Tree Traversal

Traversal of the tree in data structures is a process of visiting each node and prints their value. There are three ways to traverse tree data structure.

- Pre-order Traversal
- In-Order Traversal
- Post-order Traversal

In-Order Traversal

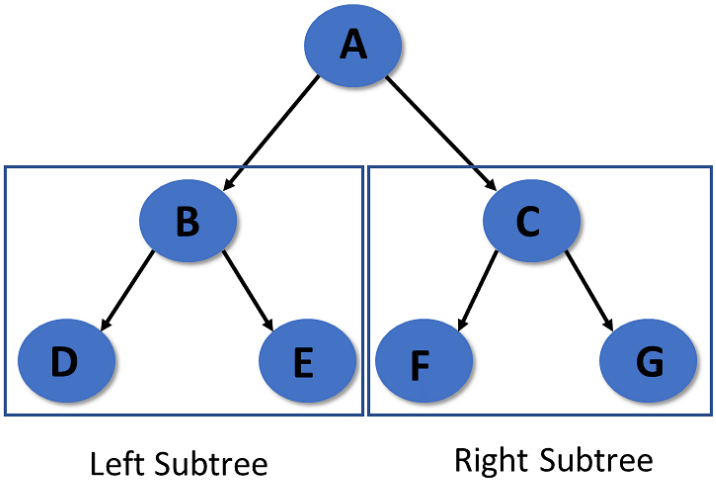
In the in-order traversal, the left subtree is visited first, then the root, and later the right subtree.

Algorithm:

Step 1- Recursively traverse the left subtree

Step 2- Visit root node

Step 3- Recursively traverse right subtree



D -> B -> E -> A -> F -> C -> G

Pre-Order Traversal

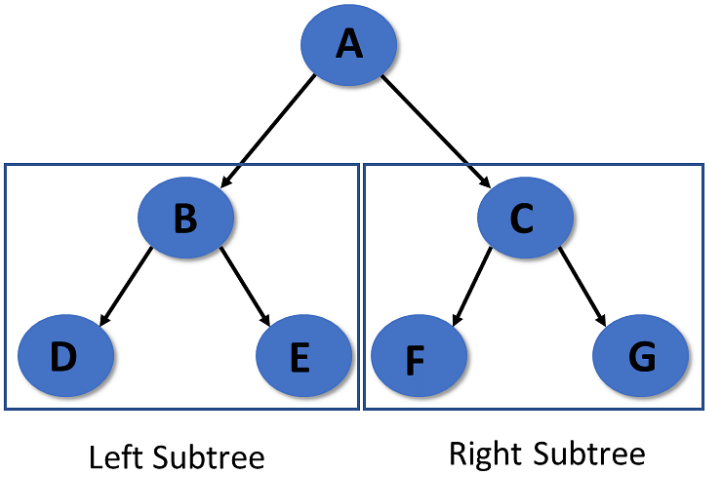
In pre-order traversal, it visits the root node first, then the left subtree, and lastly right subtree.

Algorithm:

Step 1- Visit root node

Step 2- Recursively traverse the left subtree

Step 3- Recursively traverse right subtree



A -> B -> D -> E -> C -> F -> G

Post-Order Traversal

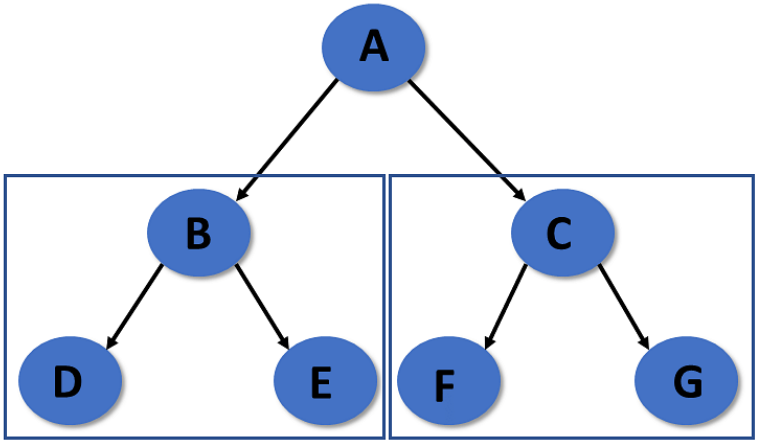
It visits the left subtree first in post-order traversal, then the right subtree, and finally the root node.

Algorithm:

Step 1- Recursively traverse the left subtree

Step 2- Visit root node

Step 3- Recursively traverse right subtree



D -> E -> B -> F -> G -> C -> A

```
#include <stdio.h>

#include <stdlib.h>

struct node {

  int data;

  struct node* left;
```

```
  struct node* right;
};
void inorder(struct node* root) {
  if (root == NULL) return;
  inorder(root->left);
  printf("%d ->", root->data);
  inorder(root->right);
}
void preorder(struct node* root) {
  if (root == NULL) return;
  printf("%d ->", root->data);
  preorder(root->left);
  preorder(root->right);
```

```
}

void postorder(struct node* root) {

  if (root == NULL) return;

  postorder(root->left);

  postorder(root->right);

  printf("%d ->", root->data);

}

struct node* createNode(item) {

  struct node* newNode = malloc(sizeof(struct node));

  newNode->data = item;

  newNode->left = NULL;

  newNode->right = NULL;

  return newNode;
```

```
}

struct node* insertLeft(struct node* root, int item) {

  root->left = createNode(item);

  return root->left;

}

struct node* insertRight(struct node* root, int item) {

  root->right = createNode(item);

  return root->right;

}

int main() {

  struct node* root = createNode(7);

  insertLeft(root, 12);

  insertRight(root, 29);
```

```
 insertLeft(root->left, 15);

 insertRight(root->left, 26);

 printf("Inorder  \n");

 inorder(root);

 printf("\nPreorder \n");

 preorder(root);

 printf("\nPostorder  \n");

 postorder(root);

}
```

Application of Tree in Data Structures

- Binary Search Tree (BST) is used to check whether elements present or not.
- Heap is a type of tree that is used to heap sort.

- Tries are the modified version of the tree used in modem routing to information of the router.
- The widespread database uses B-tree.
- Compilers use syntax trees to check every syntax of a program.