■ Report: When to Use Inorder, Preorder, and Postorder Tree Traversals

Tree traversal is the process of visiting all nodes in a binary tree systematically. The three depth-first traversals—Inorder, Preorder, and Postorder—are applied based on the task at hand.

1. Inorder Traversal (Left → Root → Right)

Use Cases:

- Retrieve data from a Binary Search Tree (BST) in sorted order.
- Verify if a tree is a valid BST (inorder should be strictly increasing).
- Convert a BST into a sorted list or array.

Example: For a BST with nodes 1–7, inorder traversal yields: 1 2 3 4 5 6 7 (sorted sequence).

2. Preorder Traversal (Root → Left → Right)

Use Cases:

- Copying or cloning trees (process root before children).
- Serialization/Deserialization of trees.
- Building expression trees (prefix notation).
- Situations where parent must be processed before children.

Example: Expression (a + b) * c → Preorder: * + a b c (prefix).

3. Postorder Traversal (Left → Right → Root)

Use Cases:

- Deleting/freeing a tree (children before parent).
- Evaluating expression trees (postfix evaluation).
- Cases where children must be processed before the parent.

Example: Expression (a + b) * c → Postorder: a b + c * (postfix).

■ Quick Reference

- Inorder → Sorted output (BST).
- Preorder → Tree cloning, serialization, prefix expressions.
- Postorder → Deletion, evaluation, child-first processing.