Module 3

Trees

TREES

Ques 1) Define trees. Give a suitable example to describe a tree.

Ans: Trees

A tree is an ideal widely-used data structure that emulates a hierarchical tree structure with a set of linked node which means that the data is organised as branches, which relates the information. A node may contain a valu or a condition or represent a separate data structure or a tree of its own. Each node in a tree has zero or mor child nodes, which are below it in the tree (by convention, trees grow down, not up as they do in nature). A node that has a child is called the child's parent node (or ancestor node, or superior). A node has at most on parent.

A tree is a finite set of one or more nodes such that:

i) There is a specially designated node called the Root.

ii) The remaining nodes are partitioned into $n \geq 0$ disjoint sets $S_1....S_n$ where each of these sets is a tree. $S_1...S_n$ is called the subtrees of the root. The condition that $S_1....S_n$ be disjoint sets prohibits subtrees from ever connecting together.

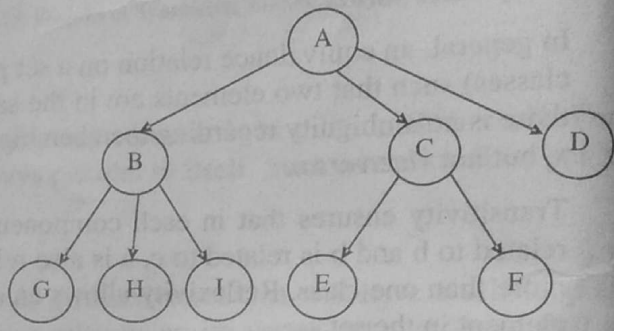


Figure 3.1: Tree Structure

For example, tree is shown in **figure 3.1**.

Ques 2) What are the basic terms used for the trees?

Ans: Terminologies used in Trees

1) **Node:** A node stands for the item of information plus the branches to other items or separate entities of a tree. **For example,** in **figure 3.2** it has 11 nodes.

2) **Root:** Specially designed data item or first node of tree, i.e., the topmost node of a tree. **For example,** in **figure 3.1,** node A is a root.

3) **Degree:** The number of subtree of a node is called its degree. **For example,** in **figure 3.1** the degree of node B is 3.

4) **Leaf or Terminal Nodes:** Nodes that have degree zero is called leaf or terminal nodes. **For example,** in **figure 3.1,** G, H, I, E, F, D are 'leaf' nodes; other nodes of tree are called 'non-leaf' nodes.

5) **Children:** The roots of the subtrees of a node i are called the children of node i and i is the 'parent' of its children. **For example,** in **figure 3.1,** node B is the parent of node G, node H, and node I.

6) **Siblings:** Children of the same parent are called 'Siblings'. **For example,** in **figure 3.1,** nodes G, H, I, are siblings.

7) **Level:** The 'level' of a node is defined by initially letting the root be at level 1. If a node is at level i, then its children are at level i+1.

8) **Height or Depth:** The height or depth of a tree is defined as the maximum level of any node in the tree. **For example,** in **figure 3.1,** the height is 3.

9) **Forest:** A 'forest' is a set of $n > 0$ disjoint trees.

Ques 3) What is a pendent vertex?

Ans: Pendent Vertex

A vertex of degree one is called a pendant vertex. A **pendant vertex** can also be found to be described as an **end vertex**.

In the context of trees, a **pendant vertex** is usually known as a **terminal node**, a leaf node or just **leaf**.

Ques 4) If T is a tree with more than 1 vertex, there are at least 2 pendant vertices.

Ans: Since T is connected, every vertex has degree at least 1. The sum of the degrees of all the vertices = $2e = 2(n-1) = 2n - 2$. If $n-1$ of the vertices of T had degree at least 2 we would get a contradiction.

Ques 5) How the distance between any two vertices can be measured? Explain using suitable example.

Ans: Distance between Two Vertices

In a connected graph G, the distance $d(V_i, V_j)$ between two its vertices V_i and V_j is the length of the shortest path (i.e., the number of edges in the shortest path) between them.

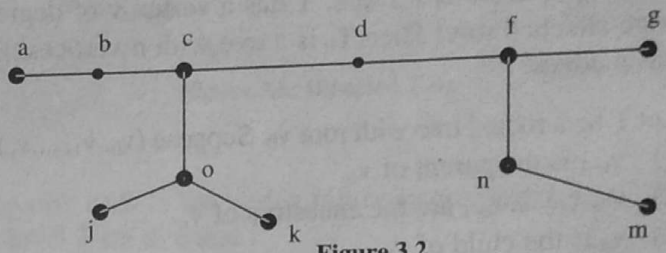


Figure 3.2

In **figure 3.2**, distance between the vertices b and o is 2, i.e., d(b, o) = 2. Since, there are two edges (b, o) and (c, o) between the vertices b and o. Similarly d(a, k) = 4, d(b, m) = 5 , d(j, f) = 4, d(k, g) = 5, d(a, m) = 6 and so on.

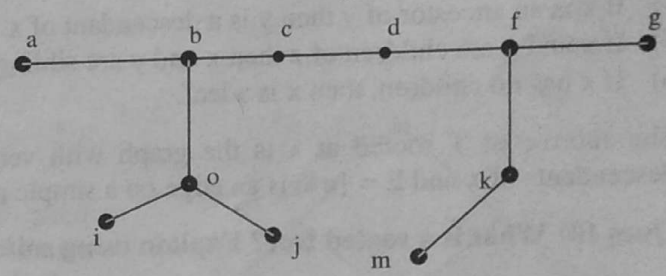


Figure 3.3

Eccentricity: Eccentricity of a vertex v of a graph G, is the distance of v from the farthest vertex (vertices). It is denoted by E(v).

Therefore $E(v) = \max d(v, v_k)$, where $v_k \in G$.

In **figure 3.3**, eccentricity of a is E(a) = 6. Since farthest vertex from a is m and d(a, m) = 6. Similarly E(b) = 5, E(c) = 4, E(d) = 4 (since i and j are the farthest vertices from d), E(f) = 5, E(m) = 7.

Ques 6) Discuss about the centres, radius and diameter in a tree.

Ans: Centers, Radius and Diameter in a Tree

1) **Centers:** A vertex of a graph G with minimum eccentricity is called the center of the graph G. If in a graph G there are two vertices having the same minimum eccentricity then both the vertices are called the centers (bi-centers) of the graph G. In **figure 3.2**, vertex d is the center of the graph and in **figure 3.3**, there are two centers c and d.
2) **Radius:** The minimum eccentricity of the center (centers) of a tree T is called the radius of T. The radius of the graph (**shown in figure 3.2**) is 3 and the radius of the graph (**figure 3.3**) is 4.
3) **Diameter:** Diameter of a tree T is the longest path between any two leaf nodes in the tree T. The diameter of the graph in the **figure 3.3** is 7. Let radius and diameter of a tree are r and d respectively. We have two relations between r and d:
   i) $d = 2r$, if tree has only one center.
   ii) $d = 2r - 1$, if tree has two centers.

Ques 7) Write down the various properties of the trees.

Ans: Properties of Trees

1) There is one and only one path between every pair of vertices in a tree, T.
2) A tree with n vertices has $(n - 1)$ edges.
3) A full m-ary tree with I internal vertices contains $n = mi+1$ vertices.
4) Any tree with at least two vertices has more than one vertex of degree 1.
5) There are at most m^h leaves in an m-ary tree of height h.

Ques 8) Prove that if any tree with more than one vertex has atleast one vertex of degree 1.

Ans: Let v_0 and v_n be two distinct vertices. Then there is a path connecting v_0, v_n. By the definition of a tree, there is only one edge incident on either v_0, v_n. Thus $\deg(v_0) = \deg(v_1) = 1$.

Ques 9) Prove the statement "A tree with n vertices has exactly (n – 1) edges".

Ans: The proof is by induction on $n \geq 1$. Let P(n) be the property. Any tree with n vertices has n – 1 edges.

Basic Step: P(1) is valid since a tree with one vertex has zero edges.

Induction Hypothesis: Suppose that P(n) holds upto $n \geq 1$.

Induction Step: We must show that any tree with n + 1 vertices has n edges. Indeed, let T be any tree with n + 1 vertices. Since $n + 1 \geq 2$, T has a vertex v of degree 1. Let T_0 be the graph obtained by removing v and the edge attached to v. Then T_0 is a tree with n vertices. By the induction hypothesis, T_0 has (n – 1) edges and so T has n edges.

Let T be a rooted tree with root v_0. Suppose $(v_0, v_1, ...v_n)$ is a simple path in T and x, y, z are three vertices. Then

1) v_{n-1} is the parent of v_n.
2) $v_0, v_1, ...v_{n-1}$ are the ancestors of v_n.
3) v_n is the child of v_{n-1}.
4) If x is an ancestor of y then y is a descendant of x.
5) If x and y are children of z then x and y are siblings.
6) If x has no children, then x is a leaf.

The subtree of T rooted at x is the graph with vertex V and edge set E, where V is x together with the descendants of x and E = {e le is an edge on a simple path from x to some vertex in V}.

Ques 10) What is a rooted tree? Explain using suitable example.

Ans: Rooted Tree

A rooted tree is a tree in which there is one designated vertex (called **root**). The vertices of degree one (other than root) in a rooted tree are called leaves or terminal nodes or pendant vertices and non-pendant vertices of a tree are called branch nodes or internal nodes or internal vertices. The level of a vertex v in a rooted tree is the length of the path to v from the root and the maximum vertex level is called depth of the rooted tree.

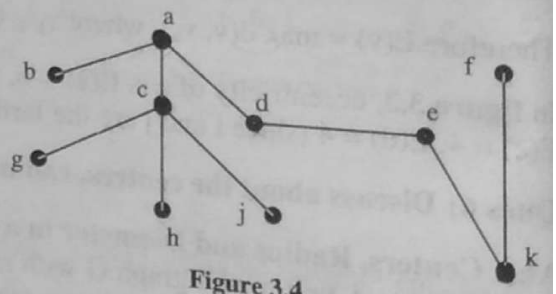


Figure 3.4

For example, in **figure 3.4** if one designate the vertex a as the root then the vertices b, g, i and e are pendant vertice or leaves and the vertices a, c, b, f, h, and j are internal vertices. Similarly, in **figure 3.4** if a is the root then:

Level (a) = 0,
Level (b) = level (f) = Level(c) = 1,
Level (g) = level (h) = level (i) = level (d) = 2,
Level (j) = 3 and Level (e) = 4.

Ques 11) Consider the rooted tree as shown figure 3.5:

1) **Find the parent of v_6.**
2) **Find the ancestors of v_{13}.**
3) **Find the children of v_3.**
4) **Find the descendants of v_{11}.**
5) **Find an example of sibling.**
6) **Find the leaves.**

Ans:

1) v_2.
2) $\{v_1, v_3, v_7\}$.
3) $\{v_7, v_8, v_9\}$.
4) None.
5) $\{v_2, v_3, v_4, v_5\}$.
6) $\{v_4, v_5, v_6, v_9, v_{10}, v_{11}, v_{12}, v_{13}\}$.

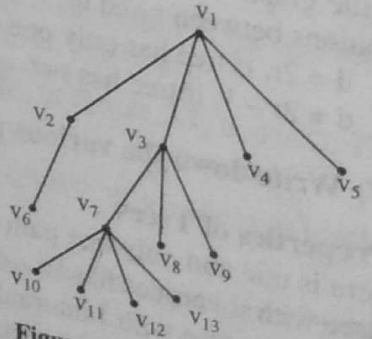


Figure 3.5: Rooted Tree

Ques 12) Describe the directed tree with an example.

Ans: Directed Tree
A directed graph is said to be a directed graph if it becomes a tree when the direction of edges are ignored.

A directed tree is an acyclic diagraph which is a tree as an undirected graph and which has one vertex called its root with in degree zero, while all other vertices have in degree one.

An isolated vertex is also a directed tree. In a directed tree, any vertex, which has out degree zero, is called a **terminal vertex** or a **leaf**. All other vertices are called branch vertices or nodes. The level of any vertex is the length of its path from the root.

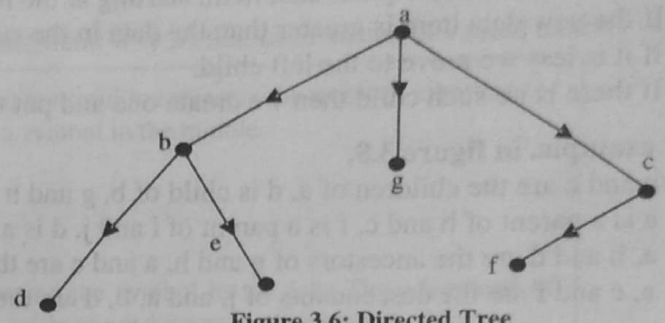


Figure 3.6: Directed Tree

For example, in **figure 3.6** T = (V, E) is a directed tree. The root of T is the vertex (since in coming degree of the vertex a is zero). The vertices at level 1 are b, g and c, at level 2 are d, e and f.

Ques 13) Give the concept of binary tree.

Ans: Binary Tree
An ordered general tree with 2 children is a Binary Tree.

A tree in which every vertex has at most two children is called a binary tree. Each child of a vertex is designated as left child or right child, the subtree rooted at the children of the tree is called left subtree or right subtree.

A tree is a binary tree if each node of it can have at the most two branches as shown in **figure 3.7**.

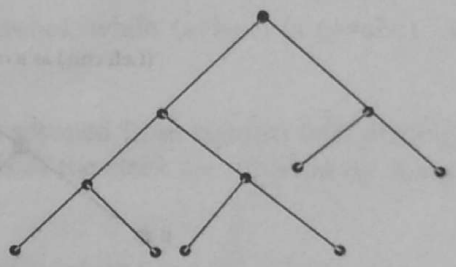


Figure 3.7: Binary Tree

In other words, it can be stated that if every node of a tree can have at most degree two, then this is called a binary tree.

In a binary tree left and right sub-tree distinguish sub-trees of a node. A Binary Tree is a finite set of elements that is either empty or is partitioned into three disjoint subsets. The first subset contains a single element called the Root of the tree. The other two subsets are themselves Binary Trees, called the left and right subtrees of the original tree. A tree, in which there is exactly one vertex of degree two, and each of the remaining vertices of degree one or three, is called a binary tree. If T is a binary tree, the vertex of degree two which is distinct from all the other vertices of T serves as a root of T. Thus, every binary tree is a rooted tree.

The maximum level occurring in a binary tree is called the height of the binary tree. A binary tree with minimum height contains maximum number of vertices at each level. The root of a binary tree is at level 0 and there can be only 1 vertex at ht 0 level. The maximum number of vertices at level 1 is 2^1, at level 2 is 2^2 and so on. By induction we can prove that the maximum number of vertices possible at level k in a binary tree is 2^k.

Ques 14) Write a short note on binary search tree.

Ans: Binary Search Tree
Binary search trees has 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. A binary search tree is a homogenous binary tree such that every item on the left subtree of every vertex v is less than v and every item on its right subtree is greater than v.

It is not necessary that a node in a binary search tree point to the nodes whose values immediately precede and follow it.

Binary Search Tree can be constructed by following way:

i) The first data chosen will be the root of the tree.
ii) Then for each subsequent data item, starting at the root we compare it to the data in the current vertex v.
iii) If the new data item is greater than the data in the current vertex then we move to the right child.
iv) If it is less we move to the left child.
v) If there is no such child then we create one and put the new data in it.

For example, in figure 3.8,

i) b and c are the children of a, d is child of b, g and h are the children of d.
ii) a is a parent of b and c, f is a parent of i and j, d is a parent of g and h.
iii) a, b and d are the ancestors of g and h, a and c are the ancestors of e and f.
iv) a, c and f are the descendants of j, and a, b, d are the descendants of g and h.

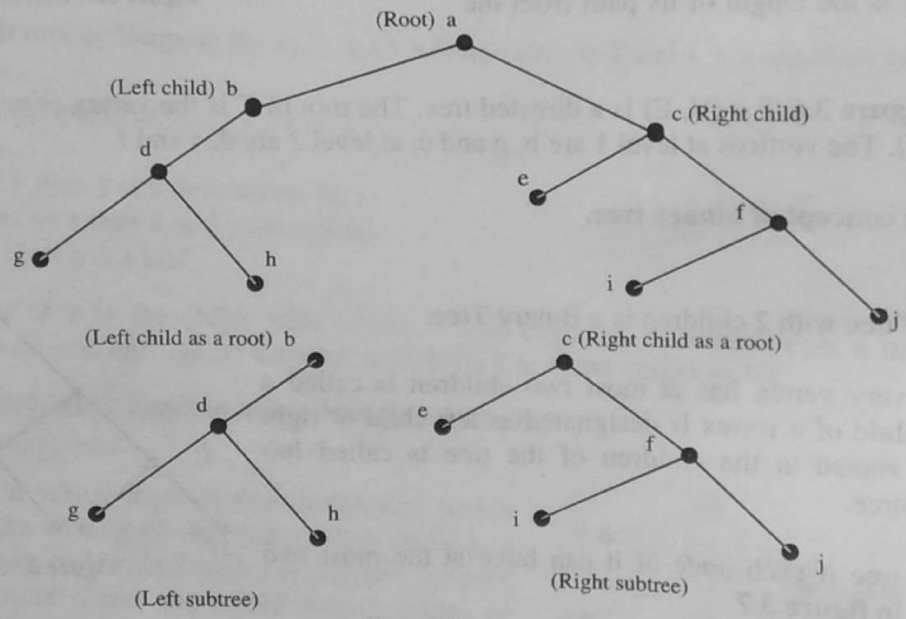


Figure 3.8

Ques 15) Explain how trees can be counted in the graph?

Ans: Counting Trees

The method of counting the trees was discovered by the scientist Arthur Cayley in 1857. He was trying to count the structural isomess numbers in molecule of the saturated hydrocarbon, i.e., C_kH_{2k+2}. He represented the molecule of C_kH_{2k+2} using the connected graph. In such a graph,

Total number of vertices, $n = 3k + 2$ and

Total number of edges is, $e = \frac{1}{2}$ (sum of degrees) $= \frac{1}{2}(4k + 2k + 2)$

$= 3k + 1$

(According to their respective chemical valencies, a carbon atom is represented by a vertex of degree 4 and a hydrogen atom by a vertex of degree 1.)

$\because$ It is a connected graph and number of edges is 1<total number of vertices

$\therefore$ It is a tree.

Hence, the structural isomers counting problem of a hydrocarbon becomes the problem of 'counting trees'. "What is the number of different trees that one can construct with n distinct vertices?" was the first question asked by Cayley. He got his reply by the theorem,

The number of labelled trees with n vertices ($n \geq 2$) is n^{n-2}

Cayley was the first person to state and prove this theorem and it was named its inventor, hence called as the **Cayley's Theorem**.

BINARY TREE TRAVERSAL

Ques 16) Using suitable example discuss the traversal in binary tree.

Ans: Binary Tree Traversal

A traversal of a tree is a process to traverse a tree in a systematic way so that each vertex is visited exactly once.

Infix Notation: Let us consider a binary operation •, it is useful to represent the result of applying the operation to two elements a, b by placing the operation symbol in the middle.

For example,

a • b.

Polish Notation: The Polish notation consists of placing the symbol to the left. The advantage of Polish notation is that it allows us to write expressions without need for parenthesis.

For example,

• a b.

Reverse Polish Notation: The reverse Polish notation consists of placing the symbol to the right:

For example,

a b •

For example, the expression a*(b+c) in Polish notation would be (*a+bc), while (a*b+c) is (+*abc). Also, Polish notation is easier to evaluate in a computer.

In order to evaluate an expression in Polish notation, the expression is scanned from right to left, placing the elements in a stack. Each time an operator is found, the two top symbols of the stack are replaced by the result of applying the operator to those elements.

For example, the expression (*+234) (which in infix notation is "(2+3)*4)" would be evaluated like this:

Expression	Stack
* + 2 3 4	
* + 2 3	4
* + 2	3 4
* +	2 3 4
*	5 4
	20

Ques 17) How an algebraic expression can be converted in the form of a tree?

Ans: Tree of an Algebraic Expression

An algebraic expression can be represented by a binary rooted tree obtained recursively in the following way. The tree for a constant or variable a has a as its only vertex. If the algebraic expression S is of the form S_L • S_R, where S_L and S_R are sub expressions with trees T_L and T_R respectively and • is an operator, then the tree T for S consists of • as root, and the sub trees T_L and T_R (**figure 3.9**).

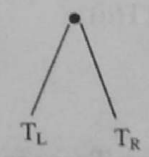


Figure 3.9: Tree of S_1 • S_2

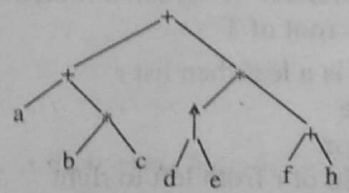


Figure 3.10: Tree for a + b * c + d ↑ e * (f + h)

For example, consider the following algebraic expression:

A + b*c + d↑e*(f + h)

Where + denotes addition, * denotes multiplication and ↑ denotes exponentiation. The binary tree for this expression is given in **figure 3.10**.

Given the binary tree of an algebraic expression, its Polish, reverse Polish and infix representations are different ways of ordering the vertices of the tree, namely in preorder, postorder and inorder respectively.

Ques 18) What are traversal algorithms? Write algorithms for preorder, postorder and inorder traversal algorithms.

Ans: Traversal Algorithms

Tree traversal literally means the visiting up of the nodes of the tree line-wise in accordance with the constraints. There are well defined traversals which are preorder, postorder and inorder in accordance with the three notations of infix, postfix and prefix.

The following are recursive definitions of several orderings of the vertices of a rooted tree T = (V, E) with root r. If T has only one vertex r, then r by itself constitutes the preorder, postorder and inorder transversal of T. Otherwise, let T_1,..., T_k the subtrees of T from left to right (**figure 3.11**).

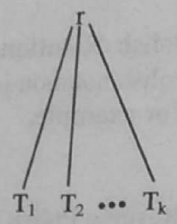


Figure 3.11: Ordering of Trees

Then,

1) **Preorder Transversal:** Pre(T) = r(root), Pre(T_1),.....Pre(T_k).
2) **Postorder Transversal:** Post(T) = Post(T_1),..., Post(T_k), r.
3) **Inorder Transversal:** If T is a binary tree with root r, left subtree T_L and right subtree T_R, then, In(T) = In(T_1), r, In(T_k).

Algorithm: Preorder Traversal

procedure **preorder**(T: ordered rooted tree)
Step 1) r := root of T
Step 2) list r
Step 3) for each child c of r from left to right
begin
T(c) := subtree with c as its root
preorder (T(c))
Step 4) end

Algorithm: Postorder Traversal

procedure **postorder** (T: ordered rooted tree)
Step 1) r := root of T
Step 2) for each child c of r from left to right
begin
T(c) := subtree with c as its root
postorder (T(c))
Step 3) end
Step 4) list r

Algorithm: Inorder Traversal

procedure **inorder** (T: ordered rooted tree)
Step1) r := root of T
Step2) if r is a leaf then list r
else
begin
l = first child of r from left to right
T(l) := subtree with l as its root
Inorder (T(l))

Step3) list r

Step4) for each child c of r except for l from left to right
T(c) := subtree with c as its root
Inorder (T(c))

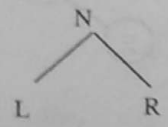
Step5) end

Ques 19) The following sequence gives the preorder and inorder traversal output of the Binary Tree T

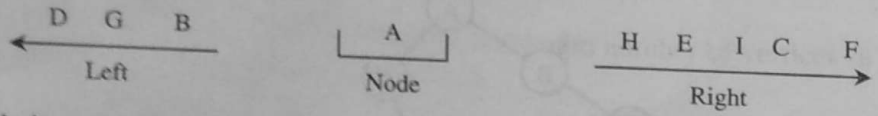
Preorder: A B D G C E H I F
Inorder: D G B A H E I C F

Draw the corresponding tree.

Ans: From the preorder it is known that A is the root as in the preorder the NLR sequence is there.

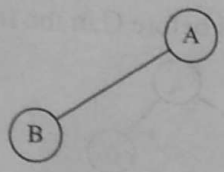


Therefore, now it can be said that the left subtree consists of elements DGB and right subtree consists of elements H E I C F from the inorder traversal.

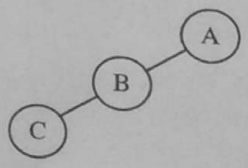


Now tracing each node in to tree step by step start with the root Node A.

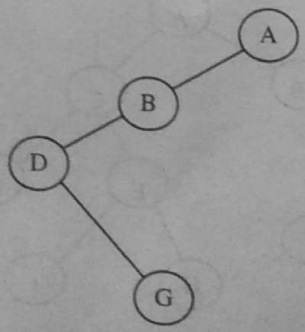
The next element in the preorder is B, i.e., next on the left tree i.e. to be traversed therefore it comes on the left of A.



The next element is D, i.e., to be fixed at proper position, according to the preorder notation the next element is D and from inorder expression the leftmost is D. Therefore we put D at left to B.

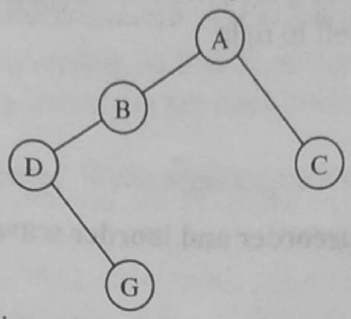


The only left element is G in the Left subtree. Therefore we think for G. As in preorder the NLR is BDG therefore it means G is on the right of D.

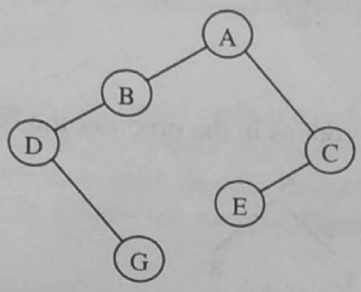


Now we have the following order:
Preorder: C E H I F
Inorder: H E I C F

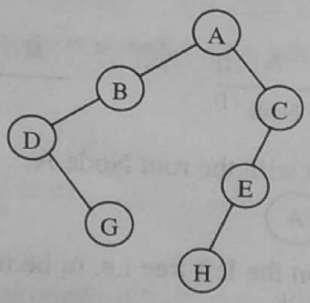
According to preorder traversal the C is the first on the right side of tree after the root Node A.



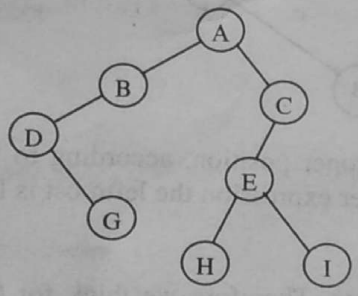
The next element is E in the preorder, i.e., when we have to Left from C then the next Node to be printed is E on the left of C:



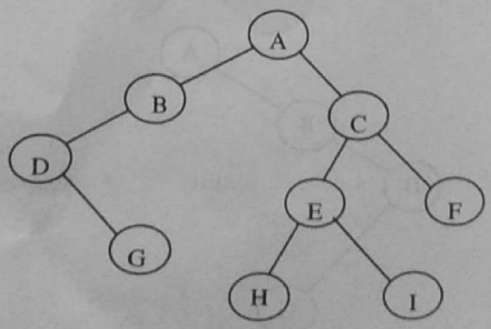
Next element is H in the preorder and H is the leftmost in the remaining Inorder expression. So it will go on the left of E and the tree is:



The next element is I in the preorder expression and before C in the Inorder expression therefore it will go to the right of E:



The last element is F both in Inorder and preorder which the rightmost element. It will be in the right of C:



To crosscheck the result, traverse the tree which yields the following results:

Preorder: A B D G C E H I F
Inorder: D G B A H E I C F
Postorder: G D B H I E F C A

COMPLETE BINARY TREE

Ques 20) Define complete binary tree.

Ans: Complete Binary Tree

A complete binary tree is a full binary tree in which all the pendent vertices are on the same level. In the complete binary tree there is only one vertex on the 0 level. There are 2 vertices on the 1 level, 4 vertices on the 2 level, 8 vertices on the 3 level and so on. Following shows a complete binary tree.

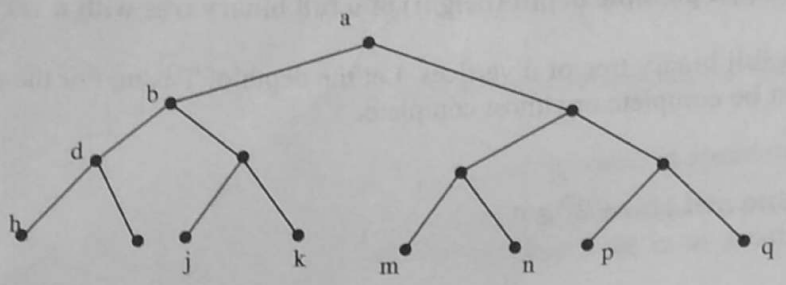


Figure 3.12

Ques 21) Find the maximum number of vertices possible in full binary tree?

Ans: Let T = (V, E) is an m-level full binary tree. For the maximum number of vertices in T, the tree T must be complete binary tree. Let number of vertices in T = n. Then

$n = 2^0 + 2_1 + 2^2 + 2^3 + \ldots\ldots\ldots\ldots\ldots\ldots + 2^m$ (a G.P)

$n = 2^0(^{2m+1} - 1) / (2 - 1)$. (Sum of m + 1terms)

$n = (2^{m+1} - 1)$.

Therefore the maximum number of vertices possible in a full binary tree= $(2^{m+1} - 1)$.

Ques 22) Find the number of internal vertices in an m-level full binary tree?

Ans: Let T = (V, E) is an m-level full binary tree. There arise two cases.

Case I: Let full binary tree is complete, then the total no of vertices in T is $(2^{m+1} - 1)$.

Let number of internal vertices = n, then number of pendent vertices = n + 1.

Therefore $n + n + 1 = 2^{m+1} - 1$.

$\Rightarrow 2n + 1 = 2^{m+1} - 1$

$\Rightarrow 2n = 2^{m+1} - 2$

$\Rightarrow n = 2^m - 1$.

Case II: Let full binary tree is not complete and T has minimum number of vertices. In this case total number of vertices = 2m + 1. Let number of internal vertices = n, then number of pendent vertices = n + 1.

Therefore, $n + n + 1 = 2m + 1$.

$\Rightarrow \quad n = m$.

Therefore the number of internal vertices in an m-level full binary tree lies between m and $2^m - 1$.

Note: Every complete binary tree is a full binary tree but converse is not true.

Ques 23) Find the maximum possible depth (height) of a full binary tree with n vertices (n is odd)?

Ans: Let T = (V, E) is a full binary tree of n vertices. Let the depth of T is m. For the maximum possible depth of the binary tree, the farthest vertices must be as far as possible from the root. In this case, the total number of vertices $n = 2m + 1$.

$$\Rightarrow m = (n - 1)/2.$$

Therefore the full maximum possible depth of a full binary tree with n vertices is $(n - 1)/2$.

Ques 24) Find the minimum possible depth (height) of a full binary tree with n vertices (n is odd)?

Ans: Let T = (V, E) is a full binary tree of n vertices. Let the depth of T is m. For the minimum possible depth of the binary tree, T must be complete or almost complete.

Therefore,

$2^0 + 2^1 + 2^2 + 2^3 + \ldots\ldots\ldots\ldots\ldots + 2^m \geq n.$

$\Rightarrow 2^{m+1} - 1 \geq n.$

$\Rightarrow 2^{m+1} \geq n + 1.$

$\Rightarrow m + 1 \geq \text{Log}_2(n + 1).$

$\Rightarrow m \geq \text{Log}_2(n + 1) - 1.$

$\Rightarrow m = [\text{Log}_2(n + 1) - 1].$ ([x] = An integer greater or equal to x)

i.e., m = An integer greater or equal to $[\text{Log}_2(n + 1) - 1]$.

SPANNING TREE

Ques 25) What is the spanning tree? Explain using a diagram.

Ans: Spanning Tree

A tree T is said to be spanning tree of a connected graph G if T is a subgraph of G and T contains all vertices of G, i.e., if $G = (V_1, E_1)$ is a connected graph, then the tree $T = (V_2, E_2)$ is a spanning tree of G if $V_1 = V_2$.

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. A spanning tree is defined only for 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 of a disconnected graph, however, does have a spanning tree.

Thus disconnected graph with k components has a spanning forest consisting of k spanning trees.

Every connected graph has a spanning tree which can be obtained by removing edges until the resulting graph becomes acyclic, i.e., where there are no cycles. **Figures 3.13 and 3.14** shows a connected graph and its spanning tree.

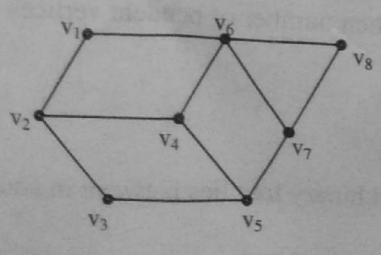


Figure 3.13: Graph G

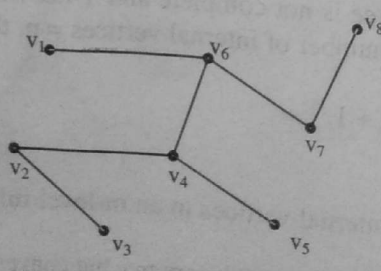


Figure 3.14: Spanning Tree of Graph G

Ques 26) Define minimum cost spanning tree.

Ans: Minimum Cost Spanning Tree

The minimum spanning tree is defined on a weighted, undirected, connected graph. A minimum-cost spanning tree in which the sum of the weights on the edges is a minimum.

It is a tree from the set of spanning trees, which has minimum weight. A minimum spanning tree of a weighted graph G is the spanning tree of G whose edges sum to minimum weight.

Ques 27) Write down the applications of the minimum spanning trees.

Ans: Applications of Minimum Spanning Trees

1) Minimum spanning trees are useful in constructing networks, by describing the way to connect a set of sites using the smallest total amount of wire.
2) Minimum spanning trees provide a reasonable way for clustering points in space into natural groups.
3) When the cities are points in the Euclidean plane, the minimum spanning tree provides a good heuristic for traveling salesman problems. The optimum traveling salesman tour is at most twice the length of the minimum spanning tree.