Advanced Programming LAB II

ASSIGNMENT - 3

Submitted by,

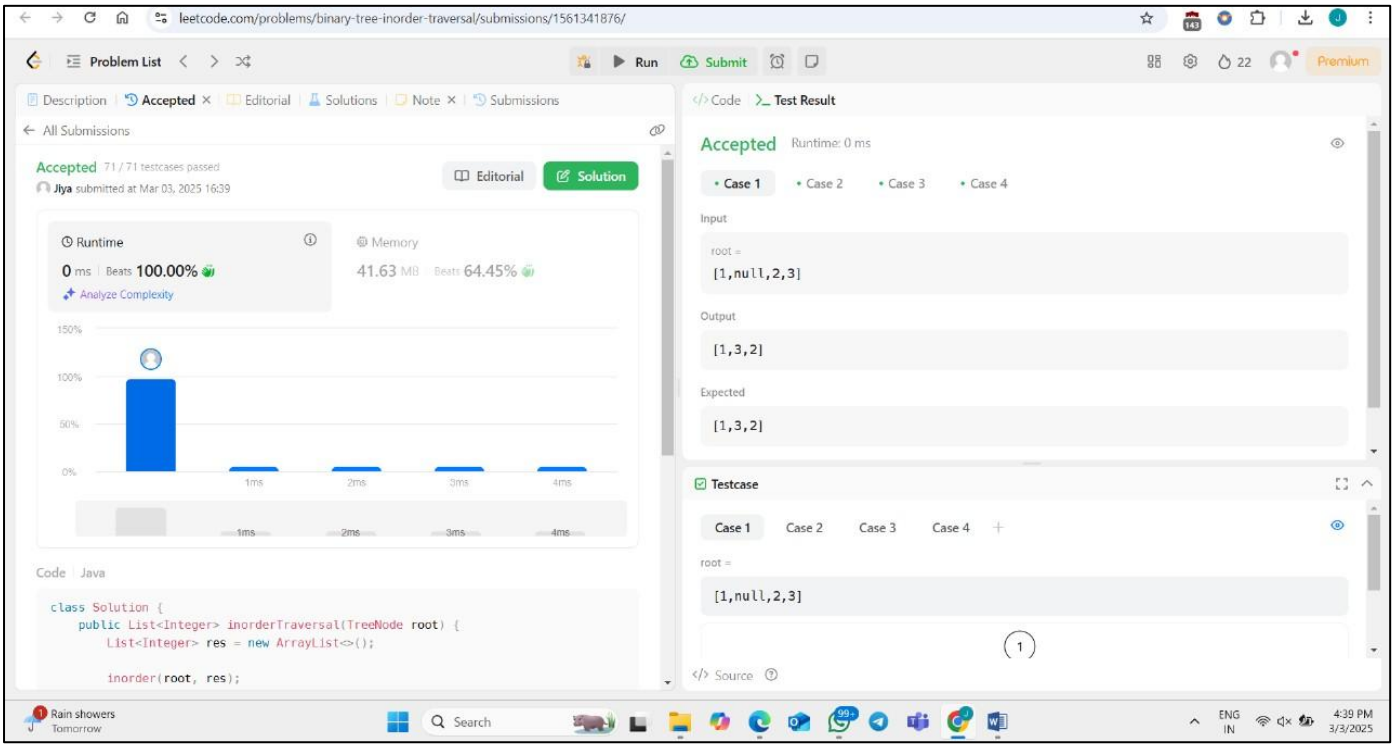
Jiya | 22BCS14856

22BCS_FL_IOT-601 (A)

94. Binary Tree Inorder Traversal

https://leetcode.com/problems/binary-tree-inorder-traversal/description/

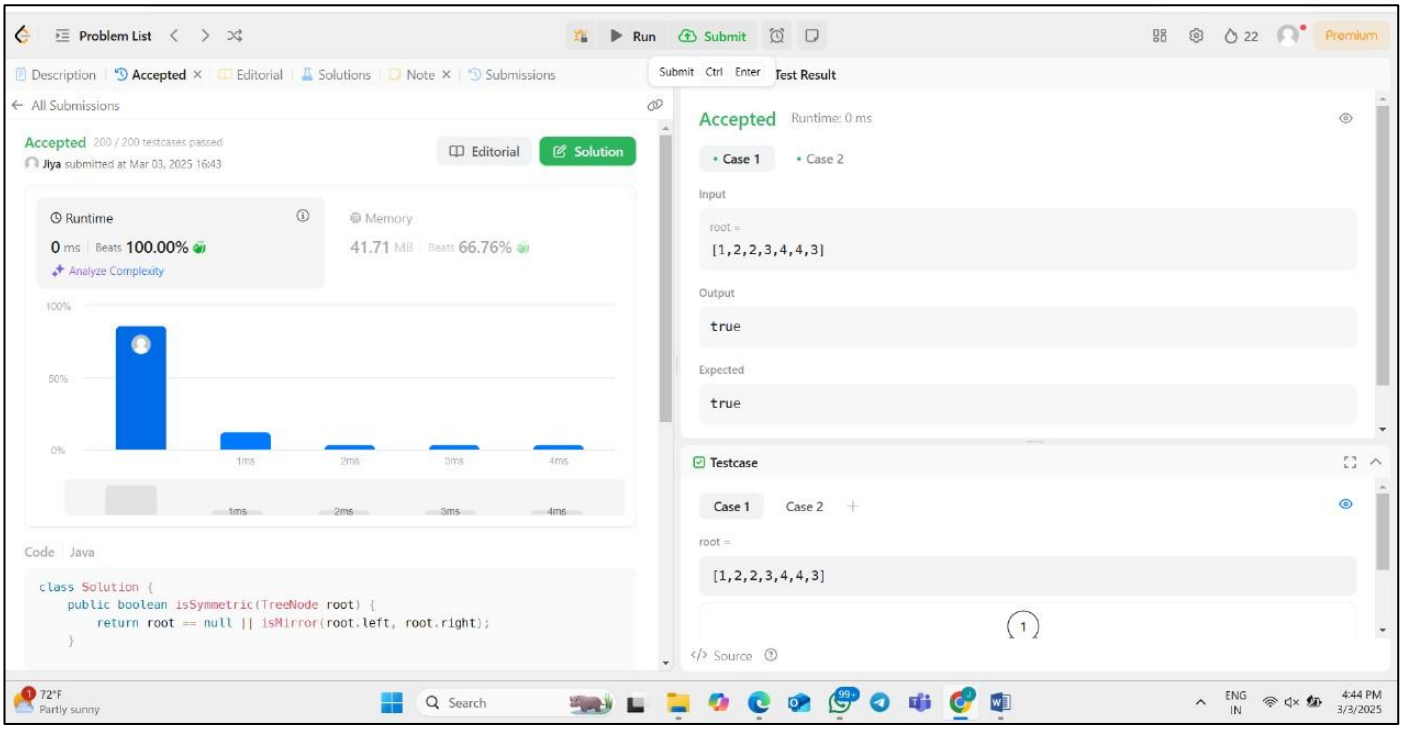
```
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> res = new ArrayList<>();

        inorder(root, res);
        return res;
    }

    private void inorder(TreeNode node, List<Integer> res) {
        if (node == null) {
            return;
        }
        inorder(node.left, res);
        res.add(node.val);
        inorder(node.right, res);
    }
}
```



101. Symmetric Tree

https://leetcode.com/problems/symmetric-tree/description/

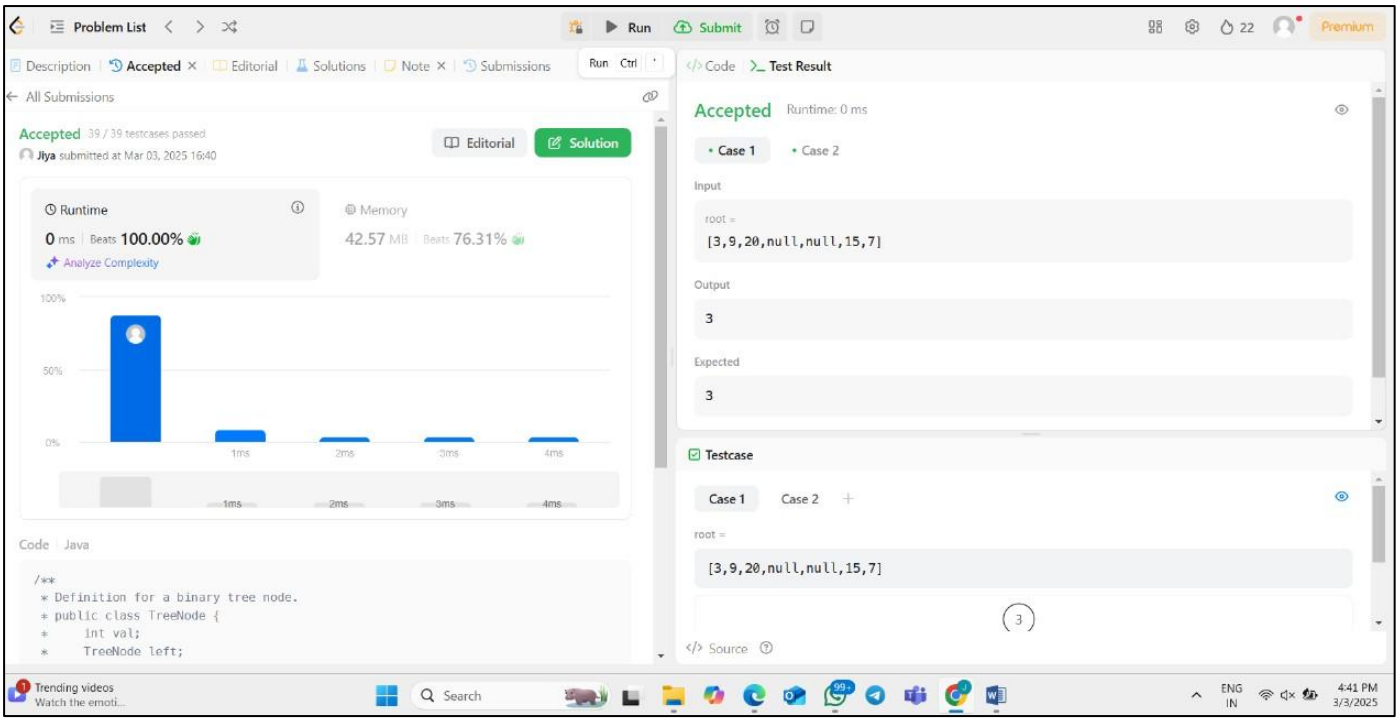
```
class Solution {
    public boolean isSymmetric(TreeNode root) {
        return root == null || isMirror(root.left, root.right);
    }

    private boolean isMirror(TreeNode a, TreeNode b) {
        if (a == null && b == null) return true;
        if (a == null || b == null || a.val != b.val) return false;
        return isMirror(a.left, b.right) && isMirror(a.right, b.left);
    }
}
```



104. Maximum Depth of Binary Tree

https://leetcode.com/problems/maximum-depth-of-binary-tree/description/

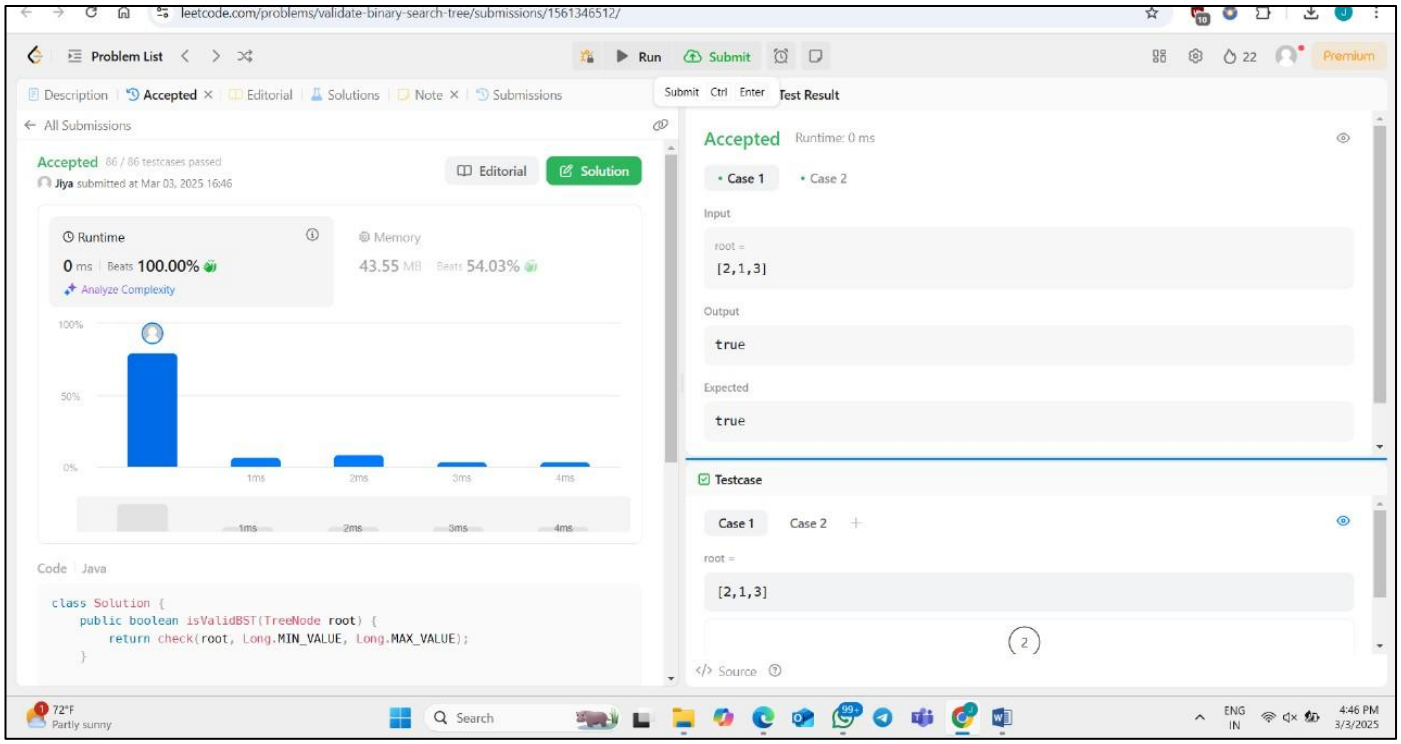
```
class Solution {
    public int maxDepth(TreeNode root) {

        if (root == null) return 0;
        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);

        return 1 + Math.max(leftDepth, rightDepth);
    }
}
```



98. Validate Binary Search Tree

https://leetcode.com/problems/validate-binary-search-tree/description/

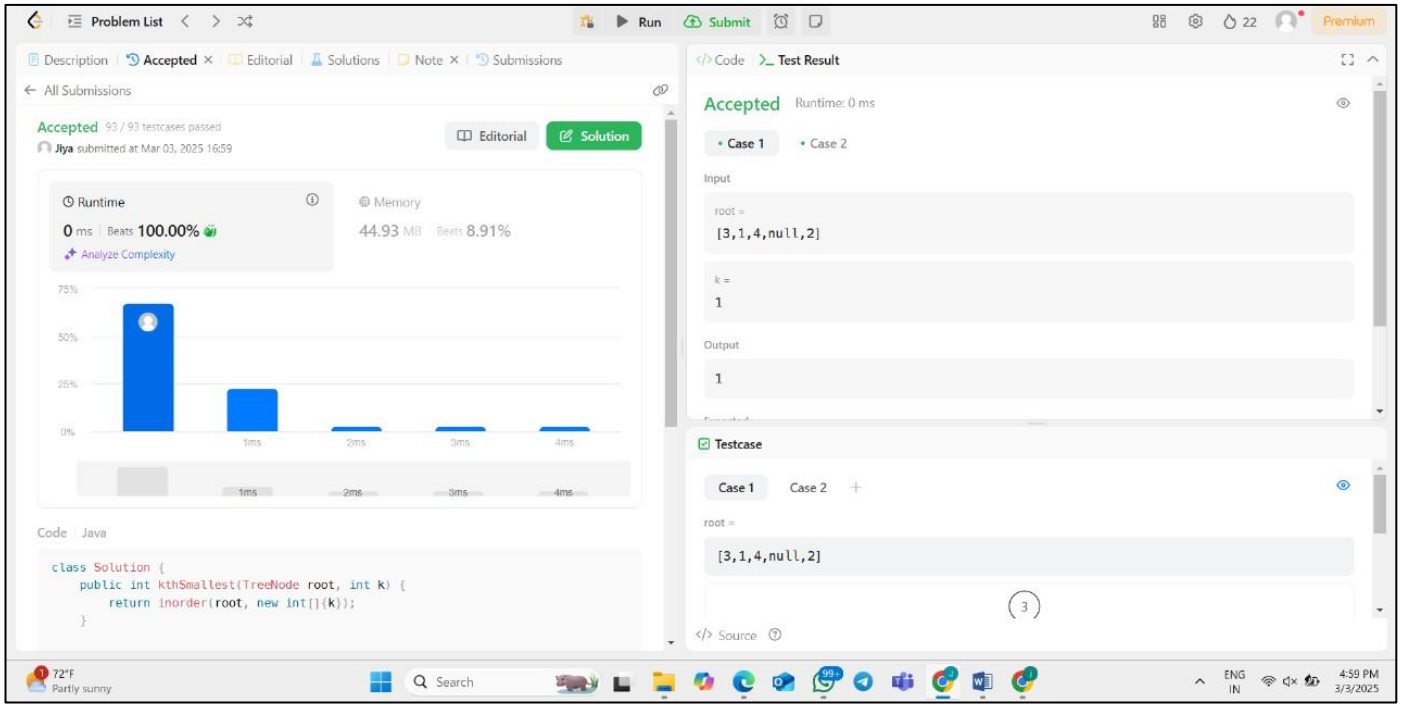
```
class Solution {
    public boolean isValidBST(TreeNode root) {
        return check(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }

    private boolean check(TreeNode node, long min, long max) {
        if (node == null) return true;
        if (node.val <= min || node.val >= max) return false;
        return check(node.left, min, node.val) && check(node.right, node.val, max);
    }
}
```



230. Kth Smallest Element in a BST

https://leetcode.com/problems/kth-smallest-element-in-a-bst/description/

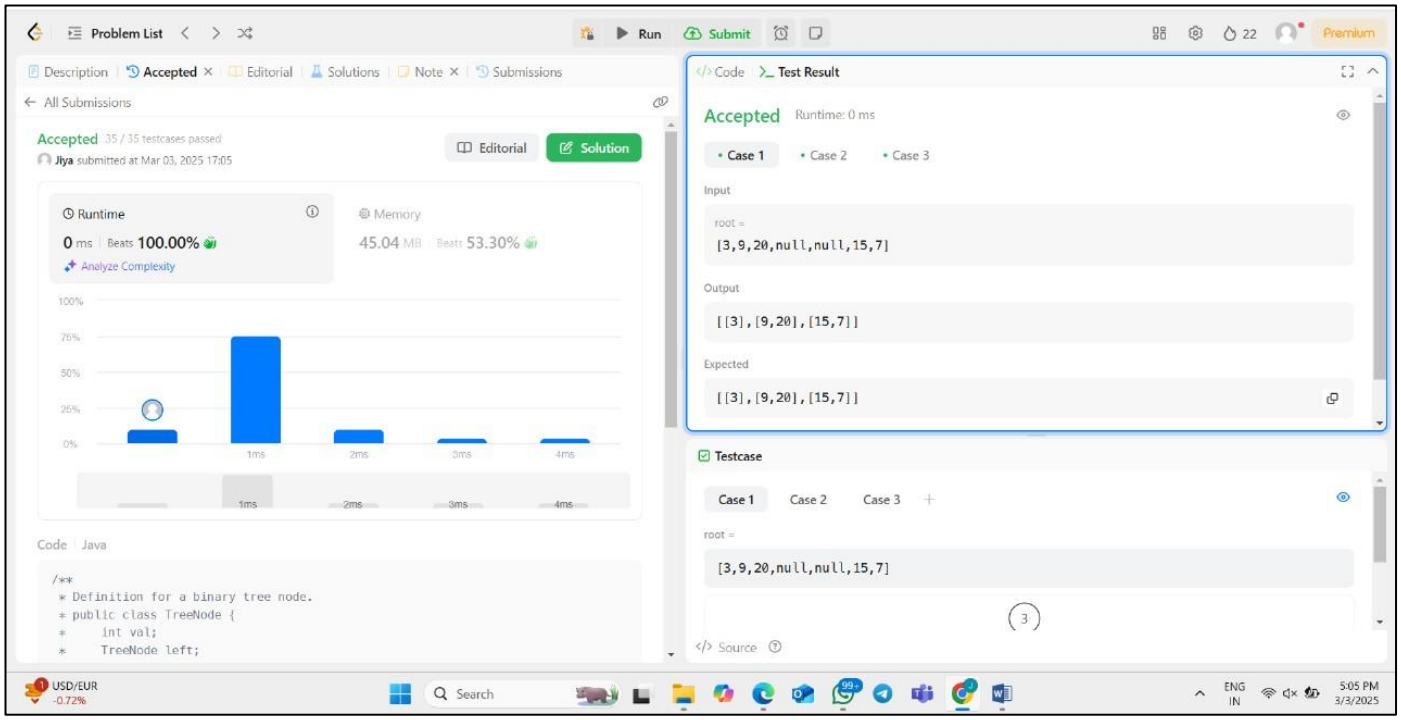
```
class Solution {
    public int kthSmallest(TreeNode root, int k) {
        return inorder(root, new int[]{k});
    }

    private int inorder(TreeNode node, int[] k) {
        if (node == null) return -1;
        int left = inorder(node.left, k);
        if (left != -1) return left;
        if (--k[0] == 0) return node.val;
        return inorder(node.right, k);
    }
}
```



102. Binary Tree Level Order Traversal

https://leetcode.com/problems/binary-tree-level-order-traversal/description/

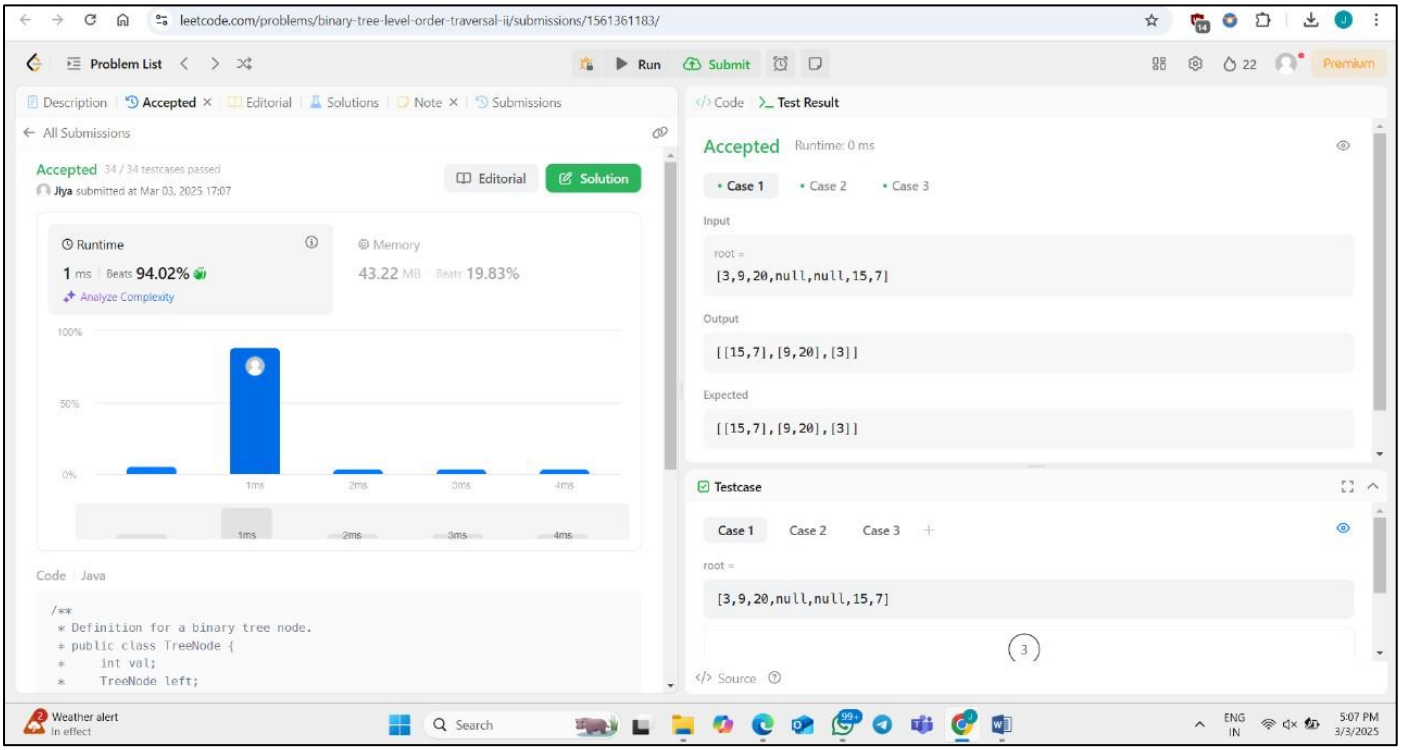
```
class Solution {
    public List<List<Integer>> levelOrder(TreeNode root)
    {
        List<List<Integer>>al=new ArrayList<>();
        pre(root,0,al);
        return al;
    }
    public static void pre(TreeNode root,int l,List<List<Integer>>al)
    {
        if(root==null)
            return;
        if(al.size()==l)
        {
            List<Integer>li=new ArrayList<>();
            li.add(root.val);
            al.add(li);
        }
        else
            al.get(l).add(root.val);
        pre(root.left,l+1,al);
        pre(root.right,l+1,al);
    }
}
```



102. Binary Tree Level Order Traversal II

https://leetcode.com/problems/binary-tree-level-order-traversal-ii/description/

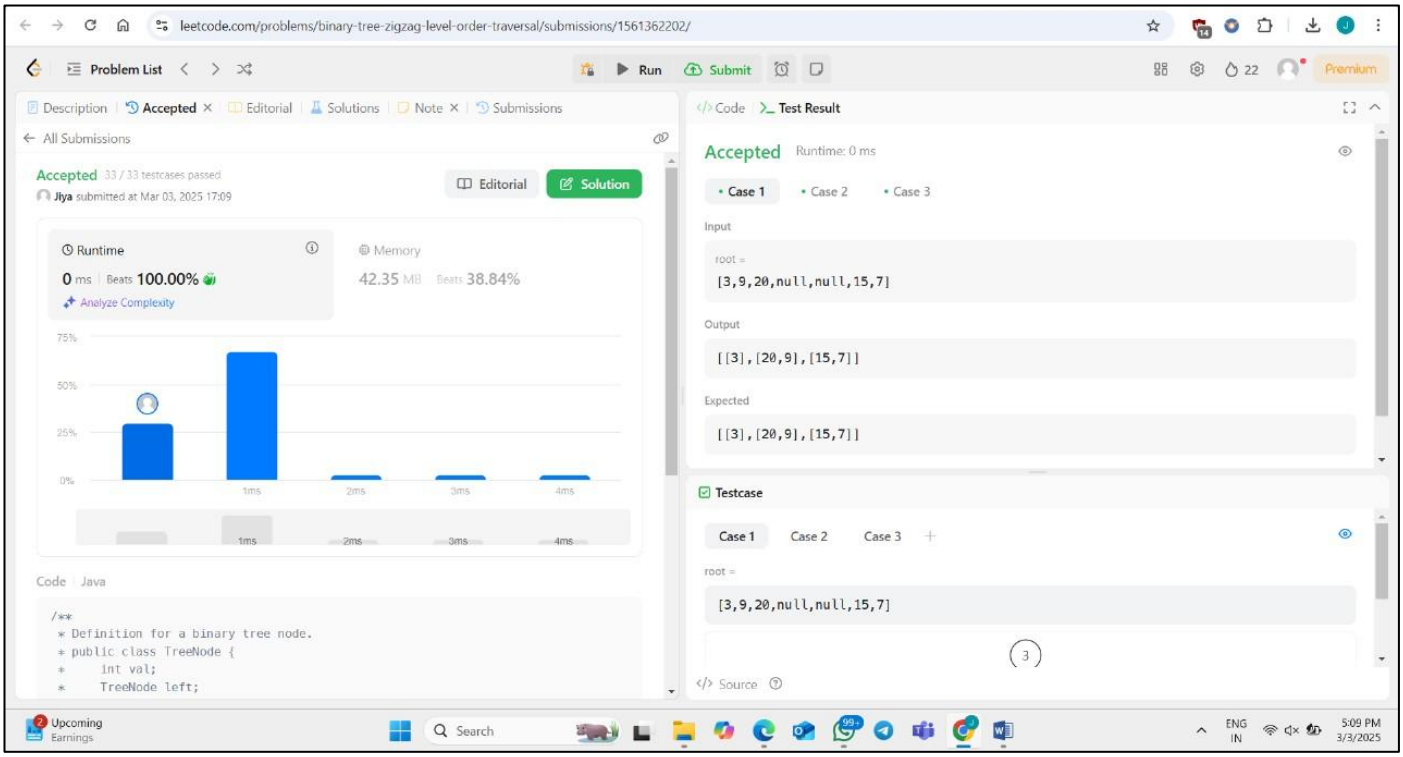
```
import java.util.;

class Solution {
    public List<List<Integer>> levelOrderBottom(TreeNode root) {
        List<List<Integer>> result = new LinkedList<>();
        if (root == null) return result;

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);

        while (!q.isEmpty()) {
            int size = q.size();
            List<Integer> level = new ArrayList<>();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                level.add(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
            result.add(0, level);
        }
        return result;
    }
}
```



103. Binary Tree Zigzag Level Order Traversal

https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/description/

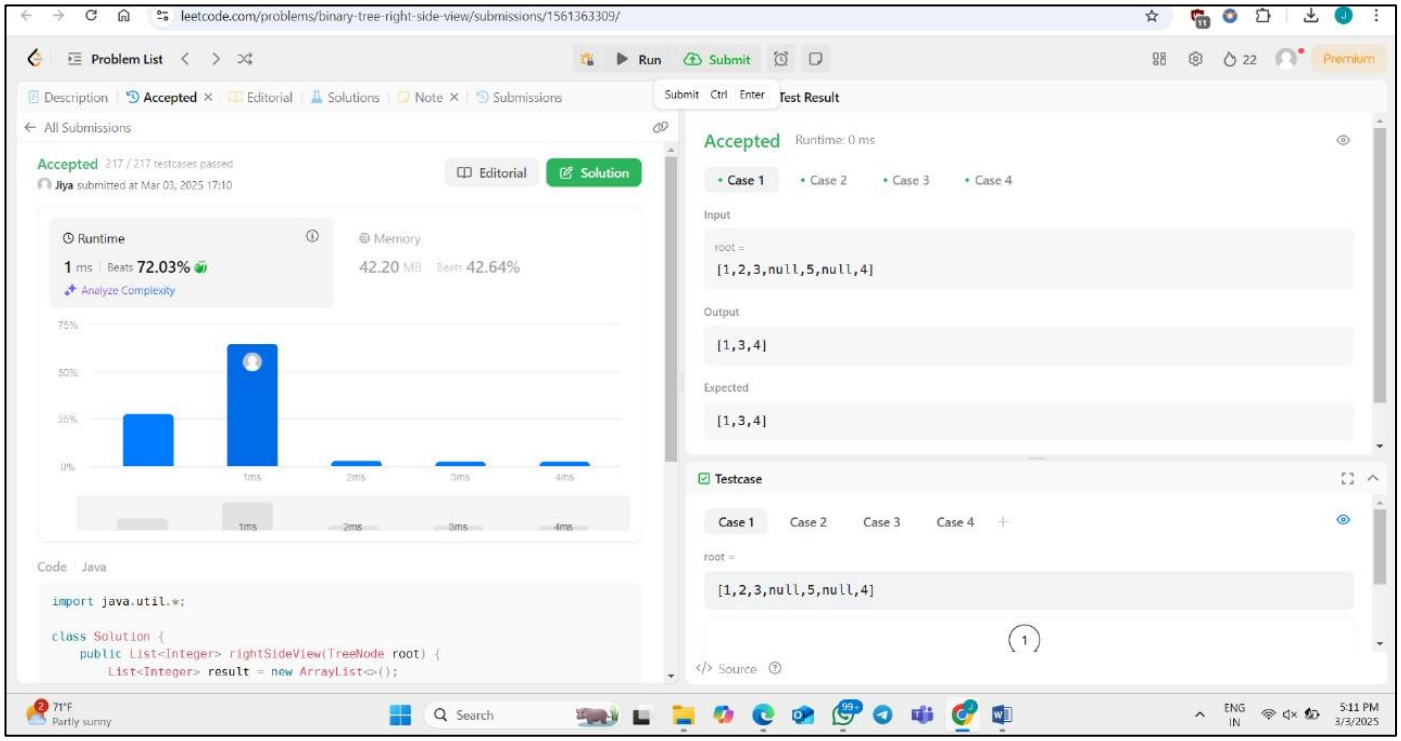
```
import java.util.*;

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 leftToRight = true;

        while (!q.isEmpty()) {
            int size = q.size();
            LinkedList<Integer> level = new LinkedList<>();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (leftToRight) level.addLast(node.val);
                else level.addFirst(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
            result.add(level);
            leftToRight = !leftToRight;
        }
        return result;
    }
}
```



199. Binary Tree Right Side View

https://leetcode.com/problems/binary-tree-right-side-view/description/

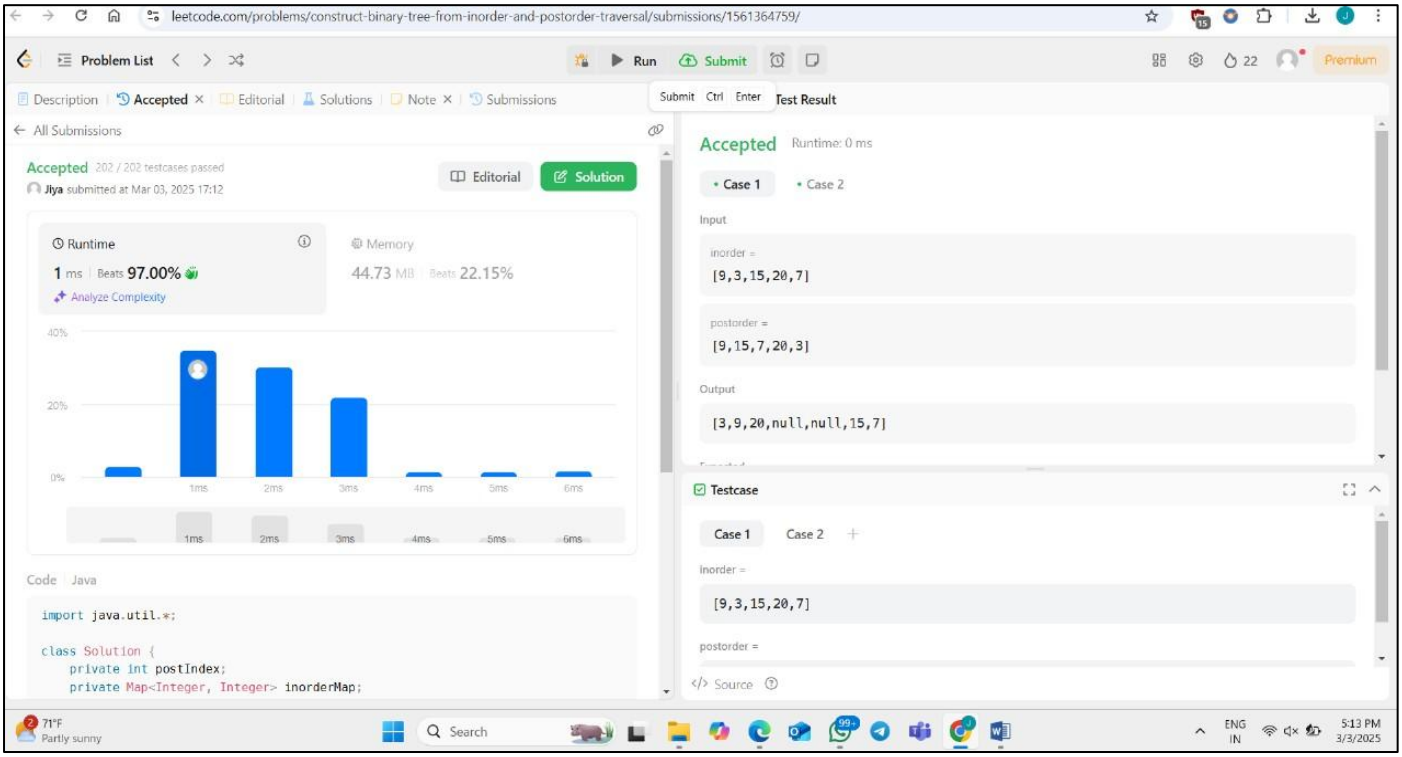
```
import java.util.*;

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

        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);

        while (!q.isEmpty()) {
            int size = q.size();
            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (i == size - 1) result.add(node.val);
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
        }
        return result;
    }
}
```



106. Construct Binary Tree from Inorder and Postorder Traversal

https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/description/

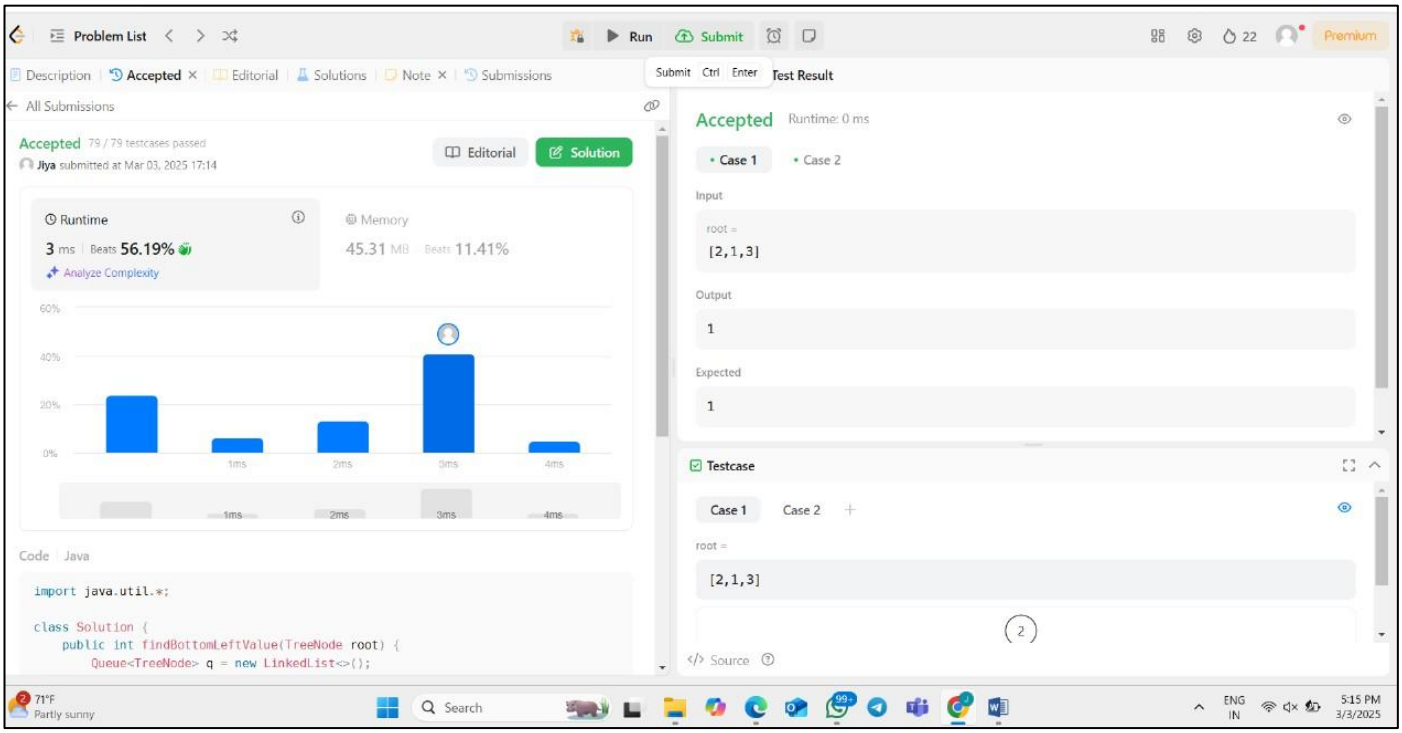
```java
import java.util.*;

class Solution {
    private int postIndex;
    private Map<Integer, Integer> inorderMap;

    public TreeNode buildTree(int[] inorder, int[] postorder) {
        postIndex = postorder.length - 1;
        inorderMap = new HashMap<>();
        for (int i = 0; i < inorder.length; i++) {
            inorderMap.put(inorder[i], i);
        }
        return build(inorder, postorder, 0, inorder.length - 1);
    }

    private TreeNode build(int[] inorder, int[] postorder, int left, int right) {
        if (left > right) return null;

        int rootVal = postorder[postIndex--];
        TreeNode root = new TreeNode(rootVal);
        int index = inorderMap.get(rootVal);

        root.right = build(inorder, postorder, index + 1, right);
        root.left = build(inorder, postorder, left, index - 1);

        return root;
    }
}
```



513. Find Bottom Left Tree Value

https://leetcode.com/problems/find-bottom-left-tree-value/description/

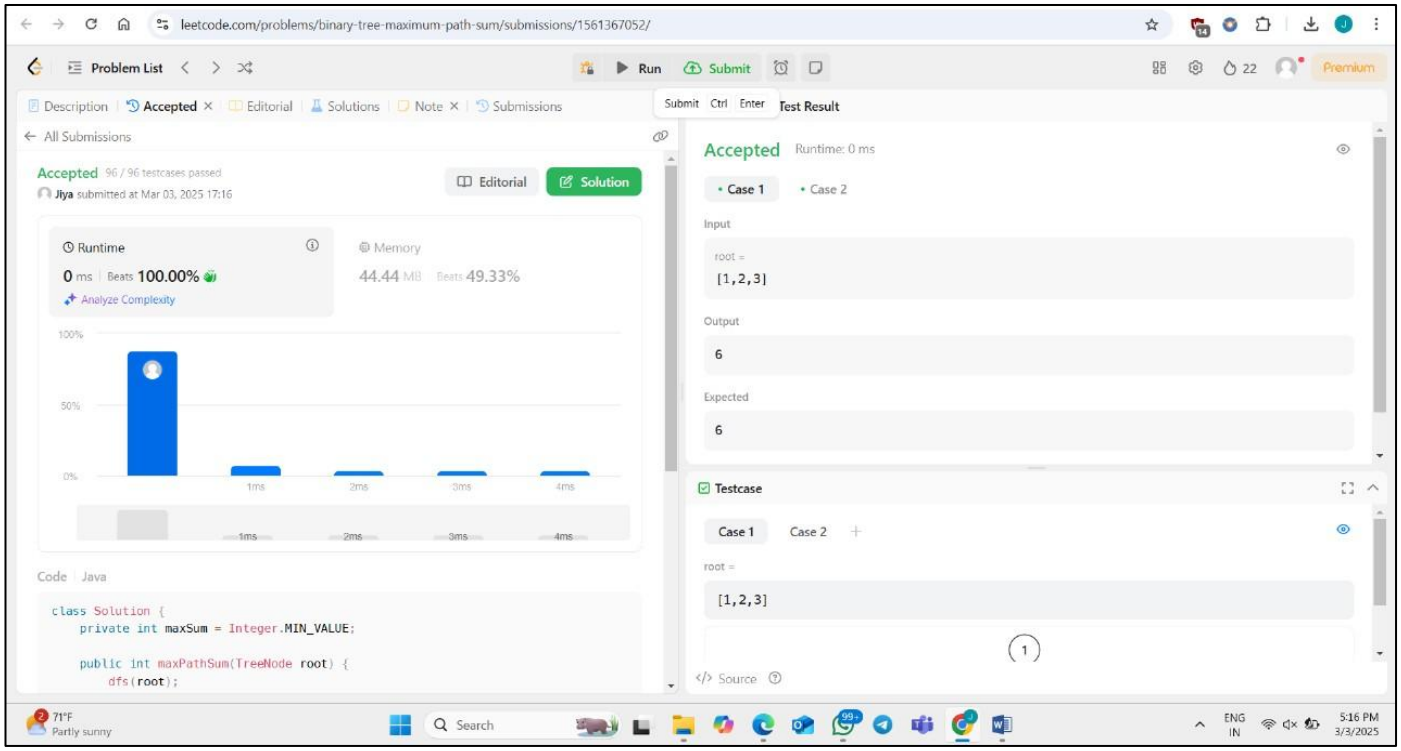
```
import java.util.*;

class Solution {
    public int findBottomLeftValue(TreeNode root) {
        Queue<TreeNode> q = new LinkedList<>();
        q.add(root);
        int leftmost = root.val;

        while (!q.isEmpty()) {
            int size = q.size();
            leftmost = q.peek().val;

            for (int i = 0; i < size; i++) {
                TreeNode node = q.poll();
                if (node.left != null) q.add(node.left);
                if (node.right != null) q.add(node.right);
            }
        }
        return leftmost;
    }
}
```



124. Binary Tree Maximum Path Sum

https://leetcode.com/problems/binary-tree-maximum-path-sum/description/

```
class Solution {
    private int maxSum = Integer.MIN_VALUE;

    public int maxPathSum(TreeNode root) {
        dfs(root);
        return maxSum;
    }

    private int dfs(TreeNode node) {
        if (node == null) return 0;

        int left = Math.max(0, dfs(node.left));
        int right = Math.max(0, dfs(node.right));

        maxSum = Math.max(maxSum, left + right + node.val);

        return Math.max(left, right) + node.val;
    }
}
```



987. Vertical Order Traversal of a Binary Tree

https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/description/

```
import java.util.*;
class Solution {
    public List<List<Integer>> verticalTraversal(TreeNode root) {
        TreeMap<Integer, TreeMap<Integer, PriorityQueue<Integer>>> map = new
TreeMap<>();
        dfs(root, 0, 0, map);
        List<List<Integer>> result = new ArrayList<>();
        for (TreeMap<Integer, PriorityQueue<Integer>> ys : map.values()) {
            List<Integer> column = new ArrayList<>();
            for (PriorityQueue<Integer> nodes : ys.values()) {
                while (!nodes.isEmpty()) {
                    column.add(nodes.poll());
                }
            }
            result.add(column);
        }
        return result;
    }
    private void dfs(TreeNode node, int x, int y, TreeMap<Integer, TreeMap<Integer,
PriorityQueue<Integer>>> map) {
        if (node == null) return;

        map.putIfAbsent(x, new TreeMap<>());
        map.get(x).putIfAbsent(y, new PriorityQueue<>());
        map.get(x).get(y).offer(node.val);

        dfs(node.left, x - 1, y + 1, map);
        dfs(node.right, x + 1, y + 1, map);
    }
}
```