Recursive Tree Traversals DSA

Give DSA notes on recursive preorder, postorder, inorder & level order tree traversal

DSA Notes: Recursive Tree Traversals (Preorder, Inorder, Postorder) & Level Order Traversal

1. Recursive Tree Traversals

Tree traversal is a process of visiting each node of a tree in a systematic manner. The most common types of depth-first traversals are:

1. **Preorder (Root → Left → Right)**
2. **Inorder (Left → Root → Right)**
3. **Postorder (Left → Right → Root)**

Preorder Traversal (DLR)

- Visit the root node.
- Recursively traverse the left subtree.
- Recursively traverse the right subtree.

Code (Java):

```java
void preorder(TreeNode root) {
    if (root == null) return;
    System.out.print(root.val + " ");  // Process node
    preorder(root.left);  // Left subtree
    preorder(root.right); // Right subtree
}
```

✅ **Use Cases:** Copying trees, prefix expressions.

Inorder Traversal (LDR)

- Recursively traverse the left subtree.
- Visit the root node.
- Recursively traverse the right subtree.

Code (Java):

```java
void inorder(TreeNode root) {
    if (root == null) return;
    inorder(root.left);  // Left subtree
    System.out.print(root.val + " ");  // Process node
    inorder(root.right); // Right subtree
}
```

✅ **Use Cases:** Sorting a BST (gives elements in ascending order).

Postorder Traversal (LRD)

- Recursively traverse the left subtree.
- Recursively traverse the right subtree.
- Visit the root node.

Code (Java):

```java
void postorder(TreeNode root) {
    if (root == null) return;
    postorder(root.left);  // Left subtree
    postorder(root.right); // Right subtree
    System.out.print(root.val + " ");  // Process node
}
```

✅ **Use Cases:** Deleting a tree, postfix expressions.

---

2. Level Order Traversal (BFS)

Unlike the depth-first recursive methods above, level-order traversal (Breadth-First Search) visits nodes level by level.

- Uses a **queue** (FIFO) to process nodes level by level.
- First, enqueue the root, then process its children.

Code (Java):

```java
void levelOrder(TreeNode root) {
    if (root == null) return;
    Queue<TreeNode> queue = new LinkedList<>();
    queue.add(root);

    while (!queue.isEmpty()) {
        TreeNode node = queue.poll();  // Dequeue front node
        System.out.print(node.val + " ");

        if (node.left != null) queue.add(node.left);
        if (node.right != null) queue.add(node.right);
    }
}
```

✅ **Use Cases:** Shortest path problems, constructing trees from arrays.

Comparison Table

Traversal	Order of Visiting	Implementation	Uses
Preorder	Root → Left → Right	Recursive	Copying trees, prefix expressions
Inorder	Left → Root → Right	Recursive	BST sorting, infix expressions
Postorder	Left → Right → Root	Recursive	Deleting trees, postfix expressions
Level Order	Level by level	Iterative (Queue)	BFS applications, shortest paths

Would you like additional variations, such as iterative implementations? 🚀