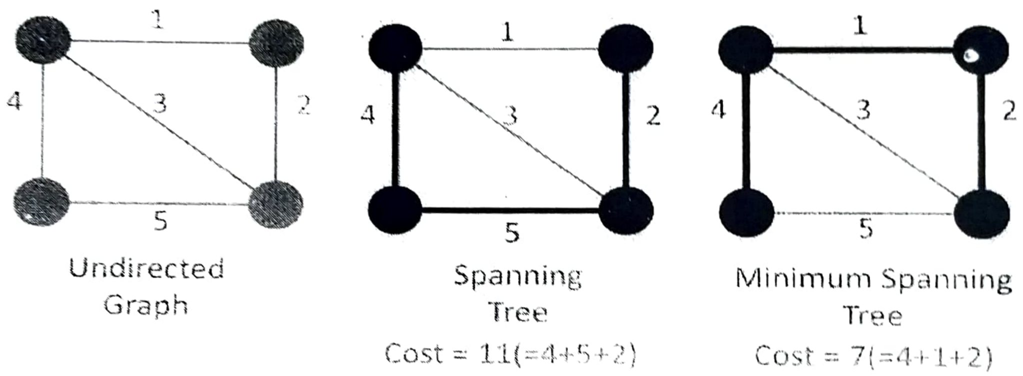
# What is a Minimum Spanning Tree?

The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees.

Minimum spanning tree has direct application in the design of networks. It is used in algorithms approximating the travelling salesman problem, multi-terminal minimum cut problem and minimum-cost weighted perfect matching. Other practical applications are:

1. Cluster Analysis
2. Handwriting recognition
3. Image segmentation



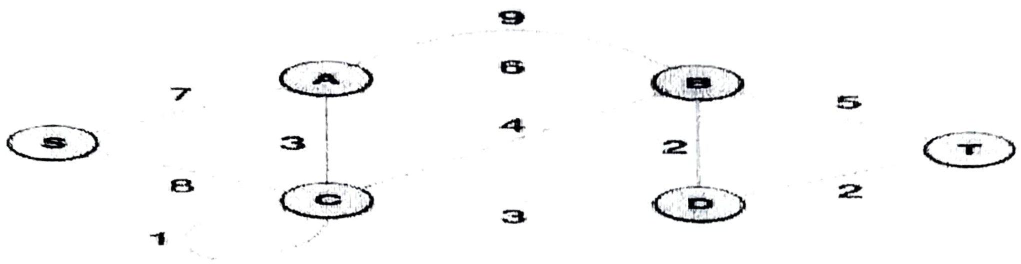

There are two famous algorithms for finding the Minimum Spanning Tree:

## Prim's Algorithm

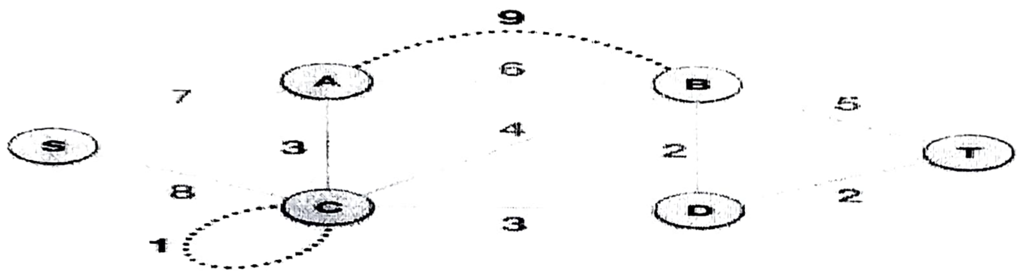
Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the shortest path first algorithms.

Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

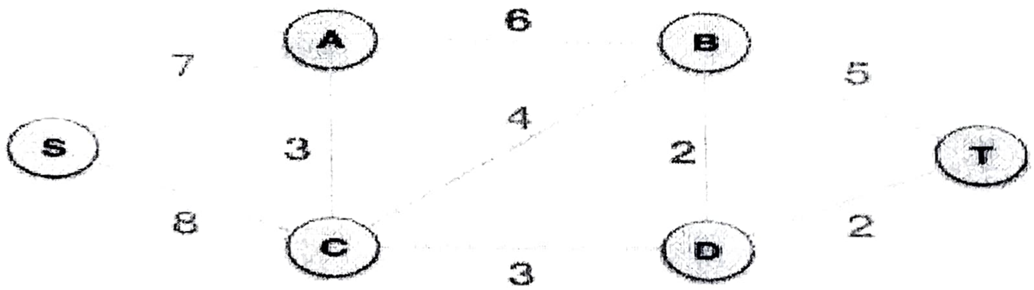
To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example –




**Step 1 - Remove all loops and parallel edges**




Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.

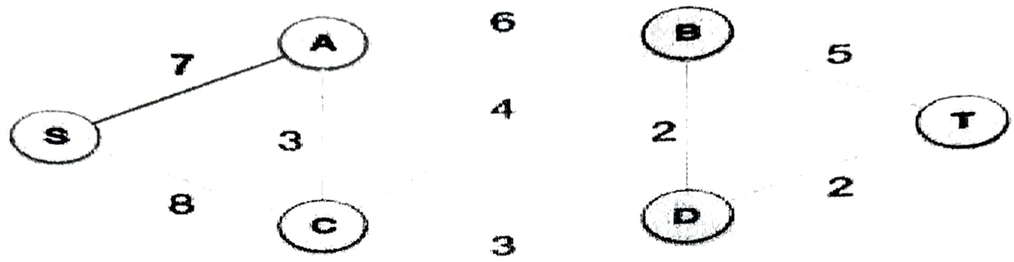



**Step 2 - Choose any arbitrary node as root node**

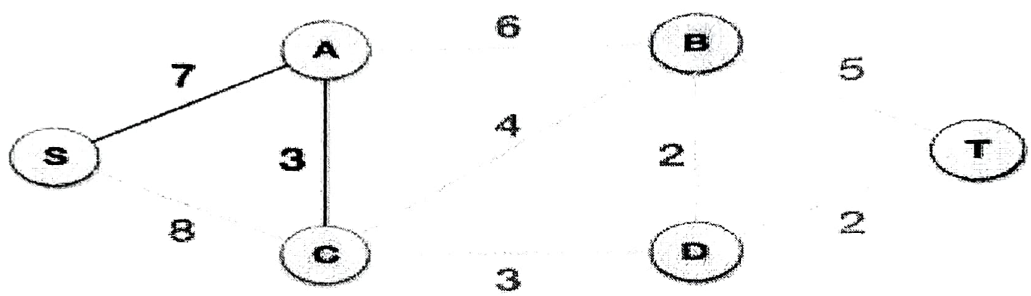
In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

**Step 3 - Check outgoing edges and select the one with less cost**

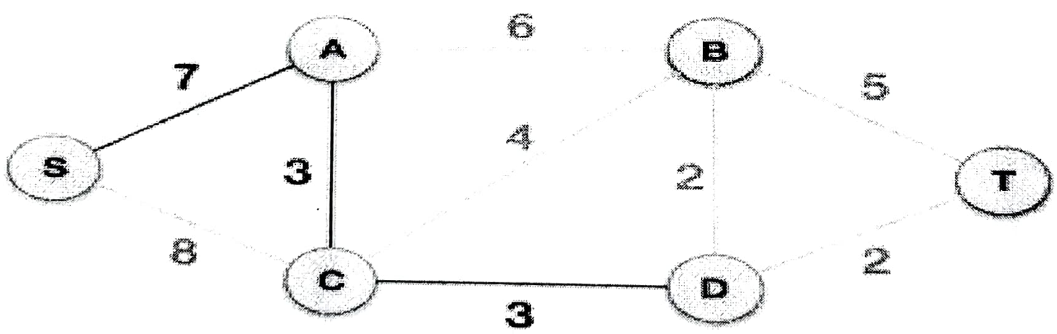
After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



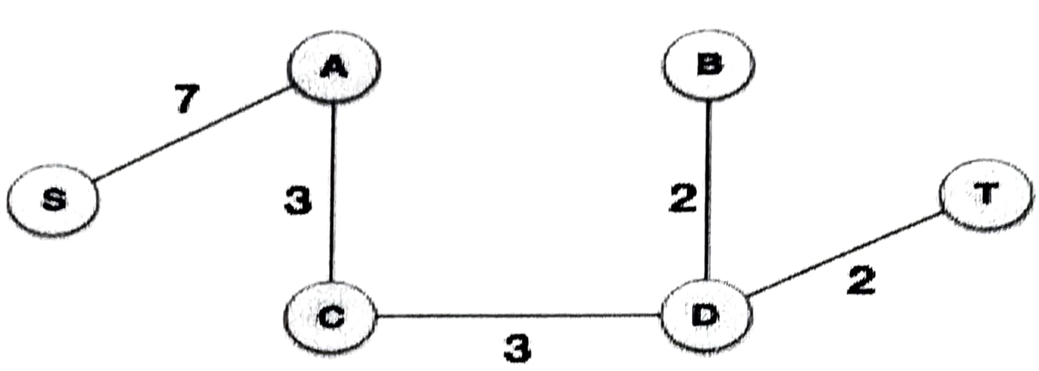

Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.




After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.




After adding node **D** to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.




We may find that the output spanning tree of the same graph using two different algorithms is same.

## Kruskal's Algorithm

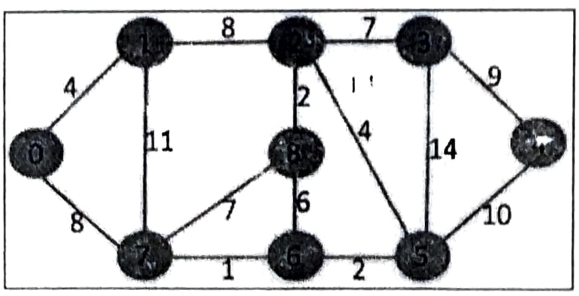
Kruskal's Algorithm builds the spanning tree by adding edges one by one into a growing spanning tree. Kruskal's algorithm follows greedy approach as in each iteration it finds an edge which has least weight and add it to the growing spanning tree.

Below are the steps for finding MST using Kruskal's algorithm

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far.
If cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are (V-1) edges in the spanning tree.

**Consider following example:**

The algorithm is a Greedy Algorithm. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far. Let us understand it with an example: Consider the below input graph.




The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having

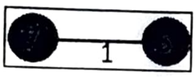
(9 – 1) = 8 edges.

After sorting:

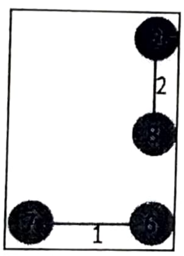
| Weight | Src | Dest |
|---|---|---|
| 1 | 7 | 6 |
| 2 | 8 | 2 |
| 2 | 6 | 5 |
| 4 | 0 | 1 |
| 4 | 2 | 5 |
| 6 | 8 | 6 |
| 7 | 2 | 3 |
| 7 | 7 | 8 |
| 8 | 0 | 7 |
| 8 | 1 | 2 |
| 9 | 3 | 4 |
| 10 | 5 | 4 |
| 11 | 1 | 7 |
| 14 | 3 | 5 |

Now pick all edges one by one from sorted list of edges

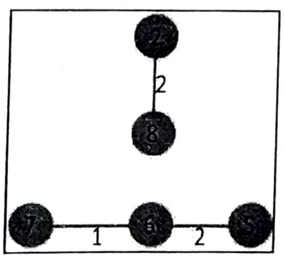
**1.** *Pick edge 7-6:* No cycle is formed, include it.



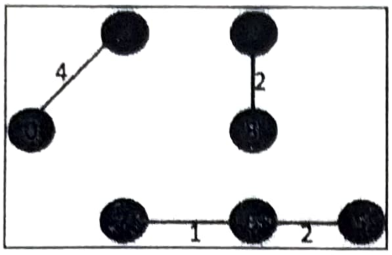

**2.** *Pick edge 8-2:* No cycle is formed, include it.



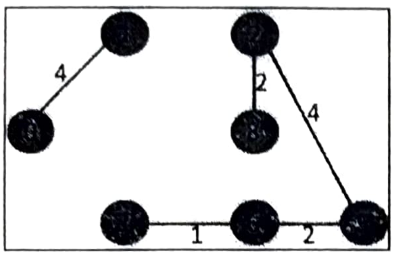

**3.** *Pick edge 6-5:* No cycle is formed, include it.




**4.** *Pick edge 0-1:* No cycle is formed, include it.

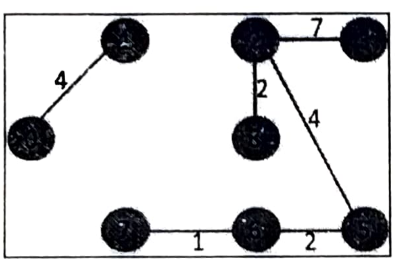



5. *Pick edge 2-5:* No cycle is formed, include it.



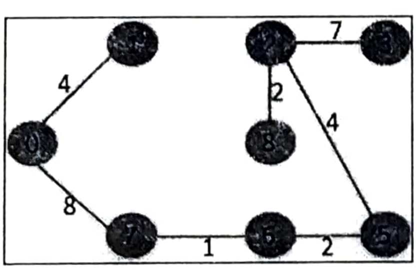

6. *Pick edge 8-6:* Since including this edge results in cycle, discard it.

7. *Pick edge 2-3:* No cycle is formed, include it.



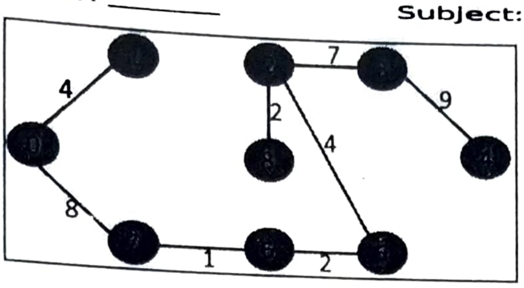

8. *Pick edge 7-8:* Since including this edge results in cycle, discard it.

9. *Pick edge 0-7:* No cycle is formed, include it.



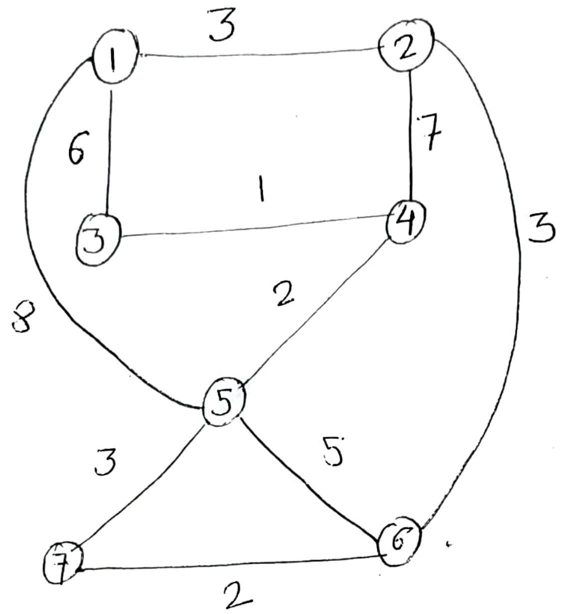

10. *Pick edge 1-2:* Since including this edge results in cycle, discard it.

11. *Pick edge 3-4:* No cycle is formed, include it.




Since the number of edges included equals (V – 1), the algorithm stops here.

# Prim's Algorithm




Find MST for the following graph using Prim's algorithm.
[10 Marks]

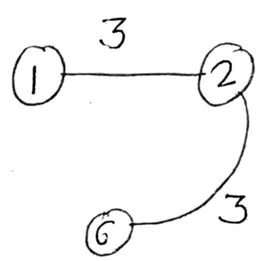
## Step 1 :-

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Distance | - | 3 | 6 | ∞ | 8 | ∞ | ∞ |
| Distance from | - | 1 | 1 | 1 | 1 | 1 | 1 |


↑ Node at minimum distance is selected

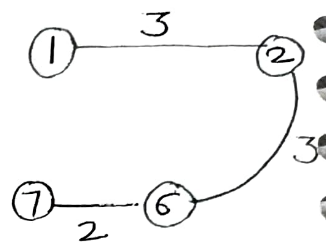
## Step 2 :-

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Distance | - | 3 | 6 | 7 | 8 | 3 | ∞ |
| Distance from | - | 1 | 1 | 2 | 1 | 2 | 1 |



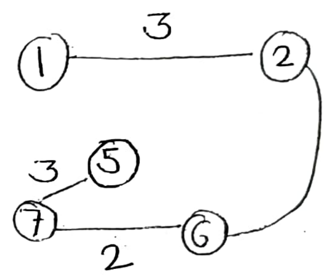
## Step 3 :-

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Distance | – | 3 | 6 | 7 | 5 | 3 | 2 |
| Distance from | – | 1 | 1 | 2 | 6 | 2 | 6 |



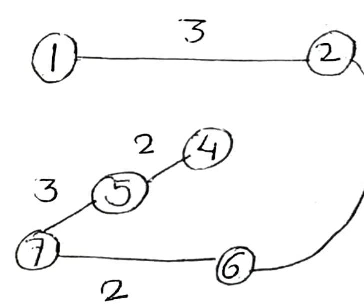

## Step 4 :-

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Distance | – | 3 | 6 | 7 | 3 | 3 | 2 |
| Distance from | – | 1 | 1 | 2 | 7 | 2 | 6 |



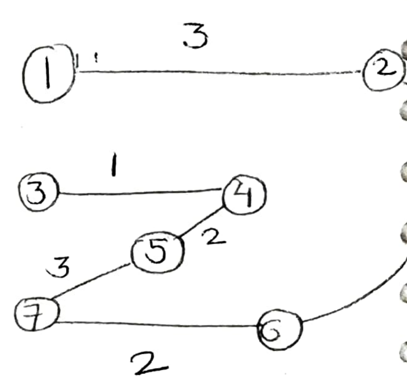

## Step 5 :-

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Distance | – | 3 | 6 | 2 | 3 | 3 | 2 |
| Distance from | – | 1 | 1 | 5 | 7 | 2 | 6 |




## Step 6 :-

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Distance | – | 3 | 1 | 2 | 3 | 3 | 2 |
| Distance from | – | 1 | 4 | 5 | 7 | 2 | 6 |

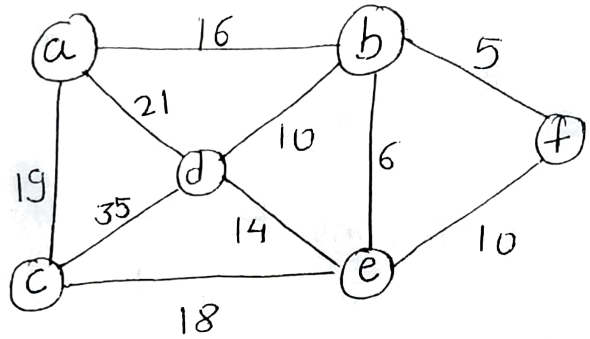


Cost of minimum spanning tree

$= 3 + 1 + 2 + 3 + 3 + 2$

$= 14$

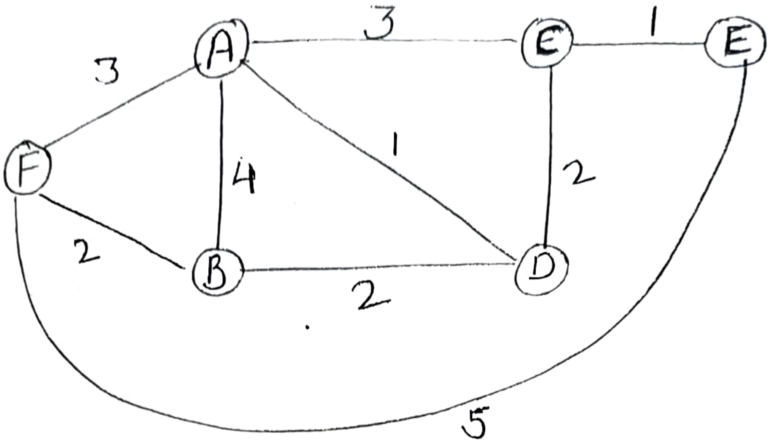
Example 2




Using Prim's Algorithm find minimum spanning tree for given graph.

[Dec 2015, 10 Marks]

# Kruskal's Algorithm




Draw the minimum cost spanning tree for the graph given below and also find its cost. Use kruskal's algorithm.

---

Step 1 :- Edges are sorted in ascending order on weight.

Edges: E1: AD 1

E2: CE 1

E3: BD 2

E4: CD 2

E5: BF 2

E6: AC 3

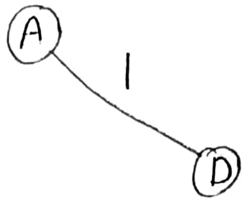
E7: AF 3

E8: AB 4

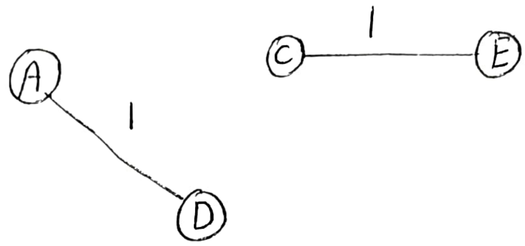
E9: EF 5

**Step 2** :- Edges are added in sequence E1 to E9 to spanning tree. If an edge forms a cycle, it is discarded.

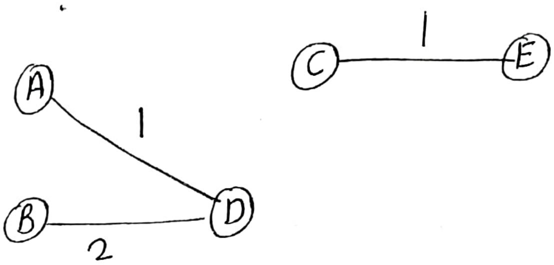
Add E1



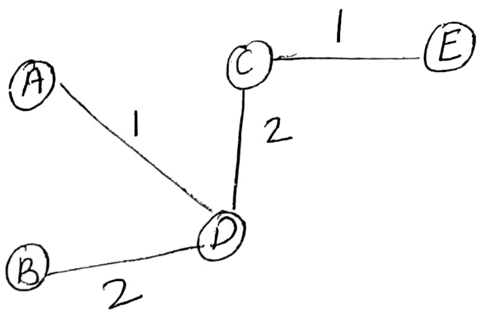

Add E2



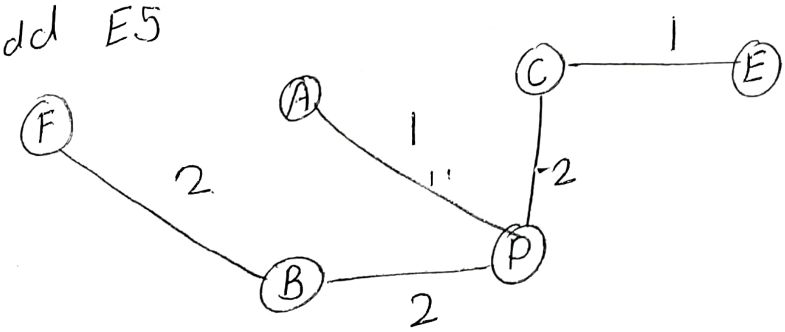

Add E3




Add E4




Add E5




As we have included $v - 1$ i.e. $6 - 1 = 5$ edges. Algorithm ~~does~~ stops at this iteration.

Cost of minimum spanning tree

$= 2 + 2 + 2 + 1 + 1$

$= 8$ unit