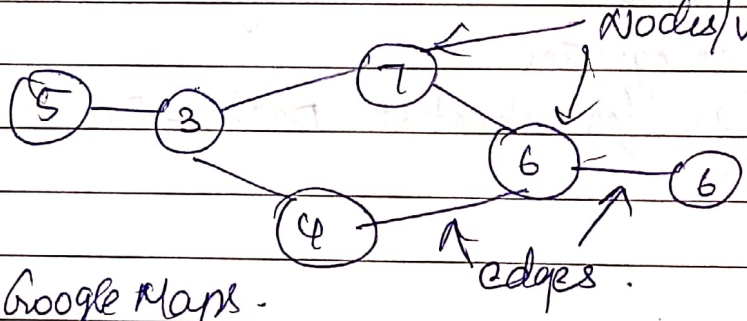
Unit - 5

- Review of basic terminology
  - Types of graphs: directed, undirected, weighted graph,
- Representation of graphs using adjacency matrix, adjacency list.
- Traversals - Depth First search (DFS) & Breadth First Search (BFS)
- Connected Components & spanning trees.
- Kruskal's & Prim's algo. for minimum spanning tree.
- Algo. for shortest path - Dijkstra's algo.

Graph:

- Non-linear Data structure
- consists of vertices & edges.
- Vertices also refered to as nodes.
- edges are lines/arcs connect any 2 nodes in a graph.
- composed of a set of vertices (V) & set of edges (E)
- is denoted by G(V, E)

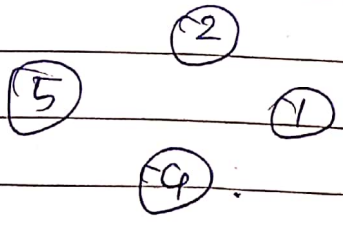


Eg: Google Maps.

Types of graph:

① Null graph:

- if there no edges in the graph.



null graph.

② trivial graph:

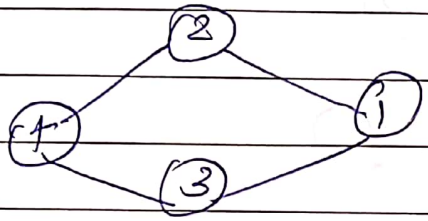
- graph having only 1 vertex
- its smallest graph as possible

(3) trivial graph.

M T W T F S S
Page No.:
Date:
YOUVA

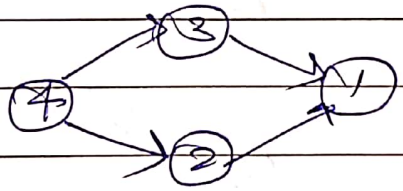
(3) Undirected graph:

- A graph in which edges do not have any direction.



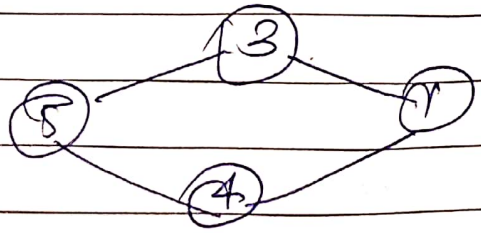
(4) Directed graph:

- A graph in which edge has direction.
- ie. the nodes are ordered pairs. in the definition of every edge.



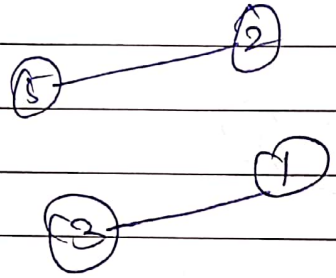
(5) connected graph:

- ~~Graph~~ ~~is connected~~ from one node can visit any other node in the graph



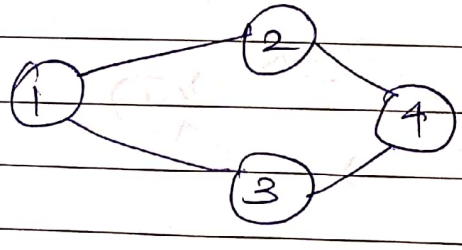
(b) disconnected graph:

Atleast one node is not reachable from a node



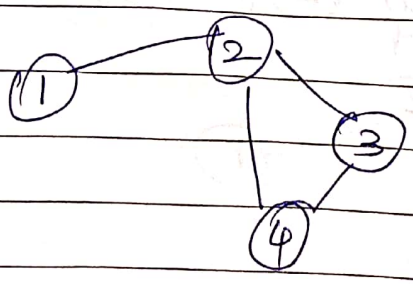
(7) cyclic graph:

- In a graph, it has a cycle itself.
- degree of each vertex is 2



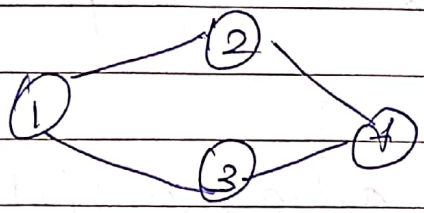
(8) Acyclic graph:

A graph containing atleast one cycle.



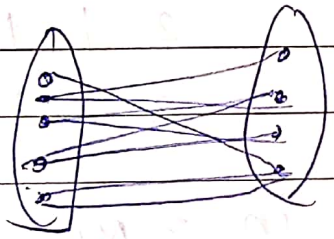
(9) Directed Acyclic graph:

- it does not contain any cycle.



(10) Bipartite graph.

In a graph, a vertex can be divided into 2 sets, that vertex in each set does not contain any edge b/w them.



(11) Weighted graph:

In a graph, the edges are specified with suitable weight it is called weighted graph.

weighted graph.

- directed weighted graph.
- undirected weighted graph.

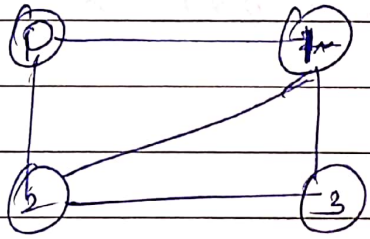
Note: Every tree will always be a graph but not all graph will be tree.

Representation of Graph: (to represent in a comp.)

— 2 ways to store a graph.

- adjacency matrix
- adjacency list.

Adjacency Matrix:



adjacency Matrix:

i \ j	0	1	2	3
0	0	1	1	0
1	1	0	1	1
2	1	1	0	1
3	0	1	1	0

- graph is stored in a matrix.
- there rows & columns denote vertices.
- each entry in the matrix denotes the weight b/w the vertices.

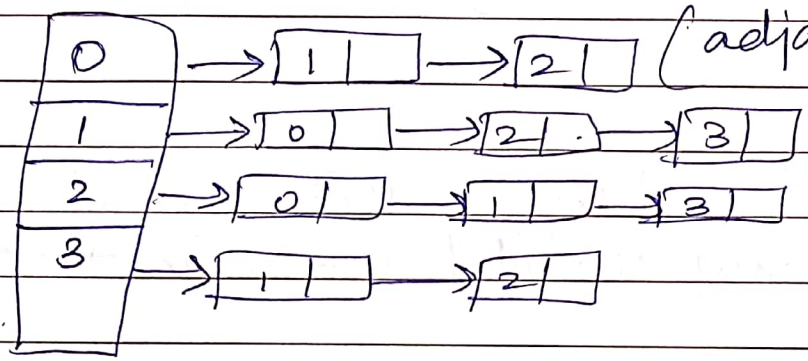
A[n][n]

if a[i][j]=1 (if i & j are adjacent otherwise it is 0).

Space complexity is $O(n^2)$ since it is n×n.

adjacent list:

Each vertices will have a linked list.



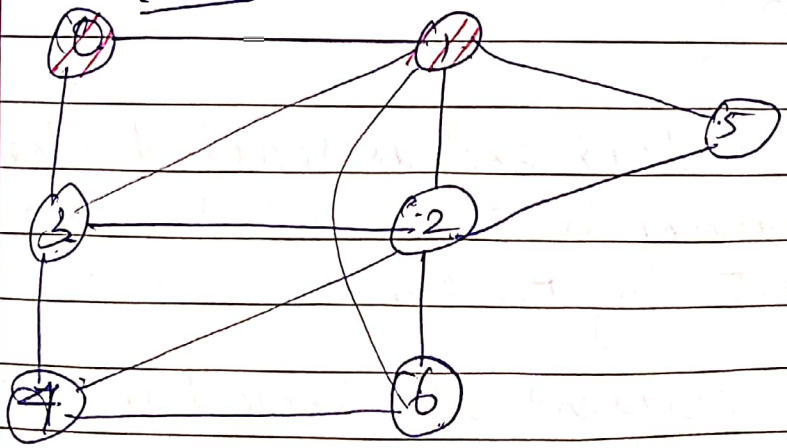
Space complexity - $O(n+2e)$

edge is return twice ie 0 to 1 & 1 to 0.

Graph traversal:

1. BFS
2. DFS

DFS: 0 1 5 2 6 4 3



V	S
0	~~0~~
1	~~1~~ 2
5	~~5~~ 6
2	~~2~~ 4
6	3
4	
3	

V	S
0	~~0~~
1	~~1~~
5	~~5~~
2	~~2~~
6	~~6~~
4	~~4~~
3	~~3~~

if '0' is considered as root node, then adjacency vertices will be visited.

M T W T F S S
Page No.:
Date:
YOUVA

BFS:

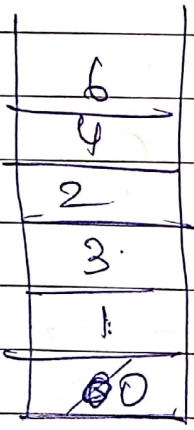
queue | 0 | 1 | 3 | 2 | 5 | 6 | 4 |

Result : 0 1 3 2 5 6 4.

1' adjacent vertices 0, 3, 5, 2, 6
add only unvisited vertices in the queue.

3 ⇒ 2, 4, 6.
2 ⇒ 1, 3, 4, 6, 5

DFS: Stack.



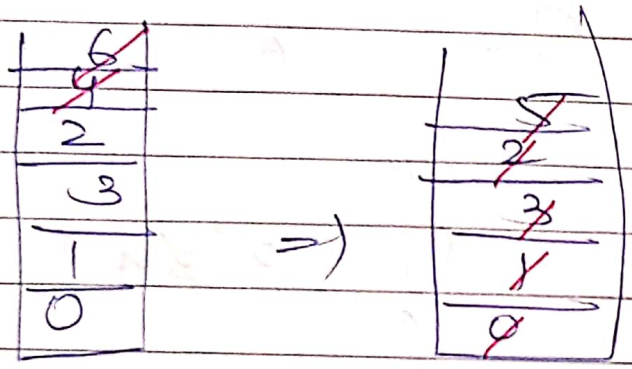
Result: 0 1 3 2 4 6

After a dead end is reached, the backtracking is performed by pop of the element of the stack.

- each element is checked whether their adjacency vertices is not is checked if its unvisited its pushed otherwise pop.

until all the vertices are pushed.

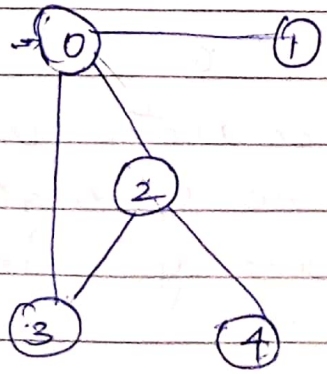
Result → elements that are printed while pushing the element into the stack.



DFS: Result: 0 1 3 2 4 6 5

DFS algo is stopped, when the stack is empty, when all elements are popped out of the stack.

DFS:



Visited: 0 1 2 4 3

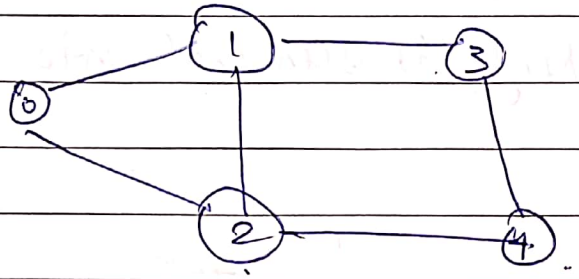
Stack: 0 1 2 4 3

top

step	V	S
	0	0
	1	1
		2
		4
		3

Visited: 0, 1, 2, 4, 3.

BFS:



visited. queue.

0. | 0 | 1 | 2 | 3 | 4 |
1 ↑ P.
2
3.
4. visited : 0, 1, 2, 3, 4.

Spanning tree:

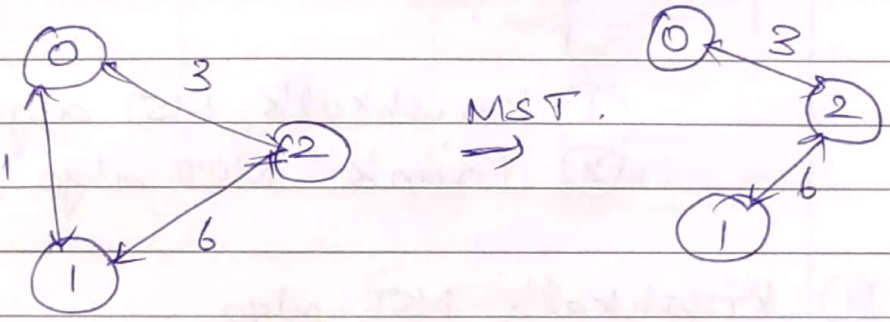
- Subset of graph G.
- Vertices are connected using minimum possible no. of edges.
- it will not have cycles in a graph.
- A graph can have more than one spanning Tree.
- it does not exist for disconnected graph.

Note: MST is subgraph. It contains all vertices, and are connected, it has no cycles. Min: Σ of weight of edges should be Min.

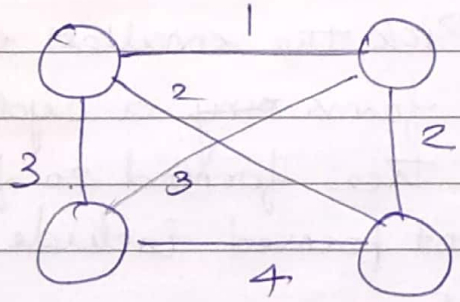
MST (Min. Spanning Tree)

the weight of a spanning tree is determined by the sum of weight of all the edges involved in it.

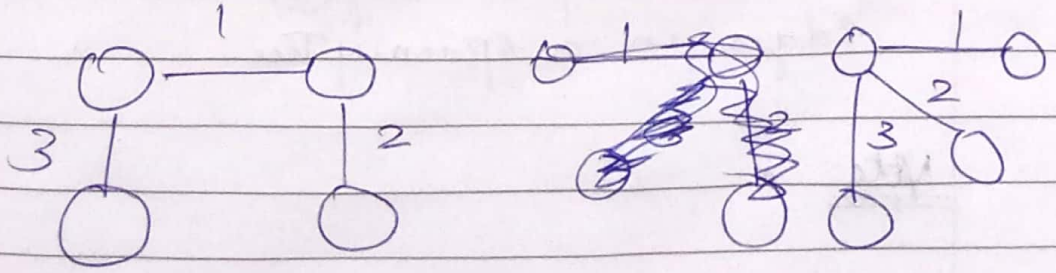
MST of directed graph.

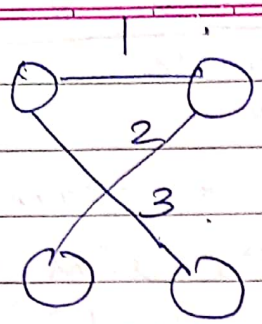


* MST → acyclic.
* MST ⇒ V vertices of a graph will have (V−1) vertices in an MST.



MST:





Diff: Algo to find minimum spanning tree:

1. Kruskal's MST algo.
2. Prim's MST algo.

1. Kruskal's MST algo:

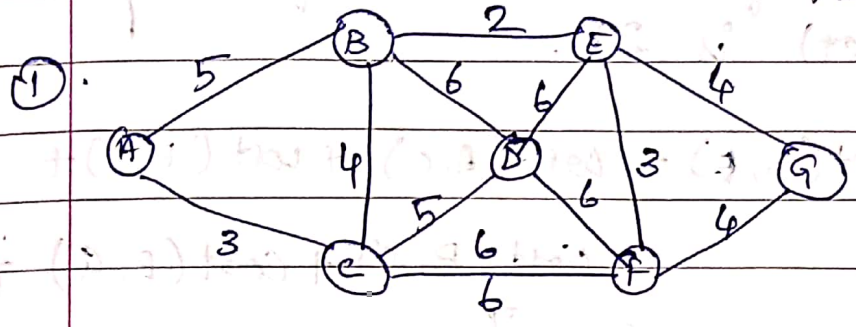
Steps:

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms ~~any~~ a cycle with a spanning tree formed so far. If the cycle is not formed include it otherwise exclude it.
3. Repeat step 2 until that V-1 edges in a spanning tree.

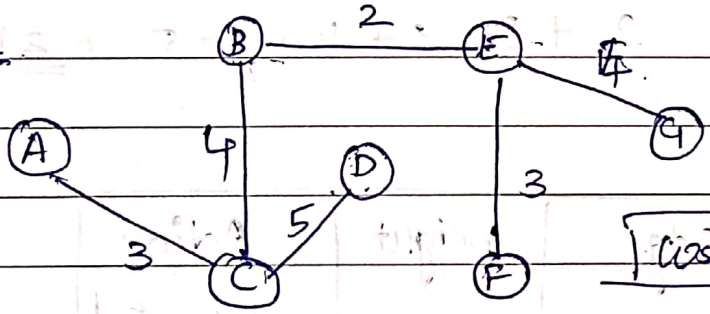
Note:

MST = $\frac{n-1}{\text{edges}} = 6$.

$n = 7$.



MST:



cost = 21.

from Given graph

Edges	weight
(B,E)	2
(A,C)	3
(E,F)	3
(E,G)	4
(F,G)	4
(B,C)	4
(A,B)	5
(C,D)	5
(C,F)	6
(D,E)	6
(B,D)	6
(D,F)	6

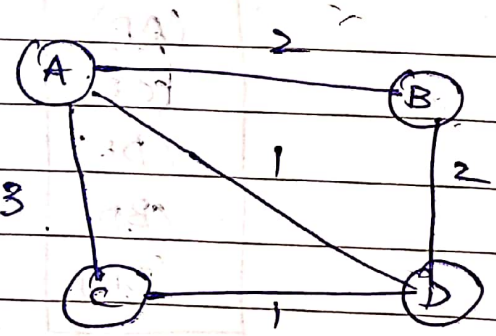
The cost of the minimum spanning tree (MST) is 21.

= [cost(B, E) + cost(A, C) + cost(E, F) + cost(B, C) + cost(E, G) + cost(C, D)]

= 2 + 3 + 3 + 4 + 4 + 5 = 21.

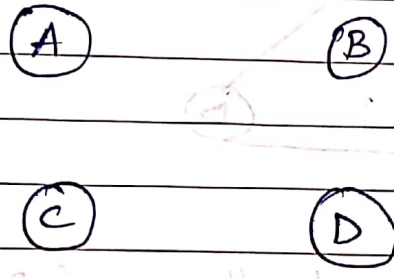
edge	Weight	Action
(B, E)	2	accepted
(A, C)	3	accepted
(E, F)	3	accepted
(B, C)	4	accepted
(E, G)	4	accepted
(C, D)	5	accepted

(2)

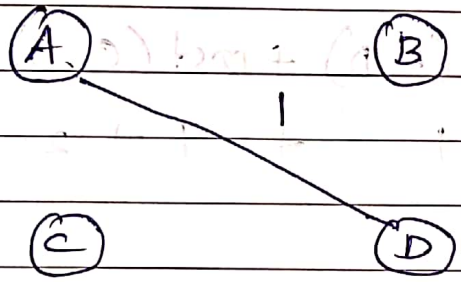


edges	Weight
(A, D)	1
(C, D)	1
(A, B)	2
(B, D)	2
(A, C)	3

Step 1: Initially each vertex is in its own set.

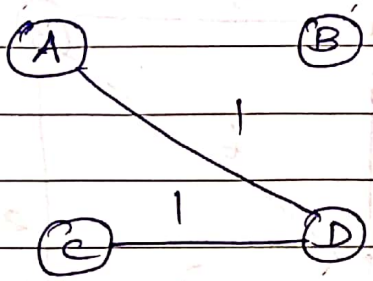


Step 2: Select the edge with minimum weight.

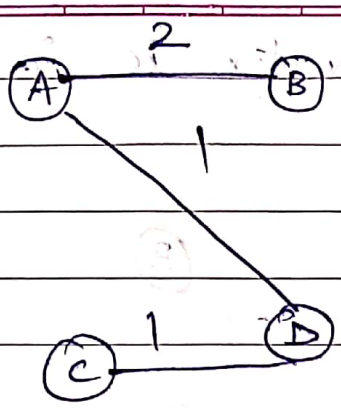


Step 3: Select the next edge, with minimum weight & check whether it forms a cycle.

⇒ If it forms a cycle, then that edge is rejected, else add the edge to the MST.



Step 4: Repeat step 3 until all vertices are included in the MST.



The cost of the minimum spanning Tree (MST) is 4.

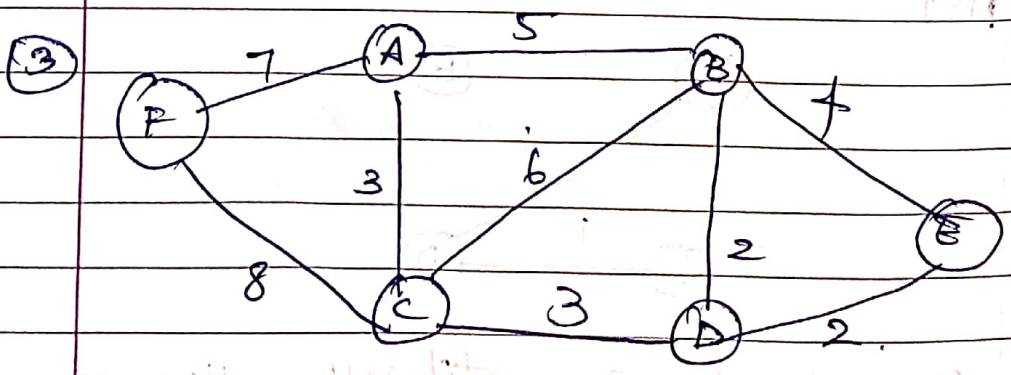
~~= cost(A, B) + c~~

$= [cost(A, D) + cost(C, D) + cost(A, B)]$

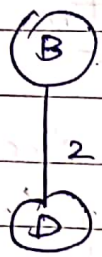
$= 1 + 1 + 2$

$= 4.$

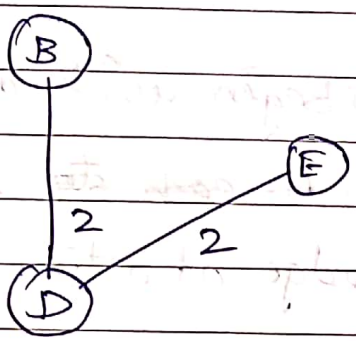
edges	weight	Action
(A, D)	1	accepted
(C, D)	1	accepted
(A, B)	2	accepted



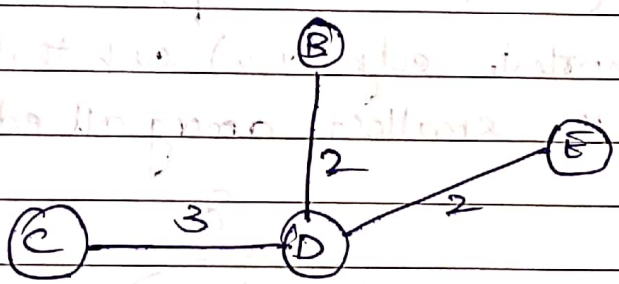
1



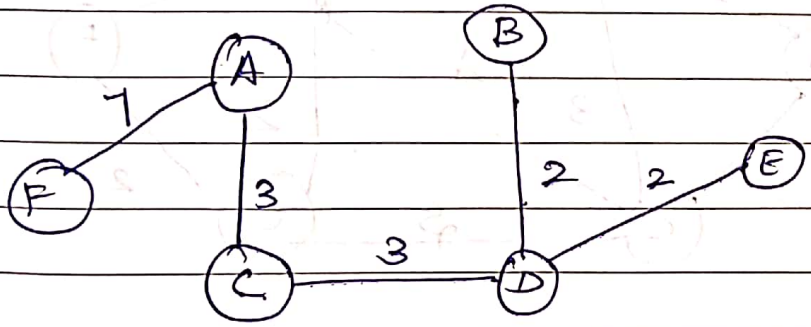
2



3



4



cost of MST = 17.

no. of edges in MST = (n-1) = 5

no. of vertices in MST is n = 6.

Prim's algorithm

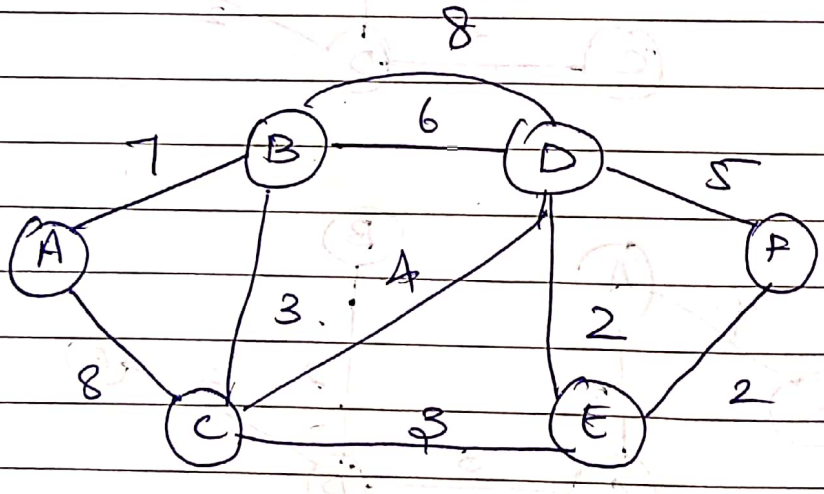
⇒ is one of the way to compute MST using a greedy technique.

Algo:

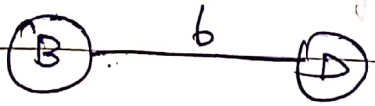
(1) Begin with a set U initialised to {1}.

(2) ~~At each step~~ It then grows a MST, one edge at a time.

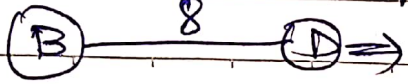
(3) At each step, it finds the shortest edge (u,v) such that cost of (u,v) is the smallest among all edges.



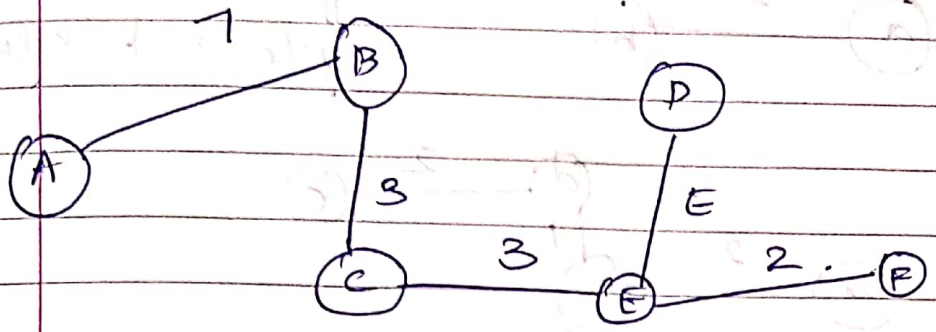
(1) Removing the edges, keep the min. edge.



is accepted.



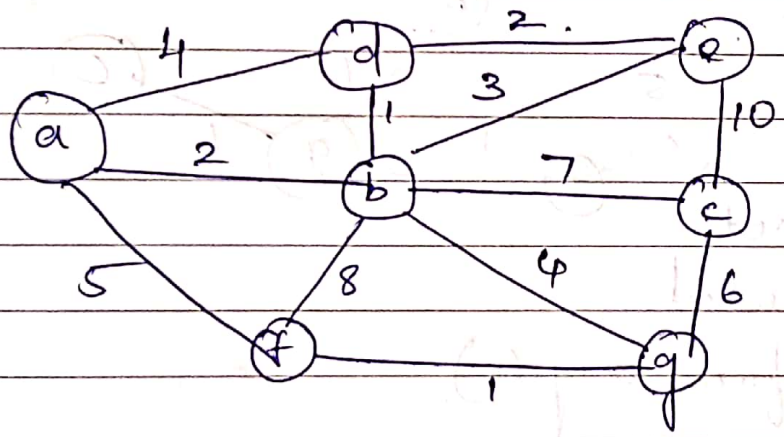
⇒ is rejected.



~~Step~~ Algo:

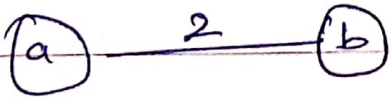
1. choose a abitrary start vertex.
2. keep including connected min edges.

Q. 2

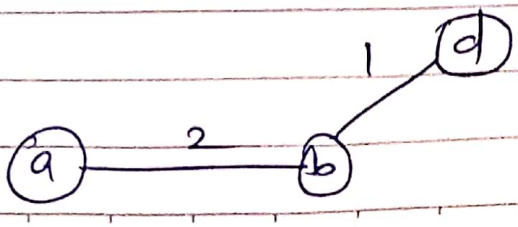


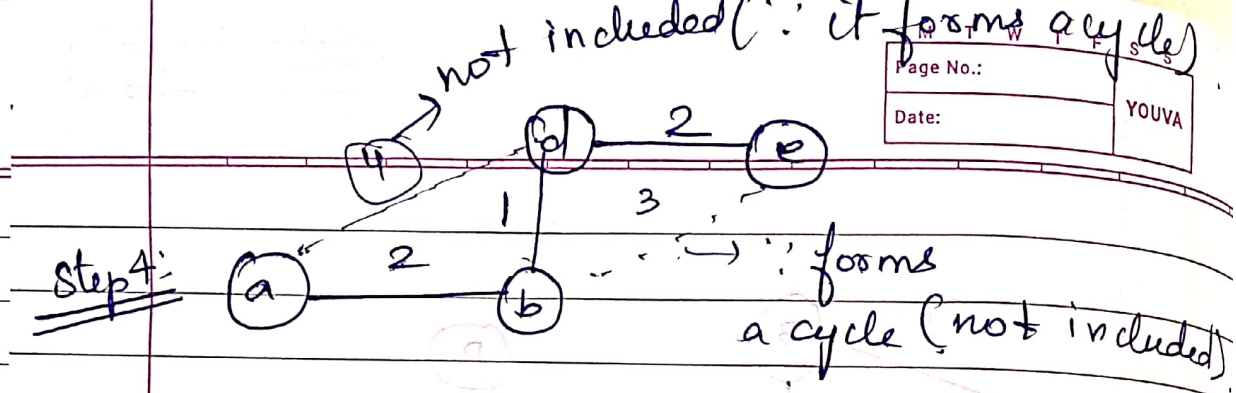
Step 1:

Step 2:



step 3:

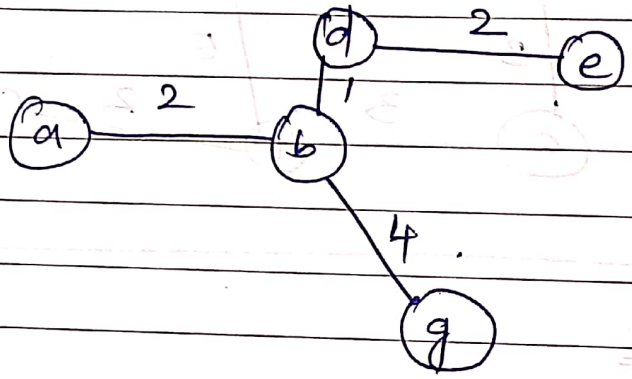




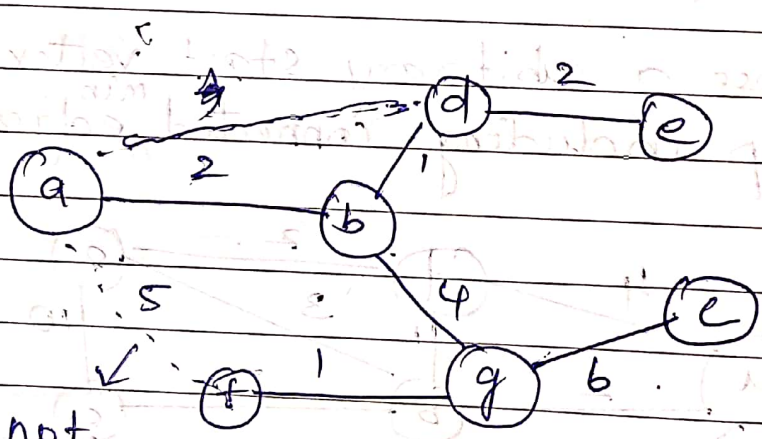
Step 4:

(∵ forms a cycle (not included)

Step 5:



Step 6:



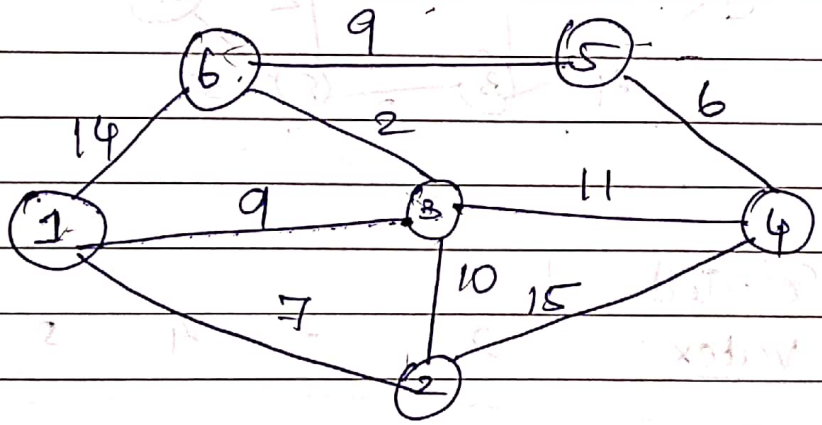
not included (∵ forms a cycle).

Single Source shortest Path.

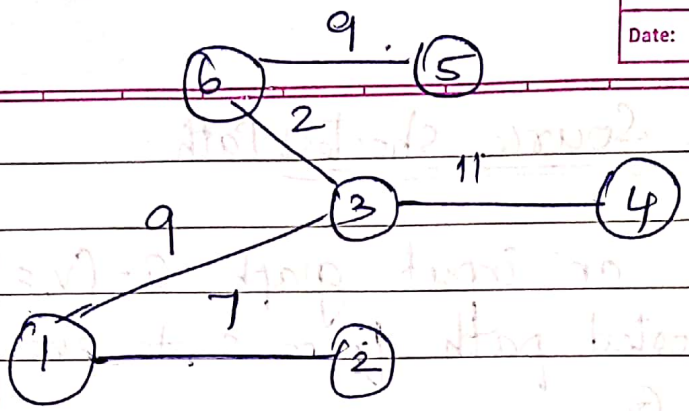
Qn. an input graph $G=(V,E)$, find the shortest path from s to every other vertex G.

* Its applied to both weighted & unweighted graph.
* Its applied directed & undirected graph.

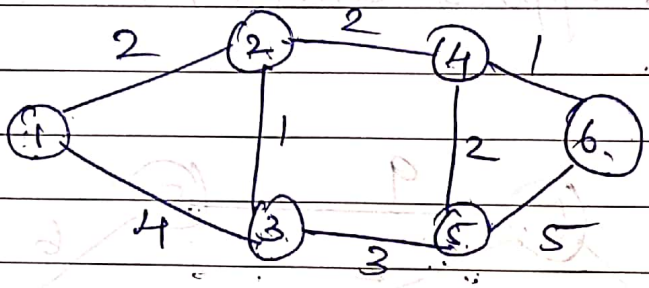
Q. (1)



Source	Destination					
	1	2	3	4	5	6
	∞	∞	∞	∞	∞	
direct link. (1,2) (1,3)	0	7	9	∞	∞	14.
1,2 thro' 2	0	7	9	22	∞	14.
1,2,3 thro' 3	0	7	9	20	∞	11.
1,2,3,6 thro' 6.	0	7	9	20.	20	11.
1,2,3,6,4.	0	7	9	20	20	11.
1,2,3,6,4,5						



Dijkstra's Algorithm:



Selected vertex:	2	3	4	5	6
1	(2)	4	∞	∞	∞
2	(2)	(3)	4	∞	∞
3	(2)	(3)	(4)	6	∞
4	2	3	4	(6)	5
5	2	3	4	6	(5)
6	2	3	4	6	5

Step 1: '1' vertex is selected as source vertex, from '1' vertex what are vertex that is directly connected, their weight is included in the table.

Rest, of the vertex is marked as '∞'.

Step 2: which weight is lesser is chosen, as the next vertex.

$E(1,3) = cost = E(1,2) + E(2,3) = 3.$

∴ the previous weight is 4, But the current weight for E(1,3) is 3 is chosen, since previous weight more than this.

Step 3: the value '4' vertex will not be changed.

i.e. $(1,4) = E(1,2) + E(2,3) + E(3,5) + E(5,4)$

$= 2 + 1 + 3 + 2$

$= 8.$

∴ the previous weight is 4 we will not choose the above cost value.

Aim: is to find minimum path from the source vertex to every other vertex.