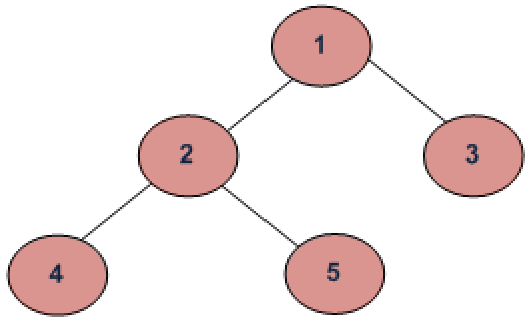
Binary Tree Traversals

Unlike linear data structures (Array, Linked List, Queues, Stacks, etc.), which have only one logical way to traverse them, trees can be traversed in different ways. Following are the generally used ways for traversing trees:



- Inorder (Left, Root, Right) : 4 2 5 1 3
- Preorder (Root, Left, Right) : 1 2 4 5 3.
- Postorder (Left, Right, Root) : 4 5 2 3 1

Let's look at each of these tree traversal algorithms in details:

- **Inorder Traversal:** In Inorder traversal, a node is processed after processing all the nodes in its left subtree. The right subtree of the node is processed after processing the node itself.

```
Algorithm Inorder(tree)
   1. Traverse the left subtree, i.e.,
      call Inorder(left->subtree)
   2. Visit the root.
   3. Traverse the right subtree, i.e.,
      call Inorder(right->subtree)
```

Example: Inorder traversal for the above-given tree is 4 2 5 1 3.

- **Preorder Traversal:** In preorder traversal, a node is processed before processing any of the nodes in its subtree.

```
Algorithm Preorder(tree)
   1. Visit the root.
   2. Traverse the left subtree, i.e.,
      call Preorder(left-subtree)
   3. Traverse the right subtree, i.e.,
      call Preorder(right-subtree)
```

Example: Preorder traversal for the above-given tree is 1 2 4 5 3.

- **Postorder Traversal:** In post order traversal, a node is processed after processing all the nodes in its subtrees.

```
Algorithm Postorder(tree)
   1. Traverse the left subtree, i.e.,
      call Postorder(left-subtree)
   2. Traverse the right subtree, i.e.,
      call Postorder(right-subtree)
   3. Visit the root.
```

Example: Postorder traversal for the above-given Tree is 4 5 2 3 1.

Dash
All
Articles
Videos
Problems
Quiz

C++ Java

```
// Java program for different tree traversals

/* Class containing left and right child of current
   node and key value*/
class Node
{
    int key;
    Node left, right;

    public Node(int item)
    {
        key = item;
        left = right = null;
    }
}

class BinaryTree
{
    // Root of Binary Tree
    Node root;

    BinaryTree()
    {
        root = null;
```

Dash

All

Articles

Videos

Problems

Quiz

```
    }

    // Method to print postorder traversal.
    void printPostorder(Node node)
    {
        if (node == null)
            return;

        // first recur on left subtree
        printPostorder(node.left);

        // then recur on right subtree
        printPostorder(node.right);

        // now deal with the node
        System.out.print(node.key + " ");
    }

    // Method to print inorder traversal
    void printInorder(Node node)
    {
        if (node == null)
            return;

        /* first recur on left child */
        printInorder(node.left);

        /* then print the data of node */
        System.out.print(node.key + " ");
```

Dash

All

Articles

Videos

Problems

Quiz

```
        /* now recur on right child */
        printInorder(node.right);
    }

    // Method to print preorder traversal
    void printPreorder(Node node)
    {
        if (node == null)
            return;

        /* first print data of node */
        System.out.print(node.key + " ");

        /* then recur on left sutree */
        printPreorder(node.left);

        /* now recur on right subtree */
        printPreorder(node.right);
    }

    // Wrappers over above recursive functions
    void printPostorder() {     printPostorder(root); }
    void printInorder() {     printInorder(root); }
    void printPreorder() {     printPreorder(root); }

    // Driver method
    public static void main(String[] args)
    {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);

        System.out.println("Preorder traversal of binary tree is ");
        tree.printPreorder();

        System.out.println("\nInorder traversal of binary tree is ");
        tree.printInorder();

        System.out.println("\nPostorder traversal of binary tree is ");
```

```
    }
}
```

Output:

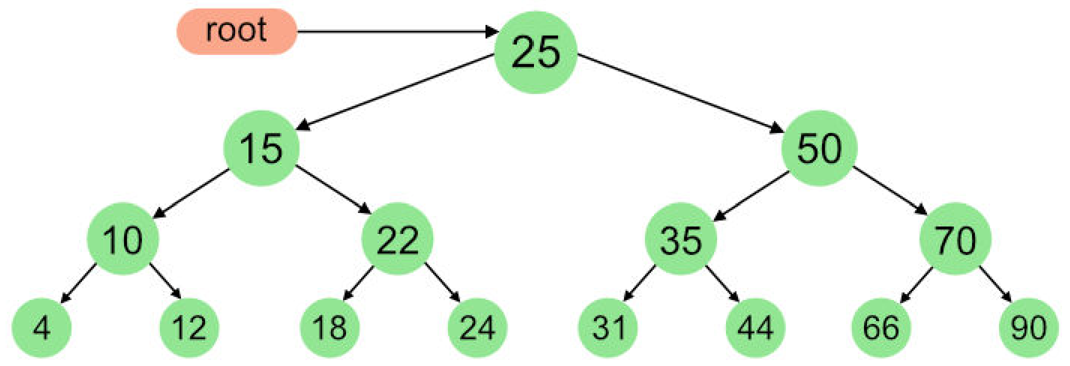
```
Preorder traversal of binary tree is
1 2 4 5 3
Inorder traversal of binary tree is
4 2 5 1 3
Postorder traversal of binary tree is
4 5 2 3 1
```

One more example:

InOrder(root) visits nodes in the following order:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

A Pre-order traversal visits nodes in the following order:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

A Post-order traversal visits nodes in the following order:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25



If you are facing any issue on this page. Please let us know.

Dash

All

Articles

Videos

Problems

Quiz