**Documentation: DFS and Tree Traversals**

**Introduction**

Tree traversal is a fundamental operation in data structures. It refers to the process of visiting all the nodes of a binary tree systematically. There are different ways to traverse a tree, each serving unique purposes in algorithms and applications.

This document explains **Depth-First Search (DFS) using Stack** and the three standard tree traversals: **Preorder, Inorder, and Postorder**.

**Depth-First Search (DFS) using Stack**

* DFS is a traversal technique that explores as far as possible along one branch before backtracking.
* The **stack-based implementation** is an iterative approach where nodes are pushed onto and popped from a stack to control the traversal order.
* The stack ensures that the left child is visited before the right child, giving a traversal order similar to Preorder (Root → Left → Right).
* This avoids recursion and makes the algorithm more memory-efficient in cases where recursion depth could be a limitation.

**Tree Traversals**

**1. Preorder Traversal (Root → Left → Right)**

* Visit the root node first.
* Traverse the left subtree recursively.
* Finally, traverse the right subtree.
* This traversal is often used to **copy trees** or **evaluate prefix expressions**.

**2. Inorder Traversal (Left → Root → Right)**

* Traverse the left subtree first.
* Visit the root node.
* Then traverse the right subtree.
* In binary search trees (BST), **inorder traversal gives sorted output**, making it highly useful.

**3. Postorder Traversal (Left → Right → Root)**

* Traverse the left subtree first.
* Then traverse the right subtree.
* Visit the root node last.
* This method is often used for **deleting trees** or **evaluating postfix expressions**.