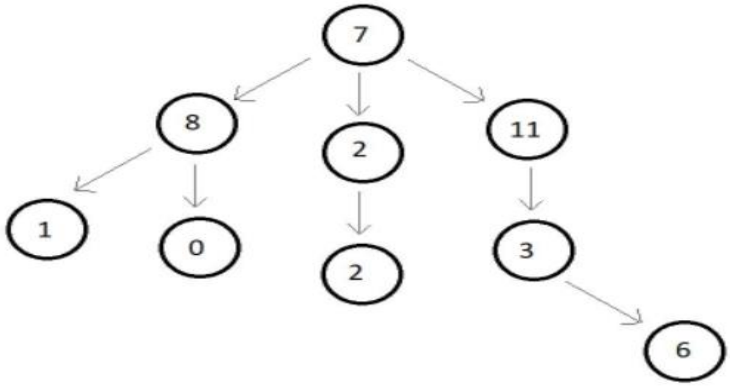
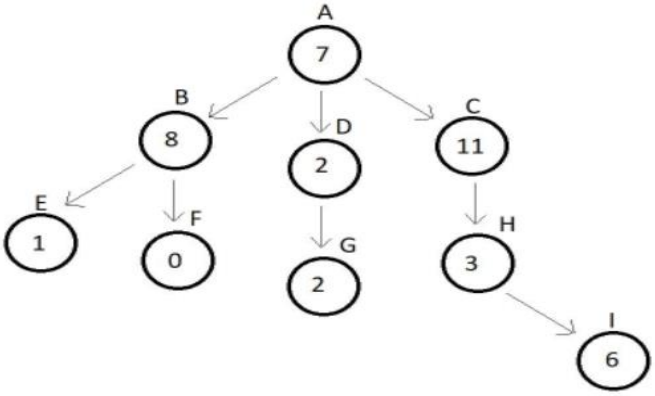
# • TRAVERSAL TECHNIQUES FOR TREES

*In this Article , we will Discuss about trees and also the Traversal techniques/Method used to solve a tree, And to find its value.*

**Tree:** *A tree usually represents the hierarchy of elements and depicts the relationships between the elements. A tree is a collection of nodes connected by directed (or undirected) edges. A tree can be empty with no nodes or a tree is a structure consisting of one node called the* ***root*** *and zero or one or more subtrees. Trees are considered as one of the largely used facets of data structures. To give you a better idea of how a tree looks like, let me give an example:*



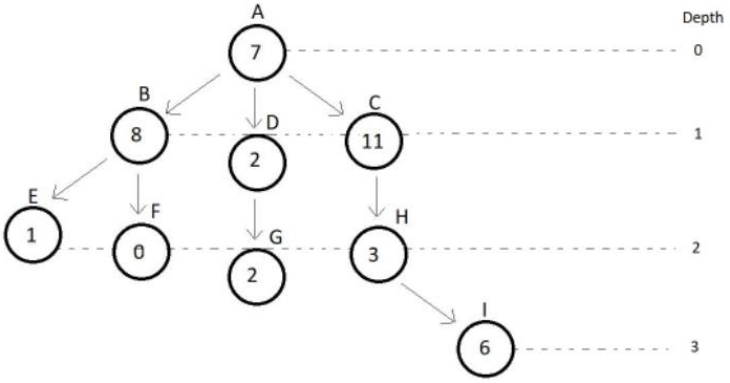

*Every circle represents a node, and every arrow represents the hierarchy.To understand this concept a little better we will name these nodes.*



# • TRAVERSAL TECHNIQUES FOR TREES

*Now, we can easily understand that the node C is the child of node A or node B is the parent of node E. There are some terminologies used in trees to get the proper concept about the trees, So we will discuss those terms below:*

1. **Root:** *The topmost node of a tree is called the root. There is no edge pointing to it, but one or more than one edge originating from it. Here, A is the root node.*
2. **Parent:** *Any node which connects to the child. Node which has an edge pointing to some other node. Here, C is the parent of H.*
3. **Child:** *Any node which is connected to a parent node. Node which has an edge pointing to it from some other node. Here, H is the child of C.*
4. **Leaf/External Node:** *Nodes which have no edge originating from it, and have no child attached to it. These nodes cannot be a parent. Here, nodes E, F, G and I are leaf nodes.*
5. **Internal Node:** *Nodes with at least one child. Here, nodes B, D and C are internal nodes.*
6. **Depth:** *Depth of a node is the number of edges from root to that node. Here, the depth of nodes A, C, H and I are 0,*



- # TRAVERSAL TECHNIQUES FOR TREES

*1, 2 and 3 respectively.*

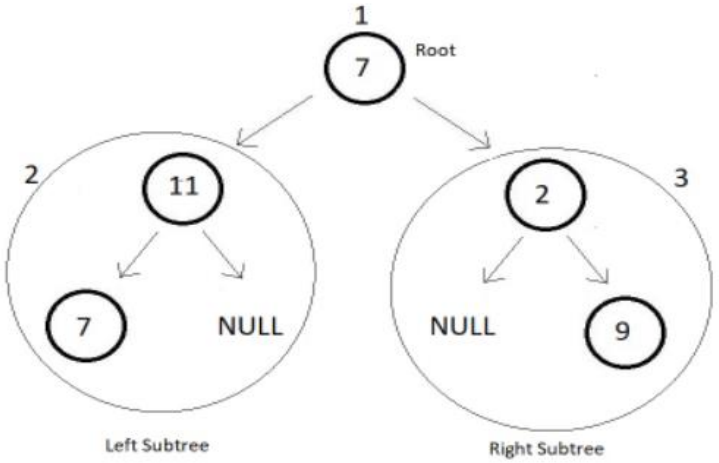
7. **Height:** *Height of a node is the number of edges from that node to the deepest leaf. Here, the height of node A is 3, since the deepest leaf from this node is node I. And similarly, height of node C is 2.*

   *Moving Further, As we have get enough explanation about trees so now we will discuss about the Techniques/ Traversal Method by which we can easily solve any tree.*

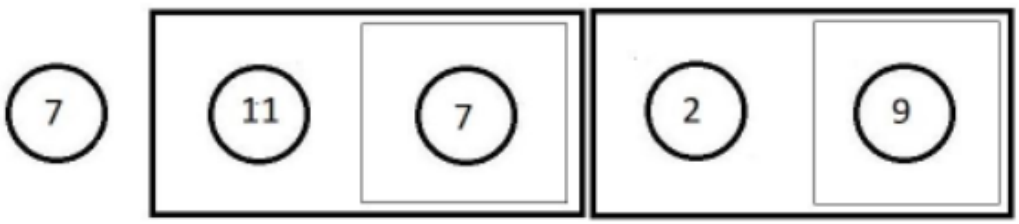
- *There Are Basically three methods to traverse a binary tree:*
  I. *Preorder Traversal.*
  II. *Inorder Traversal.*
  III. *Postorder Traversal.*

- **Preorder Traversal**: *So in this traversal technique, the first node you start with is the root node. And thereafter you visit the left subtree and then the right subtree. Taking the above example, I'll mark your order of traversal as below. You first visit section 1, then 2, and then 3.*

- # TRAVERSAL TECHNIQUES FOR TREES




*Now, this was to give you a general idea of the traverse in a binary tree using PreOrder Traversal. Each time you get a tree, you first visit its root node, and then move to its left subtree, and then to the right.So, here you first visit the root node element 7 and then move to the left subtree. The left subtree in itself is a tree with root node 11. So, you visit that and move further to the left subtree of this subtree. There you see a single element 7, you visit that, and then move to the right subtree which is a NULL. So, you're finished with the left subtree of the main tree. Now, you move to the right subtree which has element 2 as its node. And then a NULL to its left and element 9 to its right. So basically, you recursively visit each subtree in the same order. And your final traversal order is:*



- # TRAVERSAL TECHNIQUES FOR TREES

*Here, each block represents a different subtree.*

***Algorithm:***

*Until all nodes of the tree are not visited*

*Step 1 - Visit the root node*

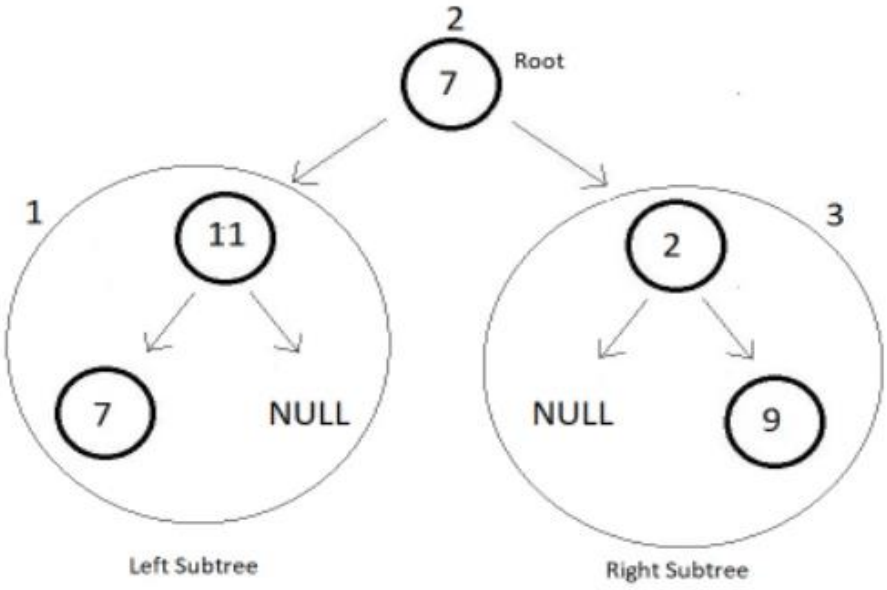
*Step 2 - Traverse the left subtree recursively.*

*Step 3 - Traverse the right subtree recursively.*

*The applications of preorder traversal include –*

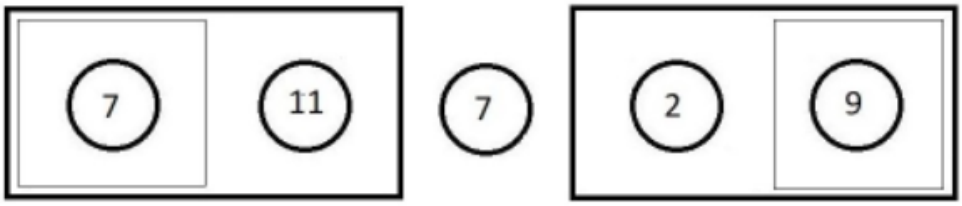
- *It is used to create a copy of the tree.*
- *It can also be used to get the prefix expression of an expression tree.*

- **Inorder Traversal:** *In this traversal technique, we simply start with the left subtree, that is you first visit the left subtree, and then go to the root node and then you'll visit the right subtree. Taking the same above example, I'll mark your order of traversal as below. You first visit section 1, then 2, and then 3.*



# • TRAVERSAL TECHNIQUES FOR TREES

*This was a general idea of you traverse in a binary tree using InOrder Traversal. Each time you get a tree, you first visit its left subtree, and then its root node, and then finally its right subtree. I expect you to write the flow of the traversal in InOrder yourself, and let me know if you could.*




*Here, each block represents a different subtree. So basically, the names PreOrder, PostOrder, and InOrder were given with respect to the position we keep our root node at while traversing in the Binary Tree. Since we first visit the root node in the first technique, that's why the name PreOrder. And we visit the root node at last in the PostOrder and in between in InOrder. Each of these techniques has one thing in common: the left subtree is traversed before the right subtree.*

***Algorithm:*** *Until all nodes of the tree are not visited*

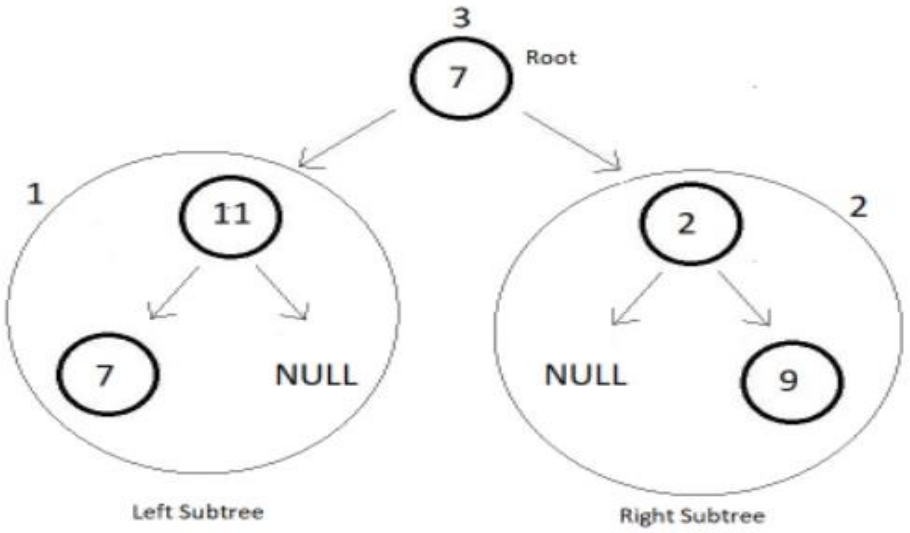
*Step 1 - Traverse the left subtree recursively.*
*Step 2 - Visit the root node.*
*Step 3 - Traverse the right subtree recursively.*

- # TRAVERSAL TECHNIQUES FOR TREES

*The applications of Inorder traversal includes –*

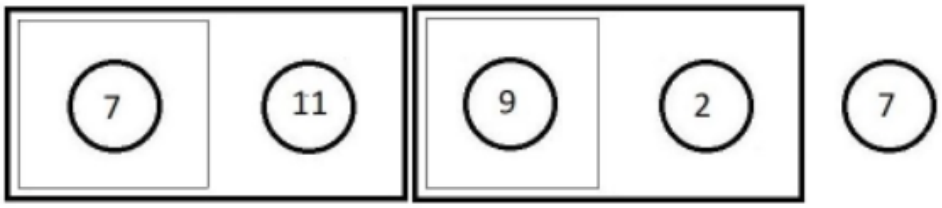
- *It is used to get the BST nodes in increasing order.*
- *It can also be used to get the prefix expression of an expression tree.*

- **Postorder Traversal**: *In this traversal technique, things are quite opposite to the PreOrder traversal. Here, you first visit the left subtree, and then the right subtree. So, the last node you'll visit is the root node. Taking the same above example, I'll mark your order of traversal as below. You first visit section 1, then 2, and then 3.*




*This was again a general idea of you traverse in a binary tree using PostOrder Traversal. Each time you get a tree, you first visit its left subtree, and then its right subtree, and then move to its root node.*
*Your final traversal order should be.*

- # TRAVERSAL TECHNIQUES FOR TREES




*Here, each block represents a different subtree.*

***Algorithm**: Until all nodes of the tree are not visited*

*Step 1 - Traverse the left subtree recursively.*
*Step 2 - Traverse the right subtree recursively.*
*Step 3 - Visit the root node.*

*The applications of postorder traversal include –*

- *It is used to delete the tree.*
- *It can also be used to get the postfix expression of an expression tree.*

---