Basic terminology of tree:-

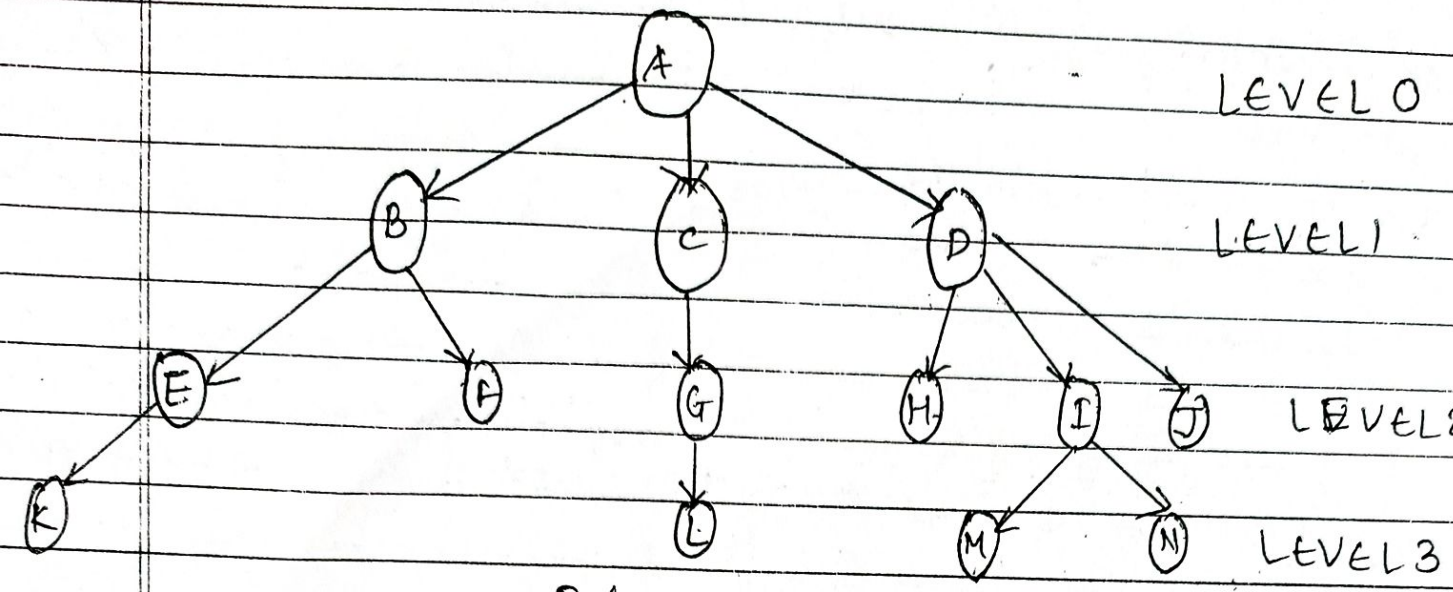


FIGURE-1 (general tree)

⊙ **Root:-** The first node of the tree is called Root node.

In figure-1 "A" is denoted as the root node

⊙ **Parent:-** Parent of a node is the immidiate predecessor of a node.

In figure-1 A is the parent of node B, C, D and B is the parent of E and F.

⊙ **child:-** Each immediate successor of a node is known as child.

In figure-1, B, C, D are the child nodes of A and L and is the child of node G.

⊙ Sibling:- If different nodes having a common parent, that different nodes is called sibling.

In figure 1, B, C, D are sibling and node E and F are also sibling.

⊙ Degree of a node:- The no. of sub trees of a node is known as degree of that node.

In figure 1, degree of node A is equals to 3. degree of node B is 2, node C is 1 and node D is 3.

⊙ Degree of a tree:- The maximum degree of nodes in a given tree is called as the degree of a tree.

In figure-1, degree of the tree is equals to 3.

⊙ Terminal Node or Leaf Node:- The node having zero degree zero is called as leaf node or Terminal node.

In Figure-1 there are 7 leaf nodes that ar K, F, L, H, M, N, J.

⊙ Non terminal or :- The nodes having degree greater than zero is called as non-terminal node.

In figure 1 there are 7 non terminal node that are A, B, E, C, G, D, I.

⊙ **Depth :-** The depth of node n_i is the length of the unique path from the root to n_i.

In figure ① the depth of node I is 2 and the depth of node A is 0 and the depth node B is 1.

⊙ **Height :-** The height of node n_i is the length of the longest path from n_i to leaf.

In figure ① height of node F is 0., height of node B is 2. (choose longest path).

◉ **Binary Trees :-**

Binary tree is a tree which can hold atmost two elements.

⊙ **Strictly Binary Tree :**

If every non-leaf node in a binary tree has non-empty left and right subtree, the tree is called as strictly binary tree.

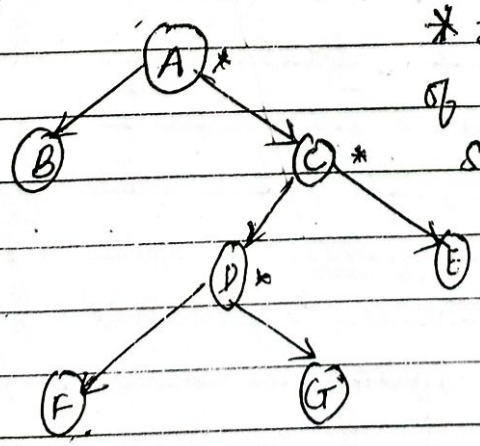


FIGURE - 2

* Remarks :-
* If there are n no. of leaf nodes in a strictly binary tree then total no. of nodes of the tree = $2n-1$

→ The above tree is defined as a strictly Binary tree.

⊙ **complete Binary Tree:-**

A binary tree of depth 'd' that contains exactly '2^d nodes' at level-d. is called as complete binary tree.

example:-

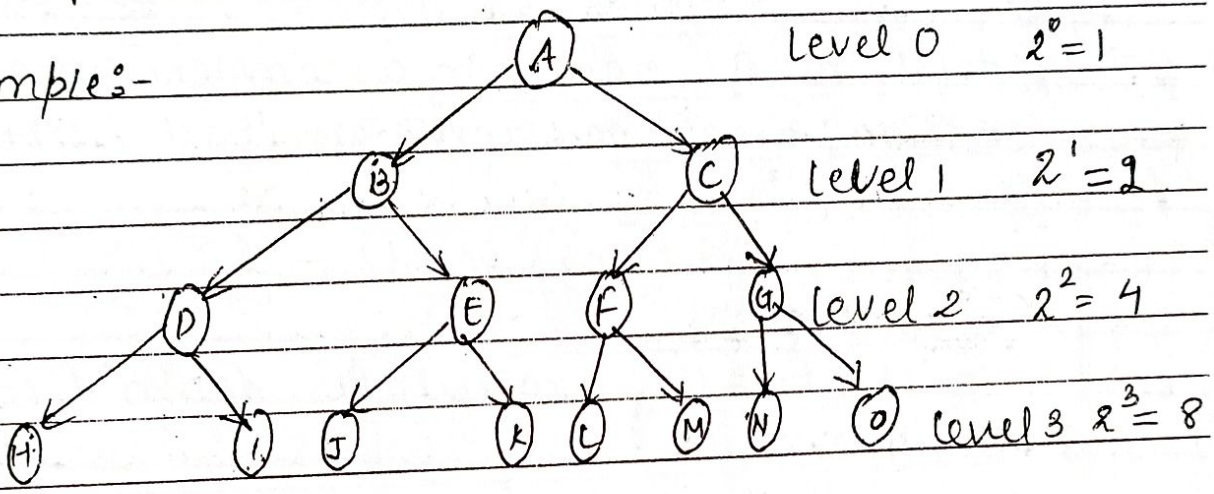


Figure-3

→ The above tree is a "complete Binary Tree".

→ the total number of nodes in a complete binary tree $\boxed{t_n = 2^{d+1} - 1}$

→ example:- If depth of a complete Binary tree is 4.

$$t_n = 2^{4+1} - 1 = 2^5 - 1 = 32 - 1 = 31$$

$$\boxed{t_n = 31}$$

No. of different binary trees with 'n' number of unlabel nodes = $\frac{(2n)!}{(n+1)!\,n!}$ ✓ Labeled $\frac{(2n)!}{(n+1)!}$

→ Since all the total number of leaf nodes will be defined as 2^d. then total no. of non-leaf nodes is defined as $2^d - 1$.

Example:- If $d = 5$ then total no. of nodes

$$t_n = 2^{5+1} - 1 = 63$$

Total no. of leaf node $= 2^d = 2^5 = 32$

Total no. of non leaf node $= 2^d - 1$

→ If total no. of nodes in a complete binary tree is known then you can calculate depth $d =$

$$\boxed{d = \log_2 (t_n + 1) - 1}$$

→ Ex:- Let $t_n = 15$; calculate depth 'd' of the given tree.

$$d = \log_2 (15+1) - 1$$

$$d = \log_2 (16) - 1$$

$$d = \log_2 (2)^4 - 1$$

$$d = 4 - 1$$

$$\boxed{d = 3}$$

→ $t_n = 127$; calculate depth 'd'.

$$d = \log_2 (127 + 1) - 1$$

$$d = \log_2 (128) - 1$$

$$d = \log_2 (2)^7 - 1 \quad = 7 - 1 = 6$$

$$\boxed{d = 6}$$

*	−*	ABD+E/*F
(	−*(	ABD+E/*F
G	−*(	ABD+E/*FG
+	−*(+	ABD+E/*FG
H	−*(+	ABD+E/*FGH
/	−*(+/	ABD+E/*FGH
K	−*(+/	ABD+E/*FGHK
)	−*	ABD+E/*FGHK/+
)		ABD+E/*FGHK/+*-

COMPETETIVE ICSVTU

Tree Traversal :-

- In tree traversing three methods are used for traversing a given tree-

1. In order traversing
2. Pre order traversing
3. Post order traversing

1. In order traversing :-

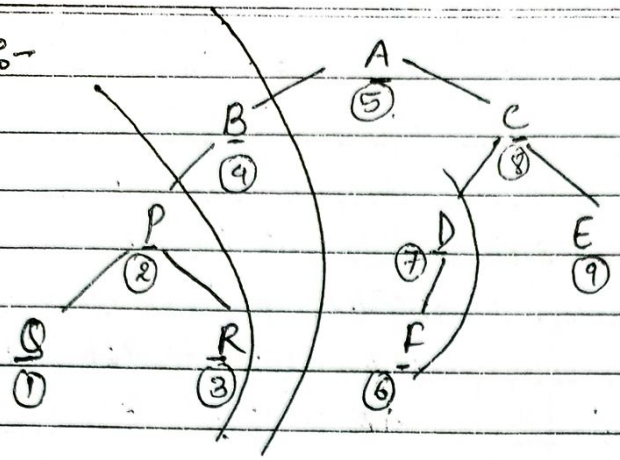
→ Algorithms for inorder traversing :-

inorder(LEFT SUBTREE, ROOT, RIGHT SUBTREE)
preorder(ROOT, LEFT SUBTREE, RIGHT SUBTREE)
postorder(LEFT SUBTREE, RIGHT SUBTREE, ROOT).

→ Algorithm :-

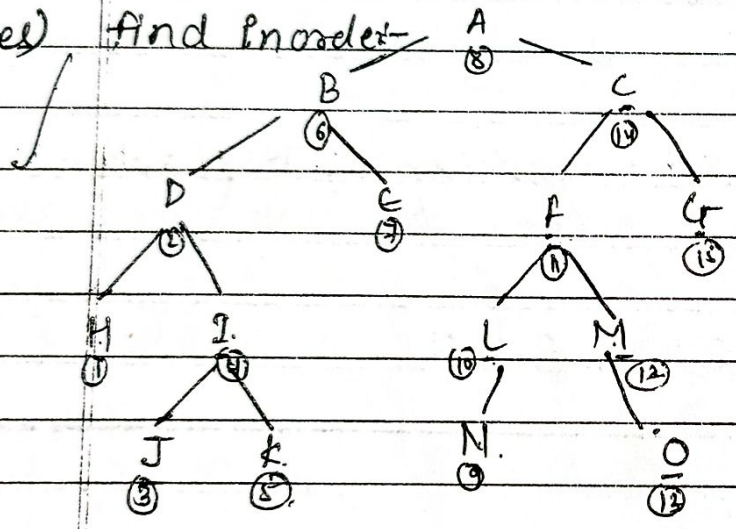
1. Traverse the left subtree element using inorder.
2. Process the ROOT element.
3. Process the right subtree using inorder.

Example:-



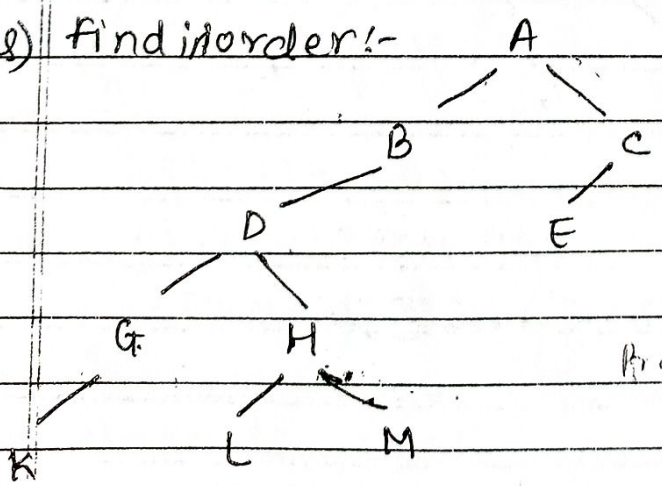
→ S P R B A F D C E Ans.

Ques) find Inorder-



→ H D J I K B E A N L F Ø M O C G Ans.

Ques) find inorder:-



Inorder:- A B D G K H L M C E

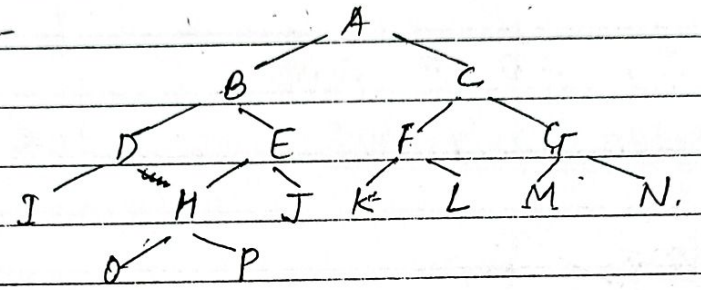
K G D L H M B A E C Ans.

- **Pre order Traversing:-**

Algorithm:-

1. Process the root element
2. Process the left subtree using pre-order.
3. Process the right subtree using pre-order.

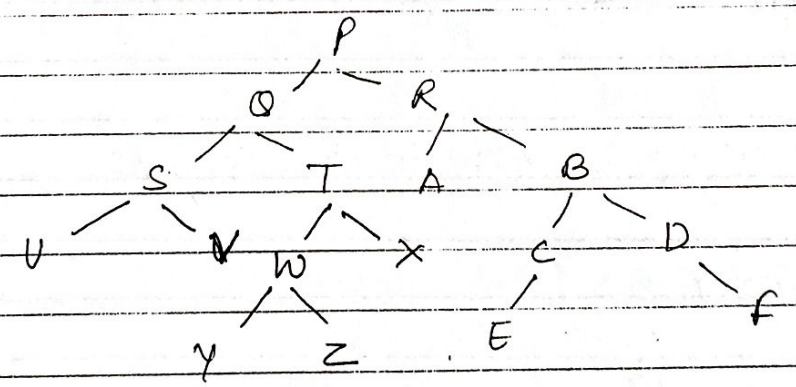
Example:-



find out the pre-order traversing of the above tree.

A B D I E H O P J C F K L G M N

Ques)



P Q S U V T W Y Z X R A B C E D F