Traversals

Four types of traversals

① InOrder (L N R)

② PreOrder (N L R)

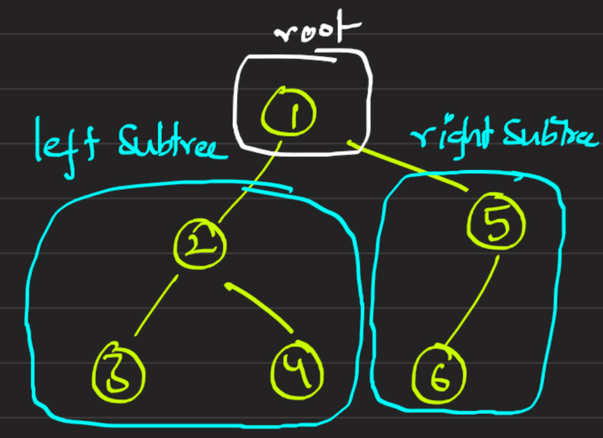
③ PostOrder (L R N)

④ Level Order : Nodes are visited level by level from left to right using a queue

InOrder :

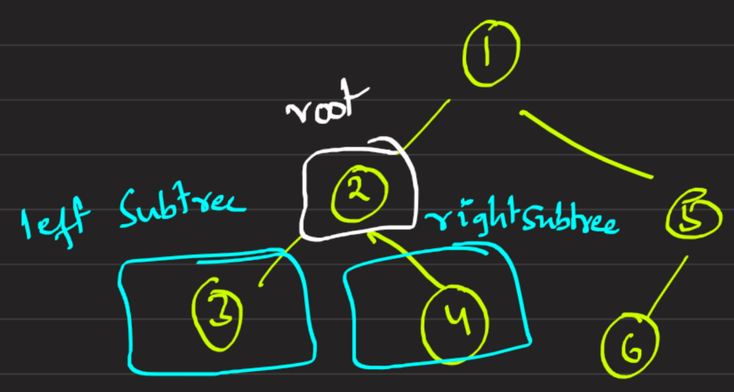
* Visit left Subtree
* Do whatever Operation you want with root node
* Visit right Subtree

This will change according to traversal

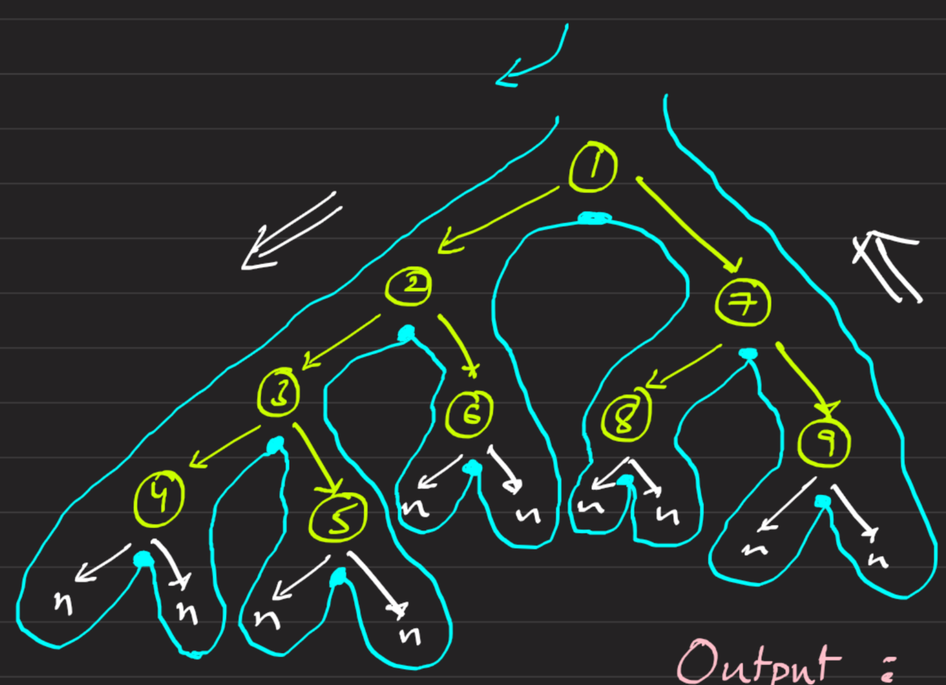


At Every node, we will have a,

① left Subtree
② Right Subtree

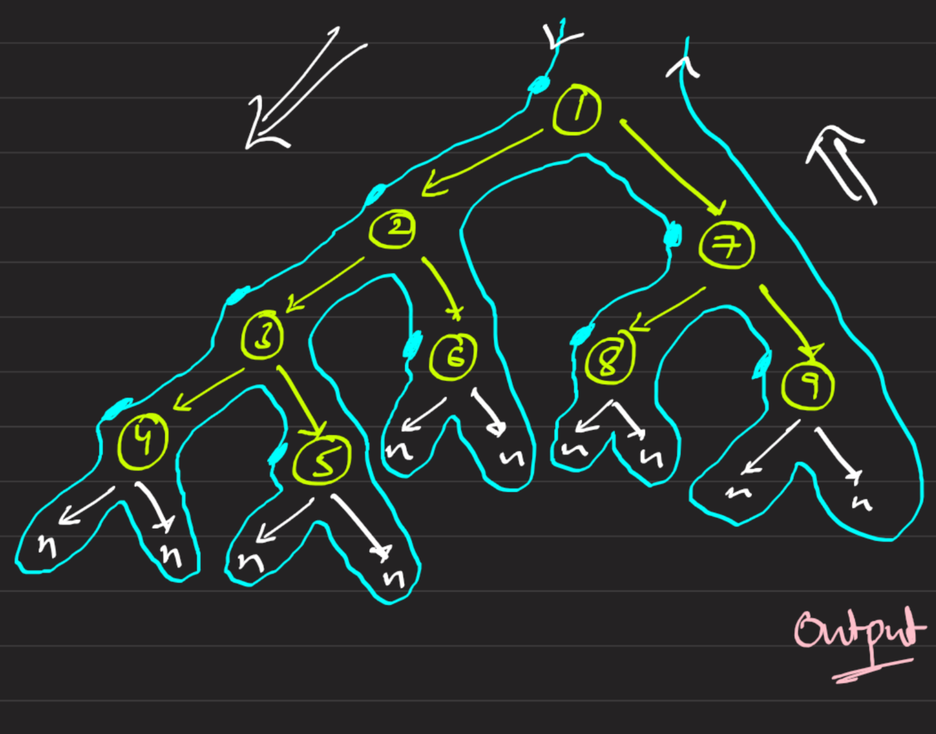


* whenever your left, right Subtrees becomes null, it states you are the leaf node



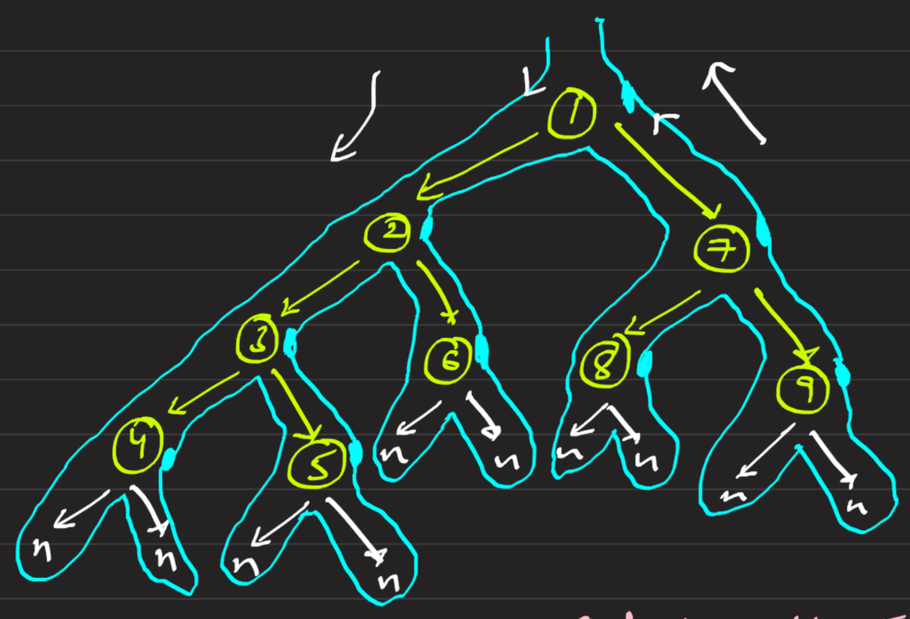
* whenever, you see the node for the second time, you print it ⟹ InOrder

Output : 4 3 5 2 6 1 8 7 9



* whenever, you see the node for the first time, you print it ⟹ PreOrder

Output : 1 2 3 4 5 6 7 8 9



* whenever, you see the node for the third time, you print it ⟹ (postOrder)

Output : 4 5 3 6 2 8 9 7 1

Pseudo Codes

```
void InOrder(Node root)
{
    if (root == null) return;
    helper(root);
}

void helper(Node root)
{
    if (root == null) return;
    helper(root.left)        leftSubtree
    operation ⟹ System.out.println(root.val + " ");
    helper(root.right);      rightSubtree
}
```

```
void preOrder(Node root)
{
    if (root == null) return;
    helper(root);
}

void helper(Node root)
{
    if (root == null) return
    operation ⟹ System.out.println(root.val + " ");
    helper(root.left)        leftSubto
    helper(root.right);      rightSubtree
}
```

```
void postOrder(Node root)
{
    if (root == null) return;
    helper(root);
}

void helper(Node root)
{
    if (root == null) return
    helper(root.left)        leftSubtree
    helper(root.right);      rightSubtree
    operation ⟹ System.out.println(root.val + " ");
}
```