Step3. Repeat steps 4 and 5 for K = 1, 2,....., M :[Updates Q.]

Step4. Repeat step 5 for I = 1; 2,....., M:

Step5. Repeat for J = 1, 2,....., M:

Set Q [I, J] := MIN (Q [I, J], Q [I, K] + Q [K, J]).

[End of step 5 loop]

[End of step 4 loop]

[End of step 3 loop]

Step6. Exit.

7.7 Spanning Tree

A spanning tree of a graph is an undirected tree consisting of only those edges necessary to connect all the nodes in the original graph. A spanning tree has the properties that for any pair of nodes, there exist only one path between them and the insertion of any edge to a spanning tree form a unique cycle. Spanning tree finds application in obtaining an independent set of circuit equations for an electric network.

A spanning tree of a connected graph G contains all the nodes and has the edges, which connects all the nodes. So number of edges will be 1 less than the number of nodes. Let us take a graph G, shown in figure 7.7(i)

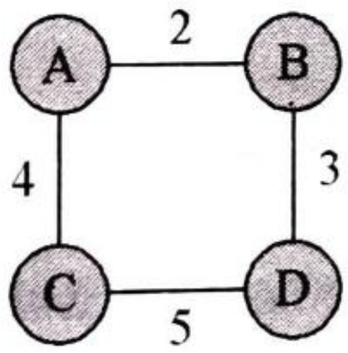


Fig. 7.7(i)

We know that a graph is a tree if there are no cycles in the graph. If we delete any one edge from the graph shown in figure 7.7(i), then we get 4 trees shown in figure 7.7.(ii).

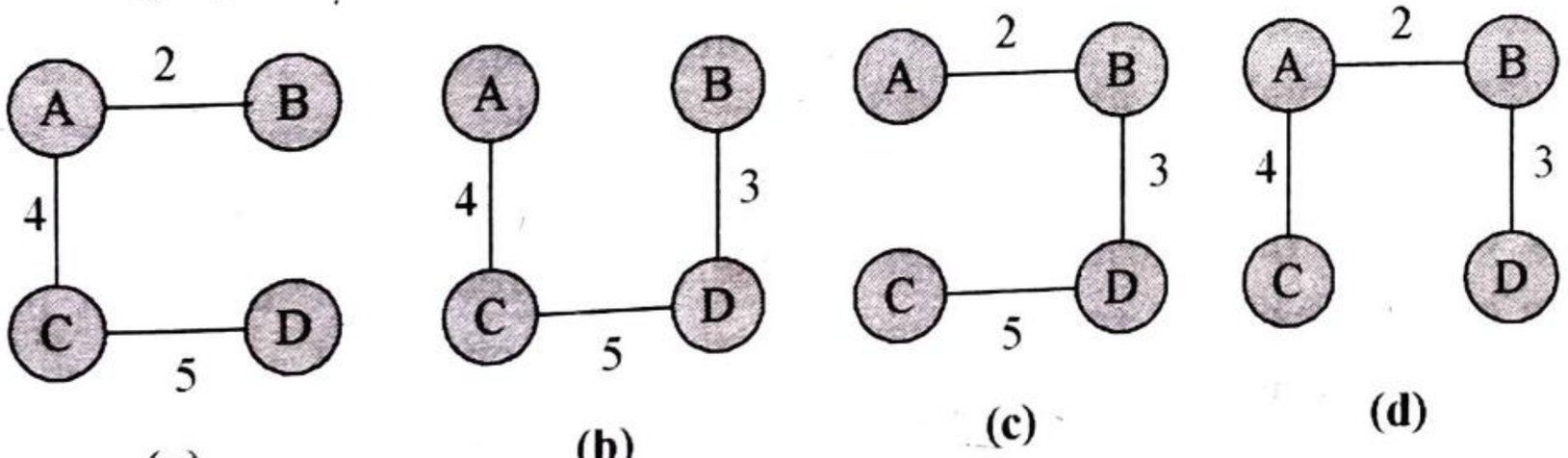


Fig. 7.7(ii)

Here all these trees are spanning trees. The number of edges are 3, which is 1 less than the number of nodes.

When determining the cost of a spanning tree of a weighted graph, the cost is simply the sum of the weights of the tree edges in figure 7.7 (ii) the costs of trees are 11, 12, 10 and 9 for trees a, b, c, and d respectively. A minimum cost spanning tree is formed when the edges are picked to minimize the total cost so, the minimum cost spanning tree among all 4 stated above is (d) with cost 9.

A method to obtain a minimum cost spanning tree builds this tree edge by edge. The next edge to include is choosen according to some optimization criterion. The simplest criterion is to choose an edge that result in a minimum increase in the sum of the cost of the edges so far included. There are two possible ways to interpret this criterion.

1. Kruskal's Algorithm

In this method the edges of the graph are considered in non decreasing order of cost. This interpretation is that the set 't' of edges so far selected for the spanning tree be such that it is possible to complete 't' into a tree. Therefore 't' may not be a tree at all stages in the algorithm. In fact, it will generally only be a forest since the set of edges 't' can be completed into a tree if and only if there are no cycles in 't'. This method is due to Kruskal.

2. Prim's Algorithm

In this method the set of edges so far included or selected form a tree. Therefore is A is the set of selected edges, then A forms a tree. The next edge (u, v) to be included in tree A is a minimum cost edge not in A with the property that (AU{u, v}) is also a tree. The corresponding algorithm is known as prim's algorithm. Let us discuss each method in details.

7.7.1 Kruskal's Algorithm

One approach to determine a minimum cost spanning tree of the graph has been given by Kruskal's algorithm. In this approach, a minimum cost spanning tree (T) is built edge by edge. Edges are considered for inclusion in tree T in node increasing order of their cost. An edge is inserted in tree T if it does not form a cycle with the edge already in tree T. Since graph G is connected and has n vertices ($n > 0$), then exactly n – 1 edges will be selected for inclusion in T.

In Kruskal's algorithm, initially E is the set of all edges in graph G. The only function, we wish to perform on this set are –

1. Determining an edge with minimum cost
2. Deleting that edge from graph and add to tree

Both of these functions can be performed efficiently if the edges in E are maintained as a sorted sequential list.

We have already discussed that a minimum cost spanning tree is formed when edges are picked to minimize the cost. Minimum spanning tree is the spanning tree in which the sum of weights (cost) on edges is minimum. It is a method in which edges of the graph are considered in increasing order of cost. Figure 7.7(a) shows a graph for which we want to create a minimum cost spanning tree using Kruskal's algorithm.

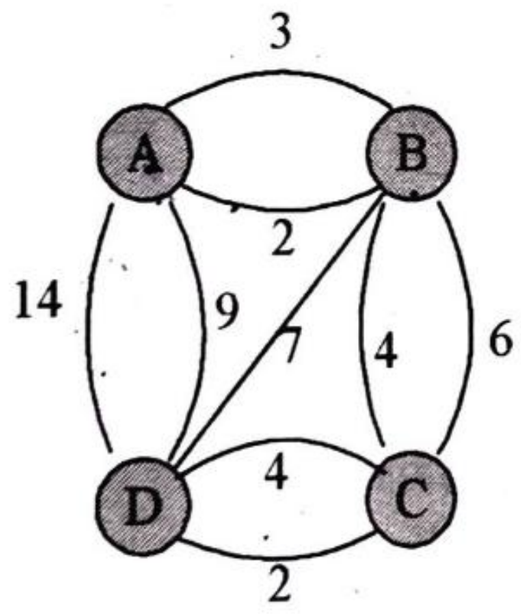


Fig. 7.7(a)

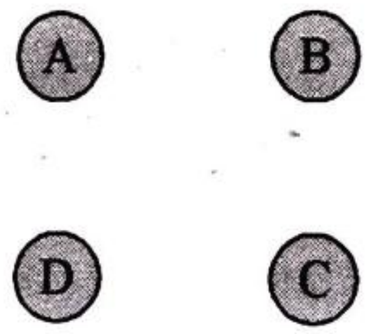
Now, at first we'll create list of edge with weight in graph.

edge	weight
AB	2
AB	3
BC	4
BC	6
BD	7
CD	4
CD	2
DA	9
DA	14

Now we'll sort this list in ascending order of weights:

edge	weight
AB	2
CD	2
AB	3
BC	4
CD	4
BC	6
BD	7
DA	9
DA	14

Now draw all the nodes (without any edge) of graph to get spanning tree:



Now, select first entry from edge list, delete it from edge list and draw it on spanning tree skeleton

edge	*weight*
~~*AB*~~	~~2~~
CD	2
AB	3
BC	4
CD	4
BC	6
BD	7
DA	9
DA	14

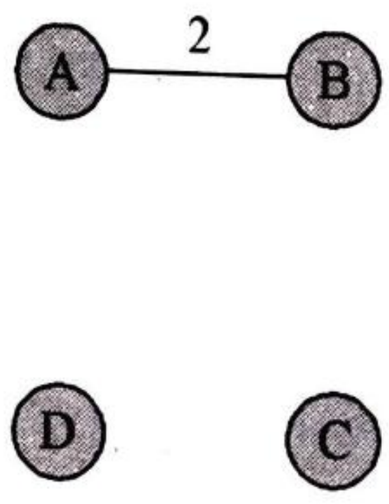


Fig. 7.7 (b)

Here, AB was shortest weighted edge and so it is drawn on skeleton for spanning tree and it has been cut from edge list.

Now, we will select second entry in edge list, which should not construct a cycle in tree. Here, CD is smallest weighted edge with weight = 2 and is not forming a cycle. Therefore it is added to tree, and cut from edge list.

edge	*Weight*
~~*AB*~~	~~2~~
~~*CD*~~	~~2~~
AB	3
BC	4
CD	4
BC	6
BD	7
DA	9
DA	14

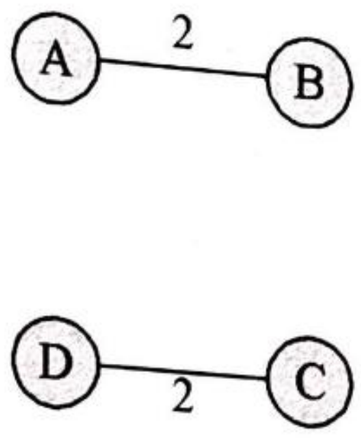


Fig. 7.7 (c)

Now, our aim is to select next smallest weighted edge but which is not forming cycle.

If we select AB, with 3, then it'll form cycle, but BC, with 4, will not from cycle. So, we will select BC as next edge and thus

edge	*Weight*
~~*AB*~~	~~2~~
~~*CD*~~	~~2~~
AB	3
~~*BC*~~	~~4~~
CD	4
BC	6
BD	7
DA	9
DA	14

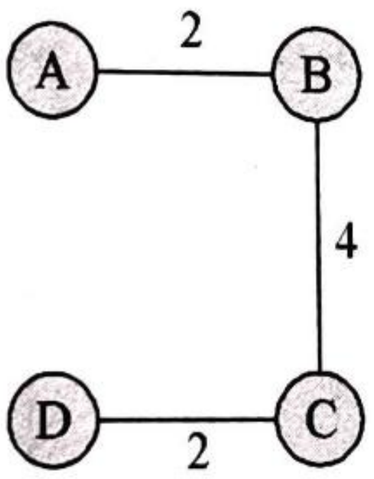


Fig. 7.7 (d)

And, this is our final spanning tree with minimum weight i.e. 2+4+ 2 = 8, and here no. of nodes = 4 so we are selecting only n–1 i.e. 3 edges.

Kruskal's Algorithm

This algorithm will create spanning tree with minimum weight, from a given weighted graph.

Step1. Begin

Step2. Create the edge list of given graph, with their weights.

Step3. Sort the edge list according to their weights in ascending order.

Step4. Draw all the nodes to create skeleton for spanning tree.

Step5. Pick up the edge at the top of the edge list (i.e. edge with minimum weights).

Step6. Remove this edge from the edge list

Step7. Connect the vertices in the skeleton with given edge. If by connecting the vertices, a cycle is created in the skeleton, then discard this edge.

Step8. Repeat steps 5 to 7, until n–1 edges are added or list of edges is over

Step9. Return

7.7.2 Prim's Algorithm

There is one more approach to determine minimum cost spanning tree given by prim's algorithm. In this case, we start with single edge of graph, and we add edges to it and finally we get minimum cost tree. In this case, as well, we have n–1 edges when number of nodes in graph are n.

In this case, we again and again add edges to tree, and tree is extended to create spening tree, while in case of Kruskal's algorithm, there may be more than one tree, which is finally connected through edge to create spanning tree.

To illustrate this method, we take figure 7.8 (a) as example. Figure 7.8 (a) is the graph for which we want to create spanning tree

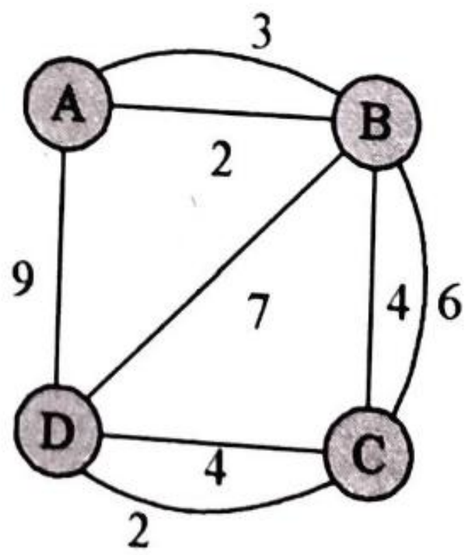


Fig. 7.8 (a)

At first we'll create edge list (with weights) of this graph.

edge	weight
AB	2
AB	3
AD	9
BC	4
BC	6
BD	7
CD	4
CD	2

and create skeleton for spanning tree

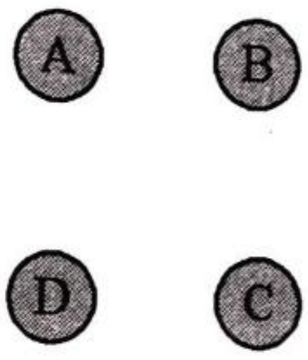


Fig. 7.8 (b)

Now, select any edge with minimum weight from edge list, and there are 2 edges AB and CD with weight 2, so we'll select any one from them, and delete it from edge list and add it to skeleton.

edge	weight
~~AB~~	~~2~~
AB	3
AD	9
BC	4
BC	6
BD	7
CD	4
CD	2

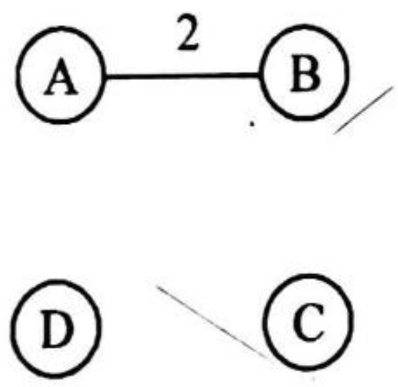


Fig. 7.8 (c)

Now, select edges, whose one end point is either A or B, and such edges are AB(3), AD(9), BC(4), BC(6), BD (7). Out of all these, AB forms cycle so discard it, and among others, BC(4) is edge with minimum weight, and so.

edge	weight
~~AB~~	~~2~~
AB	3
AD	9
~~BC~~	~~4~~
BC	6
BD	7
CD	4
~~CD~~	2

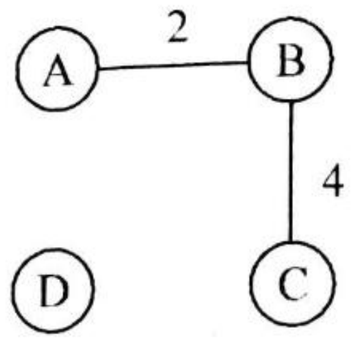


Fig. 7.8 (d)

Now, we need to select edges, whose one end point is either A, B or C and these edges are–

AB (3), AD (9), BC (6), BD (7), CD (4), CD (2)

But AB (3), BC (6), forms cycle so discard them and so, remaining edges are

AD (9), BD (7), CD (4), CD (2)

and out of all these edges, CD is edge with smallest weight i.e. 2 so we'll operate on this edge as:

edge	weight
~~AB~~	~~2~~
AB	3
AD	9
~~BC~~	~~4~~
BC	6
BD	7
CD	4
~~CD~~	~~2~~

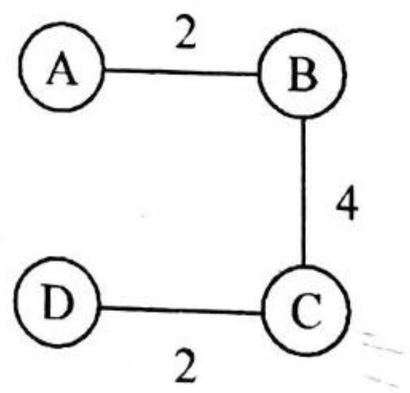


Fig. 7.8 (e)

As we have added n–1 i.e. 3 edges to tree so no more edges are needed.

Prim's Algorithm

This algorithm creates spanning tree with minimum weight from a given weighted graph.

Step1. Begin

Step2. Create edge list of given graph, with their weights.

Step3. Draw all nodes to create skeleton for spanning tree.

Step4. Select an edge with lowest weight and add it to skeleton and delete edge from edge list.

Step5. Add other edges. While adding an edge take care the one end of the edge should always be in the skeleton tree and its cost should be minimum.

Step6. Repeat step 5 until n–1 edges are added.

Step7. Return

7.8 Operations on Graph

As we have seen, a graph can be represented in two ways.

1. Adjacency matrix
2. Adjacency list

The two main operations on graph will be

1. Insertion
2. Deletion

Here insertion and deletion will be also on two things -

1. On node
2. On edge

Now we will describe insertion and deletion operation on adjacency matrix and adjacency list.

7.8.1 Insertion in Adjacency Matrix

1. Node Insertion

Insertion of node requires only addition of one row and one column with zero entries in that row and column. Let us take a graph G shown in figure 7.8(i)