Tree traversal (Inorder, Preorder an Postorder)

The term 'tree traversal' means traversing or visiting each node of a tree. There is a single way to traverse the linear data structure such as linked list, queue, and stack. Whereas, there are multiple ways to traverse a tree that are listed as follows -

- Preorder traversal
- Inorder traversal
- Postorder traversal

1. Preorder traversal

This technique follows the 'root left right' policy. It means that, first root node is visited after that the left subtree is traversed recursively, and finally, right subtree is recursively traversed. As the root node is traversed before (or pre) the left and right subtree, it is called preorder traversal.

So, in a preorder traversal, each node is visited before both of its subtrees.

The applications of preorder traversal include -

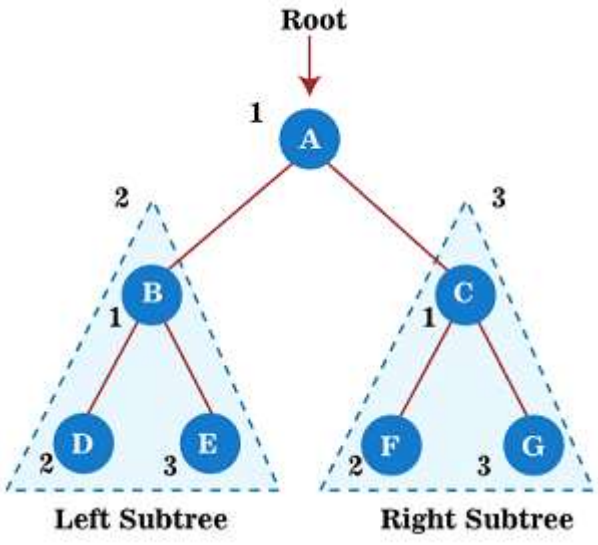
- It is used to create a copy of the tree.
- It can also be used to get the prefix expression of an expression tree.

Algorithm

1. Until all nodes of the tree are not visited
2. Step 1 - Visit the root node
3. Step 2 - Traverse the left subtree recursively.
4. Step 3 - Traverse the right subtree recursively.

Example

Now, let's see the example of the preorder traversal technique.



Now, start applying the preorder traversal on the above tree. First, we traverse the root node **A;** after that, move to its left subtree **B**, which will also be traversed in preorder.

So, for left subtree B, first, the root node **B** is traversed itself; after that, its left subtree **D** is traversed. Since node **D** does not have any children, move to right subtree **E**. As node E also does not have any children, the traversal of the left subtree of root node A is completed.

Now, move towards the right subtree of root node A that is C. So, for right subtree C, first the root node **C** has traversed itself; after that, its left subtree **F** is traversed. Since node **F** does not have any children, move to the right subtree **G**. As node G also does not have any children, traversal of the right subtree of root node A is completed.

Therefore, all the nodes of the tree are traversed. So, the output of the preorder traversal of the above tree is -

A → B → D → E → C → F → G

2. Postorder traversal

This technique follows the 'left-right root' policy. It means that the first left subtree of the root node is traversed, after that recursively traverses the right subtree, and finally, the root node is traversed. As the root node is traversed after (or post) the left and right subtree, it is called postorder traversal.

So, in a postorder traversal, each node is visited after both of its subtrees.

The applications of postorder traversal include -

- It is used to delete the tree.

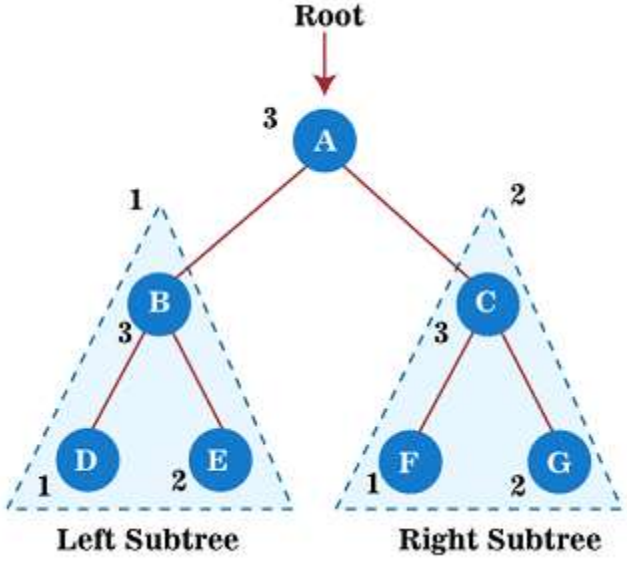
- It can also be used to get the postfix expression of an expression tree.

Algorithm

1. Until all nodes of the tree are not visited
2. Step 1 - Traverse the left subtree recursively.
3. Step 2 - Traverse the right subtree recursively.
4. Step 3 - Visit the root node.

Example

Now, let's see the example of the postorder traversal technique.



Now, start applying the postorder traversal on the above tree. First, we traverse the left subtree B that will be traversed in postorder. After that, we will traverse the right subtree **C** in postorder. And finally, the root node of the above tree, i.e., **A**, is traversed.

So, for left subtree B, first, its left subtree **D** is traversed. Since node **D** does not have any children, traverse the right subtree **E**. As node E also does not have any children, move to the root node **B.** After traversing node **B,** the traversal of the left subtree of root node A is completed.

Now, move towards the right subtree of root node A that is C. So, for right subtree C, first its left subtree **F** is traversed. Since node **F** does not have any children, traverse the right subtree **G**. As node G also does not have any children, therefore, finally, the root node of the right subtree, i.e., **C,** is traversed. The traversal of the right subtree of root node A is completed.

At last, traverse the root node of a given tree, i.e., **A**. After traversing the root node, the postorder traversal of the given tree is completed.

Therefore, all the nodes of the tree are traversed. So, the output of the postorder traversal of the above tree is -

D → E → B → F → G → C → A

3. Inorder traversal

This technique follows the 'left root right' policy. It means that first left subtree is visited after that root node is traversed, and finally, the right subtree is traversed. As the root node is traversed between the left and right subtree, it is named inorder traversal.

So, in the inorder traversal, each node is visited in between of its subtrees.

The applications of Inorder traversal includes -

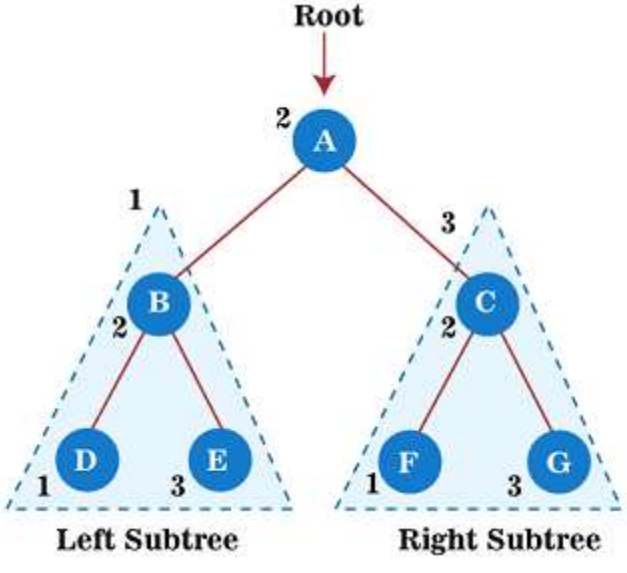
- It is used to get the BST nodes in increasing order.
- It can also be used to get the prefix expression of an expression tree.

Algorithm

1. Until all nodes of the tree are not visited
2. Step 1 - Traverse the left subtree recursively.
3. Step 2 - Visit the root node.
4. Step 3 - Traverse the right subtree recursively.

Example

Now, let's see the example of the Inorder traversal technique.



Now, start applying the inorder traversal on the above tree. First, we traverse the left subtree **B** that will be traversed in inorder. After that, we will traverse the root node **A**. And finally, the right subtree **C** is traversed in inorder.

So, for left subtree **B**, first, its left subtree **D** is traversed. Since node **D** does not have any children, so after traversing it, node **B** will be traversed, and at last, right subtree of node B, that is **E**, is traversed. Node E also does not have any children; therefore, the traversal of the left subtree of root node A is completed.

After that, traverse the root node of a given tree, i.e., **A**.

At last, move towards the right subtree of root node A that is C. So, for right subtree C; first, its left subtree **F** is traversed. Since node **F** does not have any children, node **C** will be traversed, and at last, a right subtree of node C, that is, **G**, is traversed. Node G also does not have any children; therefore, the traversal of the right subtree of root node A is completed.

As all the nodes of the tree are traversed, the inorder traversal of the given tree is completed. The output of the inorder traversal of the above tree is -

D → B → E → A → F → C → G

Complexity of Tree traversal techniques

The time complexity of tree traversal techniques discussed above is **O(n)**, where **'n'** is the size of binary tree.

Whereas the space complexity of tree traversal techniques discussed above is **O(1)** if we do not consider the stack size for function calls. Otherwise, the space complexity of these techniques is **O(h)**, where **'h'** is the tree's height.