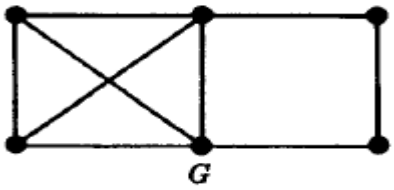
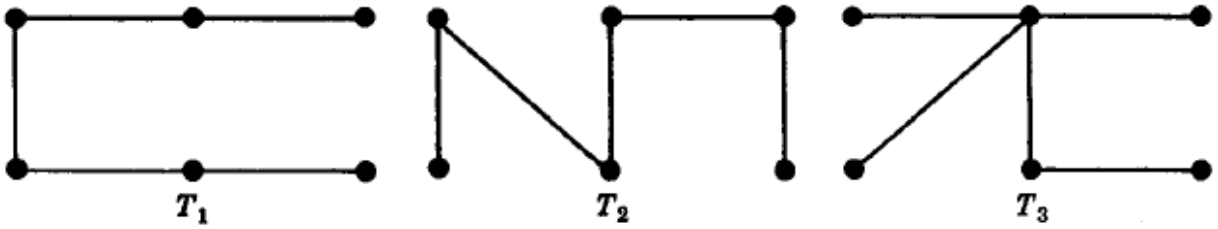
Spanning Trees

- we have discussed the tree and its properties when it occurs as a graph by itself. Now we shall study the tree as a **subgraph of another graph**.
- A given graph has numerous subgraphs, Obviously, some of these subgraphs will be trees. Out of these trees we are particularly interested in certain types of trees, called spanning trees.
- A subgraph T of a connected graph G is called a spanning tree of G if T is a tree and T includes all the vertices of G.





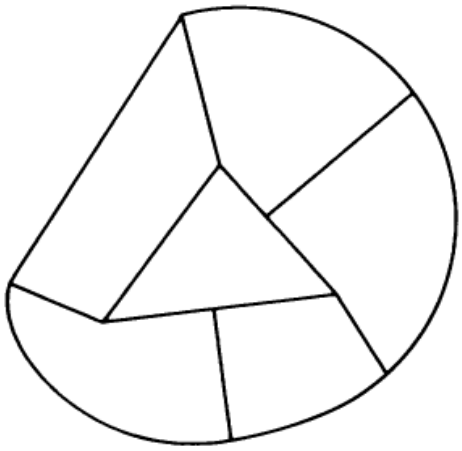
- Since the vertices of G are barely hanging together in a spanning tree, it is a sort of skeleton of the original graph G. This is why a spanning tree is sometimes referred to as a **skeleton or scaffolding of G**.
- Since spanning trees are the largest (with maximum number of edges) trees among all trees in G, it is also quite appropriate to call a spanning tree a **maximal tree subgraph** or **maximal tree** of G.
- It is to be noted that a spanning tree is **defined only for a connected graph**, because a tree is always connected, and in a disconnected graph of n vertices we cannot find a connected subgraph with n vertices. Each component (which by definition is connected) of a disconnected graph, however, does have a spanning tree. Thus, a disconnected graph with k components has a spanning forest consisting of κ spanning trees. (A collection of trees is called a forest.)
- **Finding a spanning tree** of a connected graph G: If G has no circuit, it is its own spanning tree. If G has a circuit, delete an edge from the circuit. This will still leave the graph connected. If there are more circuits, repeat the operation till an edge from the last circuit is deleted—leaving a connected, circuit-free graph that contains all the vertices of G.

Theorem: Every connected graph has at least one spanning tree.

Theorem: With respect to any of its spanning trees, a connected graph of n vertices and e edges has n-1 tree branches and e - n +1 chords.

Eg: For example, if we have an electric network with e elements (edges) and n nodes (vertices), what is the minimum number of elements we must remove to eliminate all circuits in the network? The answer is e-n+1.

Example: if we have a farm consisting of six walled plots of land, as shown in below figure, and these plots are full of water, how many walls will have to be broken so that all the water can be drained out?



Sol^n: Here n = 10 and e= 15. We shall have to select a set of six (15 — 10 + 1 =6) walls such that the remaining nine constitute a spanning tree. Breaking these six walls will drain the water out.

Minimum Spanning Trees

- Suppose G is a connected weighted graph. That is, each edge of G is assigned a **nonnegative number** called the weight of the edge.
- Then any spanning tree T of G is assigned a total weight obtained by adding the weights of the edges in T. A minimal spanning tree of G is a spanning tree whose total weight is as small as possible.
- The weight of a minimal spanning tree is unique, but the minimal spanning tree itself is not. Different minimal spanning trees can occur when two or more edges have the same weight.
- Following algorithms enable us to find a minimal spanning tree T of a connected weighted graph G where G has n vertices. (In which case T must have n − 1 edges.)

Algorithm 8.2: The input is a connected weighted graph G with n vertices.

Step 1. Arrange the edges of G in the order of decreasing weights.

Step 2. Proceeding sequentially, delete each edge that does not disconnect the graph until $n-1$ edges remain.

Step 3. Exit.

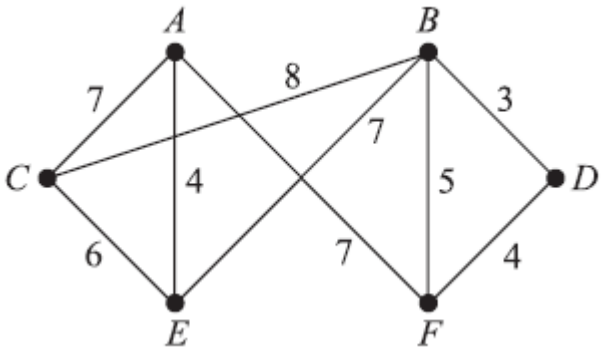
Algorithm 8.3 (Kruskal): The input is a connected weighted graph G with n vertices.

Step 1. Arrange the edges of G in order of increasing weights.

Step 2. Starting only with the vertices of G and proceeding sequentially, add each edge which does not result in a cycle until $n-1$ edges are added.

Step 3. Exit.

Example: Find a minimal spanning tree of the weighted graph Q given below:



Sol^{n}: (Using First Algo)

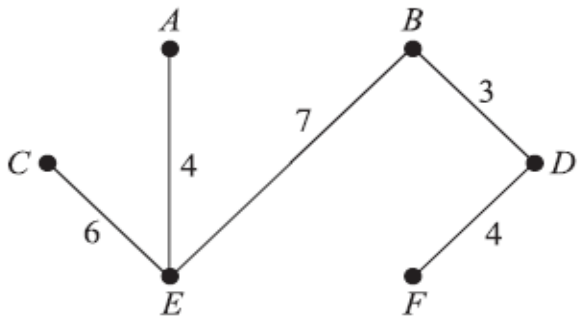
First we order the edges by decreasing weights, and then we successively delete edges without disconnecting Q until five edges remain. This yields the following data:

Edges	BC	AF	AC	BE	CE	BF	AE	DF	BD
Weight	8	7	7	7	6	5	4	4	3
Delete	Yes	Yes	Yes	No	No	Yes			

Thus, the minimal spanning tree of Q which is obtained contains the edges

BE, CE, AE, DF, BD

The spanning tree has weight 24 and it is shown below:



Sol^{n} (Using Kruskal Algo)

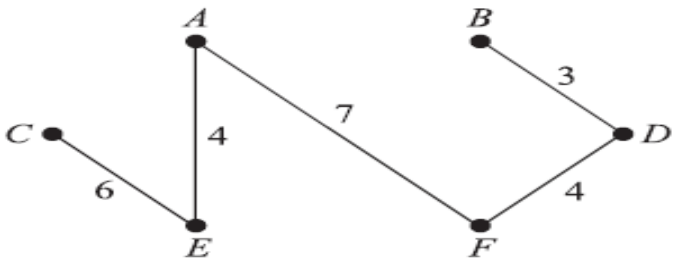
First we order the edges by increasing weights, and then we successively add edges without forming any cycles until five edges are included. This yields the following data:

Edges	BD	AE	DF	BF	CE	AC	AF	BE	BC
Weight	3	4	4	5	6	7	7	7	8
Add?	Yes	Yes	Yes	No	Yes	No	Yes		

Thus the minimal spanning tree of Q which is obtained contains the edges

$$BD, \quad AE, \quad DF, \quad CE, \quad AF$$

The spanning tree is shown below. Observe that this spanning tree is not the same as the one obtained using previous algorithm as expected it also has weight 24.



Prim's Algorithm:

Prim's Algorithm

All vertices of connected graph are included in minimum spanning tree of a graph G. Prim's algorithm starts with one vertex and grows the rest of the tree one vertex at a time, by adding associated edge. This algorithm builds a tree by iteratively adding edges until a minimal sapping tree is obtained, that is, till all nodes are added. At each iteration a minimum weight edge that does not complete a cycle is added to the existing tree.

Let $G = (V, E)$ be an original graph. Let T be a spanning tree. $T = (A, B)$, where initially A and B are empty sets. Let us select an arbitrary vertex i from V and add it to A. Now $A = \{i\}$. At each step prim's algorithm looks for the shortest possible edge $<u, v>$ such that $u \in A$ and $v \in V - A$. It then adds v to $A (A = A \cup \{v\})$ and adds edge $<u, v>$ to B. In this way, the edges in B form at any instant a minimum spanning tree for the vertices in A. We continue as long as $A \neq V$.

To illustrate the algorithm, let us consider the graph in Fig. 8.19.

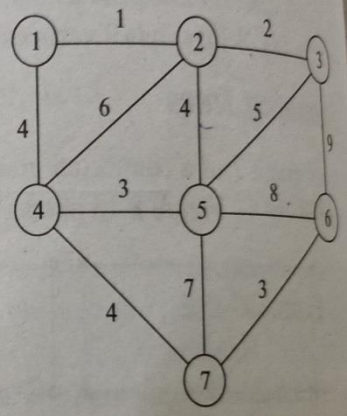


Fig. 8.19 *A weighted graph*

Let us select node 1 as the starting node. Following Table 8.1 shows the edge of minimum weight selected and set of vertices A.

Table 8.1

Step No.	*Edge <u, v>*	*Set A*
Initial	–	{1}
1	<1, 2>	{1, 2}
2	<2, 3>	{1, 2, 3}
3	<1, 4>	{1, 2, 3, 4}
4	<4, 5>	{1, 2, 3, 4, 5}
5	<4, 7>	{1, 2, 3, 4, 5, 7}
6	<7, 6>	{1, 2, 3, 4, 5, 7, 6}

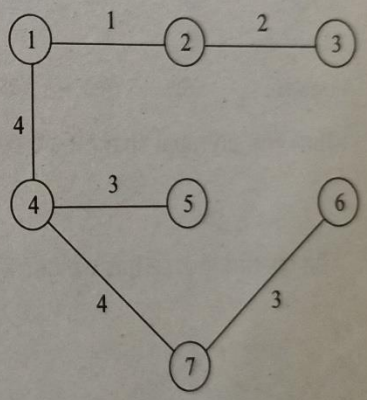


Fig. 8.20 *Minimum spanning tree*

When the algorithm stops, B contains the chosen edges $B = \{<1, 2>, <2, 3>, <1, 4>, <4, 5>, <4, 7>, <7, 6>\}$

The resultant spanning tree is drawn in Fig. 8.20 and is of weight =17.

Algorithm 8.8C (Prim's method): *The input is a connected weighted graph G of n vertices.*

{This algorithm generates minimum a spanning tree.}

Here G is graph and T is a spanning tree to be computed.

Step 1. Let $G = \{V, E\}$ and $T = \{A, B\}$

$A = \Phi$ and $B = \Phi$

Step 2. Let $i \in V$, i is a start vertex.

Step 3. $A = A \cup \{i\}$

Step 4. While $A \neq V$ do

begin

Find edge $\{u, v\} \in E$ of minimum length such that {also edge does not complete any cycle

$u \in A$ and $v \in V - A$

$A = A \cup \{v\}$ and

$B = B \cup \{u, v\}$

end

Step 5. Stop