Tree Traversal in DSA

Tree traversal refers to the process of **visiting (accessing/printing)** each node of a tree data structure **exactly once** in a systematic manner.
Unlike arrays or linked lists (which are linear), trees are **non-linear** structures, so special traversal techniques are required.

Types of Tree Traversal

Tree traversals are mainly divided into **two categories**:

1. Depth First Traversal (DFS)

In DFS, we go as deep as possible into the tree before backtracking.
It is further divided into **3 types** (based on the order of visiting root, left, and right nodes):

(a) Preorder Traversal (Root → Left → Right)

- Visit **root node** first.
- Traverse the **left subtree.**
- Traverse the **right subtree.**

Example →

Preorder = **1, 2, 4, 5, 3**

(b) Inorder Traversal (Left → Root → Right)

- Traverse the **left subtree.**
- Visit **root node.**
- Traverse the **right subtree.**

Example →

Inorder = **4, 2, 5, 1, 3**
Special: In a **Binary Search Tree (BST)**, inorder traversal gives **sorted order.**

(c) Postorder Traversal (Left → Right → Root)

- Traverse the **left subtree.**
- Traverse the **right subtree.**
- Visit **root node.**

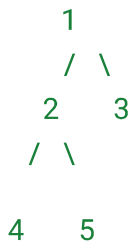
Example →

Postorder = **4, 5, 2, 3, 1**

2. Breadth First Traversal (BFS) / Level Order Traversal

- Visit nodes **level by level** (from top to bottom, left to right).
- Implemented using a **queue.**

Example →



Level Order = **1, 2, 3, 4, 5**