## **5. Binary Tree Traversals**

### **5.1 Depth-First Traversals (DFS)**

**Definition:** Traversal method where we explore as far as possible along each branch before backtracking.

**Types of DFS Traversals:**

#### **a) Inorder Traversal (Left → Root → Right)**

**Steps:**

Traverse the left subtree

Visit the root

Traverse the right subtree

**Example:**

10

/ \

5 15

/ \

2 7

Inorder: 2 5 7 10 15

#### **b) Preorder Traversal (Root → Left → Right)**

**Steps:**

Visit the root

Traverse the left subtree

Traverse the right subtree

**Example:**

Preorder: 10 5 2 7 15

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

**Steps:**

Traverse the left subtree

Traverse the right subtree

Visit the root

**Example:**

Postorder: 2 7 5 15 10

### **5.2 Breadth-First Traversal (Level-Order Traversal)**

**Definition:** Traversal method where nodes are visited **level by level** from top to bottom.

**Example:**

Level-Order: 10 5 15 2 7

**Implementation:** Usually implemented using a **queue**.

## **6. Binary Search Tree (BST)**

### **6.1 Definition and Properties**

**Definition:** A Binary Tree in which for each node:

Left subtree contains nodes with values **less than the node’s value**

Right subtree contains nodes with values **greater than the node’s value**

**Properties:**

No duplicate nodes (usually)

Inorder traversal gives **sorted order of elements**

**Applications:**

Efficient searching, insertion, and deletion

Symbol tables, databases, priority queues

### **6.2 Operations on BST**

#### **a) Insertion of a Node**

Insert at **appropriate position** maintaining BST property

Example:

Insert 6 in BST:

10

/ \

5 15

/ \

2 7

After insertion:

10

/ \

5 15

/ \

2 7

/

6

#### **b) Searching for a Node**

Start from the root and move **left/right** depending on the value

**Time Complexity:** O(h), h = height of tree

#### **c) Deletion of a Node**

Three cases:

**Leaf Node:** Simply remove it

**Node with One Child:** Remove the node and connect its child to the parent

**Node with Two Children:**

Replace node with its **inorder successor** (smallest node in right subtree)

Delete the inorder successor

**Example:**

Delete 5 from BST:

10

/ \

5 15

/ \

2 7

Inorder successor of 5 = 7

Replace 5 with 7:

10

/ \

7 15

/

2