جامـــــعة المـــــستـقبـل
AL MUSTAQBAL UNIVERSITY

قســـم الامـــــن الـــــــــسيبرانـــــــي

DEPARTMENT OF CYBER SECURITY

SUBJECT:

SEARCHING AND SORTING ALGORITHMS

CLASS:

SECOND

LECTURER:

ASST. PROF. DR. ALI KADHUM AL-QURABY

LECTURE: (5-2)

POST-ORDER TRAVERSAL

Program to Implement Postorder Traversal of Binary Tree

Below is the code implementation of the postorder traversal:

```cpp
// C++ program for postorder traversals

#include <bits/stdc++.h>
using namespace std;

// Structure of a Binary Tree Node
struct Node {
    int data;
    struct Node *left, *right;
    Node(int v)
    {
        data = v;
        left = right = nullptr;
    }
};

// Function to print postorder traversal
void printPostorder(struct Node* node)
{
    if (node == nullptr)
        return;

    // First recur on left subtree
    printPostorder(node->left);

    // Then recur on right subtree
    printPostorder(node->right);

    // Now deal with the node
    cout << node->data << " ";
}

// Driver code
int main()
{
    struct Node* root = new Node(1);
    root->left = new Node(2);
    root->right = new Node(3);
    root->left->left = new Node(4);
    root->left->right = new Node(5);
```

```
    root->right->right = new Node(6);

    // Function call
    cout << "Postorder traversal of binary tree is: \n";
    printPostorder(root);

    return 0;
}
```

Output

```
Postorder traversal of binary tree is:
4 5 2 6 3 1
```

- ## Complexity Analysis:

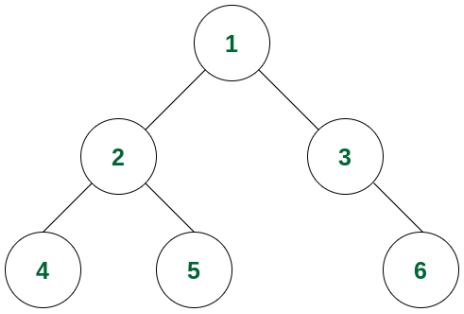
❖ **Time Complexity:** O(N) where N is the total number of nodes. Because it traverses all the nodes at least once.

❖ **Auxiliary Space:** O(1) if no recursion stack space is considered. Otherwise, O(h) where h is the height of the tree

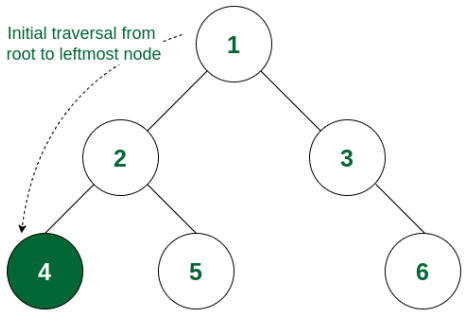
- In the worst case, **h** can be the same as **N** (when the tree is a skewed tree)
- In the best case, **h** can be the same as **logN** (when the tree is a complete tree)

How does Postorder Traversal of Binary Tree Work?

Consider the following tree:

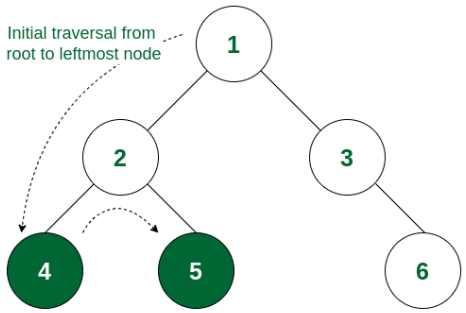


- ✓ ***Step 1:*** *The traversal will go from 1 to its left subtree i.e., 2, then from 2 to its left subtree root, i.e., 4. Now 4 has no subtree, so it will be visited.*



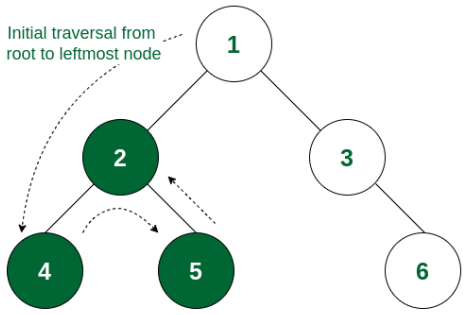
The leftmost leaf node (i.e., 4) is visited first

- ✓ ***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.*



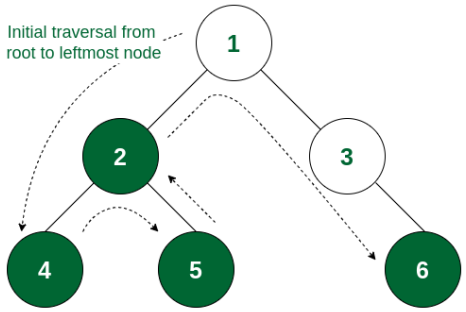
Left subtree of 2 is traversed. So 5 is visited next

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



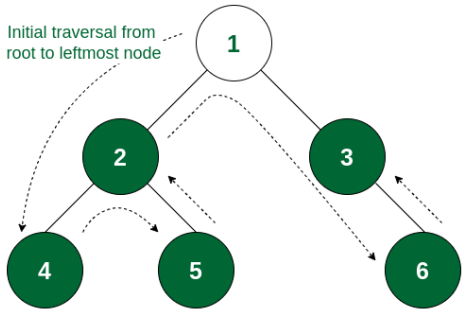
All subtrees of 2 are visited. So 2 is visited next

- ✓ ***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.*



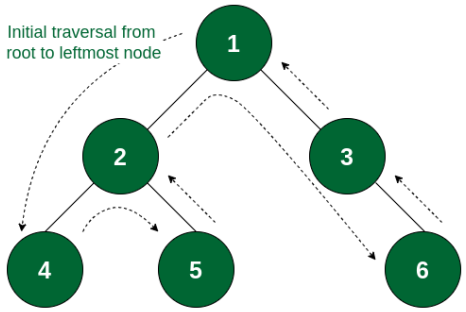
6 has no subtrees. So it is visited

- ✓ ***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.*



The root of the tree (i.e., 1) is visited

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