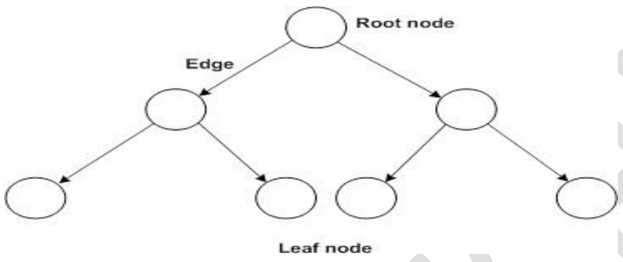
Trees

❖ 5.1 Non Linear Data Structure: Tree & Graph

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

- ✓ There is a special node called the root node R.
- ✓ The remaining nodes are divided into n > 0 disjoint sets T1, T2,.,.,. TN, where each of These sets are tree. T1, T2…., T n is called the sub tree of the root.



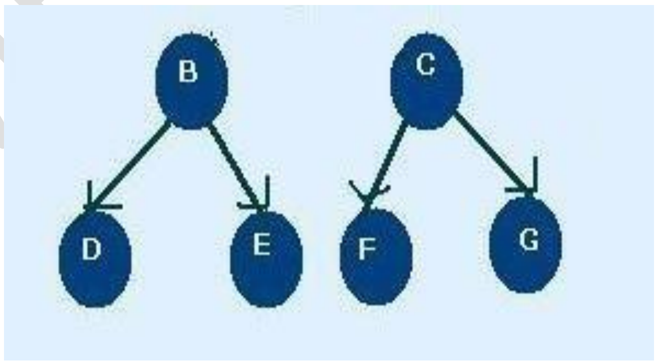
Graph: A graph is a set of vertices/nodes and edges. A tree is a set of nodes and edges. In the graph, there is no unique node which is known as root. In a tree, there is a unique node which is known as root.

❖ 5.2 Basic Terms

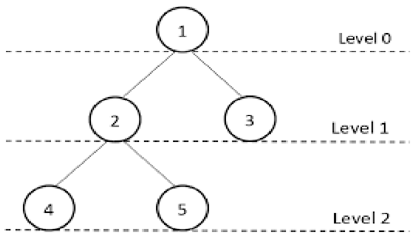
1. General Tree: A general tree is a tree where each node may have zero or more children (a binary tree is a specialized case of a general tree).

2. Forest: A forest is a set of n > 0 disjoint trees. Forest is a collection of disjoint trees.
Consider Following Figure in which a forest is a collection of two disjoint tree.

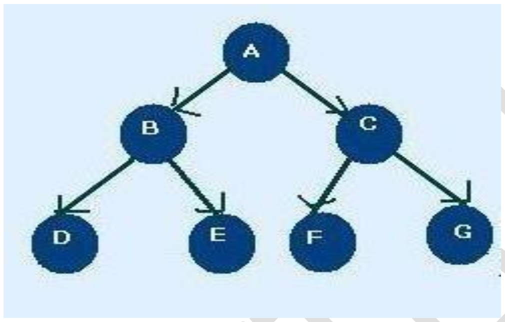
3. Binary Tree: A tree is called binary tree if each and every node has 0,1 or 2 branch(Out degree 0,1 or 2)



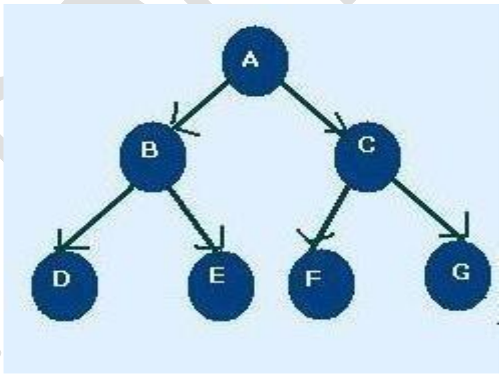
4. Level number: A level is the number of parent nodes corresponding to a given a node of the tree. It is basically the number of ancestors from that node until the root node. So, for the root node (topmost node), it's level is 0, since it has no parents.



5. Indegree: It is the number of branches terminated at a given node.
6. OutDegree: It is the number of branches emerging from a given node.
Root node having indegree always equals to 0 because it is the first node of tree.
Example: Indegree of node B is 1

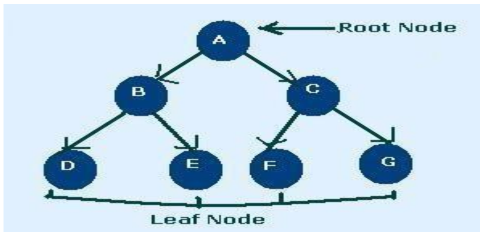


7. Out Degree: It is the number of branches emerging from a given node.
Leaf node having out degree always equals to 0 because it is the last node of tree having no furthersub tree. Example: Out degree of node B is 2.



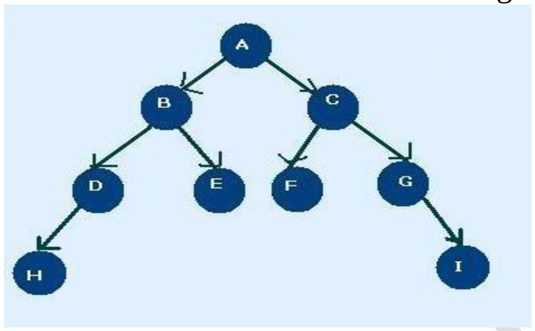
Total degree: It is sum of Indegree and Outdegree.

8. Root Node: The node at the top of the tree is called root. In a directed tree, a node which has indegree 0 is called the root node.



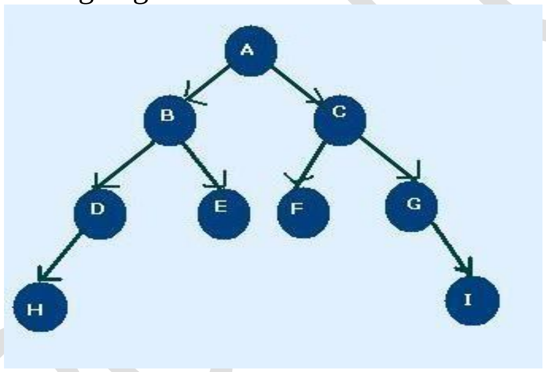
9. Leaf Node: A leaf node is any node that has two empty children. An internal node is any node that has at least one non-empty child.

10. Directed Edge: An edge of directed tree is called as directed edge.

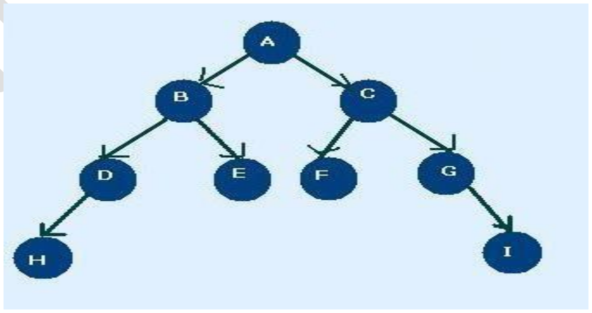


Here in above example an edge from A to B is directed edge.

11. Path: A sequence of connecting edges from one node to another node is called Path.

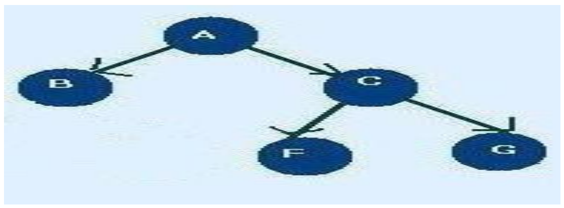


12. Depth: The length of the path from Root Node to particular Node V is called depth of node V. Ina tree, Maximum level no is called depth of the tree

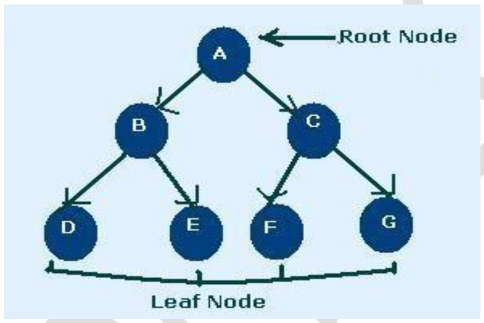


❖ 5.3 Binary Search Tree

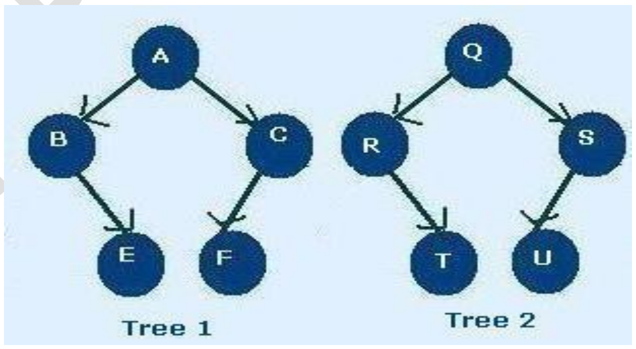
- ✓ Binary Tree: A tree is called binary tree if each and every node has 0,1 or 2 branch(Out degree 0,1 or 2)
- ✓ Strictly binary tree: A tree is called Strictly binary tree if each and every node has 0 or 2 branch(Out degree 0 or 2)



- ✓ Complete Binary Tree: A binary tree is called Complete binary tree , if all level except last havethe maximum number of possible nodes.($2^{levelno}$)

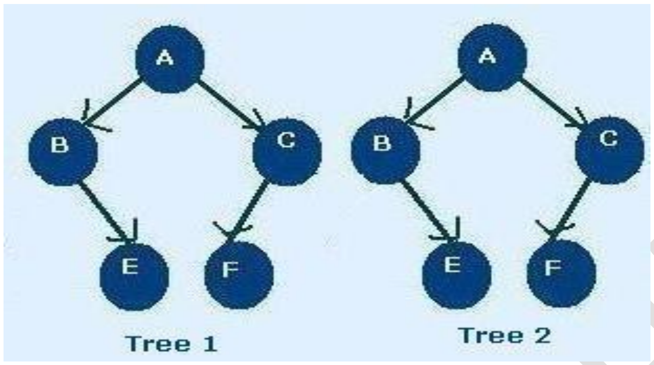


- ✓ Similar binary trees: If two binary trees are similar in structure then they are said to be **similar binary trees**. Consider following figure:

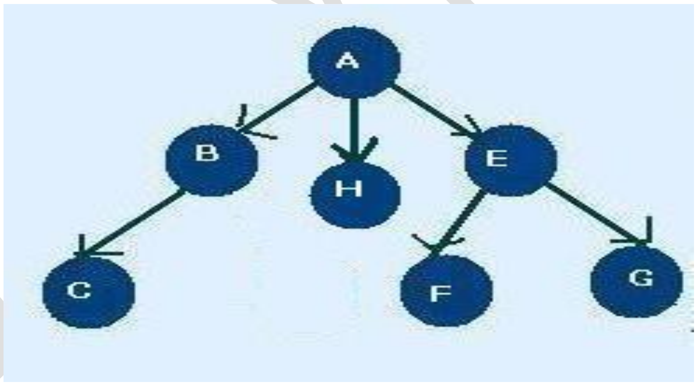


- ✓ Copies of binary trees: If two binary trees are similar in structure and their corresponding nodes having same value then they are said to be **copies of binary trees**.

Consider following figure:



- ✓ General Tree: A Tree in which each node having either 0 or more child nodes is called **generaltree**.
  - ✓ So we can say that a Binary Tree is a specialized case of General tree.
  - ✓ General Tree is used to implement File System.
  - ✓ Example:



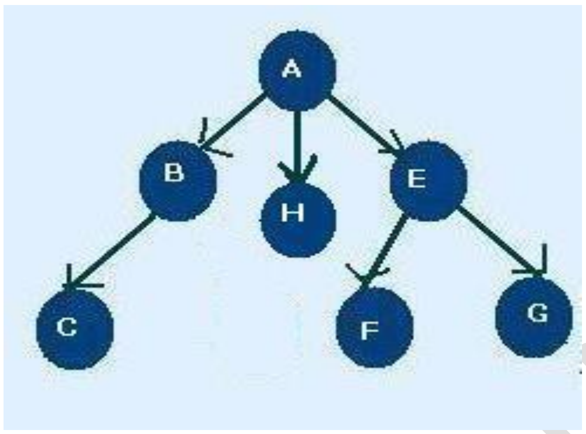
Conversion of General Tree to Binary Tree

- The process of converting general tree in to binary tree is given below:

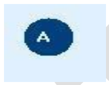
1. Root node of general tree becomes root node of Binary Tree.
2. Now consider T1, T2, T3 ... Tn are child nodes of the root node in general tree. The left most child (T1) of the root node in general tree becomes left most child of root node in the binary tree. Now Node T2 becomes right child of Node T1, Node T3 becomes right child of Node T2 and so on in binary tree.
3. The same procedure of step 2 is repeated for each leftmost node in the general tree.

Example:

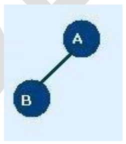
General tree is given in the figure.



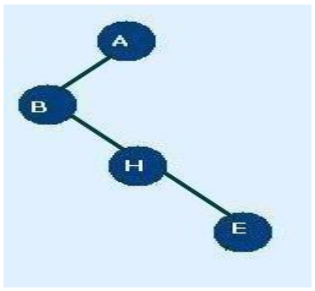
Step 1: Root Node of General tree becomes the Root node of binary tree.



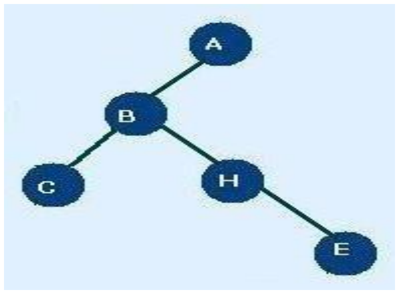
Step 2: Now Root Node (A) has three child Nodes (B, H, E) in general tree. The leftmost node (B) of the root node (A) in the general tree becomes the left most node of the root node (A) in binary tree.



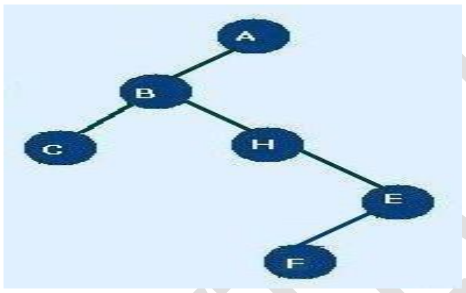
Step 3: Now Node H becomes the right node of B and Node E becomes the right node of H.



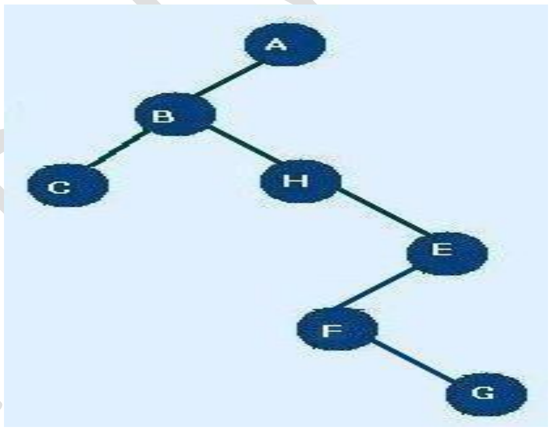
Step 4: Now Node B has only one left child node, which is C in general tree. So NodeC becomes left child of Node B in binary tree.



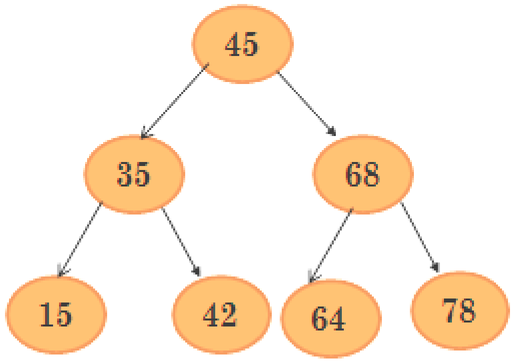
Step 5: Now Node E has two child nodes (F, G). The leftmost node (F) of the node (E) in the general tree becomes the left most node of the node E in the binary tree.



Step 6**:** Now **Node** G becomes right node of Node F in binary tree.



- **What is binary Search Tree?**
  - ✓ A tree is called binary search tree if each and every node can have 0,1 or 2 branches.Andit should contain following characteristics.
  - ✓ All the nodes to the left of the root node have value less than the value of the root node.
  - ✓ All the nodes to the right of the root node have value greather than the value of the root node.

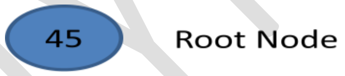


- ✓ On binary search tree, we can perform several operations like,
  - Insertion of a node in Binary Tree
  - Deletion of a node from Binary tree
  - Searching a node from Binary Tree

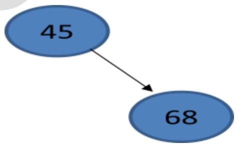
➢ **Insertion of a node in binary Search tree**

Suppose we want to construct binary search tree for following set of data:45 68 35 42 15 64 78

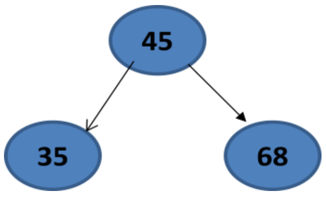
- ✓ Step 1: First element is 45 so it is inserted as a root node of the tree.



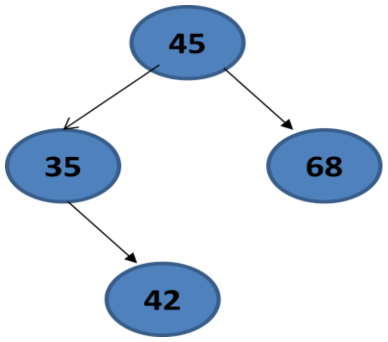
- ✓ Step 2: Now we have to insert 68. First we compare 68 with the root node which is 45. Since the value of 68 is greater then 45 so it is inserted to the right of the root node.



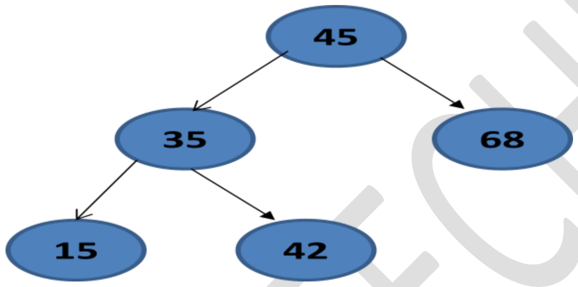
- ✓ Step 3: Now we have to insert 35. First we compare 35 with the root node which is 45. Since the value of 35 is less then 45 so it is inserted to the left of the root node.



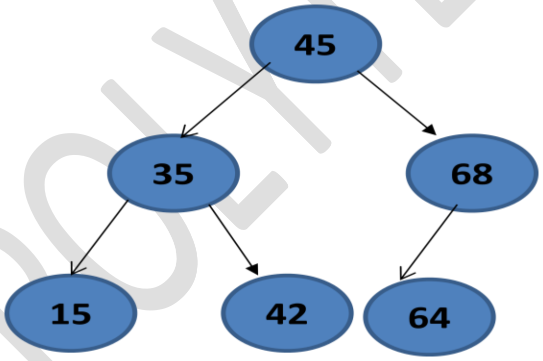
- ✓ Step 4: Now we have to insert 42.



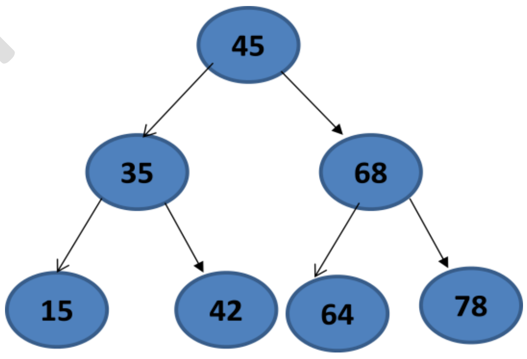
- ✓ Step 5: Now we have to insert 15



- ✓ Step 6: Now we have to insert 64.

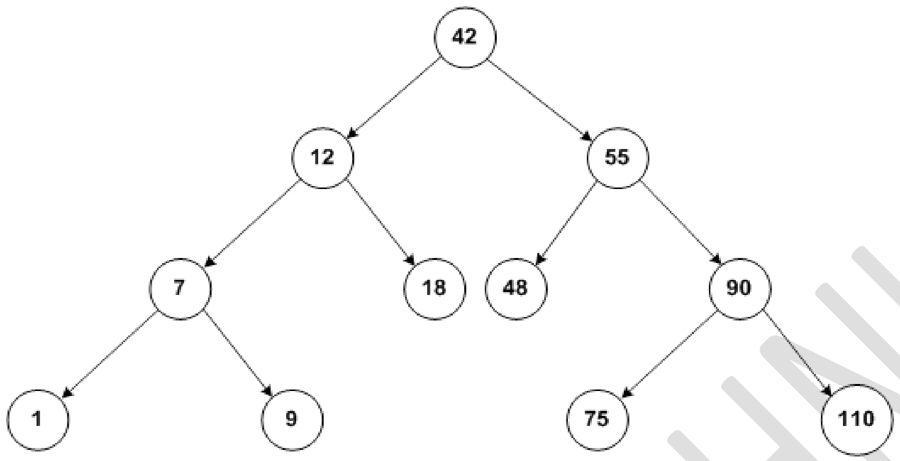


- ✓ Step 7: Now we have to insert 78.



- ➢ **Deletion of a node from binary Search tree**

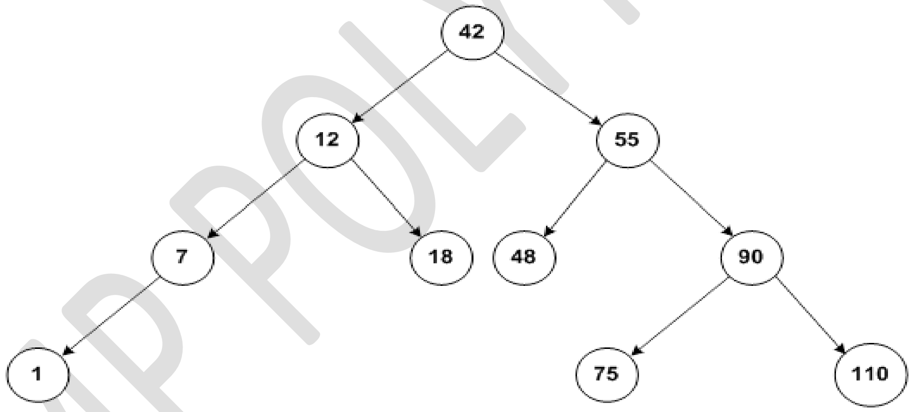
Consider the following binary tree:



There are three possibilities when we want to delete an element from binary search tree.

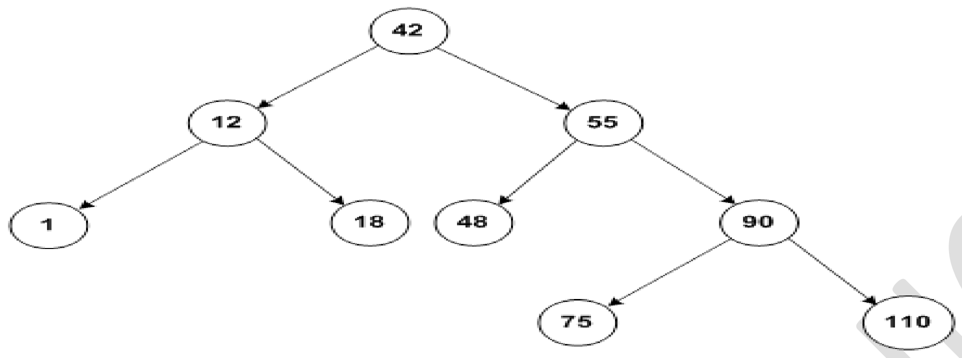
(1) If a node to be deleted has empty left and right sub tree then a node is deleted directly.

Suppose we want to delete node 9



(2) If a node to be deleted has only one left sub tree or right sub tree then the sub tree of the deleted node is linked directly with the parent of the deleted node.

Suppose we want to delete node 7



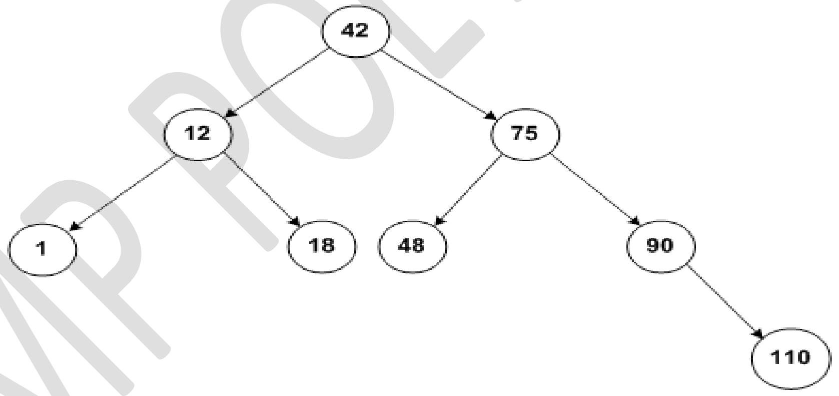
(3)If a node to be deleted has left and right sub tree then we have to do following steps:

(a) Find next maximum (in order successor) of the deleted node.

(b) Append the right sub tree of the in order successor to its grandparent.

(c) Replace the node to be deleted with its next maximum (in order successor).

Suppose we want to delete node 55.



➢ Searching a node in binary Search tree

First the key to be searched is compared with root node. If it is not in the root node then we haveto possibilities:

(1) If the key to be searched having value less than root node then we search the key in left subtree.

(2) If the key to be searched having value greater then root node then we search the key in rightsub tree.

```
Step 1:

        If ROOT == NULL then
                Write "Tree is Empty. Search Un Successful"

         Exit

Step 2:

        If X=ROOT->INFO then

              Write "Search is Successful"

        Return (ROOT)

Step 3:

        If X  <  ROOT->INFO then

              Call SEARCH (ROOT->LPTR, X)

        Else

              Call SEARCH (ROOT->RPTR, X)
```

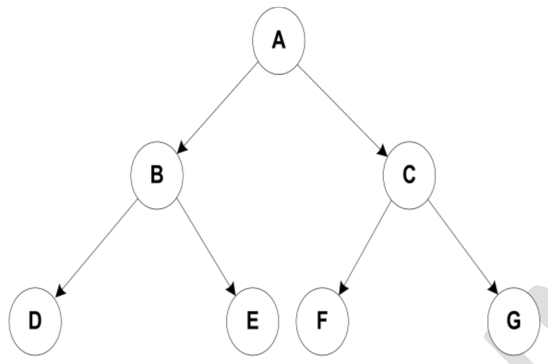
❖ 5.4 Binary tree traversal

- ✓ Traversal is the method of processing every nodes in the tree exactly once in a systematic manner.
- ✓ Types of Traversal:
- ✓ There are three different types of tree traversal.
  1. Preorder Traversal
  2. In order Traversal
  3. Post order Traversal

1. Preorder Traversal

- ✓ The Preorder traversal follows the three steps:
  1. Process the root node. (Vertices or Node)

2. Traverse the left sub tree in Post order. (Left node)
3. Traverse the right sub tree in Post order. (Right node)



Preorder Traversal: A B D E C F G

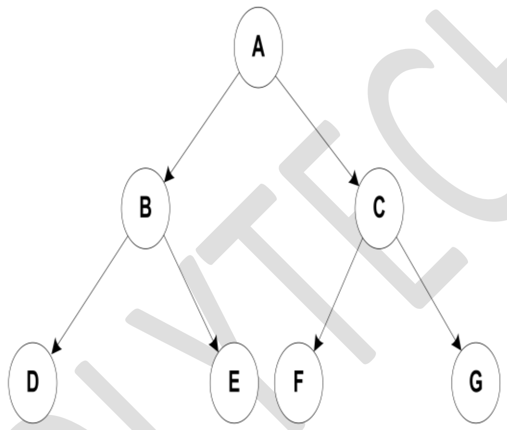
```
PREORDER(T)

Step 1:[Check for empty tree]
        If T == NULL then
                Write "Tree is empty"Exit
Step 2:[Process the root node]
        Write (DATA(T))
Step 3:[Process the left sub tree]
        If LPTR (T) ≠ NULL then
                Call PREORDER (LPTR (T))
Step 4:[Process the right sub tree]
        If RPTR (T) ≠ NULL then
                Call PREORDER (RPTR (T))
Step 5:[Finished]
            Exit
```

2. Inorder Traversal

- ✓ The Inorder traversal follows the three steps:
    1. Traverse the left sub tree in Post order. (*L*eft node)
    2. Process the root node. (*V*ertices or Node)
    3. Traverse the right sub tree in Post order. (*R*ight node)

Example:



Inorder Traversal: D B E A F C G

INORDER (T)

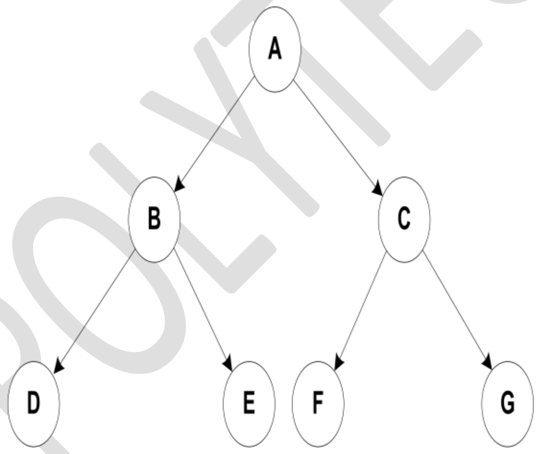
```
    Step 1: [Check for empty tree]
                If T == NULL then
                        Write "Tree is empty"
                Exit
    Step 2: [Process the left sub tree]
                If LPTR (T) ≠ NULL then
                            Call INORDER (LPTR (T))
    Step 3: [Process the root node]
                Write (DATA (T))
    Step 4: [Process the right sub tree]
```

```
        If RPTR (T) ≠ NULL then
                Call INORDER (RPTR (T))
Step 5:[Finished]
            Exit
```

3.Post order Traversal

✓ The Post order traversal follows the three steps:

1. Traverse the left sub tree in Post order. (Left node)
2. Traverse the right sub tree in Post order. (Right node)
3. Process the root node. (Vertices or Node)



Post Order Traversal: D E B F G C A

POSTORDER (T)

```
Step 1: [Check for empty tree]
        If T == NULL then
            Write "Tree is empty"
                ExitStep
2: [Process the left sub tree]
```

```
            If LPTR (T) ≠ NULL then
                    Call POSTORDER (LPTR (T))
    Step 3: [Process the right sub tree]
            If RPTR (T) ≠ NULL then
                    Call POSTORDER (RPTR (T))
    Step 4: [Process the root node]
            Write (DATA (T))
    Step 5: [Finished]
                  Exit
```

❖ 5.5 Write application of binary tree.

- ✓ Manipulation of arithmetic expressions
  Binary tree is used to represent formulas in prefix or postfix notation.For

  Example: Infix- A+B
- ✓ Syntax Analysis

  It deals with the use of grammars in syntax analysis or parsing.
- ✓ Construction and maintenance of symbol table

  Binary tree will formulate the algorithms that will maintain a stack implemented tree structured symbol table.