



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Advance Programming

Student Name: Deshveer Singh

UID: 22BCS14259

Branch: CSE

Section/Group: 612-B

Semester: 6th

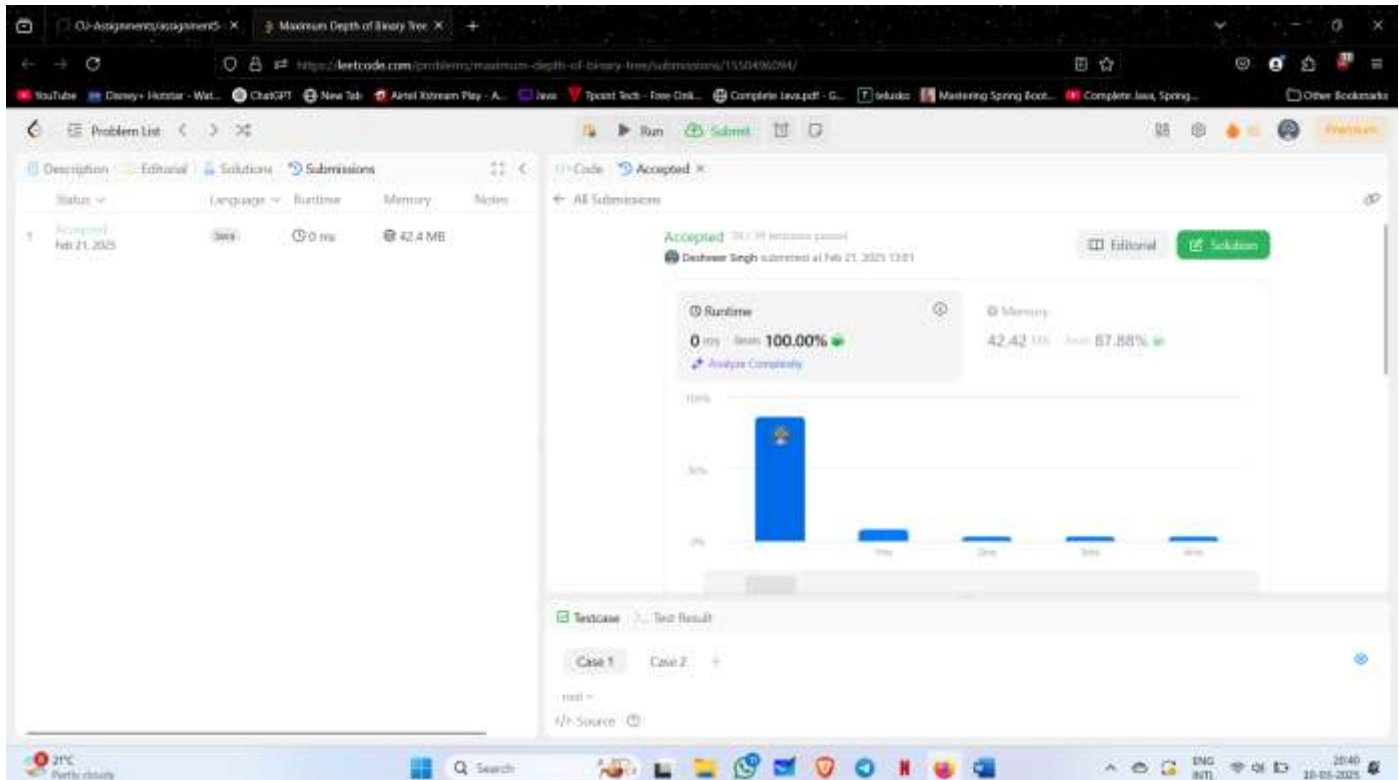
Date of Performance: 20/02/25

Subject Name: Advance Programming

Subject Code: 22CSP-367

1. 104:

```
class Solution {  
    public int maxDepth(TreeNode root) {  
        if(root == null) return 0;  
  
        int nodeRight = maxDepth(root.left);  
        int nodeLeft = maxDepth(root.right);  
  
        return 1 + Math.max( nodeLeft, nodeRight);  
    }  
}
```



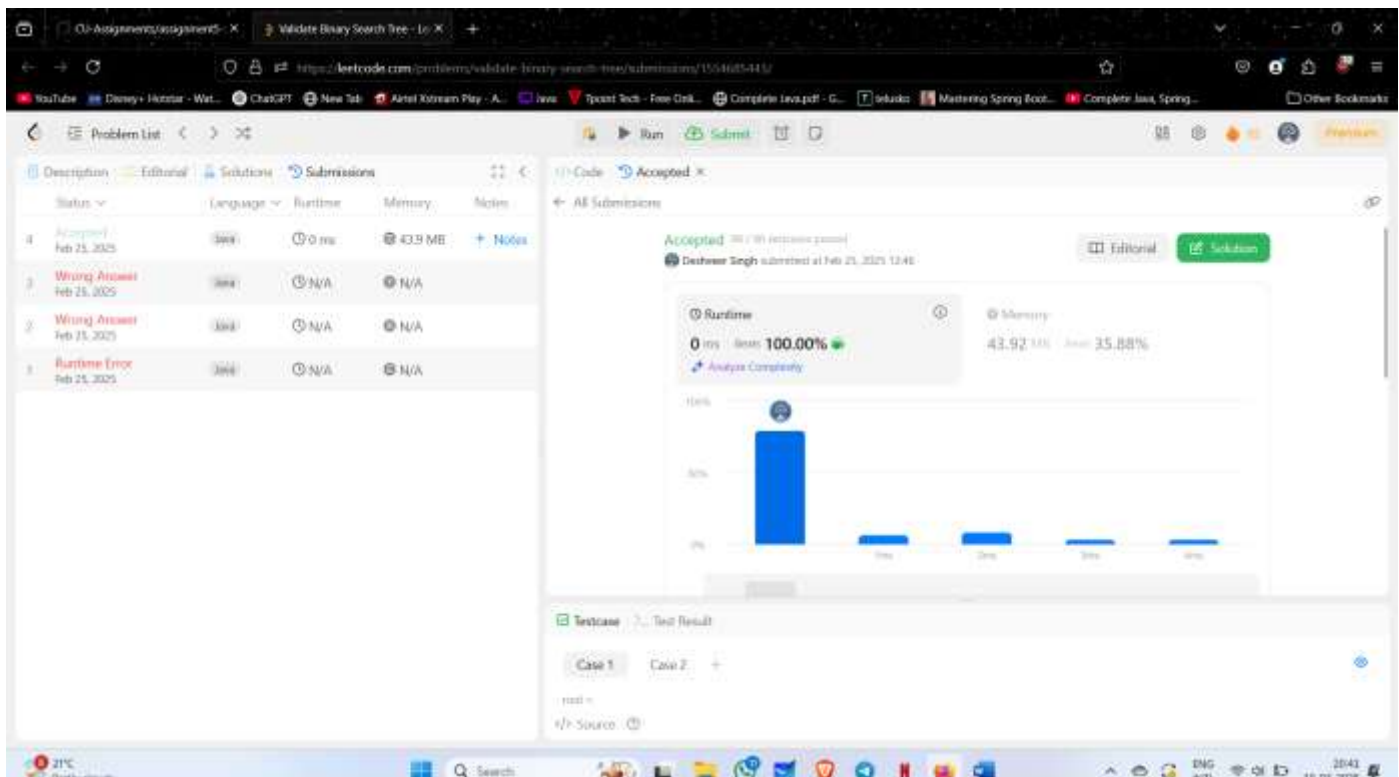
2. 98:

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

    private boolean valid(TreeNode root, long min, long max) {
        if (root == null) {
            return true;
        }

        if (root.val <= min || root.val >= max) {
            return false;
        }

        return valid(root.left, min, root.val) && valid(root.right, root.val, max);
    }
}
```



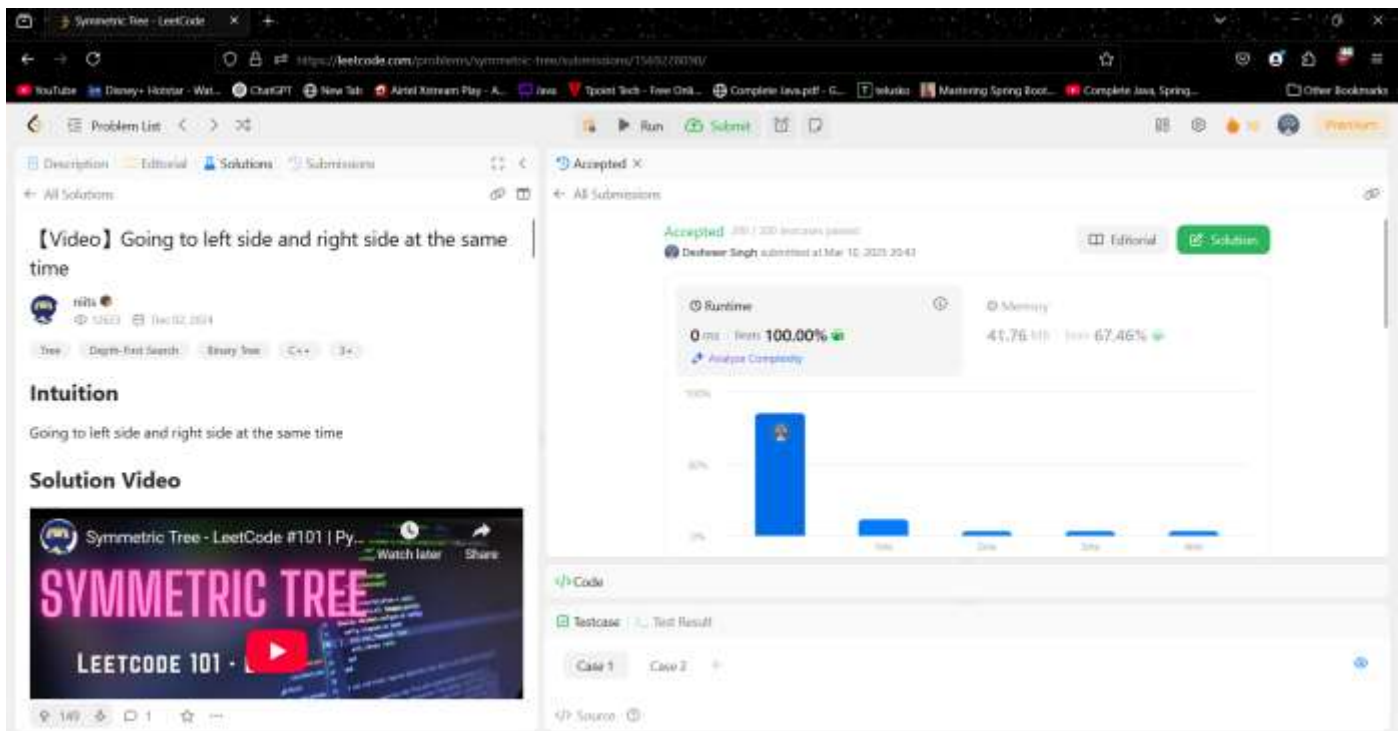
3. 101:

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

    private boolean isMirror(TreeNode n1, TreeNode n2) {
        if (n1 == null && n2 == null) {
            return true;
        }

        if (n1 == null || n2 == null) {
            return false;
        }

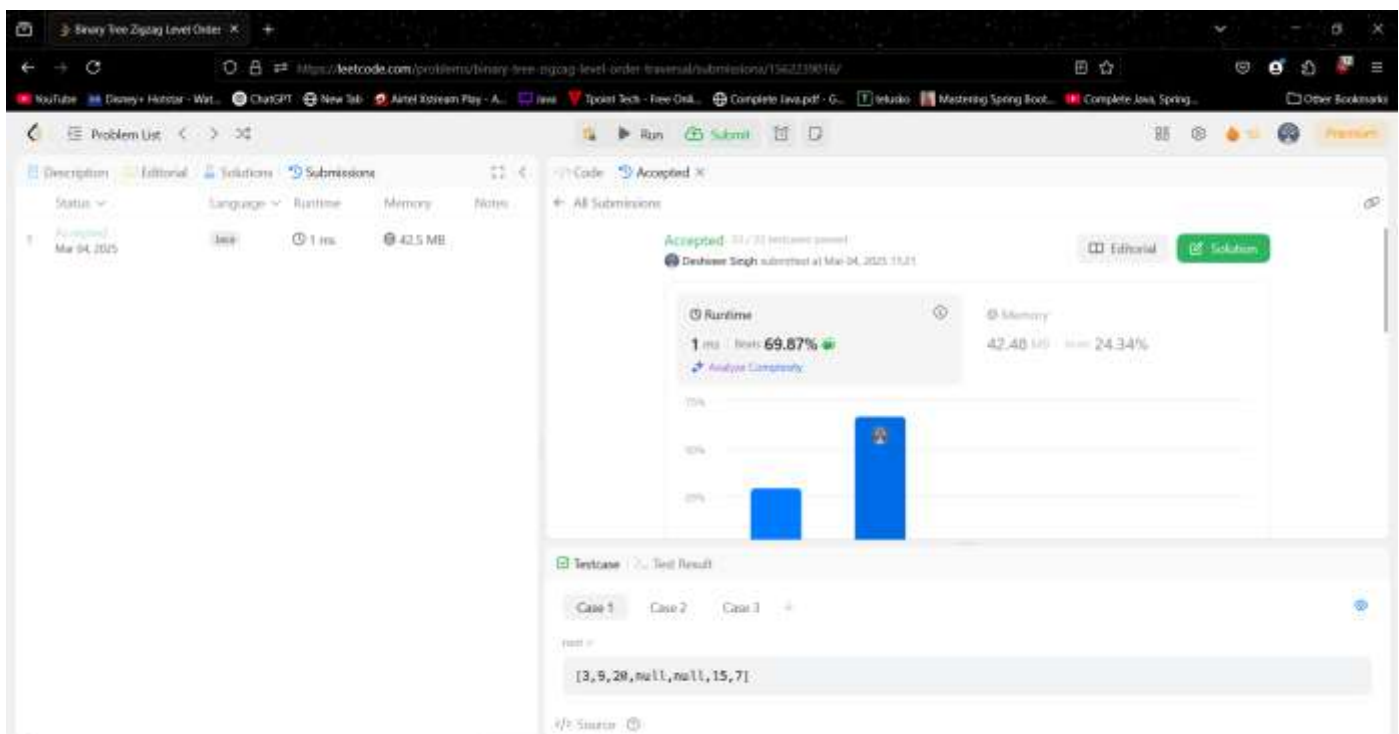
        return n1.val == n2.val && isMirror(n1.left, n2.right) && isMirror(n1.right, n2.left);
    }
}
```



The screenshot shows the LeetCode interface for the 'Symmetric Tree' problem (ID 101). On the left, there's a video player titled 'Symmetric Tree - LeetCode #101 | Py...' with a thumbnail showing the text 'SYMMETRIC TREE' and 'LEETCODE 101'. Below the video, there's a section for 'Intuition' and 'Solution Video'. The main content area on the right shows the problem description, a 'Run' button, and a submission status of 'Accepted'. The submission details indicate that the code was submitted by 'Devesh Singh' at 10:20:43 on Mar 10, 2021. The performance metrics show 0 ms runtime (beats 100.00%) and 41.76 MB memory (beats 67.46%). A bar chart shows the distribution of runtime performance across different languages. At the bottom, there's a 'Code' section with a 'Testcase' dropdown and buttons for 'Case 1' and 'Case 2'.

4. 103:

```
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        Queue<TreeNode> st = new LinkedList<>();
        List<List<Integer>> ans = new ArrayList<>();
        if(root!=null)st.add(root);
        int k = 0;
        while(!st.isEmpty()){
            ArrayList<Integer> arr = new ArrayList<>();
            int n = st.size();
            // PUSHING TILL QUEUE.SIZE()
            for(int i = 0 ; i<n ; i++){
                TreeNode temp = st.poll();
                if(temp==null)break;
                arr.add(temp.val);
                if(temp.left!=null)st.add(temp.left);
                if(temp.right!=null)st.add(temp.right);
            }
            // REVERSING IF K IS ODD
            if(k%2!=0){
                Collections.reverse(arr);
                ans.add(new ArrayList<>(arr));
            }else ans.add(new ArrayList<>(arr));
            // INCREASING COUNTER
            k++;
        }return ans;
    }
}
```



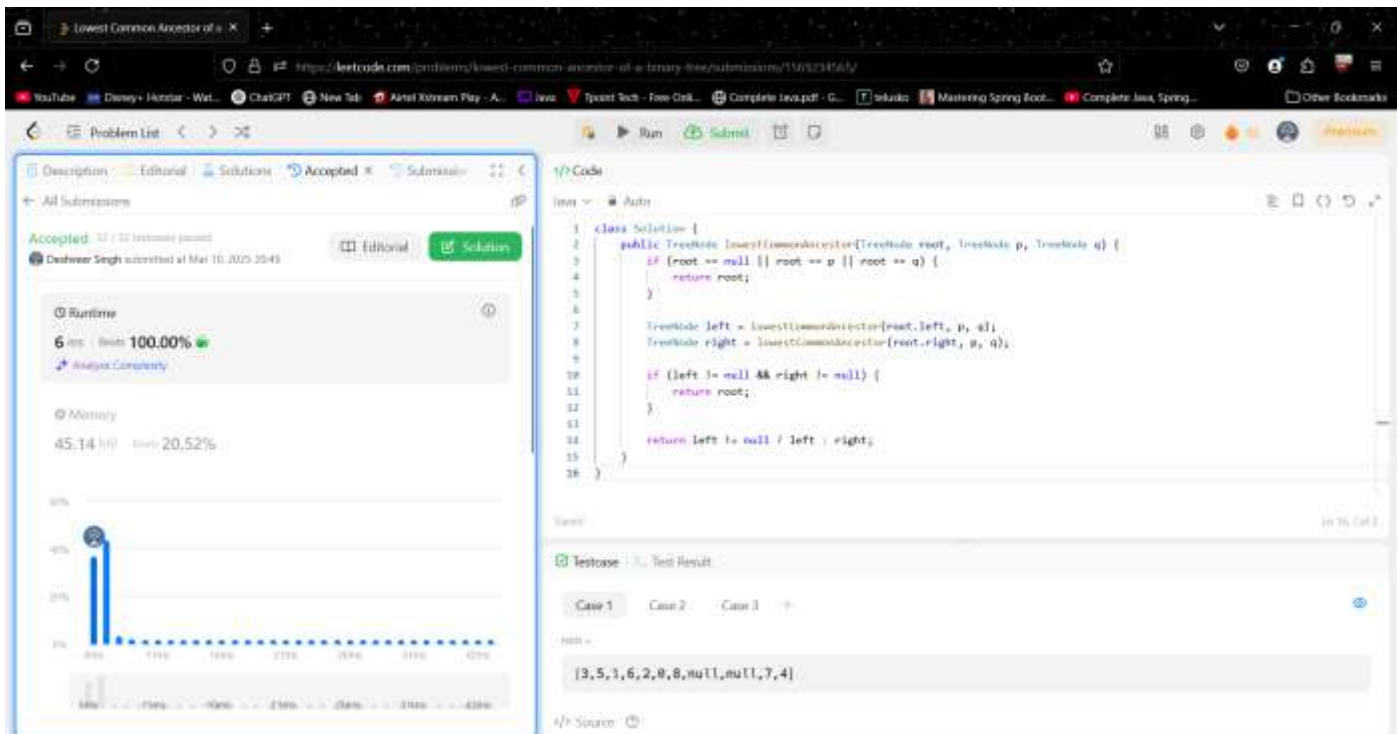
5. 236:

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if (root == null || root == p || root == q) {
            return root;
        }

        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p, q);

        if (left != null && right != null) {
            return root;
        }

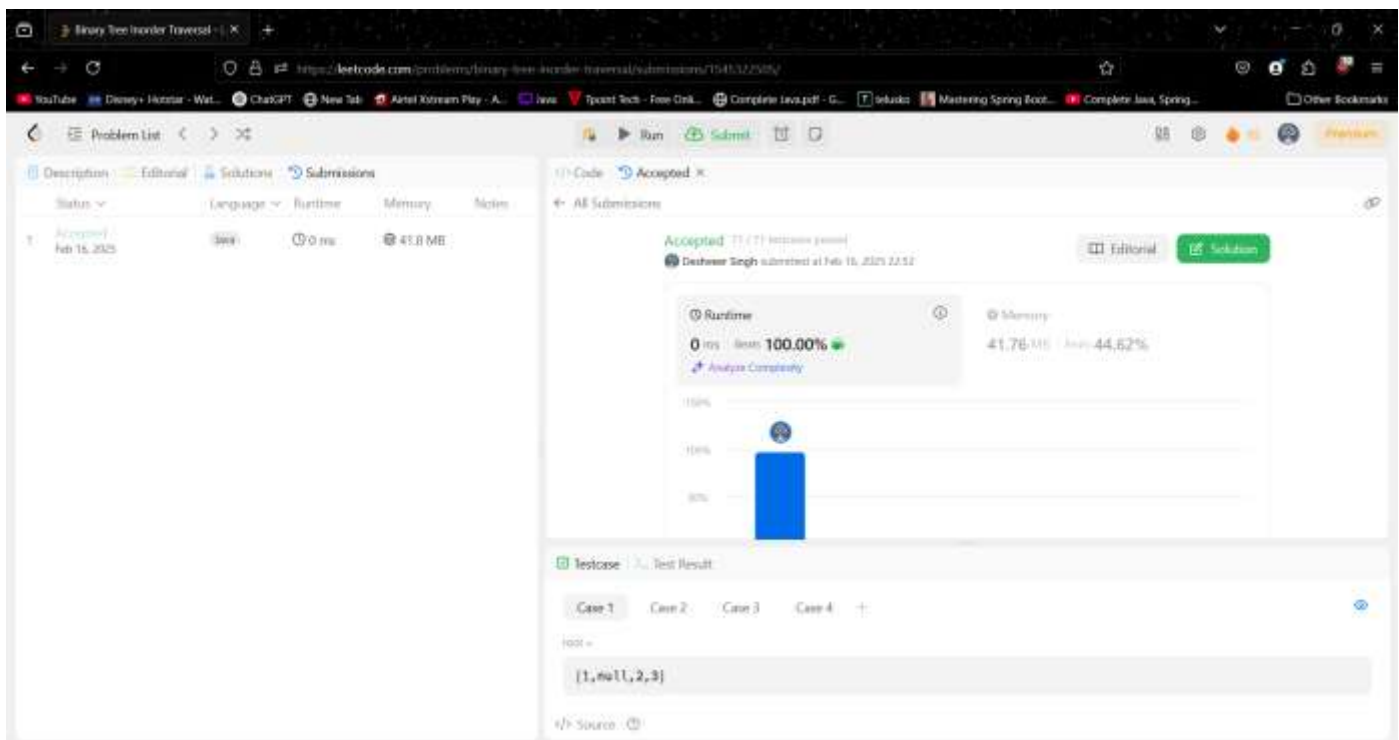
        return left != null ? left : right;
    }
}
```



6. 94:

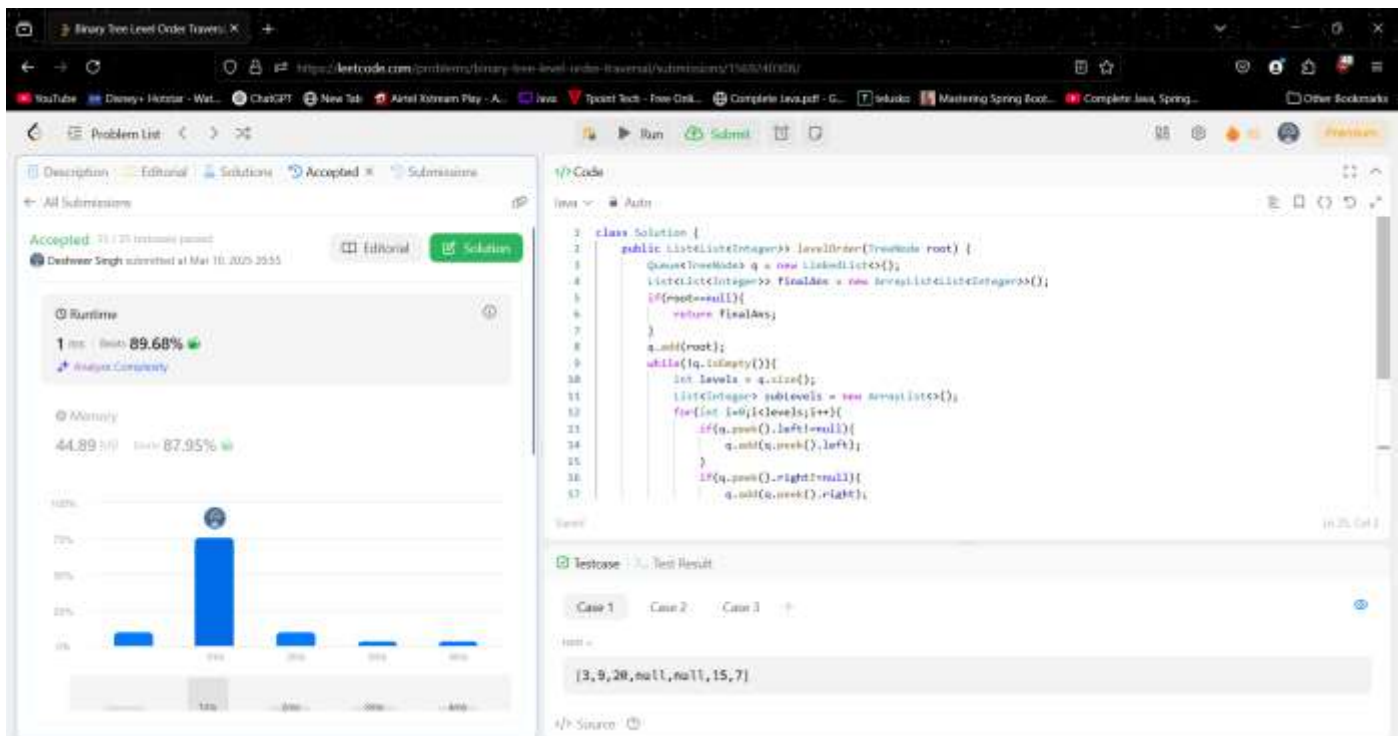
```
class Solution {
    private List<Integer> res = new ArrayList<>();
    public List<Integer> inorderTraversal(TreeNode root) {
        traverse(root);
        return res;
    }

    private void traverse(TreeNode root) {
        if (root == null) {
            return;
        }
        traverse(root.left);
        res.add(root.val);
        traverse(root.right);
    }
}
```



7. 102:

```
class Solution {
    public List<List<Integer>> levelOrder(TreeNode root) {
        Queue<TreeNode> q = new LinkedList<>();
        List<List<Integer>> finalAns = new ArrayList<List<Integer>>();
        if(root==null){
            return finalAns;
        }
        q.add(root);
        while(!q.isEmpty()){
            int levels = q.size();
            List<Integer> subLevels = new ArrayList<>();
            for(int i=0;i<levels;i++){
                if(q.peek().left!=null){
                    q.add(q.peek().left);
                }
                if(q.peek().right!=null){
                    q.add(q.peek().right);
                }
                subLevels.add(q.remove().val);
            }
            finalAns.add(subLevels);
        }
        return finalAns;
    }
}
```



8. 230:

```
class Solution {
    private int count = 0; // Counter for visited nodes

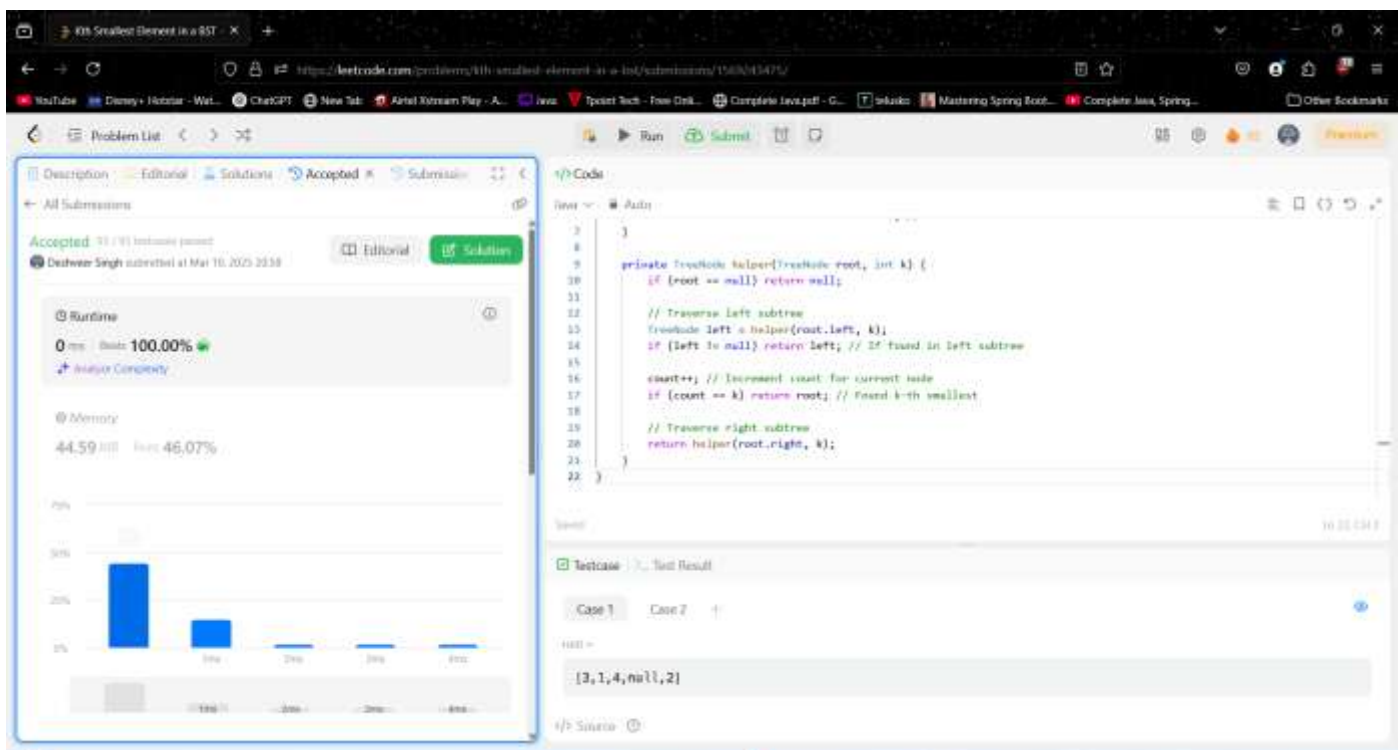
    public int kthSmallest(TreeNode root, int k) {
        TreeNode result = helper(root, k);
        return result != null ? result.val : 0; // Return value or 0 if not found
    }

    private TreeNode helper(TreeNode root, int k) {
        if (root == null) return null;

        // Traverse left subtree
        TreeNode left = helper(root.left, k);
        if (left != null) return left; // If found in left subtree

        count++; // Increment count for current node
        if (count == k) return root; // Found k-th smallest

        // Traverse right subtree
        return helper(root.right, k);
    }
}
```



9. 116:

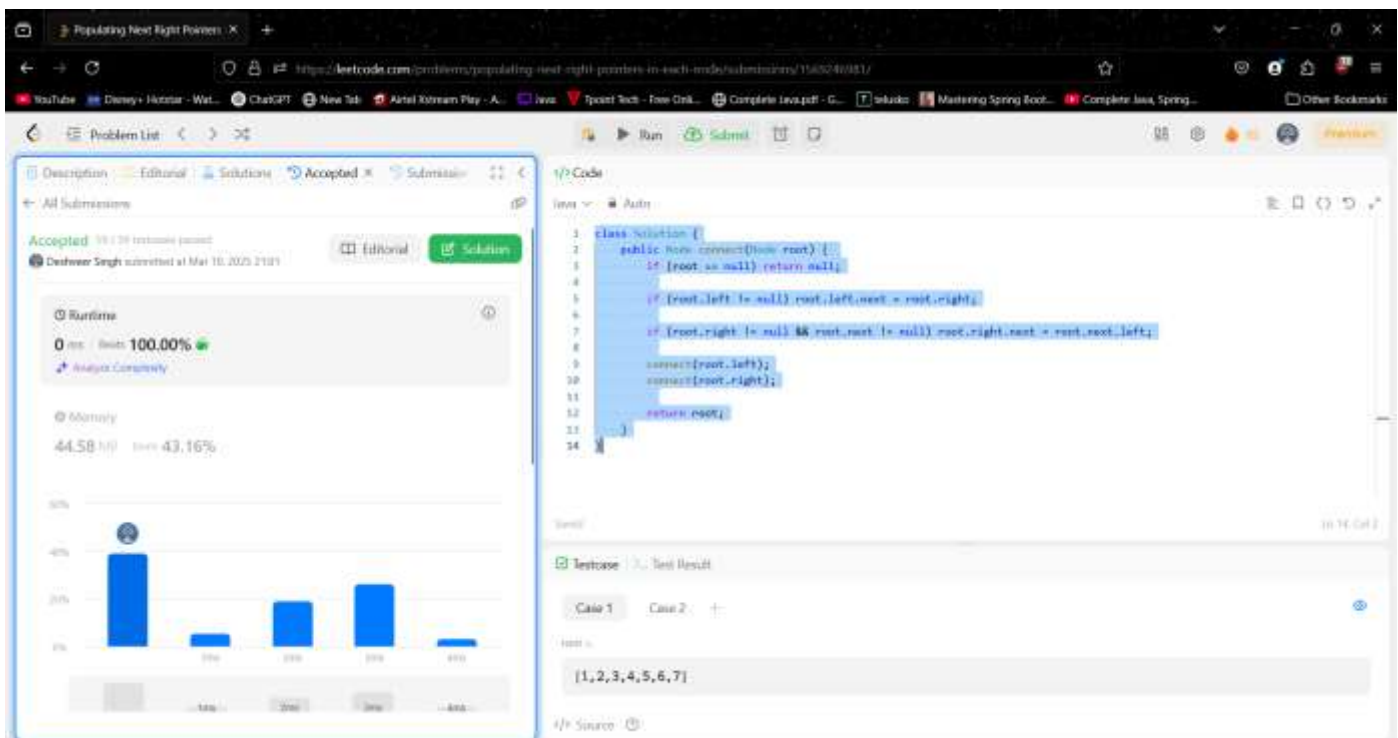
```
class Solution {
    public Node connect(Node root) {
        if (root == null) return null;

        if (root.left != null) root.left.next = root.right;

        if (root.right != null && root.next != null) root.right.next = root.next.left;

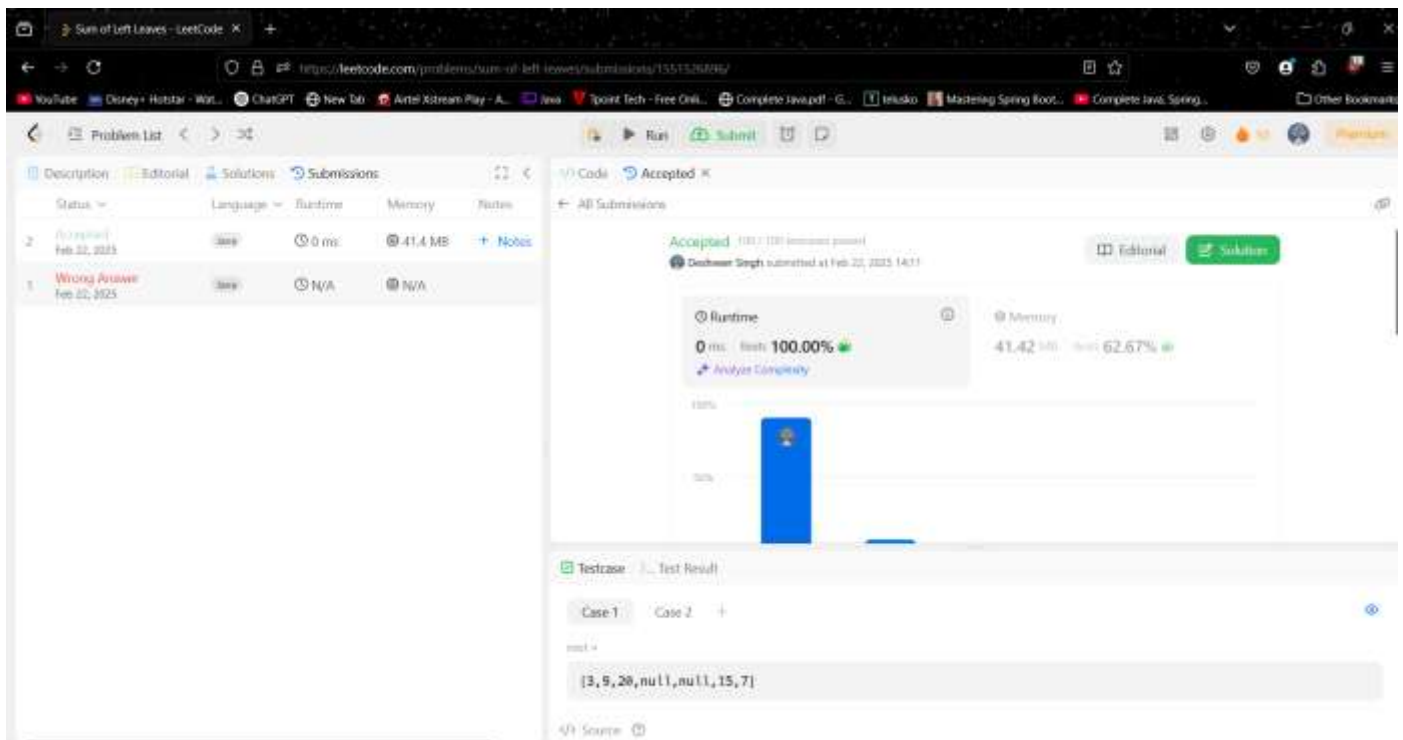
        connect(root.left);
        connect(root.right);

        return root;
    }
}
```



10. 404:

```
class Solution {  
    private int sum = 0;  
    public int sumOfLeftLeaves(TreeNode root) {  
        sumTree(root, false);  
        return sum;  
    }  
    private void sumTree(TreeNode node, boolean flag){  
        if(node == null) return;  
  
        if(node.left == null && node.right == null && flag == true){  
            sum += node.val;  
        }  
  
        sumTree(node.left, true);  
        sumTree(node.right, false);  
  
        return;  
    }  
}
```



The screenshot displays the LeetCode interface for the problem "Sum of Left Leaves". The left sidebar shows the problem list with a submission status of "Wrong Answer" for the test case "[3, 9, 20, null, null, 15, 7]". The main area shows the submission details for a user named "Dishwan Singh" who submitted the solution on Feb 22, 2023, at 14:11. The submission is marked as "Accepted" and shows a runtime of 0 ms and memory usage of 41.42 MB. The test case input is "[3, 9, 20, null, null, 15, 7]".