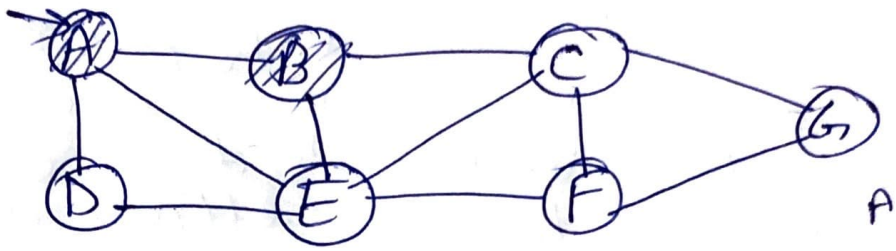
4- Repeat step3 until there is no new vertex to be visited from the vertex which is at the top of the stack.

5- When there is no new vertex to visit then use backtracking and pop one vertex from the stack.

6- Repeat steps 3, 4 & 5 until stack becomes empty.

7- when stack becomes empty, then produce final spanning tree by removing unused edges from the graph.

e.g.



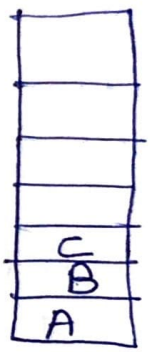
A B C G

Step1 - Select the vertex A as starting point (visit A)

- Push A on to the stack.

Step2 - visit any adjacent vertex of A i.e. B Push B →

Step3 (C) Push C →



BFS – BFS traversal of a graph produces a spanning tree as final result Spanning tree is a graph without loops. We use queue data structure with max. size of total number of vertices in the graph to implement BFS traversal.

1- Define a queue of size total no. of vertices in the graph.

2- Select any vertex as starting point for traversal. Visit that vertex and insert into the queue.

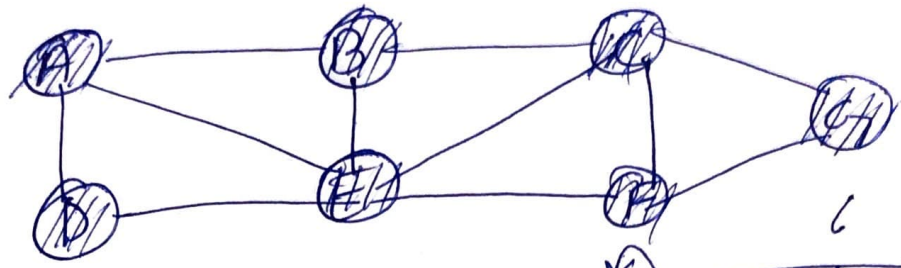
3- Visit all the non-visited adjacent vertices of the vertex which is at front of the queue and insert them into the queue.

4- When there is no new vertex to be visited from the vertex which is at front of the queue then delete that vertex.

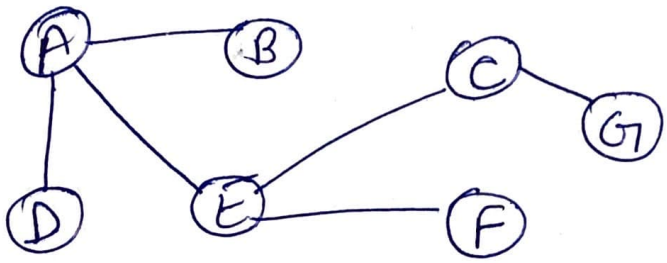
5- Repeat step 3 and 4 until queue becomes empty.

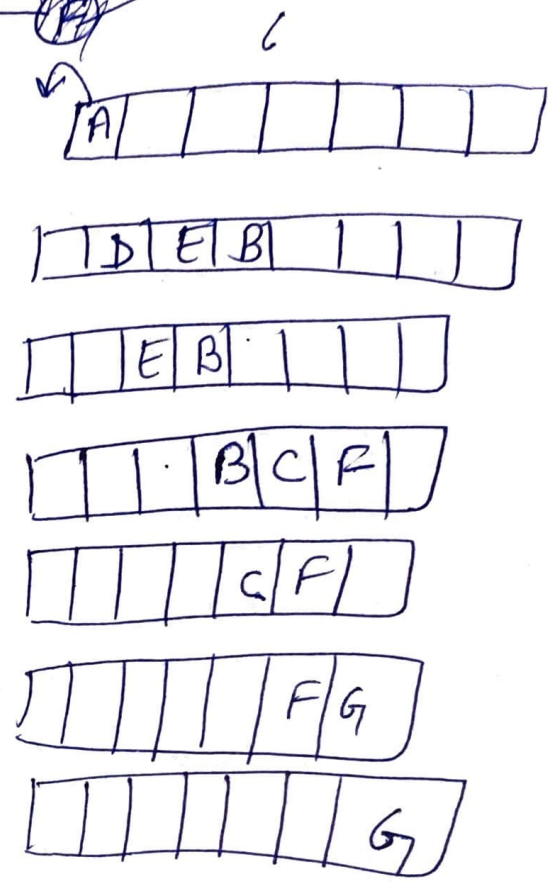
6- When queue becomes empty, then produce final spanning tree by removing unused edges from the graph.

16



17





18