Trees - traversals

DFS iterative

DFS Iterative Traversal

Inorder

```
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> list = new ArrayList<>();
        Stack<TreeNode> stack = new Stack<>();
        while(stack.size() > 0 || root != null) {
            while(root != null) {
                stack.add(root);
                root = root.left;
            }
            root = stack.pop();
            list.add(root.val);
            root = root.right;
        }

        return list;
    }
}
```

Preorder traversal

Preorder

```
class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> list = new ArrayList();
        if(root == null)
            return list;
        Stack<TreeNode> stack = new Stack();
        stack.add(root);
        while(!stack.isEmpty()) {
            root = stack.pop();
            list.add(root.val);
            if(root.right != null)
                stack.add(root.right);
            if(root.left != null)
                stack.add(root.left);
        }

        return list;
    }
}
```

Postorder

```
class Solution {
    public List<Integer> postorderTraversal(TreeNode root) {
        List<Integer> list = new ArrayList();
        Stack<TreeNode> stack = new Stack();
        while(!stack.isEmpty() || root != null) {
            if(root != null) {
                stack.add(root);
                root = root.left;
            } else {
                TreeNode temp = stack.peek().right;
                if(temp == null) {
                    temp = stack.pop();
                    list.add(temp.val);
                    while(!stack.isEmpty() && temp ==  stack.peek().right) {
                        temp = stack.pop();
                        list.add(temp.val);
                    }
                } else {
                    root = temp;
                }
            }
        }

        return list;
    }
}
```

DFS recursive

DFS Recrsive Traversal

Inorder

```java
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> list = new ArrayList();
        dfs(root, list);
        return list;
    }

    private void dfs(TreeNode root, List<Integer> list) {
        if(root == null)
            return;
        dfs(root.left, list);
        list.add(root.val);
        dfs(root.right, list);
    }
}
```

Preorder

```java
class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> list = new ArrayList();
        dfs(root, list);
        return list;
    }

    private void dfs(TreeNode root, List<Integer> list) {
        if(root == null)
            return;
        list.add(root.val);
        dfs(root.left, list);
        dfs(root.right, list);
    }
}
```

Postorder

```
class Solution {
    public List<Integer> postorderTraversal(TreeNode root) {
        List<Integer> list = new ArrayList();
        dfs(root, list);
        return list;
    }

    private void dfs(TreeNode root, List<Integer> list) {
        if(root == null)
            return;
        dfs(root.left, list);
        dfs(root.right, list);
        list.add(root.val);
    }
}
```

Other traversals

BFS / Level Order Traversal

Level Order Traversal

```
class Solution {
    public List<Integer> levelOrder(TreeNode root) {
        List<Integer> result = new ArrayList();
        if(root == null)
            return result;

        Queue<TreeNode> q = new LinkedList();
        q.add(root);
        while(q.size() > 0) {
            root = q.poll();
            result.add(root.val);
            if(root.left != null)
                q.add(root.left);
            if(root.right != null)
                q.add(root.right);
        }

        return result;
    }
}
```

Level Order Level By Level https://leetcode.com/problems/binary-tree-level-order-traversal/
Application - https://leetcode.com/problems/average-of-levels-in-binary-tree/

```
class Solution {
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> result = new ArrayList();
        if(root == null)
            return result;

        Queue<TreeNode> q = new LinkedList();
        q.add(root);
        while(q.size() > 0) {
            int size = q.size();
            List<Integer> level = new ArrayList();
            while(size-- > 0) {
                root = q.poll();
                level.add(root.val);
                if(root.left != null)
                    q.add(root.left);
                if(root.right != null)
                    q.add(root.right);
            }
            result.add(level);
        }

        return result;
    }
}
```

ZigZag Level Order https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/

```
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        List<List<Integer>> result = new ArrayList();
        if(root == null)
            return result;

        Queue<TreeNode> q = new LinkedList();
        q.add(root);
        boolean isLevelOdd = false;
        while(q.size() > 0) {
            int size = q.size();
            List<Integer> level = new ArrayList();
            while(size-- > 0) {
                root = q.poll();
                level.add(root.val);
                if(root.left != null)
                    q.add(root.left);
                if(root.right != null)
                    q.add(root.right);
            }
            if(isLevelOdd)
                Collections.reverse(level);
            result.add(level);
            isLevelOdd = !isLevelOdd;
        }

        return result;
    }
}
```

Leetcode articles

https://leetcode.com/discuss/general-discussion/937307/iterative-recursive-dfs-bfs-tree-traversal-in-pre-post-levelorder-views

https://leetcode.com/discuss/study-guide/1212004/binary-trees-study-guide