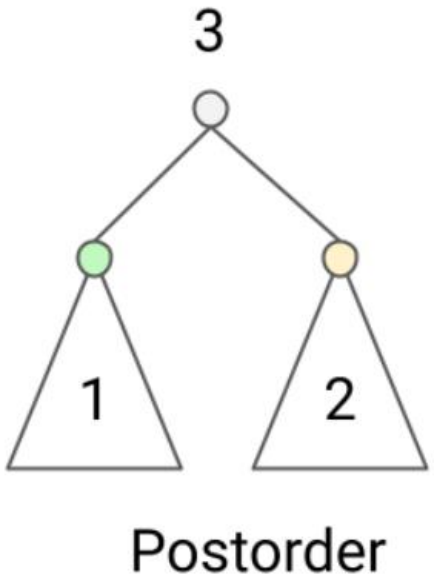
# Data Structures and Algorithms

**Lecture 27:** Trees: Post-order Traversal

# Post – Order Traversal

Post-order traversal is defined as a type of tree traversal which follows the Left-Right-Root policy such that for each node:

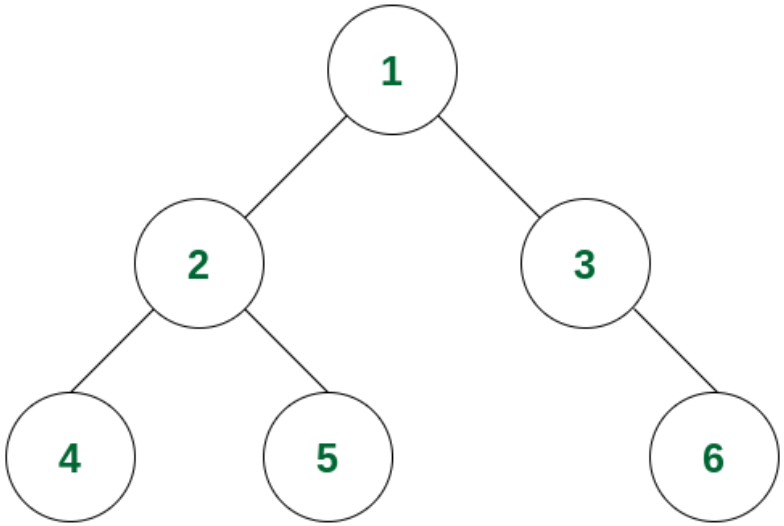
- The left subtree is traversed first
- Then the right subtree is traversed
- Finally, the root node of the subtree is traversed



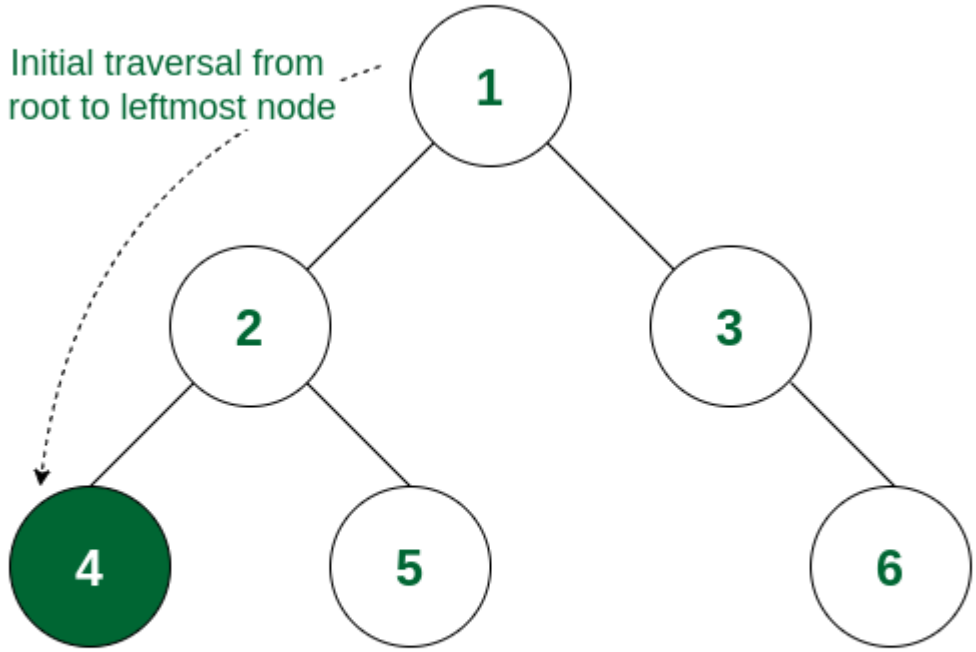
**Algorithm for In-order Traversal of Binary Tree**

Post-order(root):

1. Follow step 2 to 4 until root != NULL
2. Post-order (root -> left)
3. Post-order (root -> right)
4. Write root -> data
5. End loop

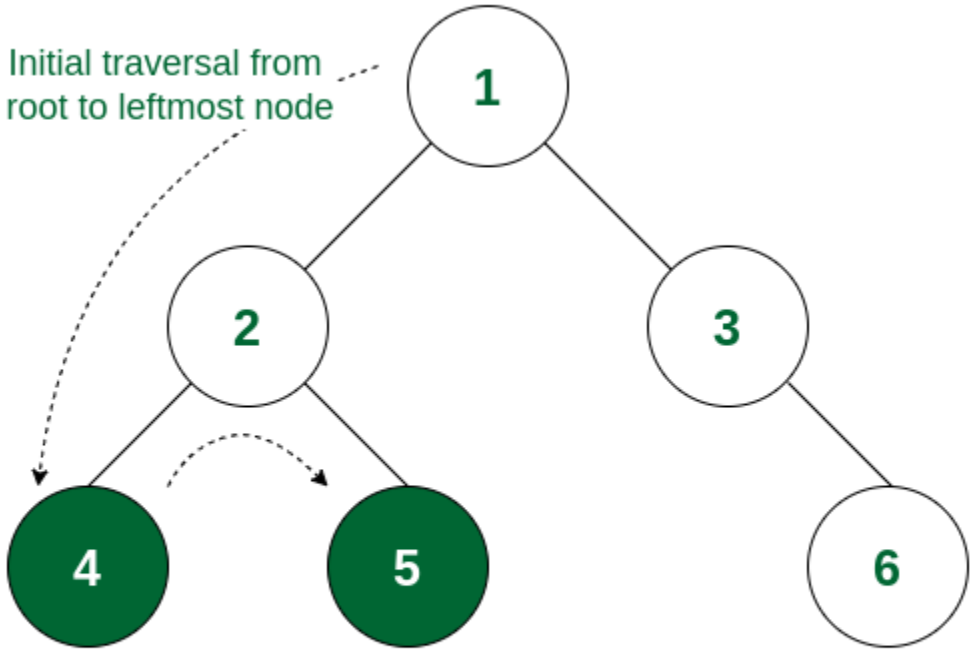


***Step 1:*** *The traversal will go from 1 to its left subtree i.e., 2, then from 2 to its left subtree root, i.e., 4. Now 4 has no subtree, so it will be visited.*



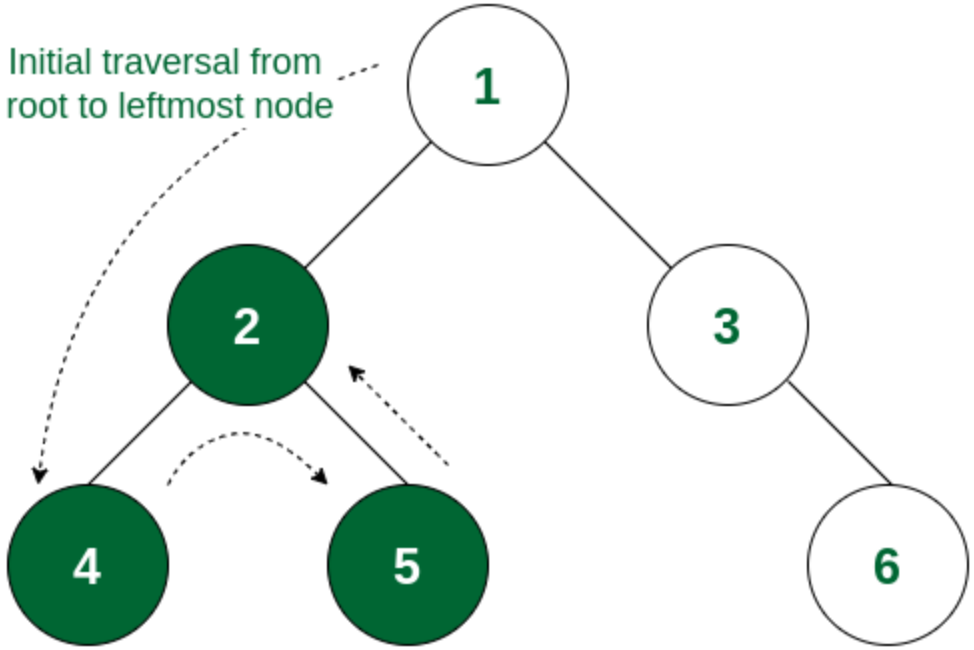

**The leftmost leaf node (i.e., 4) is visited first**

***Step 2:*** *As the left subtree of 2 is visited completely, now it will traverse the right subtree of 2 i.e., it will move to 5. As there is no subtree of 5, it will be visited.*



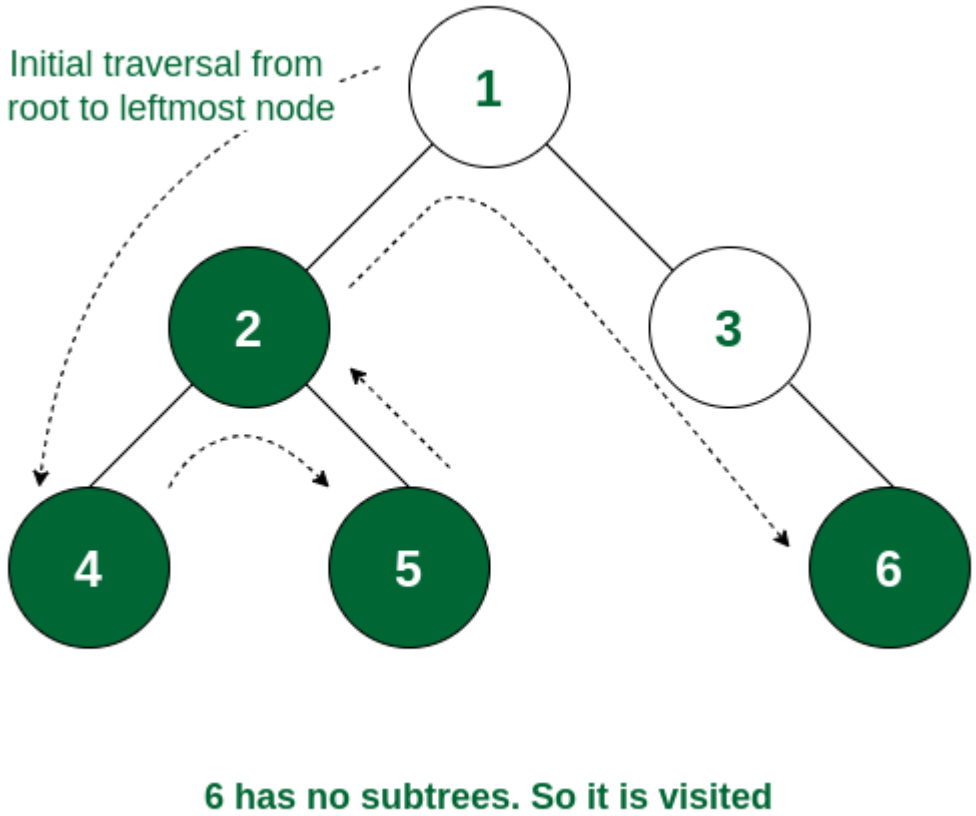

**Left subtree of 2 is traversed. So 5 is visited next**

***Step 3:*** *Now both the left and right subtrees of node 2 are visited. So now visit node 2 itself.*

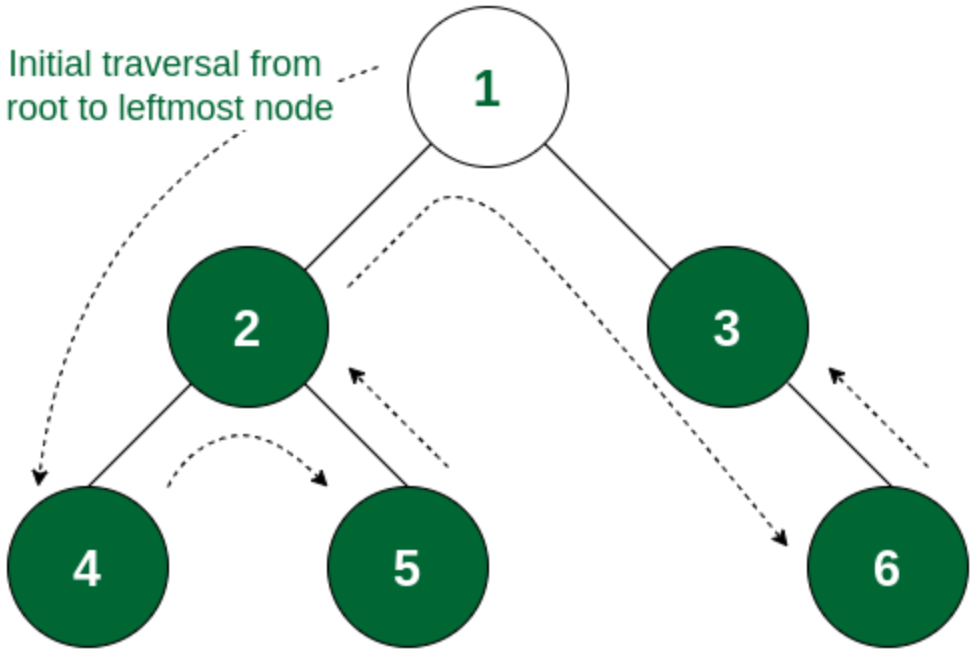



**All subtrees of 2 are visited. So 2 is visited next**

***Step 4:*** *As the left subtree of node 1 is traversed, it will now move to the right subtree root, i.e., 3. Node 3 does not have any left subtree, so it will traverse the right subtree i.e., 6. Node 6 has no subtree and so it is visited.*

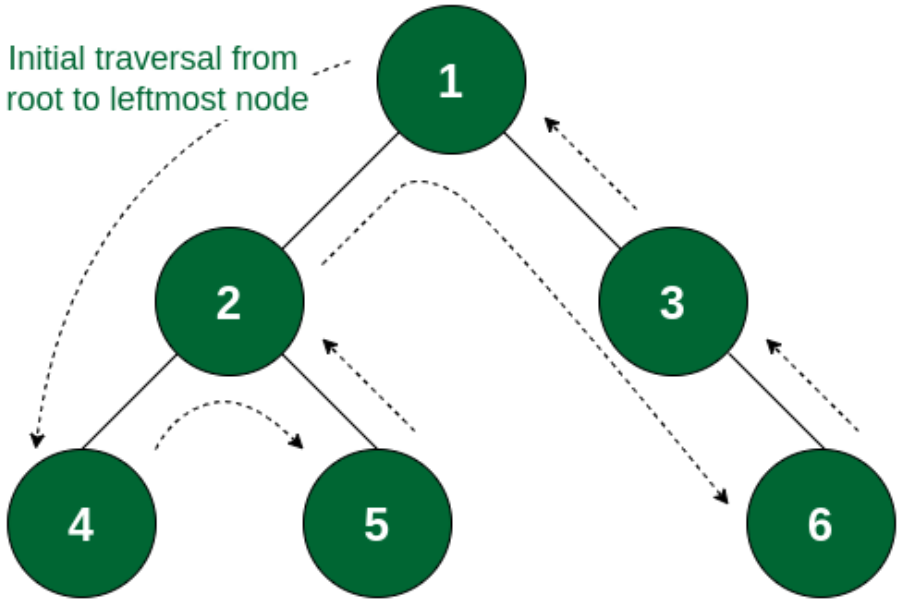


***Step 5:*** *All the subtrees of node 3 are traversed. So now node 3 is visited.*




**3 is visited after all its subtrees are traversed**

***Step 6:*** *As all the subtrees of node 1 are traversed, now it is time for node 1 to be visited and the traversal ends after that as the whole tree is traversed.*




*So the order of traversal of nodes is **4 -> 5 -> 2 -> 6 -> 3 -> 1**.*

```
function post_order(root, nodes) {
    if (root && root.left) {
        post_order(root.left, nodes);
    }
    if (root && root.right) {
        post_order(root.right, nodes);
    }
    nodes.push(root.data);
    return nodes;
}
```

## Use cases of Post-order Traversal:

- This is used for tree deletion.
- It is also useful to get the postfix expression from an expression tree.