Preorder, Inorder, and Postorder Traversals

Mr. Vikas Kumar
Assistant Professor
Industry Embedded Program

Lesson Plan

Subject/Course	**Competitive Coding**
Lesson Title	**Preorder, Inorder, and Postorder Traversals**

Lesson Objectives
A tree is a hierarchical data structure consisting of nodes.
The topmost node is called the root.
Each node may have child nodes, Nodes without children are called leaf nodes.
Traversal means visiting each node of the tree exactly once.

Problem Statement:

Write a Program to Understand and implement Tree traversals i.e. **Pre-Order Post-Order**, **In-Order**.

Concept

- Tree traversal means visiting every node of a tree in a specific order (once only).
- There are mainly three depth-first traversals:

Traversal Type	**Order of Visit**	**Example (for tree below)**
Preorder	Root → Left → Right	A B D E C F
Inorder	Left → Root → Right	D B E A F C
Postorder	Left → Right → Root	D E B F C A

Algorithm/Logic

Concept:

Traversal = visiting every node **once** in a specific order using **recursion**.

◆ **Preorder (Root → Left → Right)**

Steps:

1. Visit Root
2. Traverse Left
3. Traverse Right

Output: A B D E C F

Algorithm/Logic

◆ **Inorder (Left → Root → Right)**

Steps:

1. Traverse Left
2. Visit Root
3. Traverse Right

Output: D B E A F C

Algorithm/Logic

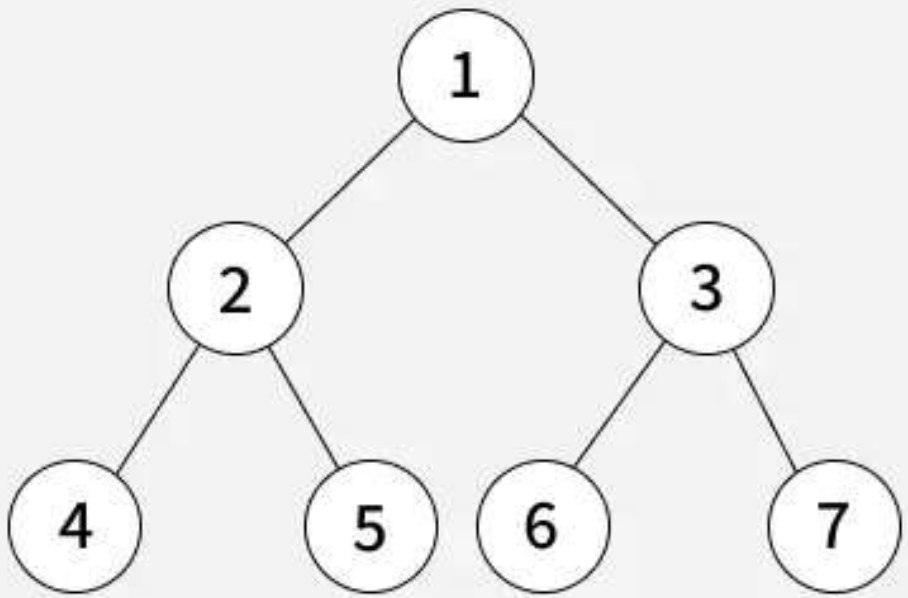
◆ **Postorder (Left → Right → Root)**

Steps:

1. Traverse Left
2. Traverse Right
3. Visit Root

Output: D E B F C A

Visualization



Inorder Traversal

4	2	5	1	6	3	7

Preorder Traversal

1	2	4	5	3	6	7

Postorder Traversal

4	5	2	6	7	3	1

Level order Traversal

1	2	3	4	5	6	7

Code Implementation

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node *left, *right;
    Node(int val) : data(val), left(NULL), right(NULL) {}
};

void preorder(Node* root) {
    if (!root) return;
    cout << root->data << " ";
    preorder(root->left);
    preorder(root->right);
}

void inorder(Node* root) {
    if (!root) return;
    inorder(root->left);
    cout << root->data << " ";
    inorder(root->right);
}

void postorder(Node* root) {
    if (!root) return;
    postorder(root->left);
    postorder(root->right);
    cout << root->data << " ";
}
```

```
int main() {
    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);

    cout << "Preorder: "; preorder(root); cout << endl;
    cout << "Inorder: "; inorder(root); cout << endl;
    cout << "Postorder: "; postorder(root); cout << endl;
}
```

Output:

```
Preorder: 1 2 4 5 3
Inorder: 4 2 5 1 3
Postorder: 4 5 2 3 1
```

Time & Space Complexity

Traversal Type	Time Complexity	Space Complexity (Recursion Stack)	Explanation
Preorder	O(n)	O(h)	Each node is visited once; recursion depth = tree height (h).
Inorder	O(n)	O(h)	Visits every node exactly once; stack used for recursion.
Postorder	O(n)	O(h)	Each node processed after its children; same depth as others.

Summary

1.Tree traversal means visiting every node exactly once in a specific order.

2.There are three main Depth-First Traversals:

- **Preorder** (Root → Left → Right): Used to create or copy a tree.
- **Inorder** (Left → Root → Right): Retrieves nodes in sorted order for BSTs.
- **Postorder** (Left → Right → Root): Used to delete or free the tree.

3.All traversals have Time Complexity = O(n) and Space Complexity = O(h),

4.where h is the height of the tree.

5.Traversals are the foundation of tree operations like searching, expression evaluation, and serialization.

Practice Questions:

1. **Binary Tree Inorder Traversal — LeetCode #94**

🔗 *https://leetcode.com/problems/binary-tree-inorder-traversal/*

Concept:
Implement **Inorder Traversal (Left → Root → Right)** of a binary tree using **recursion or an iterative stack-based approach**.

Why Practice:

- Reinforces understanding of traversal order.
- Tests recursion and stack manipulation.
- Foundation for solving BST and expression tree problems.

Practice Questions:

2. Binary Tree Preorder Traversal — LeetCode #144

🔗 *https://leetcode.com/problems/binary-tree-preorder-traversal/*

Concept:

Perform **Preorder Traversal (Root → Left → Right)** of a binary tree.

You can solve it **recursively** or **iteratively using a stack**.

Why Practice:

- Builds clear understanding of visiting order.
- Improves recursive thinking and stack-based problem solving.
- Prepares for advanced problems like serialization/deserialization of trees.

Thanks