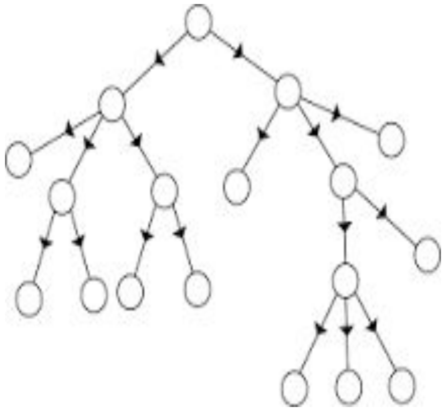
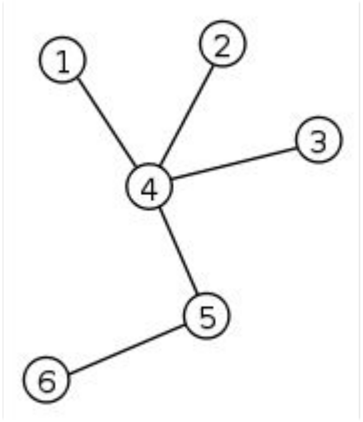
Trees

- Trees-these are connected undirected graph with no simple circuit
- Rooted trees –Is a tree in which one vertex is designated as root.





Tree examples

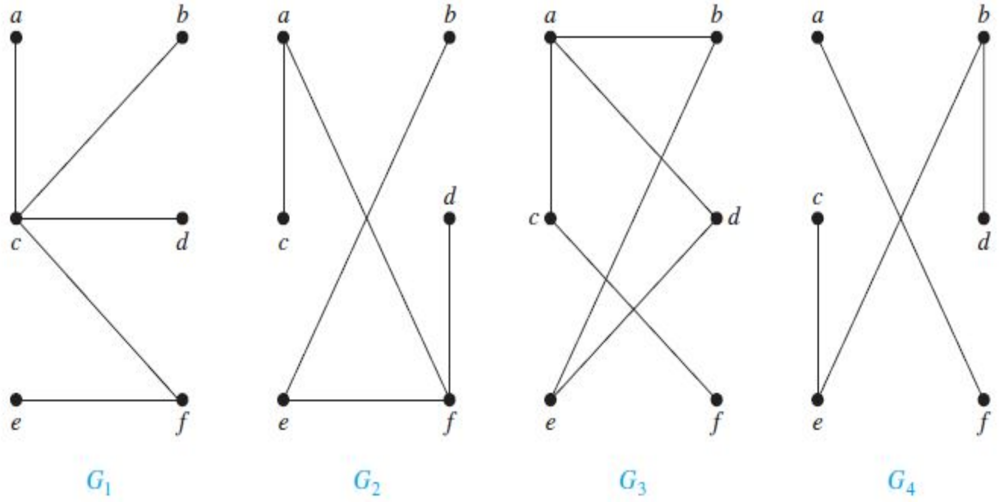
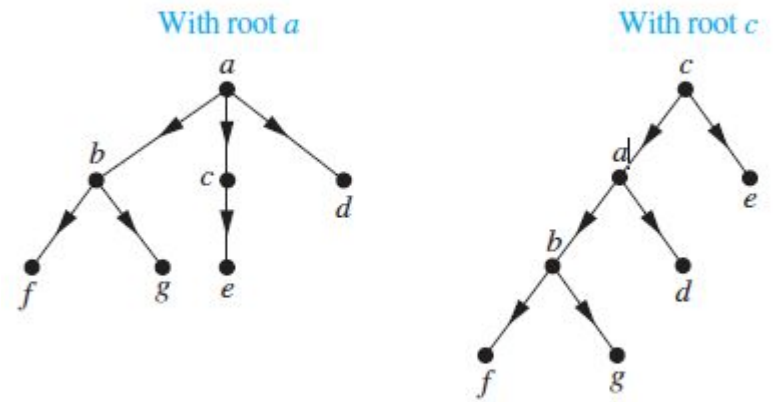


FIGURE 2 **Examples of trees and graphs that are not trees.**

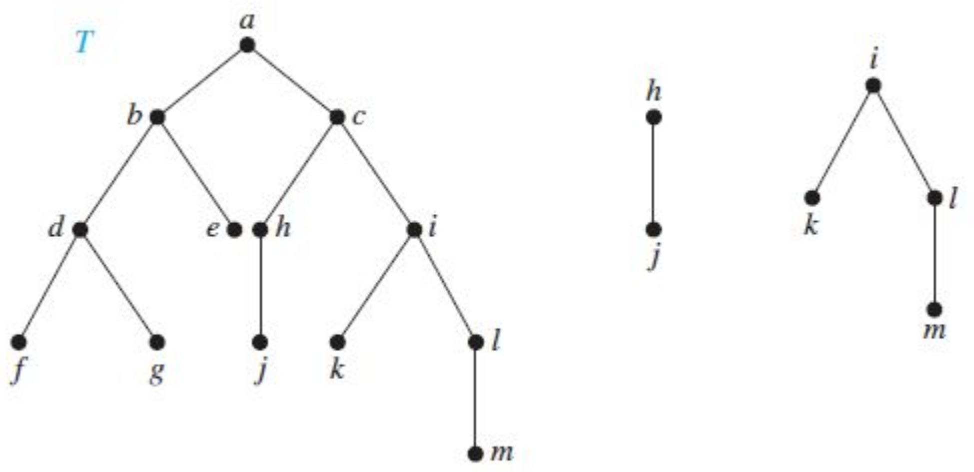
Solution: G_1 and G_2 are trees, because both are connected graphs with no simple circuits. G_3 is not a tree because e, b, a, d, e is a simple circuit in this graph. Finally, G_4 is not a tree because it is not connected.

- An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.



A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children. The tree is called a *full m-ary tree* if every internal vertex has exactly m children. An m-ary tree with $m = 2$ is called a *binary tree*.

Find the left and right child of d and subtree of c



(a) (b) (c)

Tree as model for organization

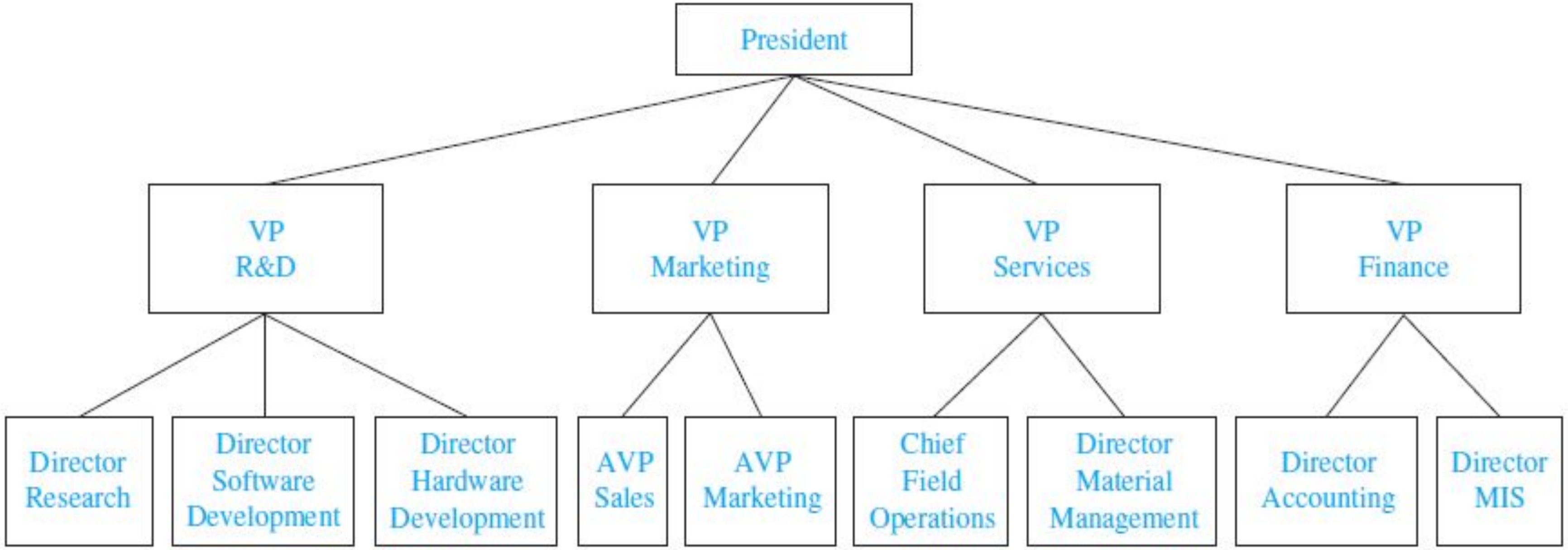
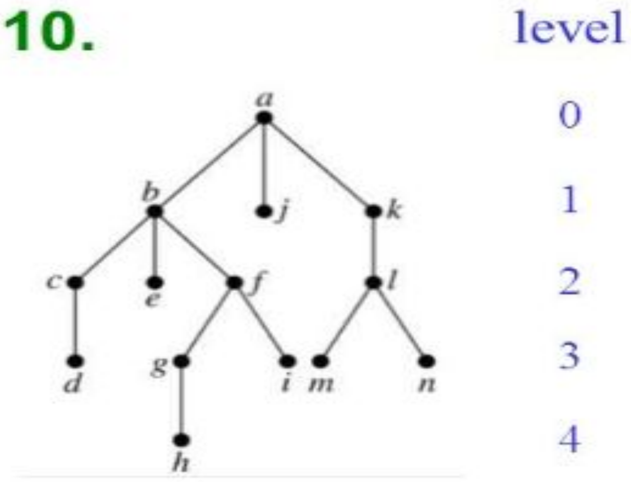


FIGURE 10 An organizational tree for a computer company.

- A tree with n vertices has n-1 edges.
- A full m-ary tree with I internal vertices contains n=mi+1 vertices.

Def: The level of a vertex v in a rooted tree is the length of the unique path from the root to this vertex. The level of the root is defined to be zero.
The height of a rooted tree is the maximum of the levels of vertices.

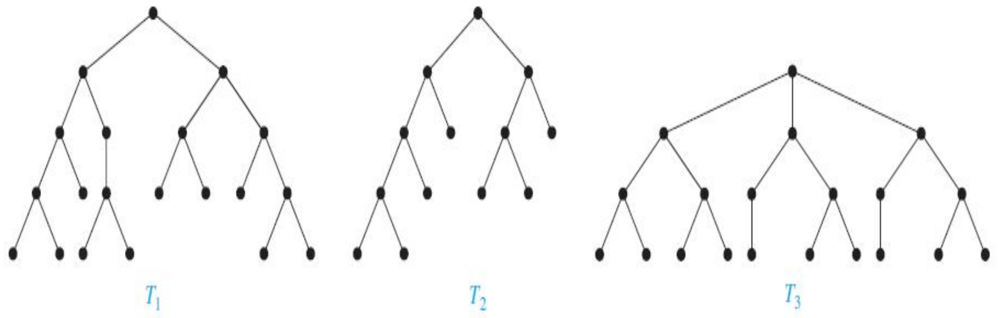
Example 10.



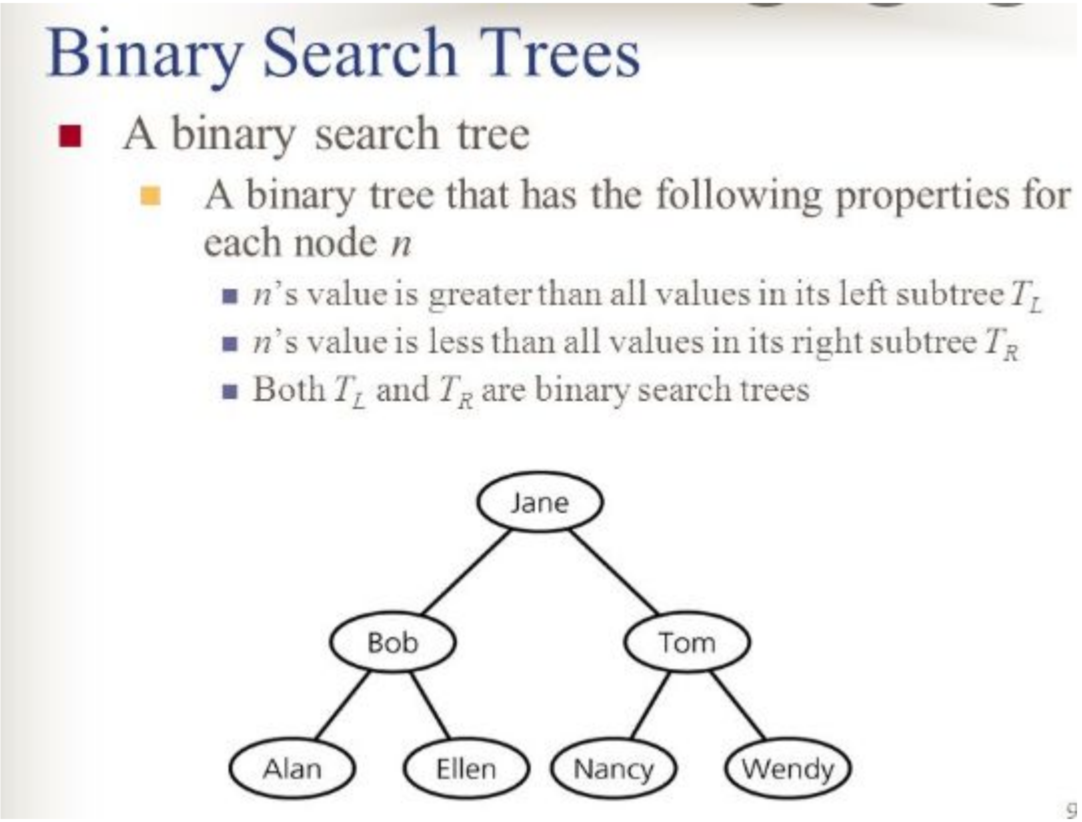
height $= 4$

Balanced tree

- A rooted m-ary tree of height h is balanced if all leaves are at level h or h-1
- In fig T1 is balanced as all leaves are present at level 3 &4.
- T2 is not balanced
- T3 is balanced as all leaves are at level 3.



Binary Search Tree



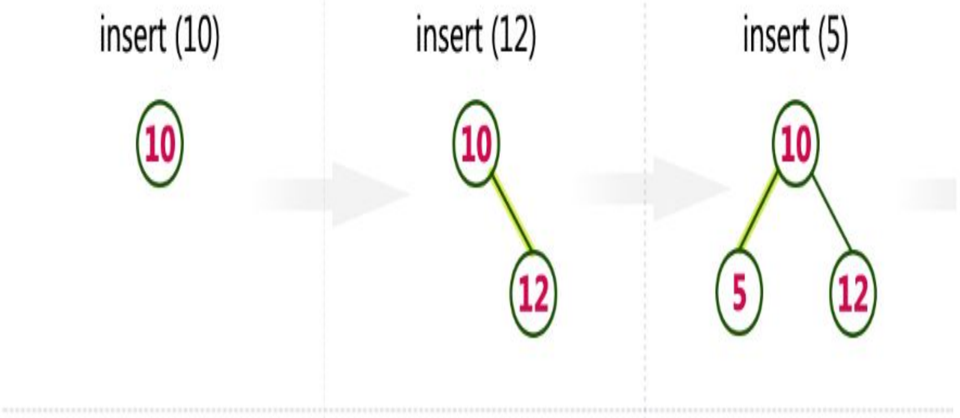
- Binary search trees have the property that the node to the left contains a smaller value than the node pointing to it and the node to the right contains a larger value than the node pointing to it.

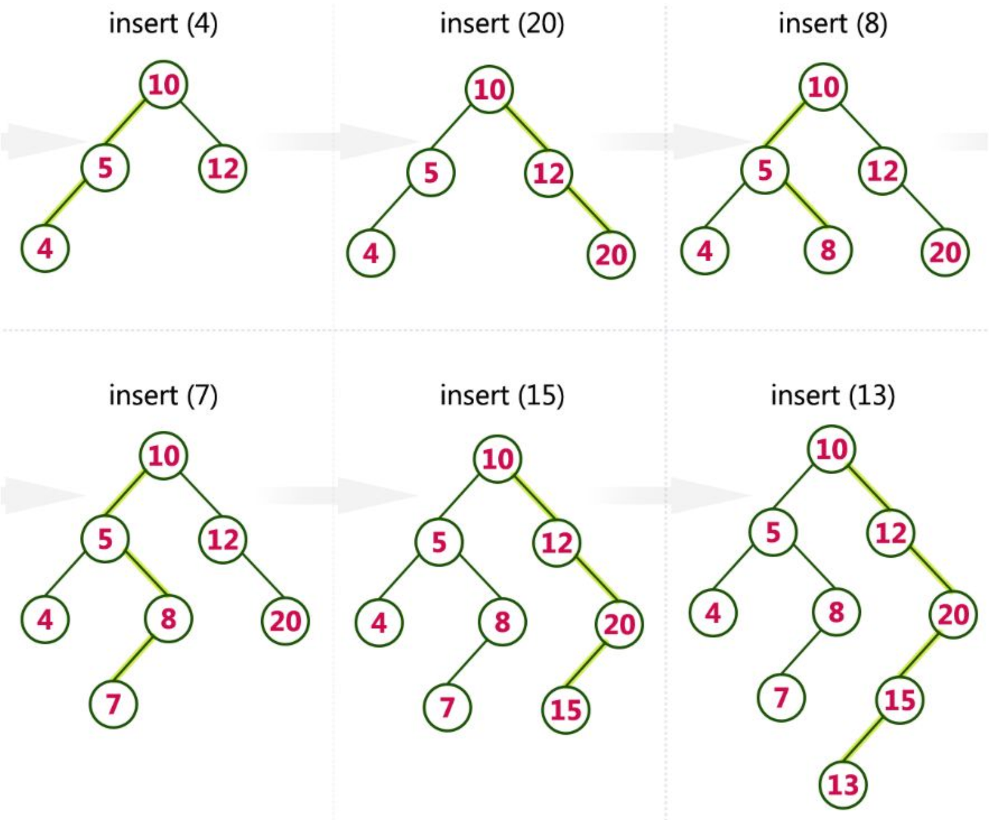
Example

Construct a Binary Search Tree by inserting the following sequence of numbers...

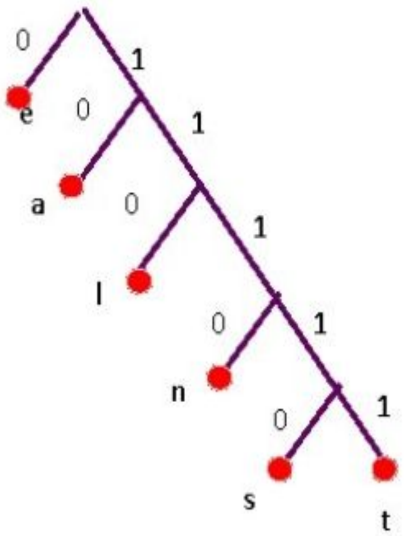
10,12,5,4,20,8,7,15 and 13

Above elements are inserted into a Binary Search Tree as follows...





Prefix codes



In typical English texts, e is most frequent, followed by, l, n, s, t ... The prefix tree assigns to each letter of the alphabet a code whose length depends on the frequency:

e = 0, a = 10, l= 110, n = 1110 etc

Such techniques are popular for data compression purposes. The resulting code is a variable-length code.

- Encoding of alphabets using bitstrings of different length.
- A prefix code can be represented using binary tree where characters are labels of the tree
- Tree representing a code can be used to decode a bit string

Huffman coding

- Used for data compression
- Symbols and frequencies are given as input and as output generate prefix code.
- Construct the rooted binary tree where symbols are the labels of trees
- Procedure
- At each step, we combine two trees having the least total weight into a single tree
- by introducing a new root and placing the tree with larger weight as its left subtree and the tree
- with smaller weight as its right subtree. Furthermore, we assign the sum of the weights of the
- two subtrees of this tree as the total weight of the tree. (Although procedures for breaking ties by
- choosing between trees with equal weights can be specified, we will not specify such procedures

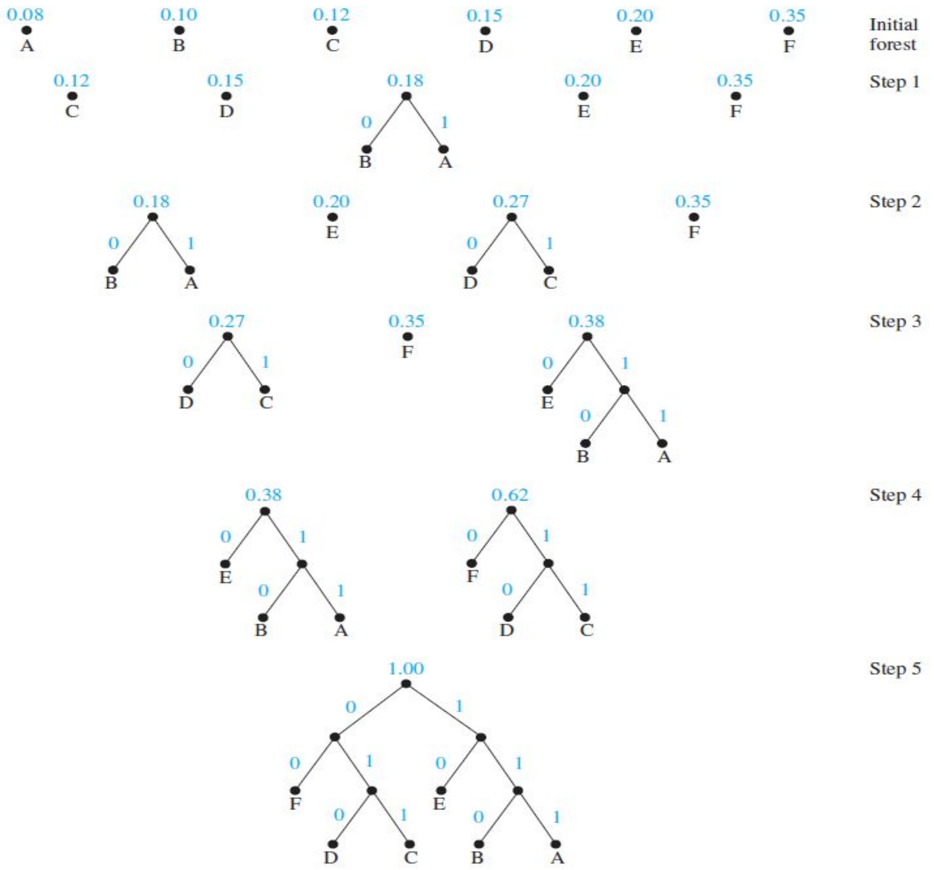
ALGORITHM 2 Huffman Coding.

procedure *Huffman*(C: symbols a_i with frequencies w_i, $i = 1, \ldots, n$)
$F :=$ forest of n rooted trees, each consisting of the single vertex a_i and assigned weight w_i
while F is not a tree
  Replace the rooted trees T and T' of least weights from F with $w(T) \geq w(T')$ with a tree having a new root that has T as its left subtree and T' as its right subtree. Label the new edge to T with 0 and the new edge to T' with 1.
  Assign $w(T) + w(T')$ as the weight of the new tree.
{the Huffman coding for the symbol a_i is the concatenation of the labels of the edges in the unique path from the root to the vertex a_i}

Example: **Use Huffman coding to encode the following symbols with the frequencies listed: A: 0.08, B: 0.10, C: 0.12, D: 0.15, E: 0.20, F: 0.35. What is the average number of bits used to encode a character?**

Solution: Figure 6 displays the steps used to encode these symbols. The encoding produced encodes A by 111, B by 110, C by 011, D by 010, E by 10, and F by 00. The average number of bits used to encode a symbol using this encoding is

$$3 \cdot 0.08 + 3 \cdot 0.10 + 3 \cdot 0.12 + 3 \cdot 0.15 + 2 \cdot 0.20 + 2 \cdot 0.35 = 2.45.$$



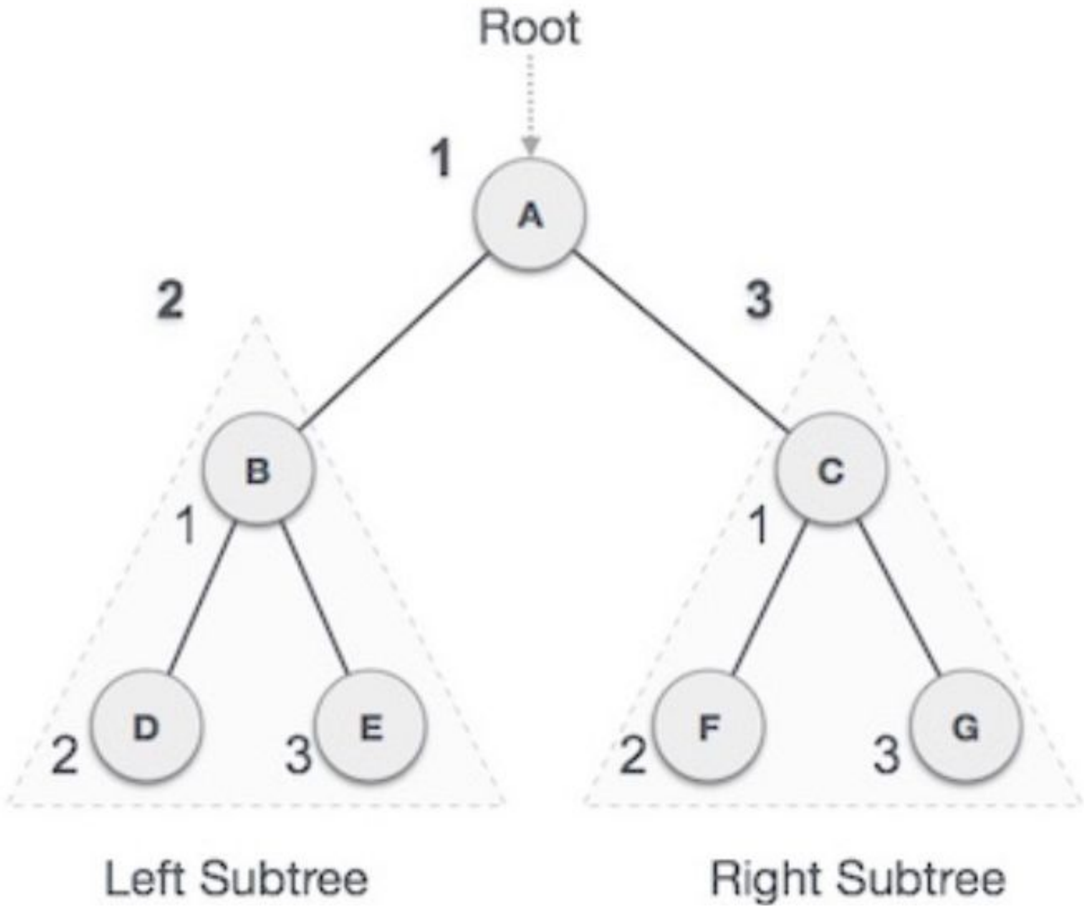
Tree Traversal

- Procedure for systematically visiting every vertex of ordered rooted tree are traversal algorithms.
- Pre ordered traversal
- In ordered Traversal
- Post ordered Traversal

Pre ordered traversal

- In this traversal method root is visited first then Left subtree then Right subtree.
- Algorithm
- Until all nodes are traversed
- Step 1:Visit Root node
- Step 2:Recursively traverse Left subtree
- Step 3: Recursively traverse Right subtree

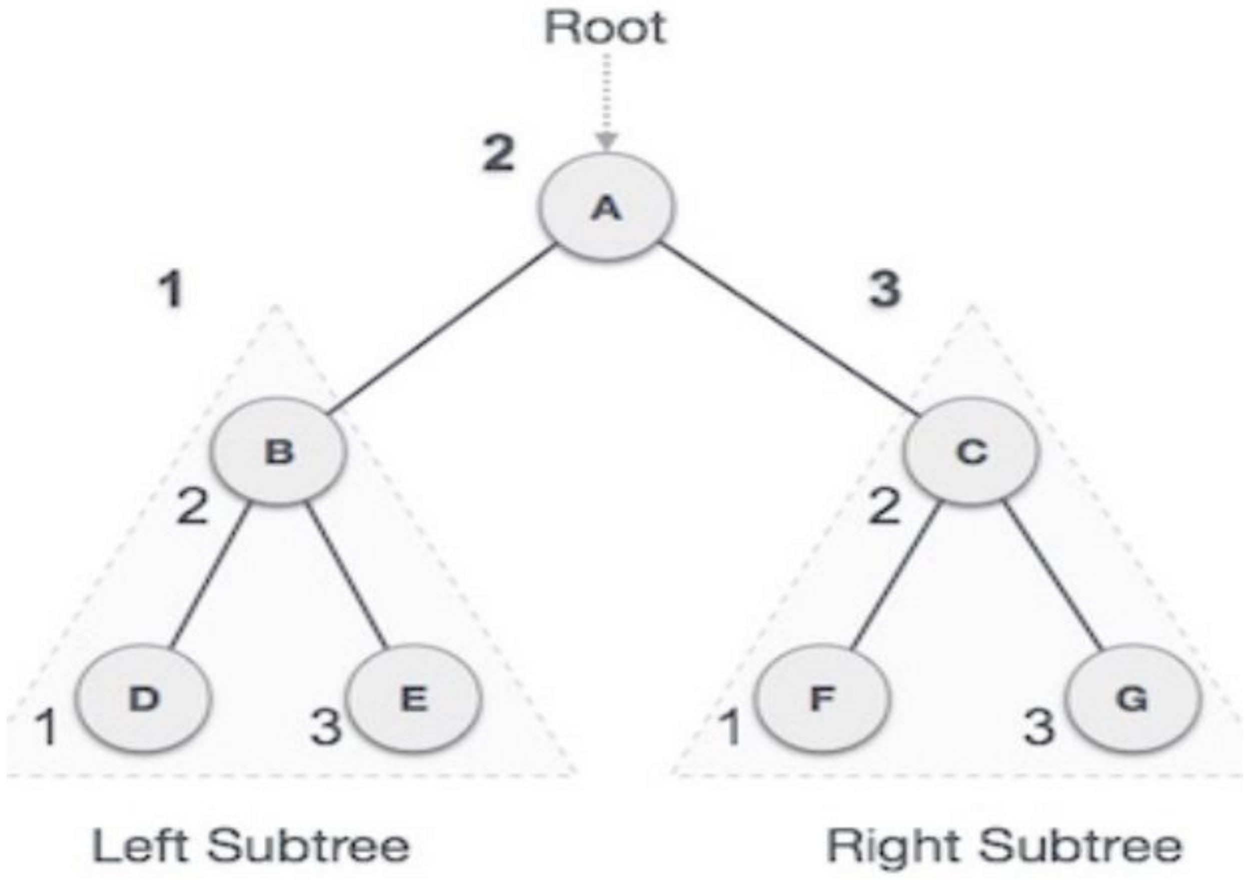
A->B->D->E->C->F->G



In ordered Traversal

- In this Traversal Left subtree is visited first then Root then Right subtree. Every node may have subtree.
- If binary tree is traversed in In ordered the output will produce key values in ascending order.
- Algorithm
- Until all nodes are traversed
- Step 1:Recursively traverse Left subtree
- Step 2:Visit Root node
- Step 3: Recursively traverse Right subtree

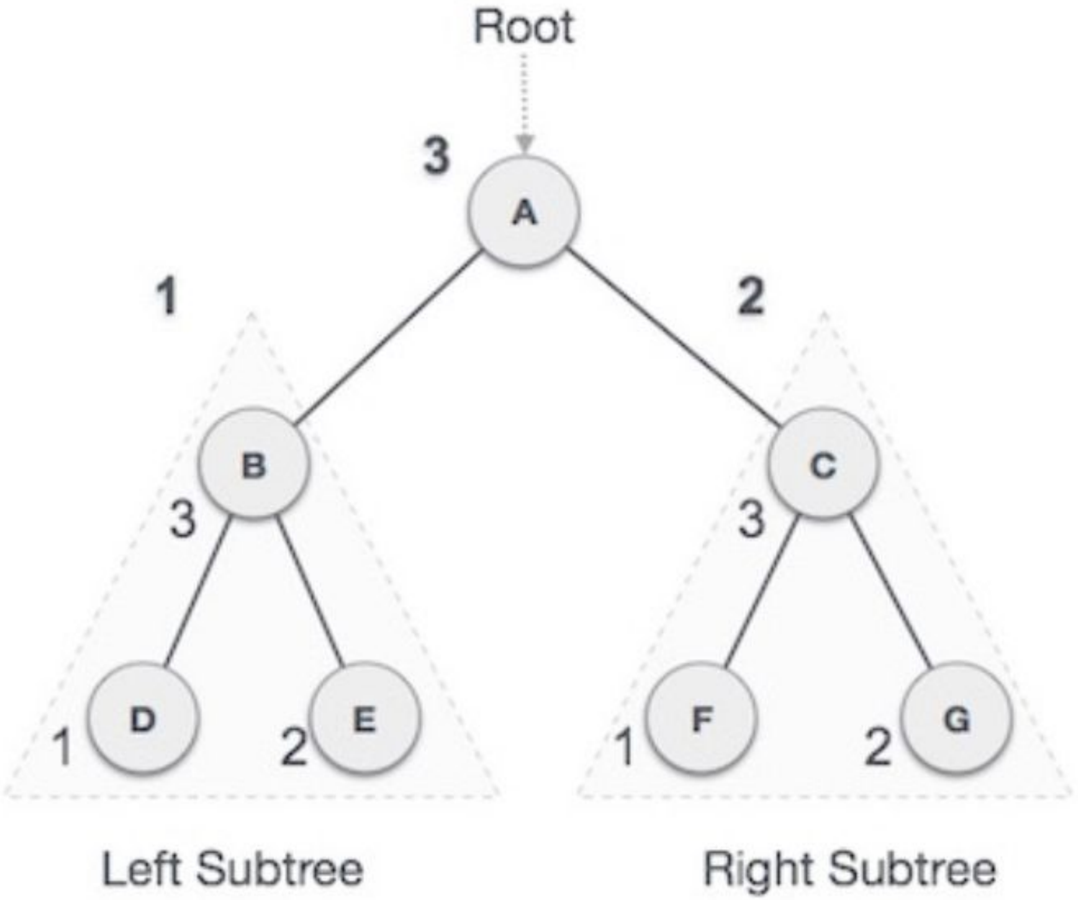
D->B->E->A->F->C->G



Post order Traversal

- In this method of traversal First we visit left subtree then we visit Right subtree
- Algorithm
- Until all nodes are traversed
- Step 1:Recursively traverse Left subtree
- Step 2: Recursively traverse Right subtree
- Step 3:Visit Root node

D->E->B->F->G->C->A



Example

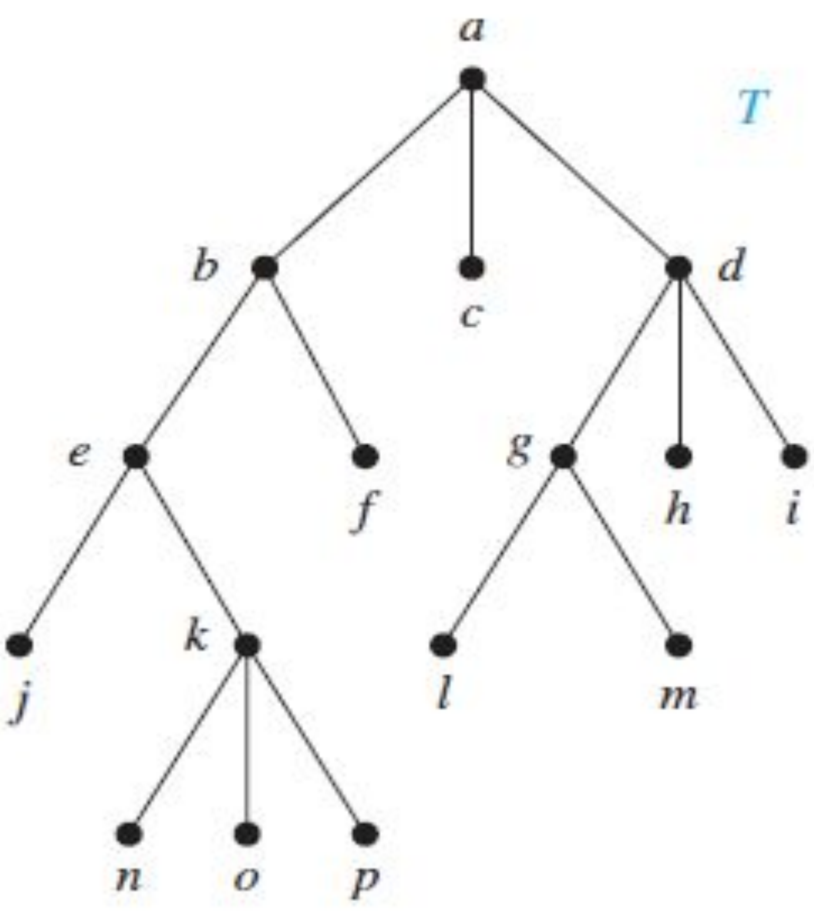


FIGURE 3 The ordered rooted tree T.

- **Pre ordered=**
- **a,b,e,j,k,n,o,p,f,c,d,g,l,m,h,i**
- **In ordered=**
- **J,e,n,k,o,p,b,f,a,c,l,g,m,d,h,i**
- **Post orderd**
- **J,n,o,p,k,e,f,b,c,l,m,g,h,i,d,a**

Spanning tree

- It is a subset of graph G with minimum edges and all vertices.
- Every connected simple graph has a spanning tree.

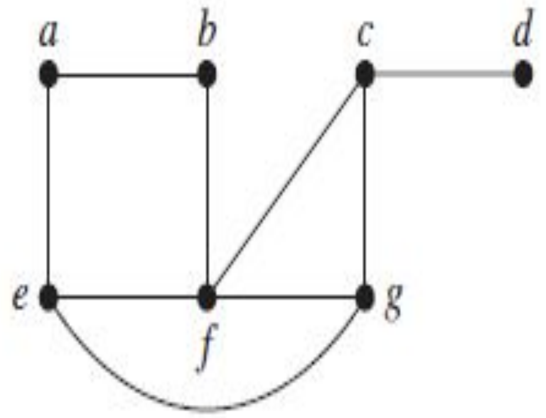
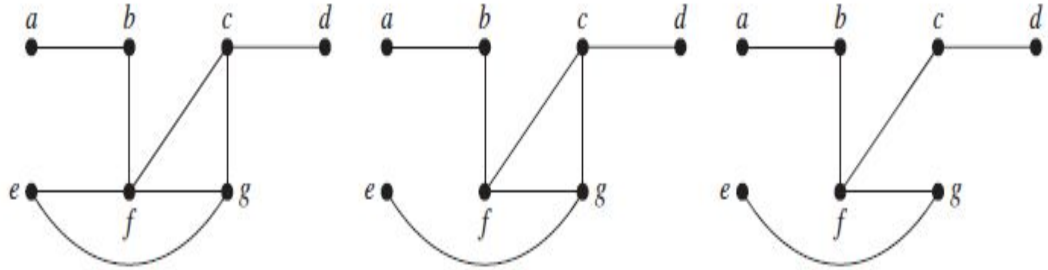


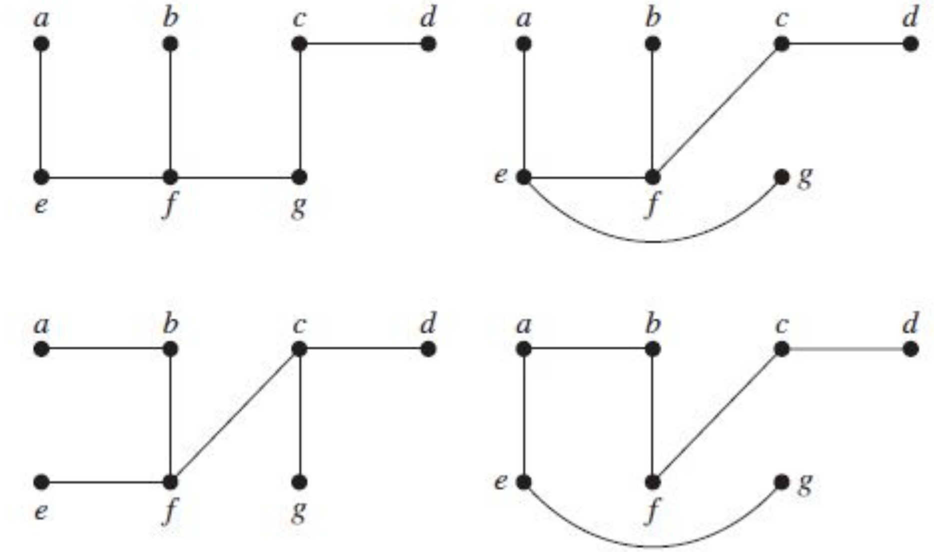
FIGURE 2 The simple graph G.

solution



Edge removed: $\{a, e\}$ $\{e, f\}$ $\{c, g\}$

(a) (b) (c)



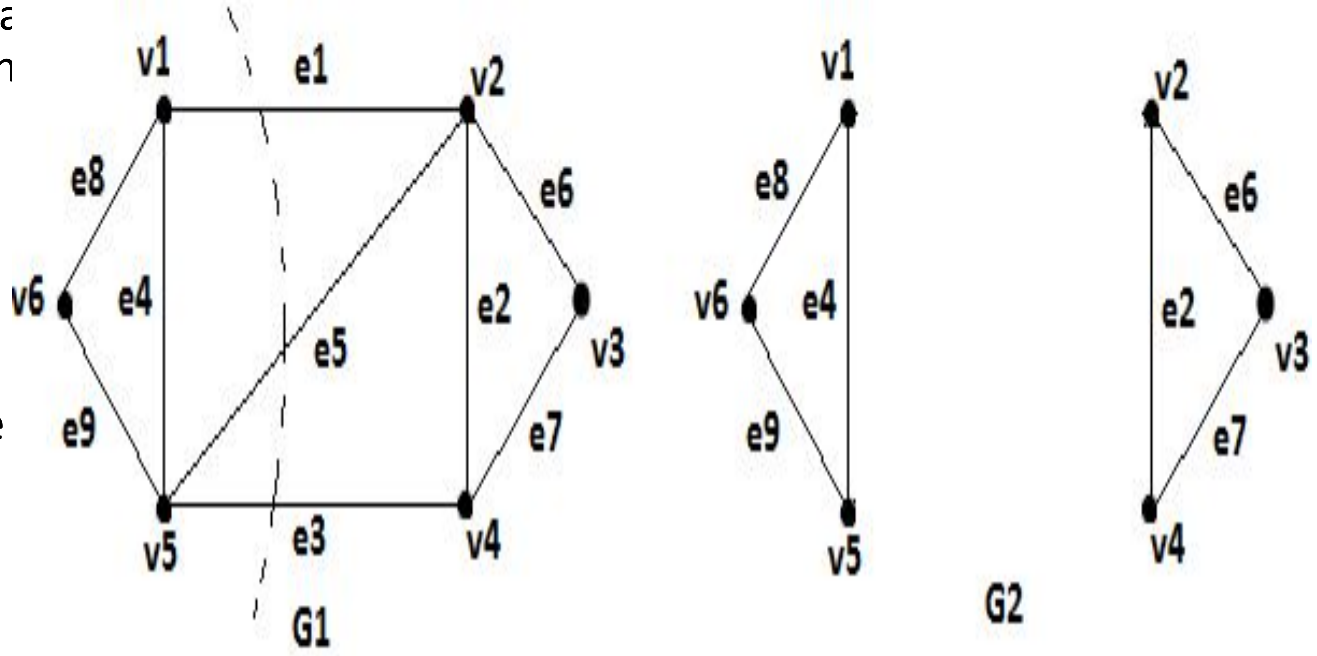
Cut set

- A **cut set** is a **minimal set** of edges whose remova disconnects the graph & increases the componen of graph by **one**.

•For Example,

•For instance, in Fig. $\{e_1, e_4, e_8\}$ is a cut set whereas $\{e_1, e_4, e_8, e_9\}$ is **not** a cut set because its subset $\{e_1, e_4, e_9\}$ is also a cut set.

•Other cut sets in graph $\{e_6, e_7\}$, $\{e_8, e_9\}$ & $\{e_1, e_3, e_5\}$.



MST

- A Minimum spanning tree in a connected graph is a spanning tree that has the smallest possible sum of weights of its edges. There are 2 algorithms for constructing a MST.
- A wide variety of problems can solved by finding MST of weighted graph.

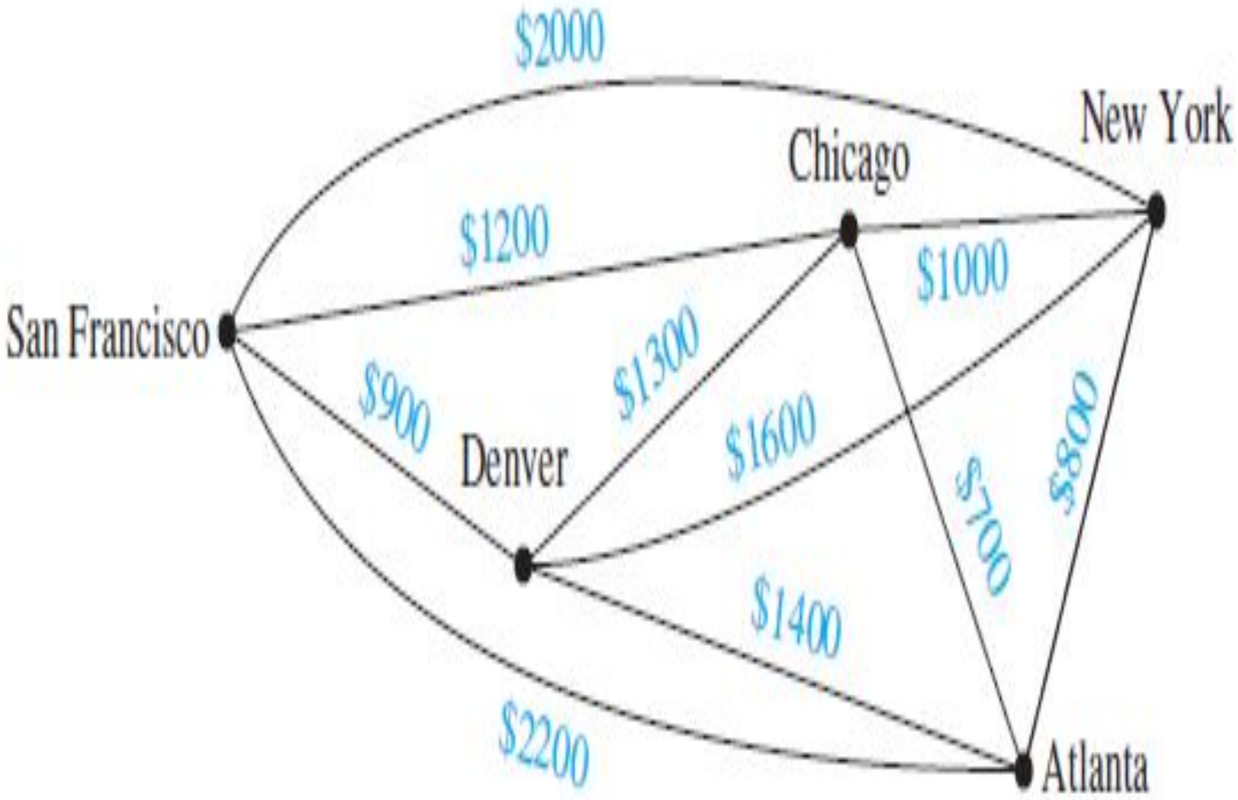
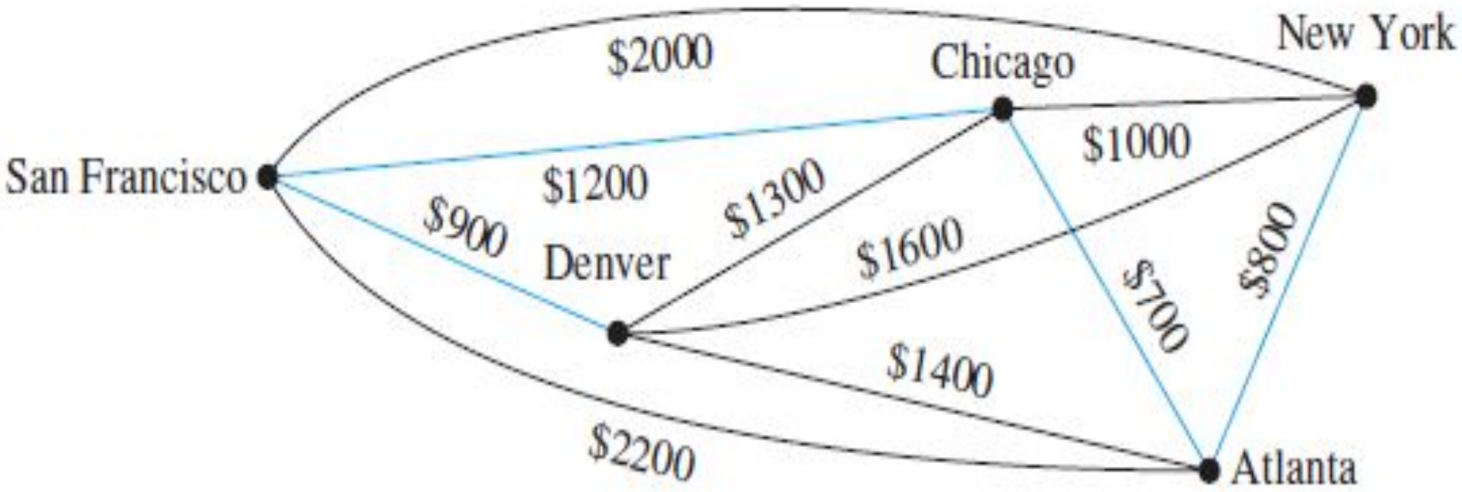


FIGURE 1 A weighted graph showing monthly lease costs for lines in a computer network.

MST for above weighted graph



Choice	Edge	Cost
1	{Chicago, Atlanta}	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$1200
4	{San Francisco, Denver}	$ 900
	Total:	$3600

Prims Algorithm