Tree

Introduction

There are many situations in which information has a hierarchical or nested structure like that found in family trees or organization charts. The abstraction that models hierarchical structure is called a tree and this data model is among the most fundamental in computer science. It is the model that underlies several programming language , including Lisp.

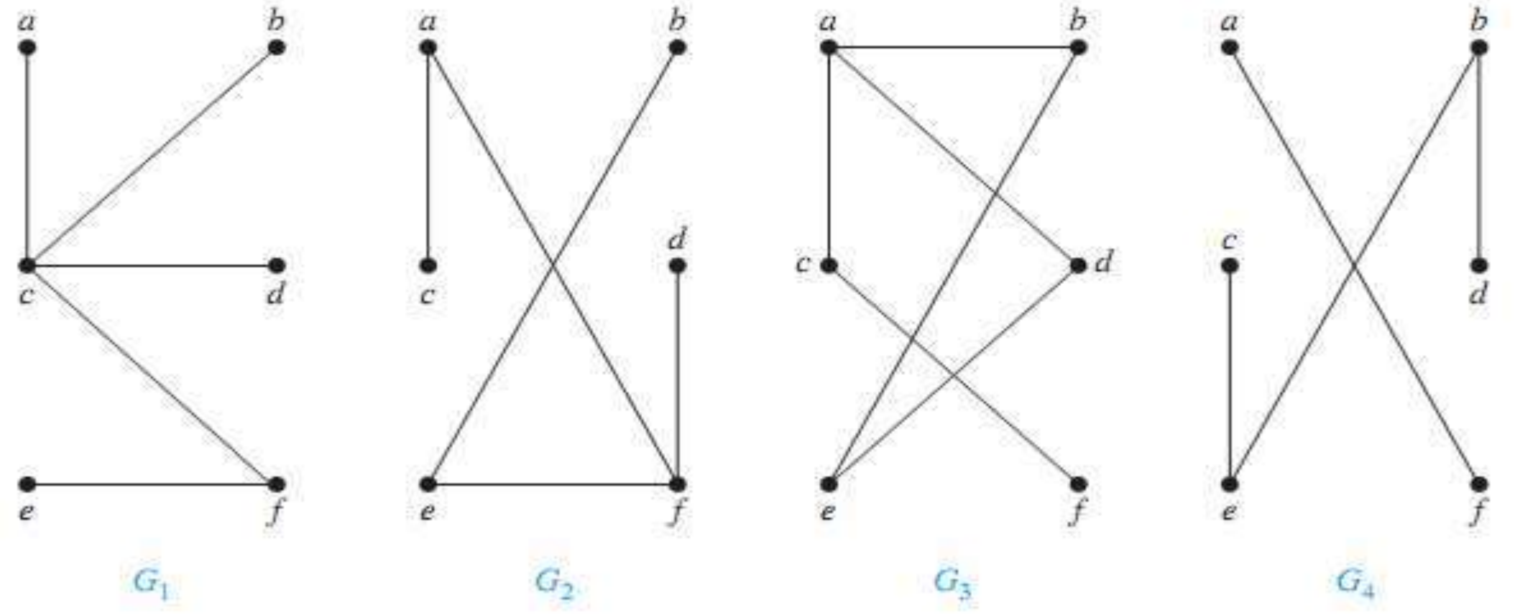
Outlines

- Definition of Tree.
- Tree Terminologies.
- An m-ary Tree.
- Ordered Rooted Tree.
- Tree Traversal.
- Algebraic Expressions and Polish Notation
- Spanning Trees

Definition of Tree

A **tree** is an undirected connected graph with no simple circuits, no multiple edges, no loops. Therefore any tree must be a simple graph. An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Ex : Which of the graphs shown in Figure are trees?



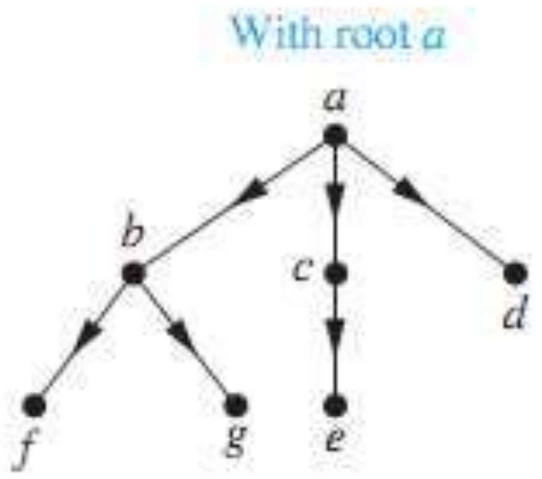
Sol:

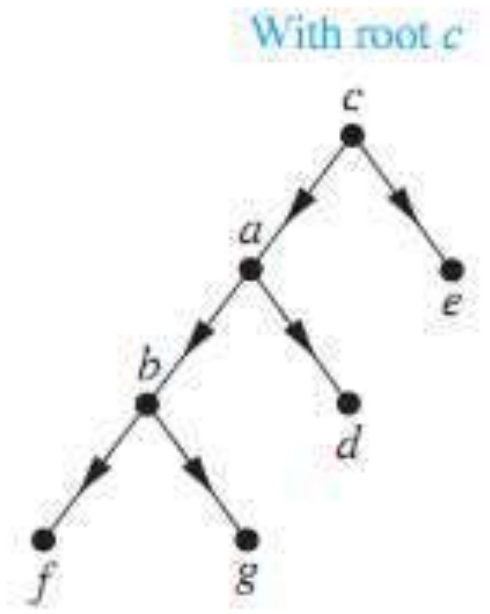
G1 and G2 are trees, because both are connected graphs with no simple circuits.

G3 is not a tree because e, b, a, d, e is a simple circuit in this graph.

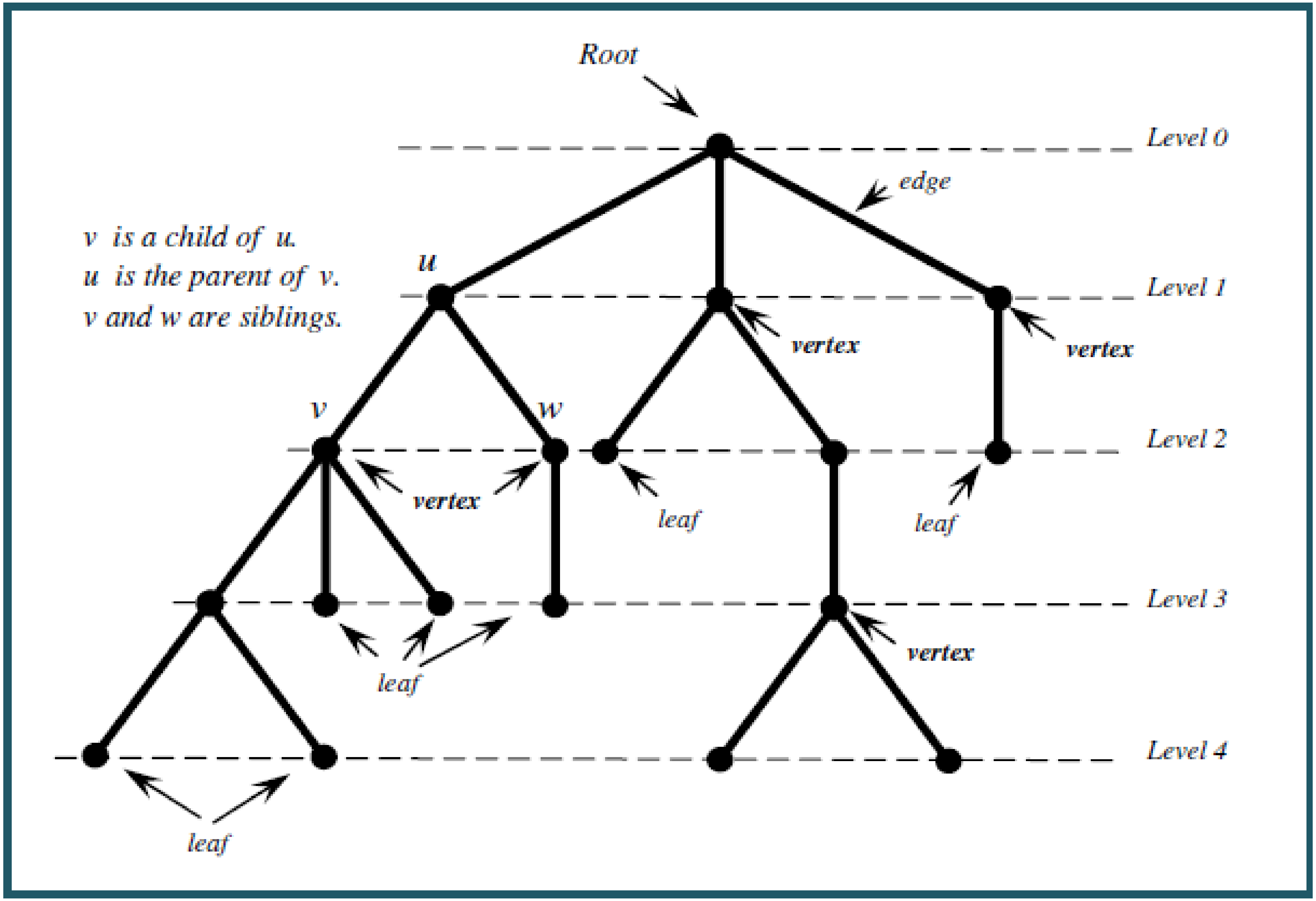
G4 is not a tree because it is not connected.

A **rooted tree** is a tree in which one vertex has been designated as the root and every edge is directed away from the root.





Tree Terminologies



Parent: the parent of v is the unique vertex u such that there is a directed edge from u to v.

Child: When u is the parent of v, v is called a child of u.

Siblings إخوة : siblings are vertices with the same parent.

Leaf: A vertex that has no children.

Internal Vertices: they are the vertices that have children.

Level of a Vertex: is the length of the path from the root to this vertex. **The level of the root is defined to be zero.**

The Height of a Rooted Tree(or depth) : is the maximum of the levels of vertices.(the height of a rooted tree is the length of the longest path from the root to any vertex).

Ex: In the rooted tree T shown in the figure below, find the root, the parent of c, the children of g, the siblings of h,, all internal vertices, and all leaves.

Sol:

The root is a.

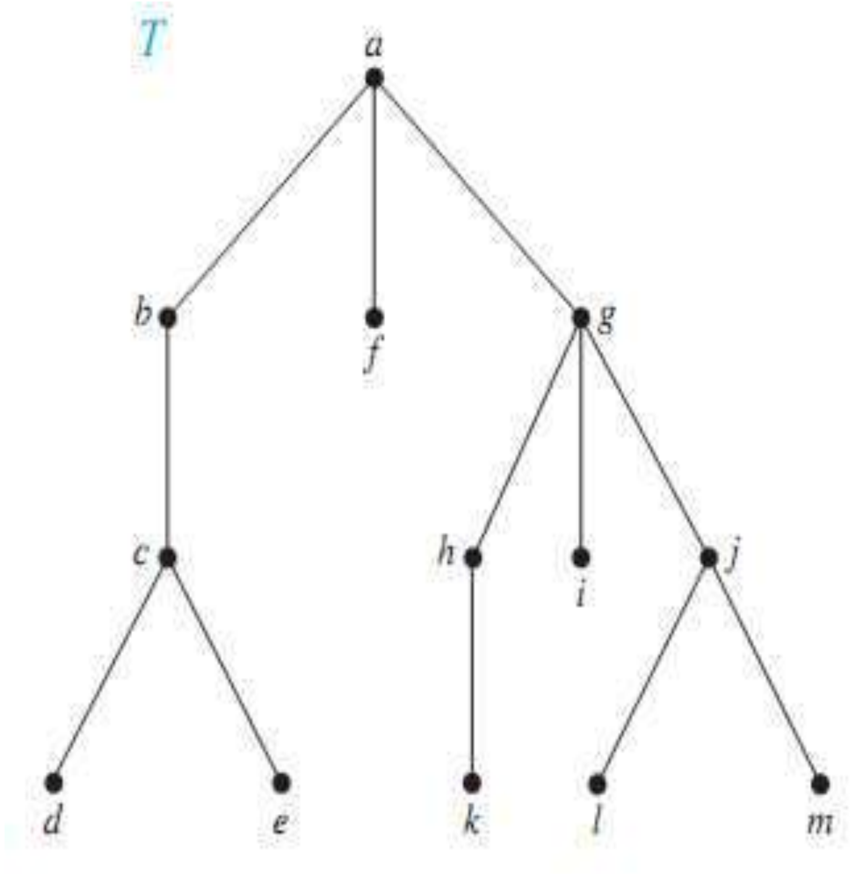
The parent of c is b.

The children of g are h, i, and j.

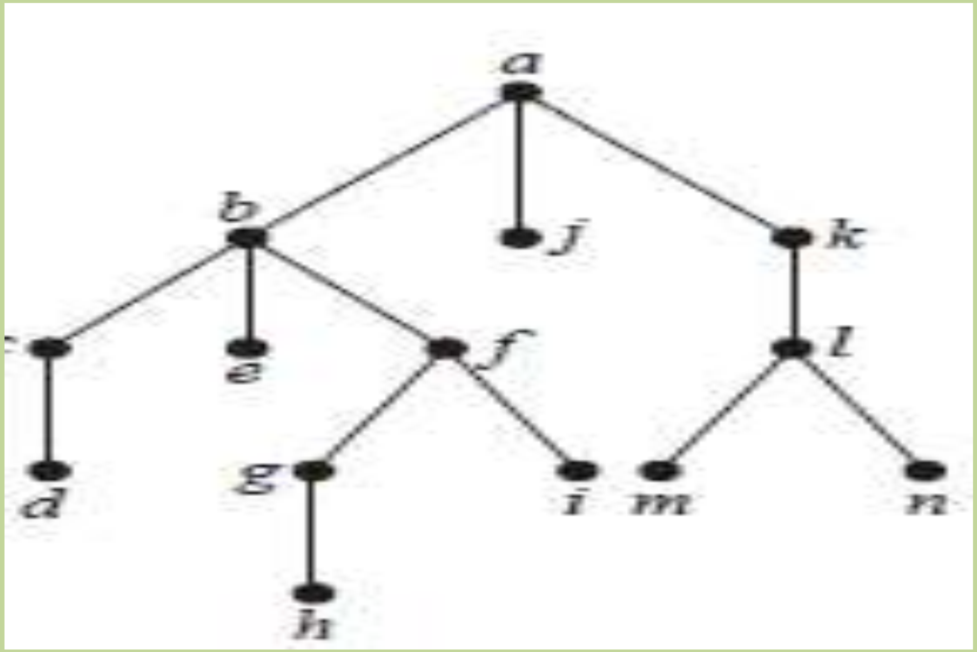
The siblings of h are i and j.

The internal vertices are a, b, c, g, h, and j.

The leaves are d, e, f , i, k, l, and m.



Ex: Find the level of each vertex in the rooted tree. What is the height of this tree?



Solution: The root *a* is at level 0. Vertices *b*, *j* , and *k* are at level 1. Vertices *c*, *e*, *f* , and *l* are at level 2.Vertices *d*, *g*, *i*, *m*, and *n* are at level 3. Finally, vertex *h* is at level 4. Because the largest level of any vertex is 4, this tree has height 4.

Binary trees: are a special type of rooted trees.

There are common types of binary trees—full , complete and perfect binary trees.

The binary data tree is full if each node has 0 or 2 child nodes. A full binary data tree can also be defined as a tree in which every node has two child nodes except for the leaf nodes.

A binary data tree is complete if all levels in it are fully populated except for the last level, where most keys are to the left as far as possible.

A binary data tree is perfect if all inner nodes have two child nodes and all leaf nodes are on the same level.

Binary trees are one of the most important types of structures in computer science. in algorithms for compressing data, and in many other applications.

```
      18
     /   \
    15    20
   / \
  40   50
 /  \
30   50
```

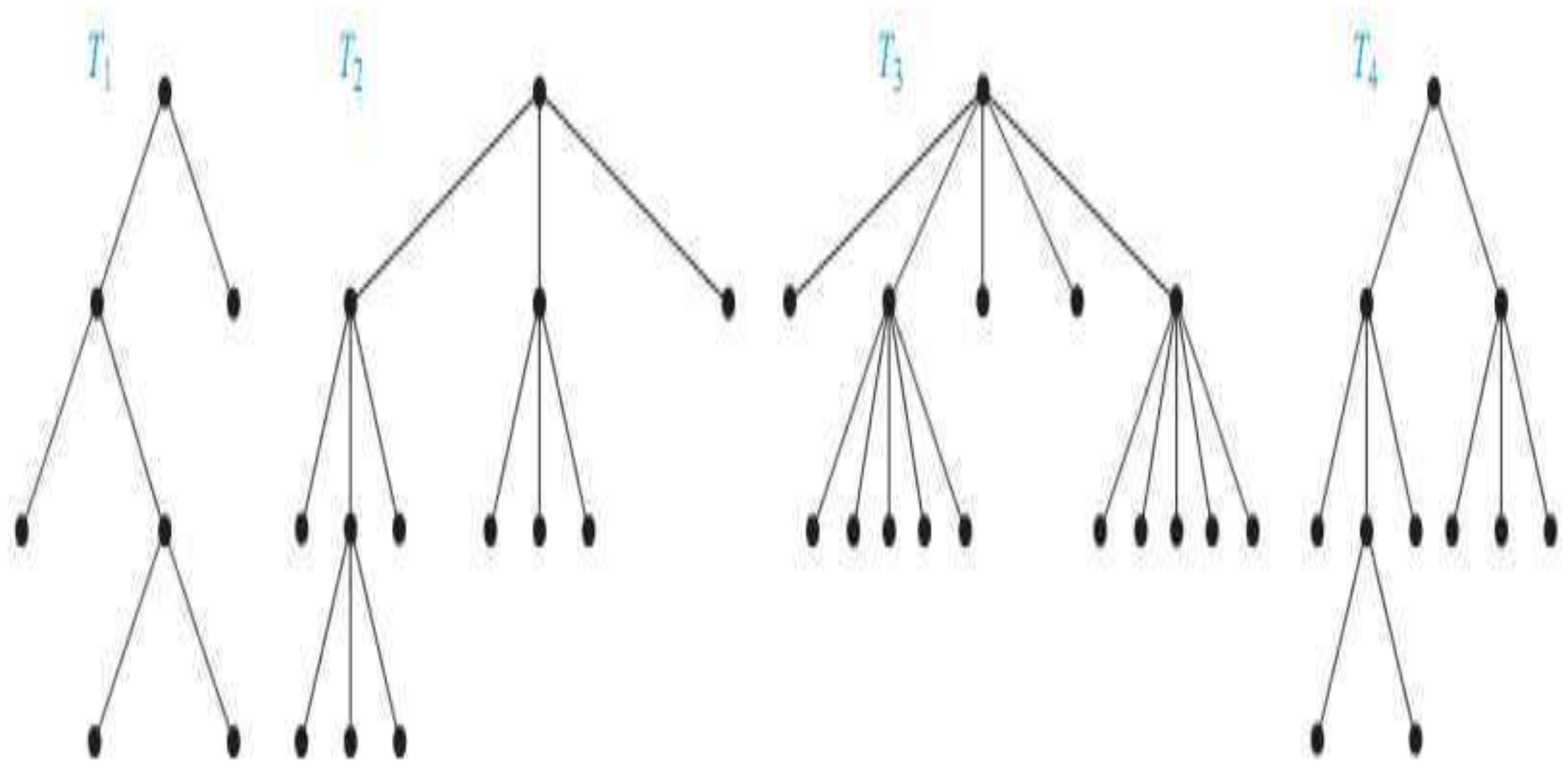
```
      18
     /      \
    15        30
   / \       / \
  40   50   100   40
```

```
       18
     /      \
    15        30
   / \       / \
  40   50   100   40
 / \  /
8   7  9
```

```
      18
     /      \
    15        30
   / \       / \
  40   50   100   40
```

• An m-ary Tree

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*.

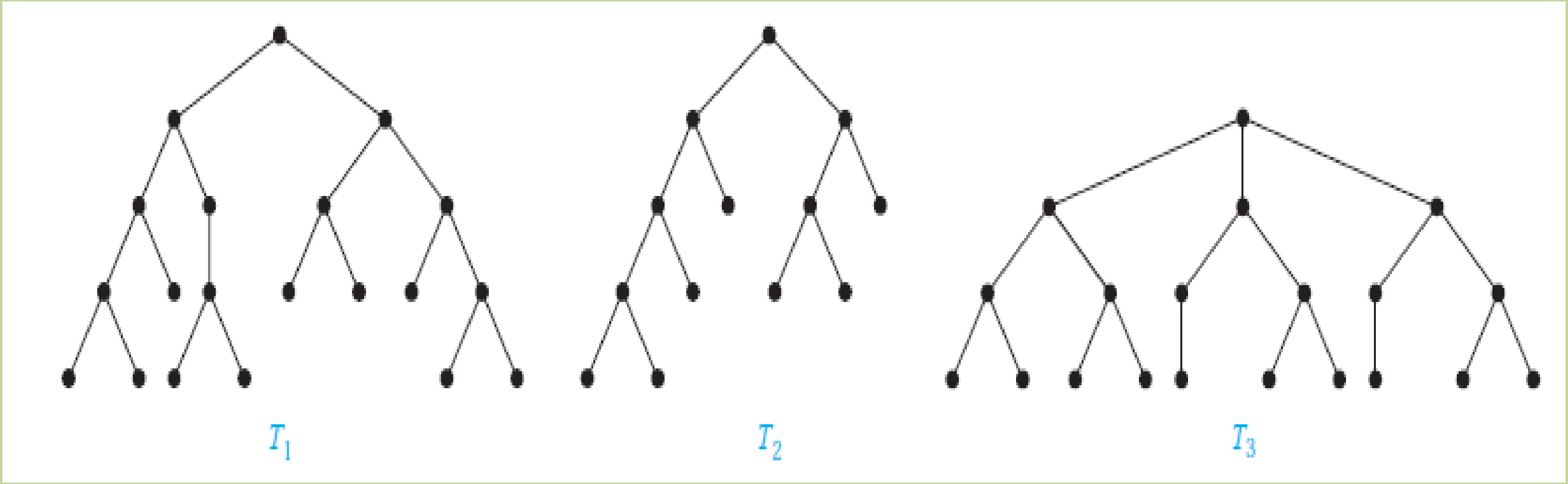


From the figure, we notice that it includes different m-ary trees as follows:

- T1: is a full 2-ary tree; because each of its internal vertices has two children (m=2).
- T2: is a full 3-ary tree; because each of its internal vertices has three children(m=3).
- T3: is a full 5-ary tree; because each internal vertex has five children (m=5).
- T4: is not a full m-ary tree for any m; because some of its internal vertices have two children and others have three children. it is just 3-ary tree.

BALANCED *m*-ARY TREES that the subtrees at each vertex contain paths of approximately the same length

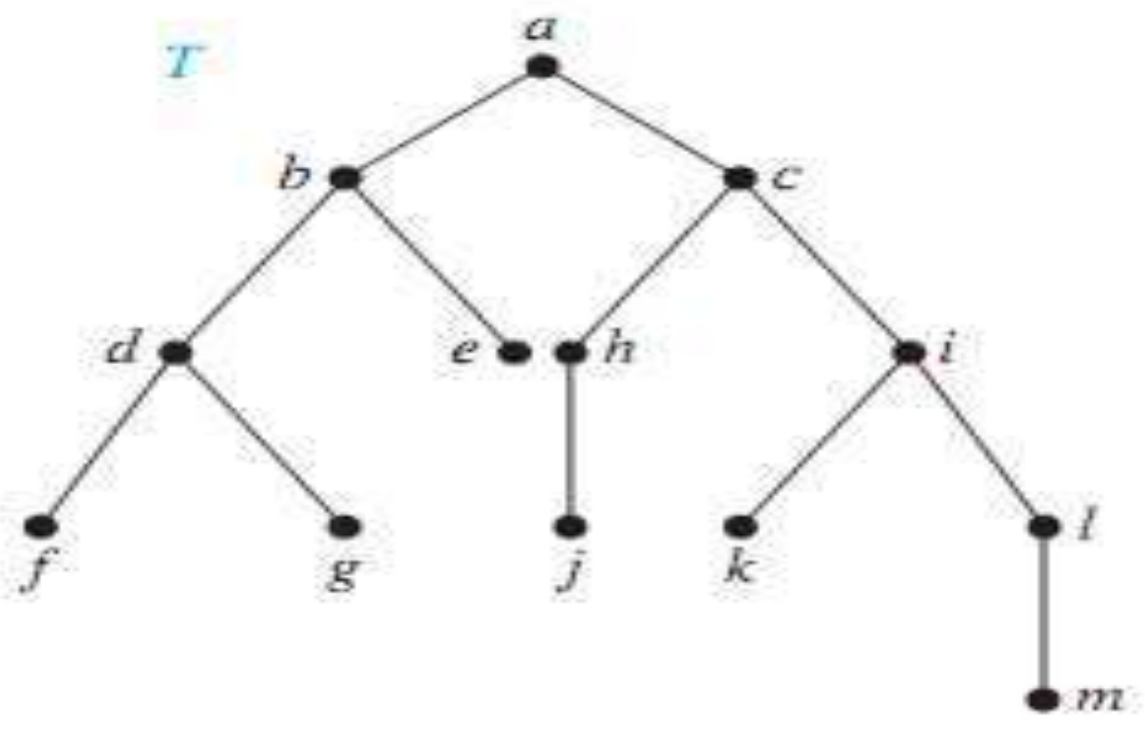
Ex: Which of the rooted trees shown in the following figure are balanced?



*Solution: T*1 is balanced, because all its leaves are at levels 3 and 4. However, *T*2 is not balanced, because it has leaves at levels 2, 3, and 4. Finally, *T*3 is balanced, because all its leaves are at level 3.

Ordered rooted Tree

An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered from left to right.



Degree of tree: The largest number of children in the vertices of the tree .

Tree Traversal

Tree Traversal is a procedures for visiting each vertex of an ordered rooted tree in order to process the data in that vertices and building a list of the results.

Ordered rooted trees can also be used to represent various types of expressions, such as arithmetic expressions involving numbers, variables, and operations.

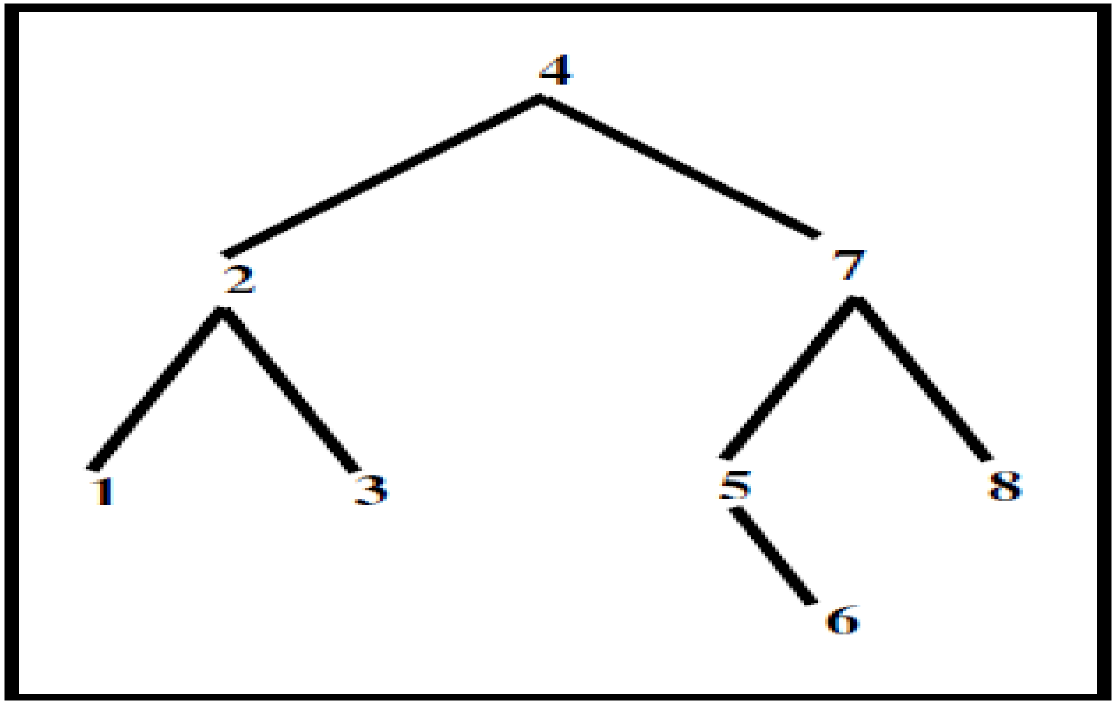
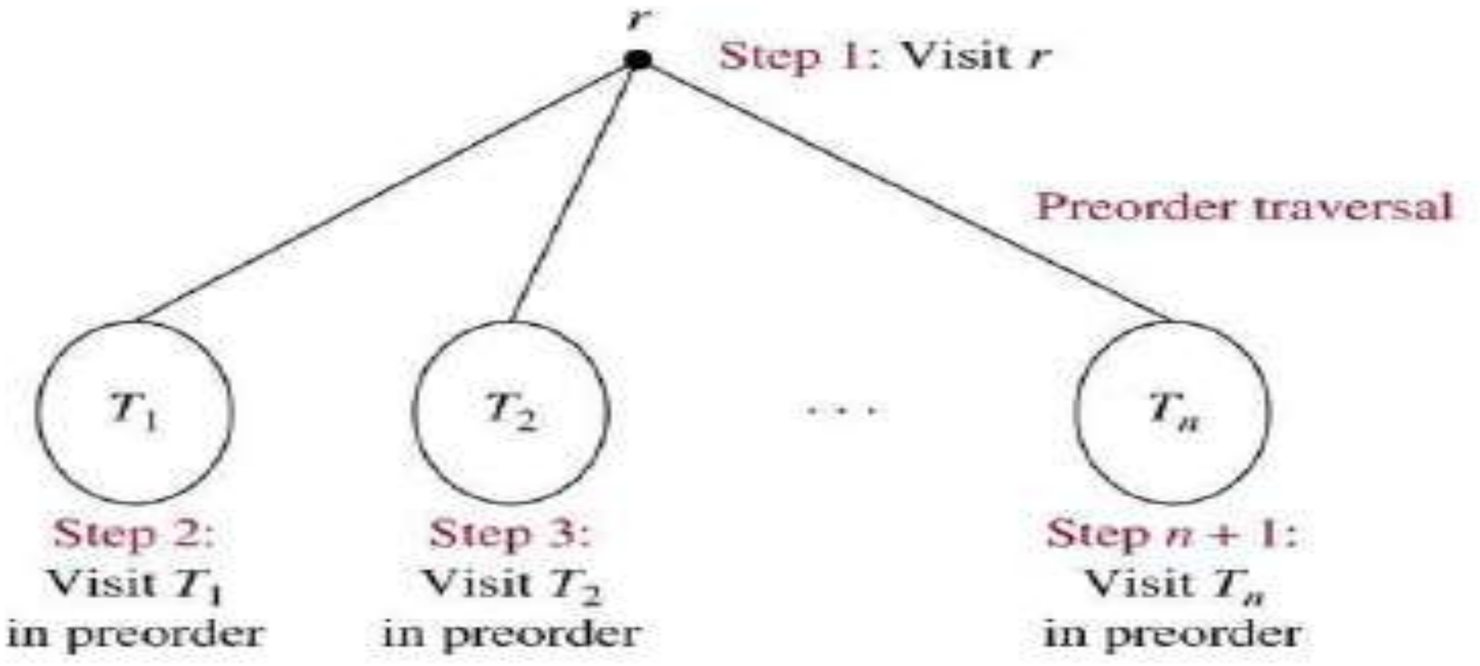


figure 1

The three common traversal Algorithms are as follows:

Preorder traversal:

In this algorithm, the root is firstly visited, then traverse the left subtree, then traverse the right subtree.



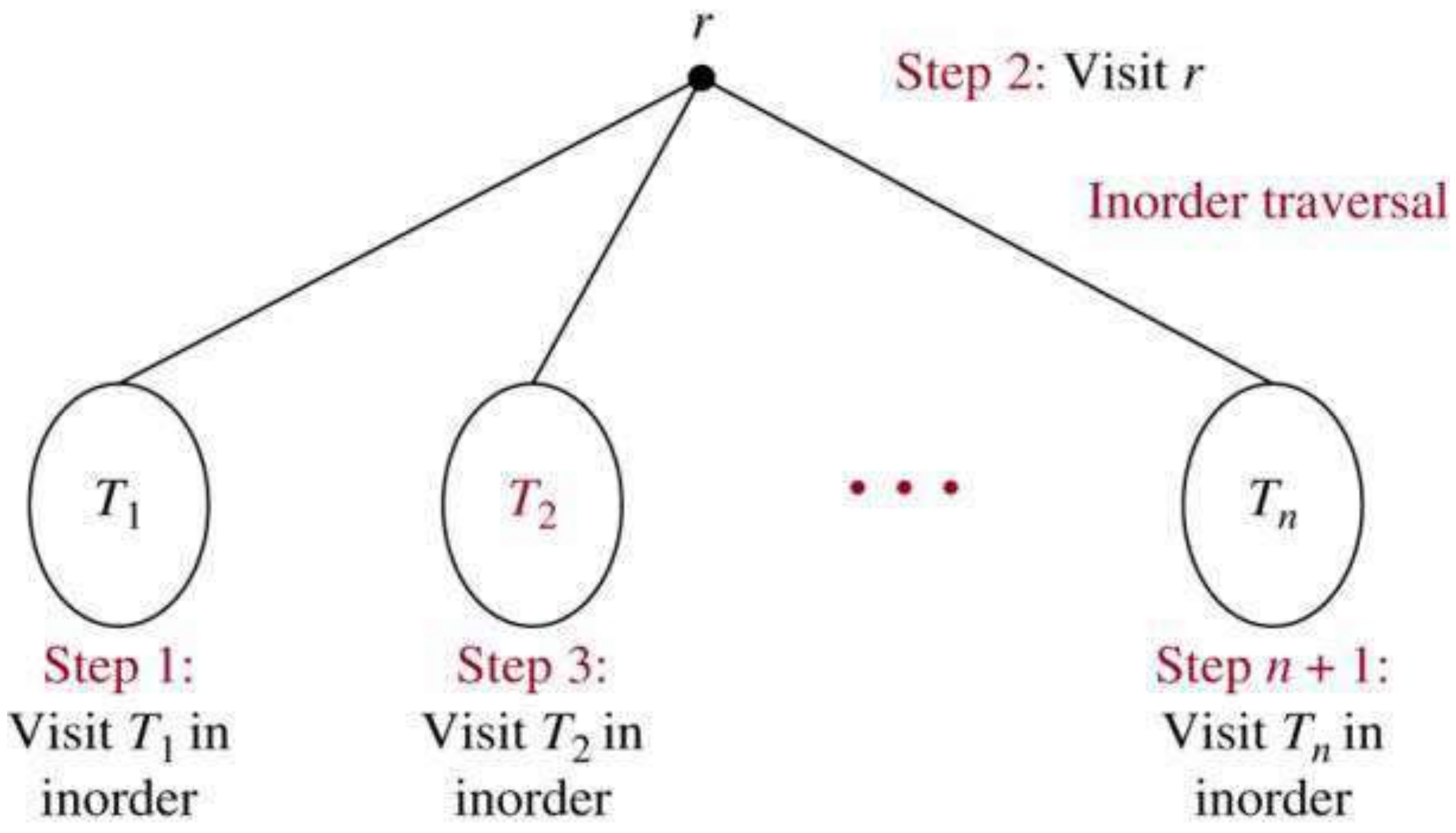
r
Step 1: Visit r
Preorder traversal
T_1
T_2
. . .
T_n
Step 2: Visit T_1 in preorder
Step 3: Visit T_2 in preorder
Step n + 1: Visit T_n in preorder

A preorder traversal of the tree in figure 1 is: 4, 2, 1, 3, 7, 5, 6, 8.

Inorder traversal:

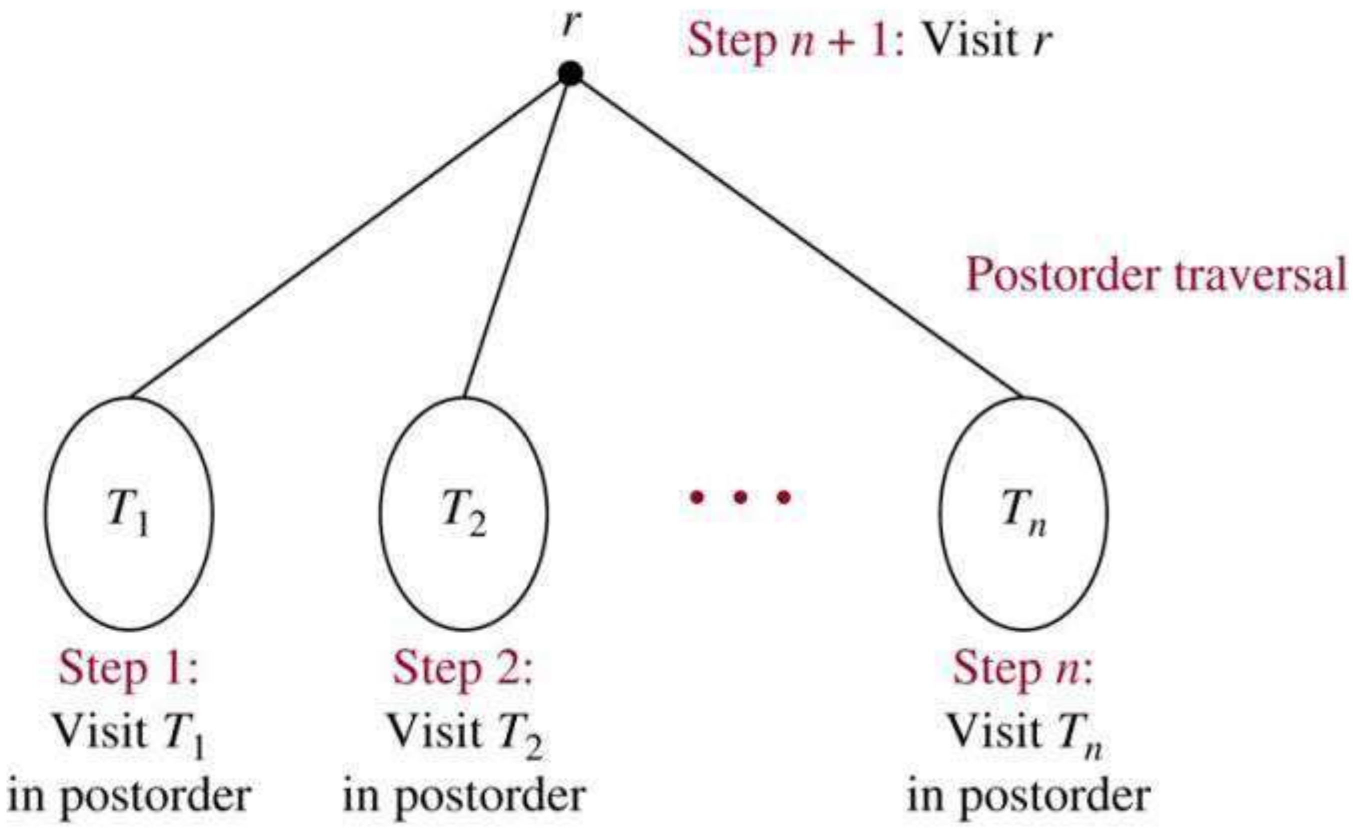
In this algorithm, the left subtree is firstly visited, then the root, then the right subtree.

An inorder traversal of the tree in this figure 1 is: 1, 2, 3, 4, 5, 6, 7, 8.


r
Step 2: Visit r
Inorder traversal
T_1
T_2
T_n
Step 1: Visit T_1 in inorder
Step 3: Visit T_2 in inorder
Step $n + 1$: Visit T_n in inorder

Postorder traversal:

In this algorithm, traverse the left subtree, then the right subtree, and finally visit the root.


r
Step $n + 1$: Visit r
Postorder traversal
T_1
T_2
T_n
Step 1: Visit T_1 in postorder
Step 2: Visit T_2 in postorder
Step n: Visit T_n in postorder

A postorder traversal of the tree in figure 1 is: 1, 3, 2, 6, 5, 8, 7, 4.

Tree Traversal

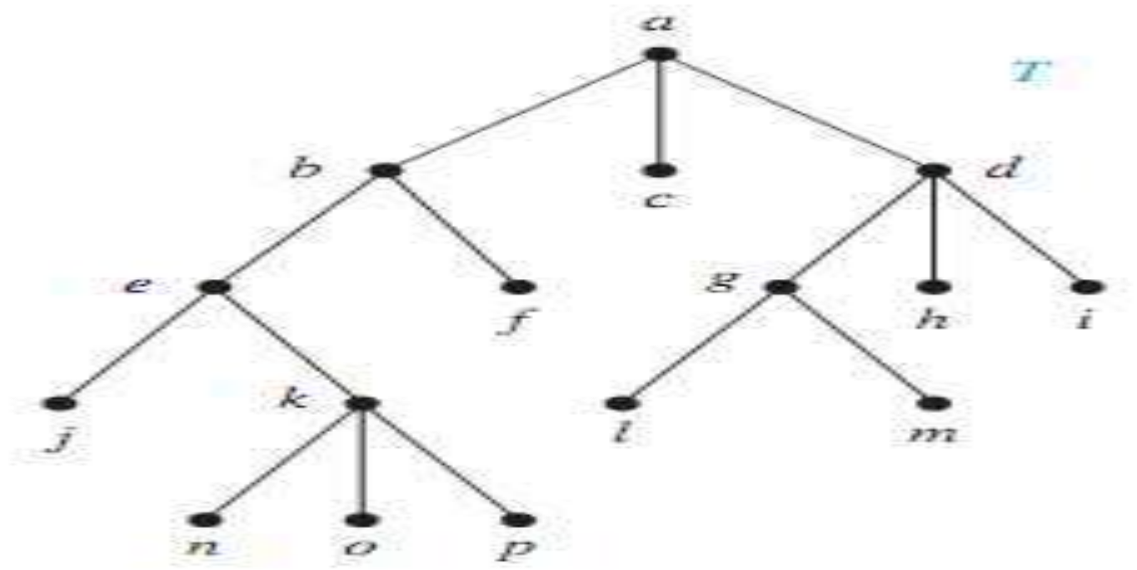
Converts hierarchical data into a linear sequence

Preorder: root, left, right

Inorder: left, root, right

Postorder: left, right, root

Ex: Show the preorder, inorder, postorder traversal for the tree shown below.



Sol:

Preorder: a b e j k n o p f c d g l m h i.

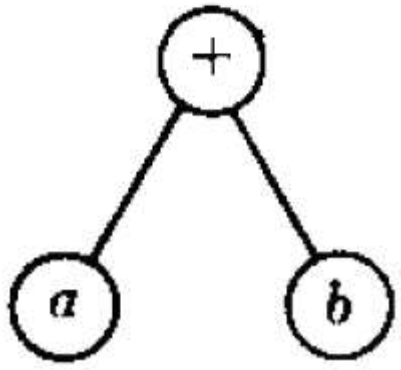
Inorder: j e n k o p b f a c l g m d h i.

Postorder: j n o p k e f b c l m g h I d a.

Algebraic Expressions and Polish Notation

Any algebraic expression involving binary operations +, -, ×, ÷ can be represented by an order rooted tree (ORT).

the binary rooted tree for a+b is :



Polish notation:

The polish notation form of an algebraic expression represents the expression unambiguously with out the need for parentheses

1) a + b (infix)
2) + a b (prefix)
3) a b + (postfix)

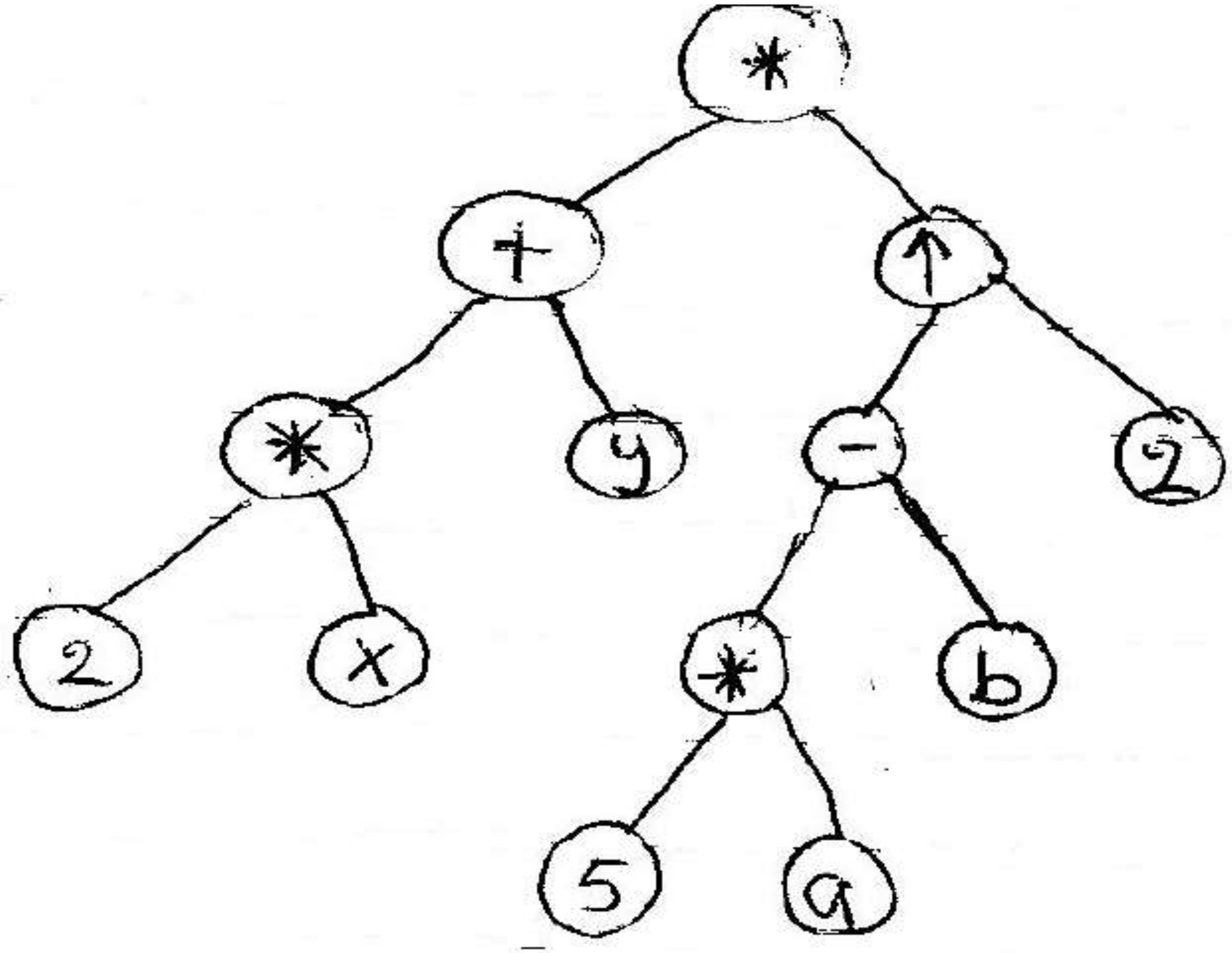
Examples of infix to prefix and post fix

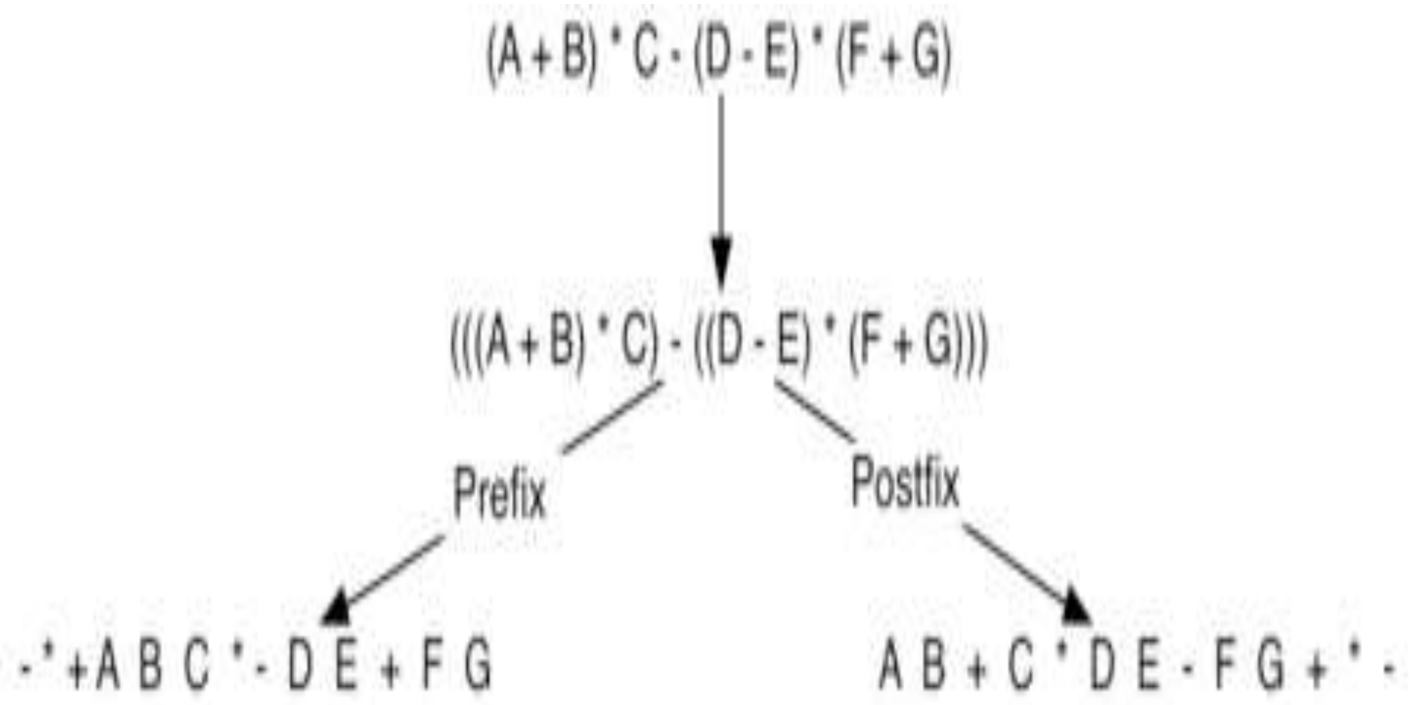
Infix	Postfix	Prefix
A+B	AB+	+AB
(A+B) * (C + D)	AB+CD+*	*+AB+CD
A-B/(C*D^E)	ABCDE^*/-	-A/B*C^DE

Example :

infix polish notation is : (2 * x + y).(5 * a – b)^2

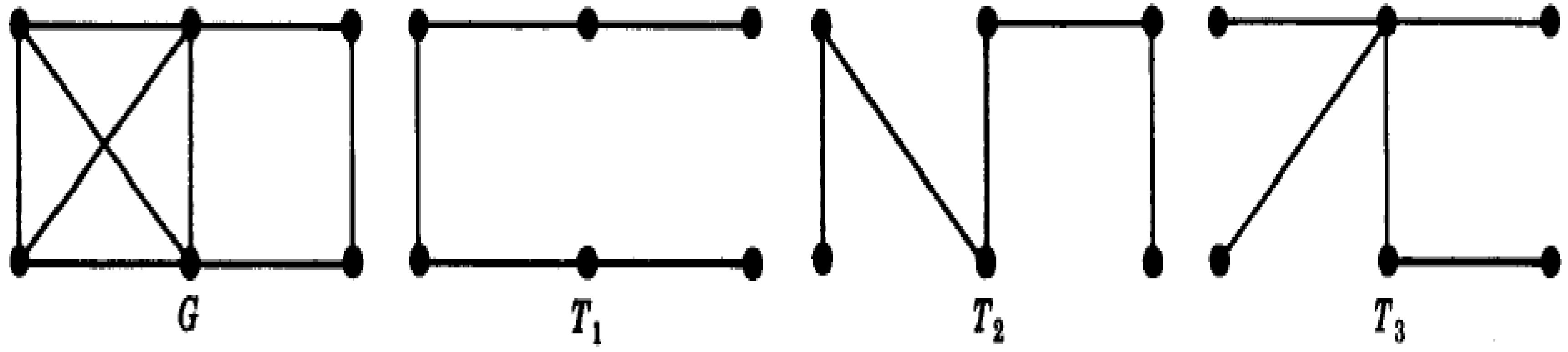
prefix polish notation : * + * 2 x y ^ - * 5 a b 2

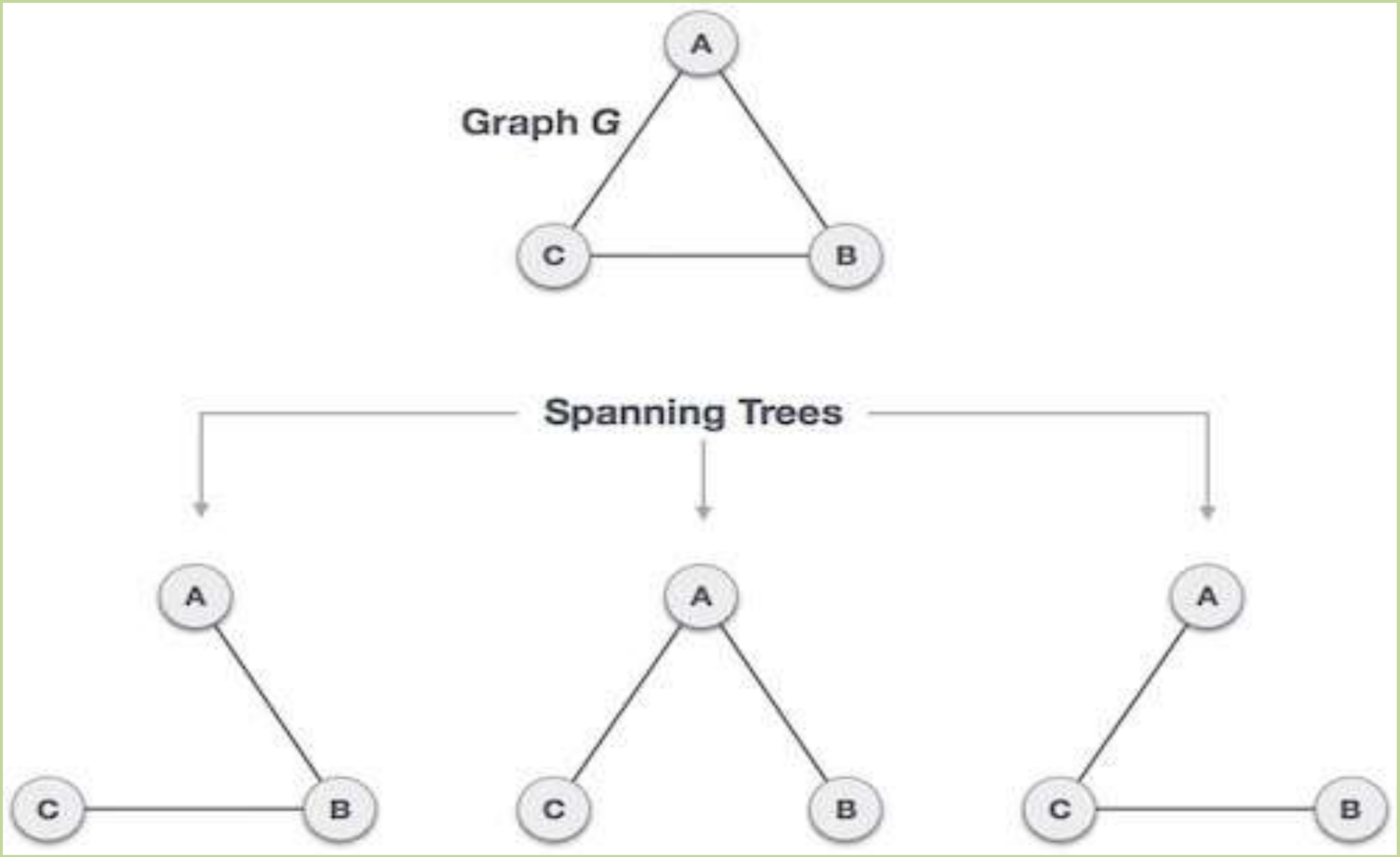



(A + B) * C - (D - E) * (F + G)
(((A + B) * C) - ((D - E) * (F + G)))
Prefix
Postfix
- * + A B C * - D E + F G
A B + C * D E - F G + * -

Spanning Trees

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected..




Graph G
A
C
B
Spanning Trees
A
C
B
A
C
B
A
C
B

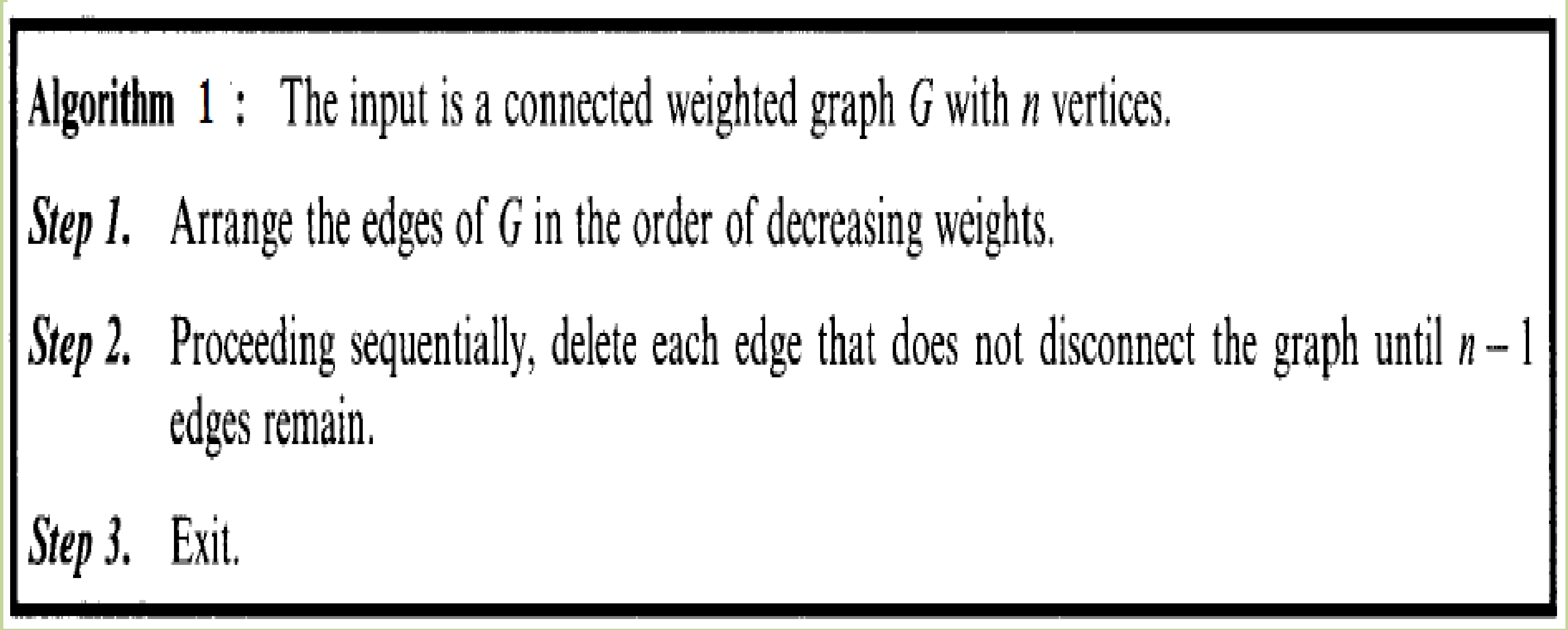
General Properties of Spanning Tree

One graph can have more than one spanning tree. Following are a few properties of the spanning tree connected to graph G :

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.
- Spanning tree has **n-1** edges, where **n** is the number of nodes (vertices).

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.



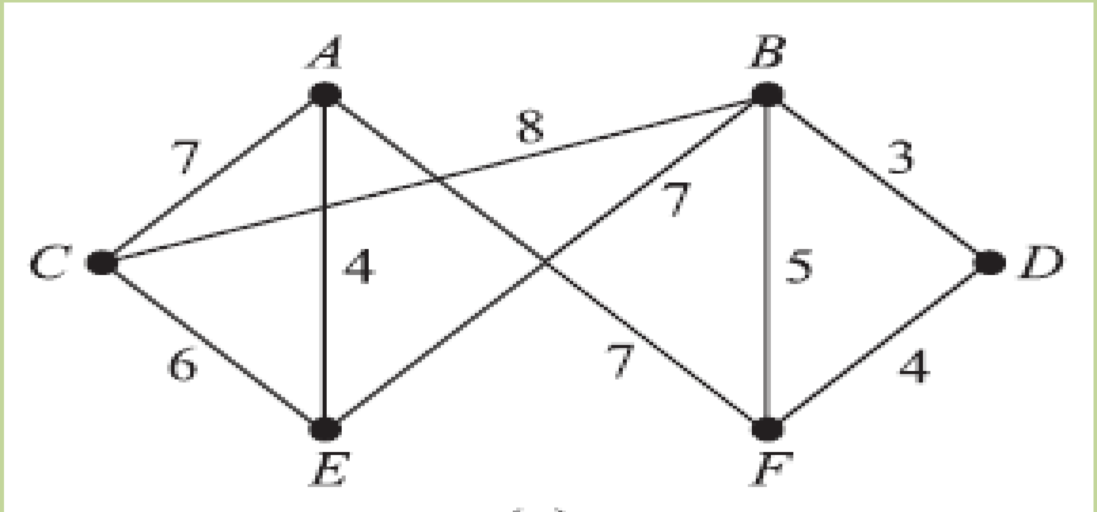
Algorithm 1 : The input is a connected weighted graph G with n vertices.

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.

EXAMPLE: Find a minimal spanning tree of the weighted graph Q, Note that Q has six vertices, so a spanning tree will have five edges.

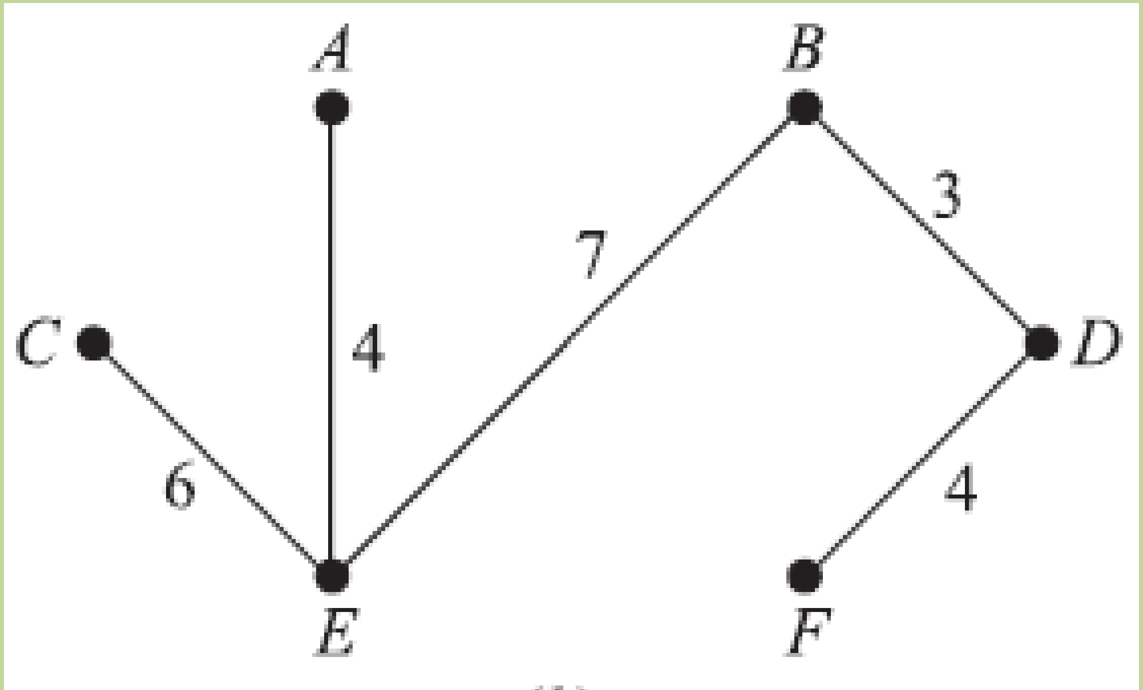


(a)

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

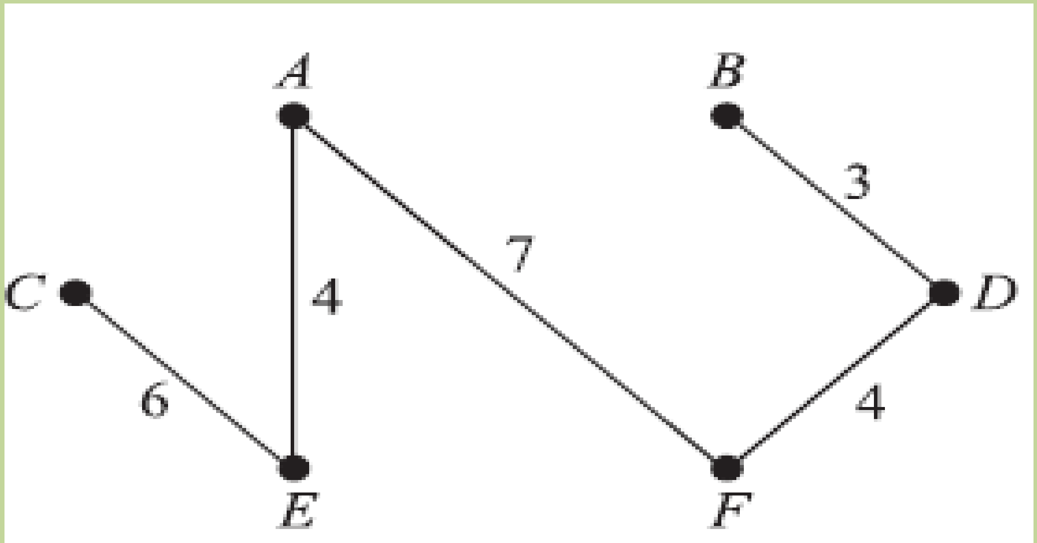


(b)

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, AE, DF, CE, AF
Observe that this spanning tree it also has weight 24.



(c)