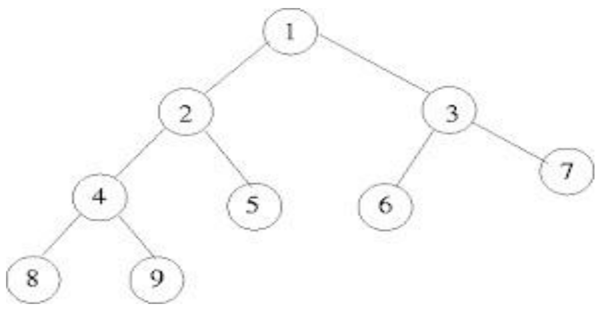
Binary Tree

Binary trees are types of data structures that have many uses. They can be applied in search, 3D video games, high-bandwidth network routers, some peer-to-peer programs and cryptography. In this lesson, we'll explore the basics and application of binary trees.

What Is a Binary Tree?

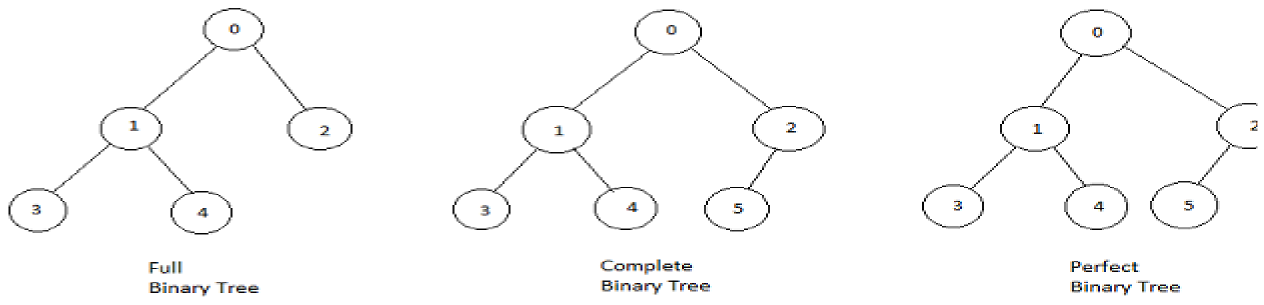
A normal tree has no restrictions on the number of children each node can have. Binary trees, on the other hand, can have at most two children for each parent. Every node contains a 'left' reference, a 'right' reference, and a data element. The topmost node in the tree is called the root node. Nodes with children are parent nodes, and the child nodes contain references to their parents. A node with no children is called a leaf node. Thus, each node in a binary tree can have 0, 1 or 2 children as shown in Figure.



Types & Implementation

There are three different types of binary trees that will be discussed in this lesson:

- **Full binary tree**: Every node other than leaf nodes has 2 child nodes.
- **Complete binary tree**: All levels are filled except possibly the last one, and all nodes are filled in as far left as possible.
- **Perfect binary tree**: All nodes have two children and all leaves are at the same level.



Types of Binary Trees

Binary trees can be implemented using pointers. A tree is represented by a pointer to the top-most node in the tree. If the tree is empty, then the value of the root is NULL.

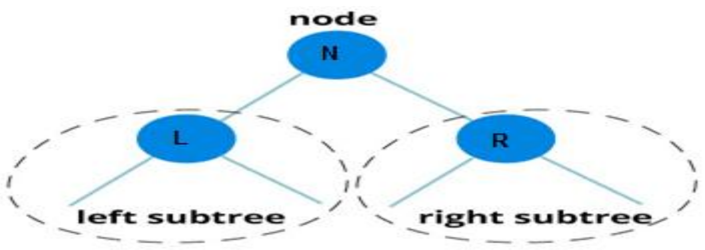
A binary tree has the following parts:

- info
- Pointer to the left child (*lchild)
- Pointer to the right child (*rchild)

Traversing Binary Trees

Any node in a binary tree data structure can be reached by starting at the root node and repeatedly following references to either the left or the right child. The degree, or number of children a node has, is a maximum of two.

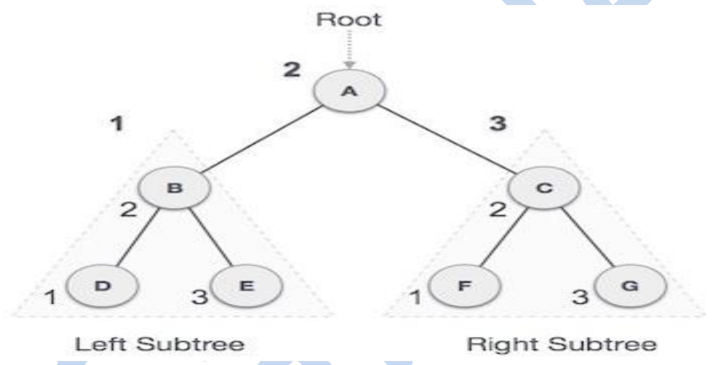
The process for visiting all the nodes is called a **traversal**, and consists of three types



In-order Traversal (LNR)

In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.

If a binary tree is traversed **in-order**, the output will produce sorted key values in an ascending order.



We start from **A**, and following in-order traversal, we move to its left subtree **B**. **B** is also traversed in-order. The process goes on until all the nodes are visited. The output of inorder traversal of this tree will be −

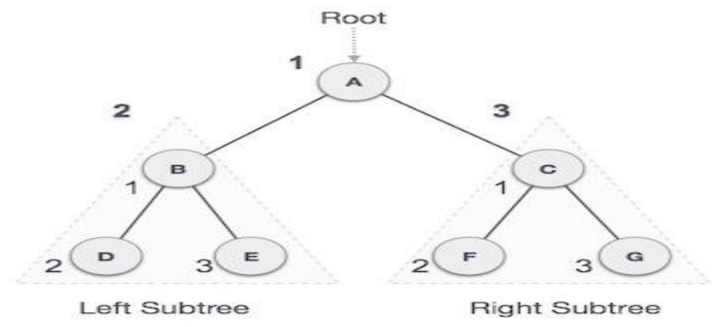
$$D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G$$

Algorithm

Until all nodes are traversed −
Step 1 − Recursively traverse left subtree.
Step 2 − Visit root node.
Step 3 − Recursively traverse right subtree.

Pre-order Traversal (NLR)

In this traversal method, the root node is visited first, then the left subtree and finally the right subtree.



We start from **A**, and following pre-order traversal, we first visit **A** itself and then move to its left subtree **B**. **B** is also traversed pre-order. The process goes on until all the nodes are visited. The output of pre-order traversal of this tree will be −

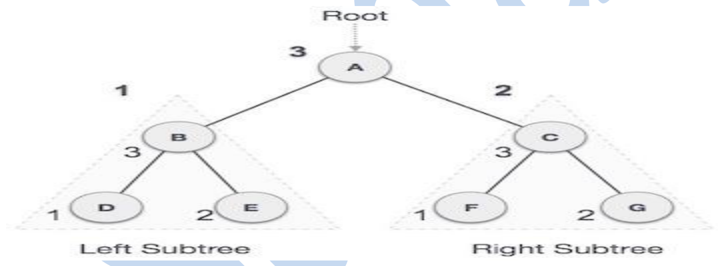
$$A \to B \to D \to E \to C \to F \to G$$

Algorithm

Until all nodes are traversed −
Step 1 − Visit root node.
Step 2 − Recursively traverse left subtree.
Step 3 − Recursively traverse right subtree.

Post-order Traversal(LRN)

In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.



We start from **A**, and following Post-order traversal, we first visit the left subtree **B**. **B** is also traversed post-order. The process goes on until all the nodes are visited. The output of post-order traversal of this tree will be −

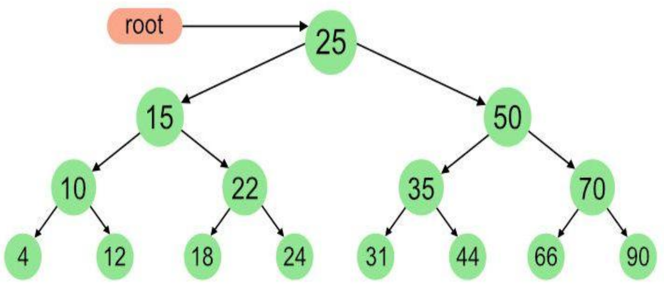
$$D \to E \to B \to F \to G \to C \to A$$

Algorithm

Until all nodes are traversed −
Step 1 − Recursively traverse left subtree.
Step 2 − Recursively traverse right subtree.
Step 3 − Visit root node.

Assignments

Q:- Find the sequence of nodes through all three different traversal mechanisms.



Q:- Construct the tree

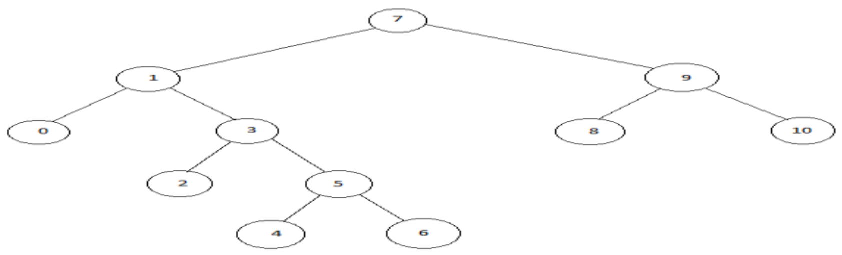
Inorder sequence: D B E A F C
Preorder sequence: A B D E C F

Q:- Construct the tree

inorder = {4, 8, 2, 5, 1, 6, 3, 7}

postorder = {8, 4, 5, 2, 6, 7, 3, 1}

Q:- Find the sequence of nodes through all three different traversal mechanisms.



The most disastrous thing that you can ever learn is your first programming language. Then you can save many disastrous through that language.

With Regards

Yours Sam Sir