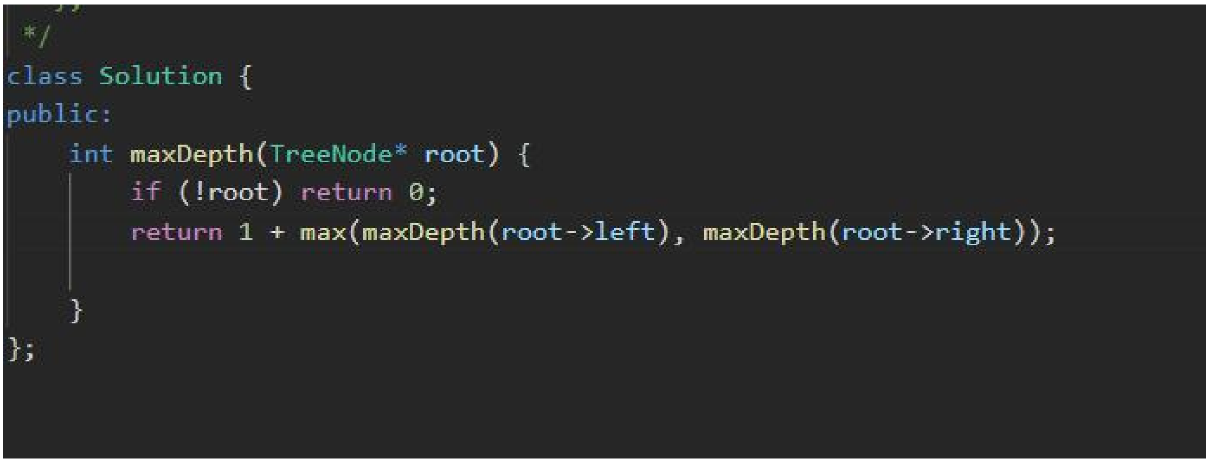
Advanced Pragramming

ASSIGNMENT 05

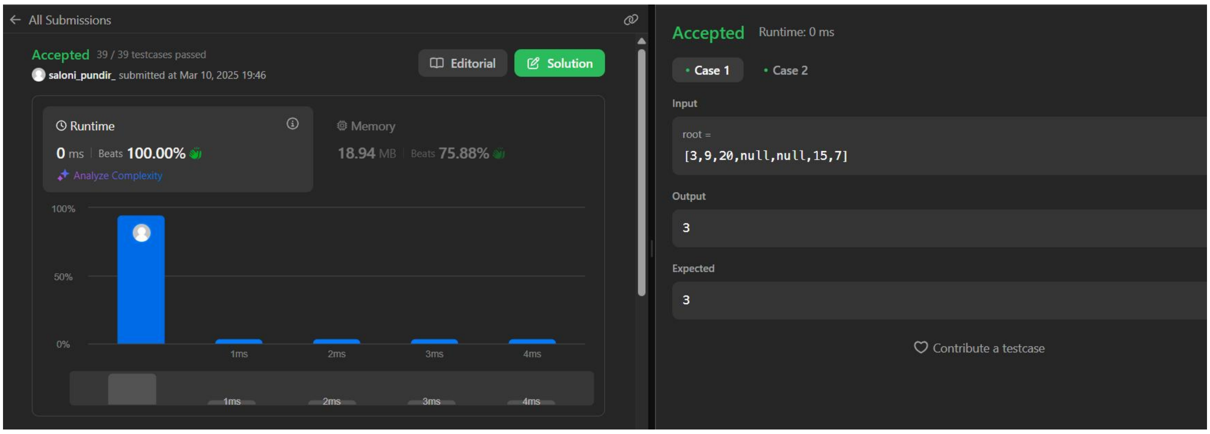
Q1. Maximum Depth of Binary Tree

Code:



```
 */
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if (!root) return 0;
        return 1 + max(maxDepth(root->left), maxDepth(root->right));

    }
};
```

Output:



Accepted 39 / 39 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:46

Runtime

0 ms | Beats 100.00%

Memory

18.94 MB | Beats 75.88%

Accepted Runtime: 0 ms

Input

```
root =
[3,9,20,null,null,15,7]
```

Output

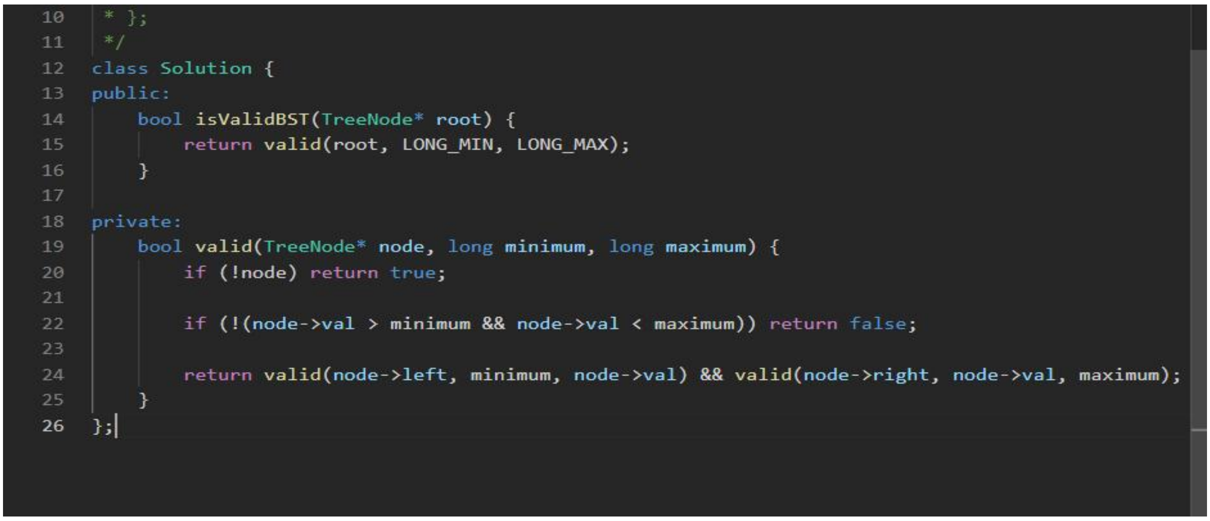
```
3
```

Expected

```
3
```

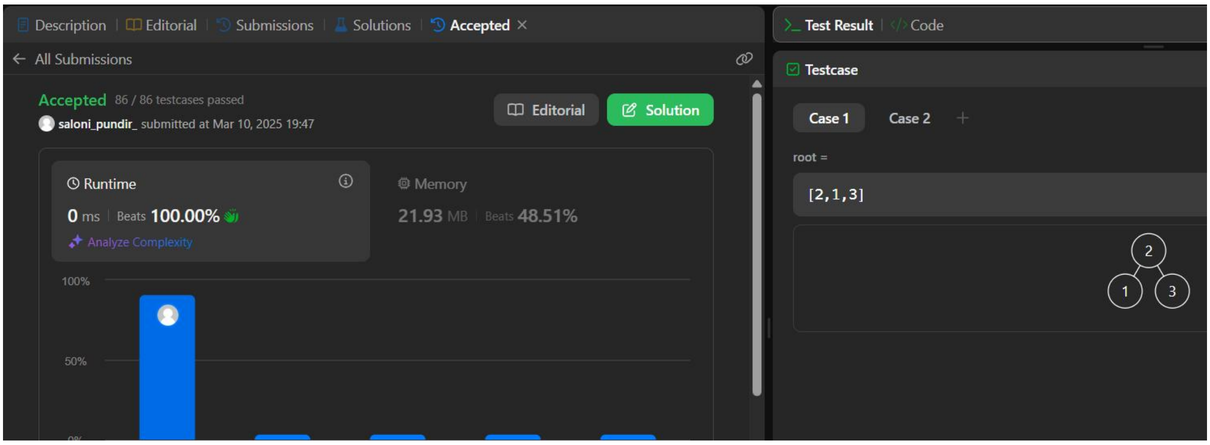
Q2. Validate Binary Search Tree

Code:



```
 * };
 */
class Solution {
public:
    bool isValidBST(TreeNode* root) {
        return valid(root, LONG_MIN, LONG_MAX);
    }

private:
    bool valid(TreeNode* node, long minimum, long maximum) {
        if (!node) return true;

        if (!(node->val > minimum && node->val < maximum)) return false;

        return valid(node->left, minimum, node->val) && valid(node->right, node->val, maximum);
    }
};
```

Output:



Accepted 86 / 86 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:47

Runtime

0 ms | Beats 100.00%

Memory

21.93 MB | Beats 48.51%

Testcase

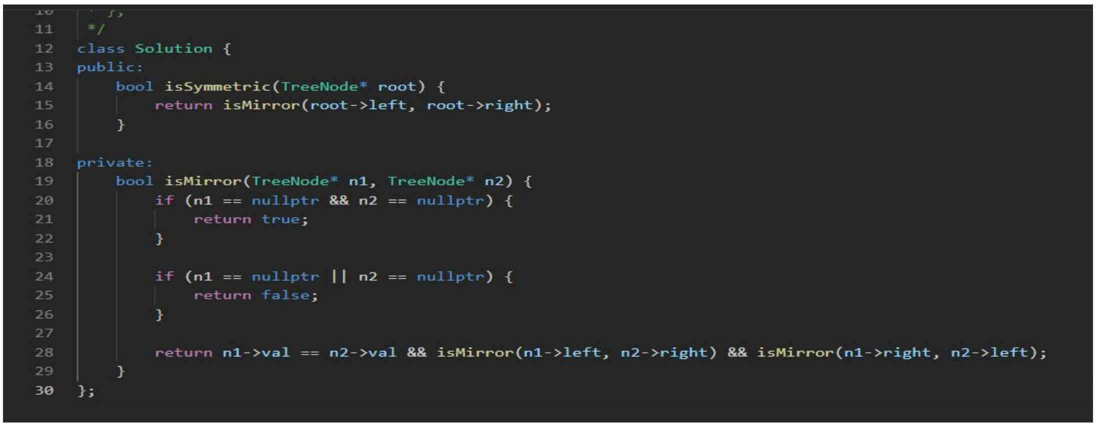
Case 1 Case 2 +

root =

[2,1,3]

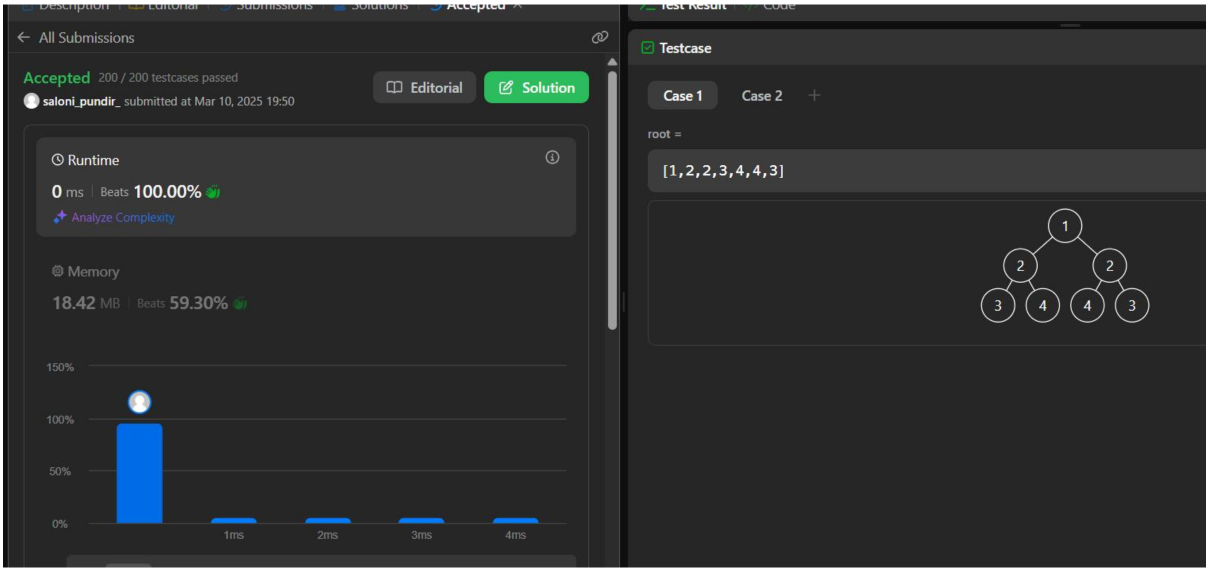
Q3. Symmetric tree

Code:



```
 */
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        return isMirror(root->left, root->right);
    }

private:
    bool isMirror(TreeNode* n1, TreeNode* n2) {
        if (n1 == nullptr && n2 == nullptr) {
            return true;
        }

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

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

Output:



Accepted 200 / 200 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:50

Runtime

0 ms | Beats 100.00%

Memory

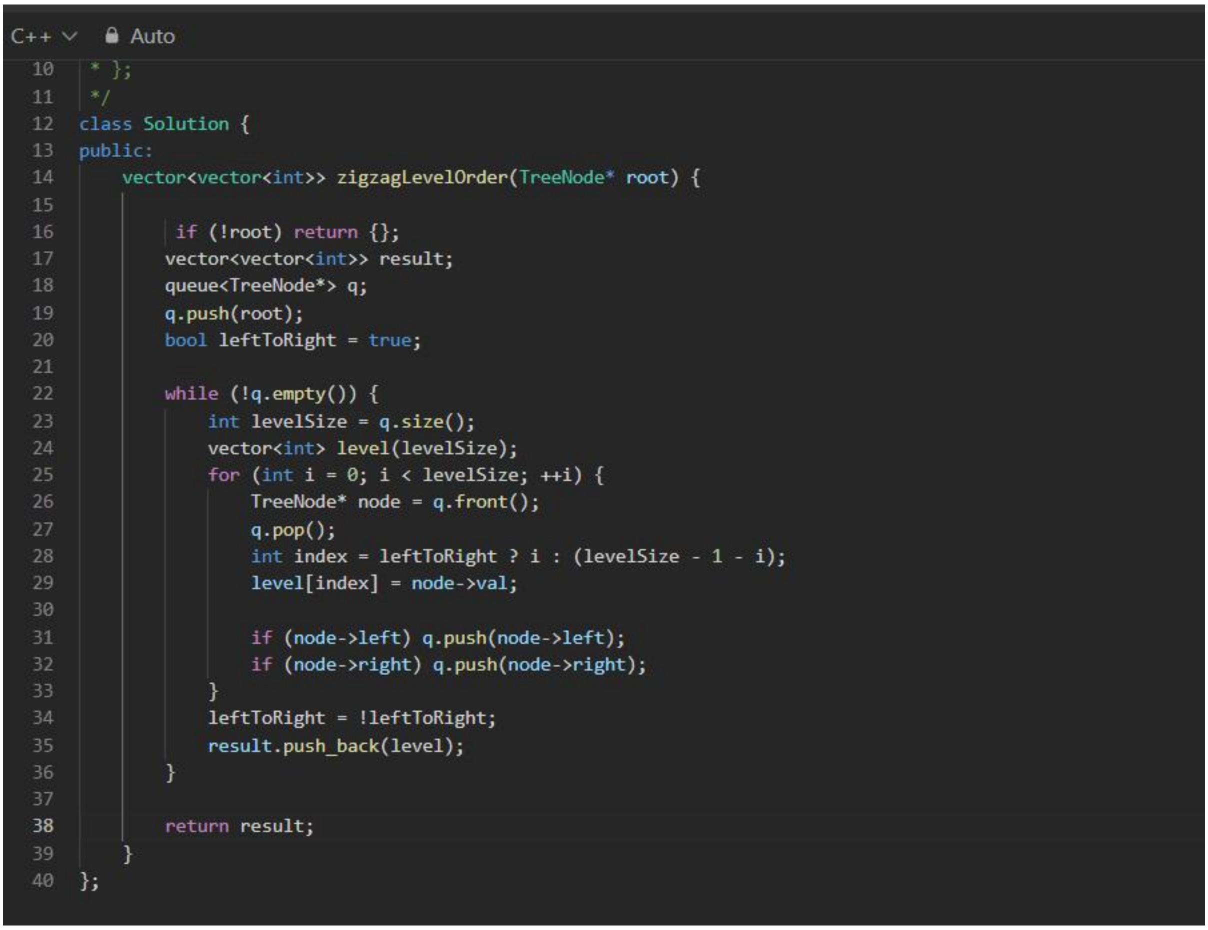
18.42 MB | Beats 59.30%

root =

```
[1,2,2,3,4,4,3]
```

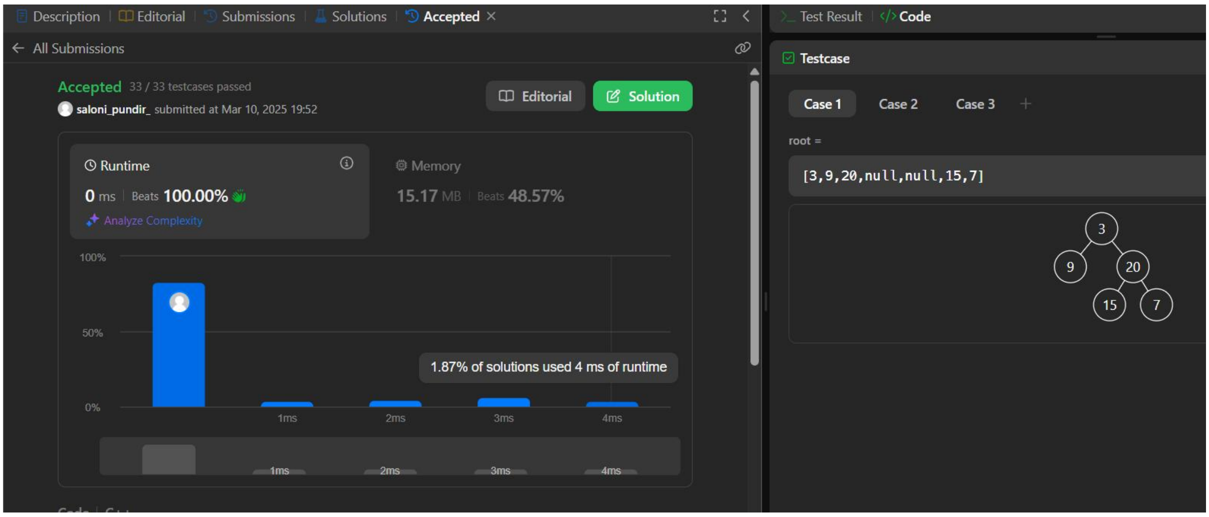
Q4. Binary Tree ZigZag Level Order Traversal

Code:



```cpp
 * };
 */
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {

        if (!root) return {};
        vector<vector<int>> result;
        queue<TreeNode*> q;
        q.push(root);
        bool leftToRight = true;

        while (!q.empty()) {
            int levelSize = q.size();
            vector<int> level(levelSize);
            for (int i = 0; i < levelSize; ++i) {
                TreeNode* node = q.front();
                q.pop();
                int index = leftToRight ? i : (levelSize - 1 - i);
                level[index] = node->val;

                if (node->left) q.push(node->left);
                if (node->right) q.push(node->right);
            }
            leftToRight = !leftToRight;
            result.push_back(level);
        }

        return result;
    }
};
```

Output:



Accepted 33 / 33 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:52

Runtime

0 ms | Beats 100.00%

Memory

15.17 MB | Beats 48.57%

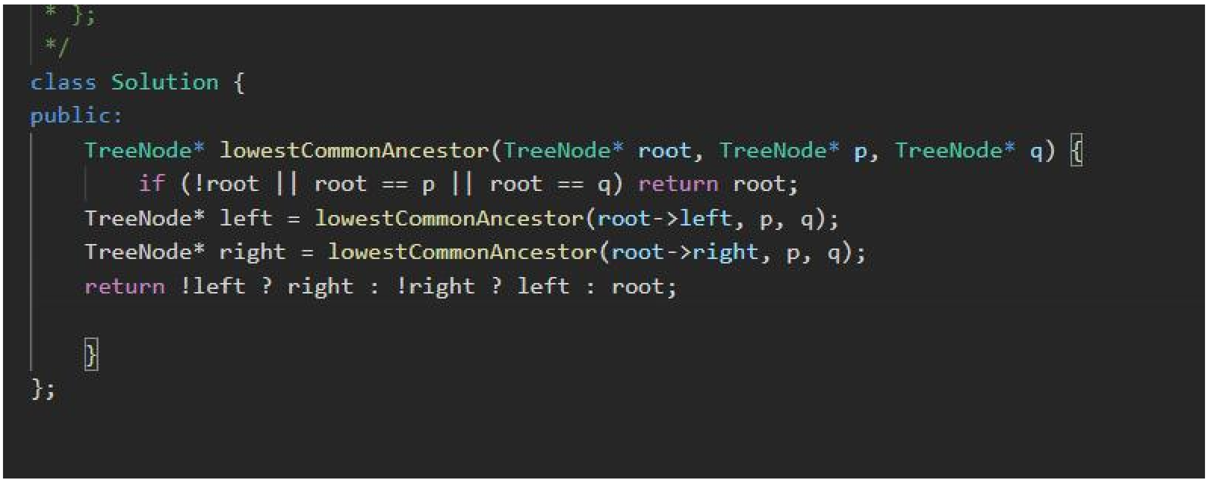
1.87% of solutions used 4 ms of runtime

root =

```
[3,9,20,null,null,15,7]
```

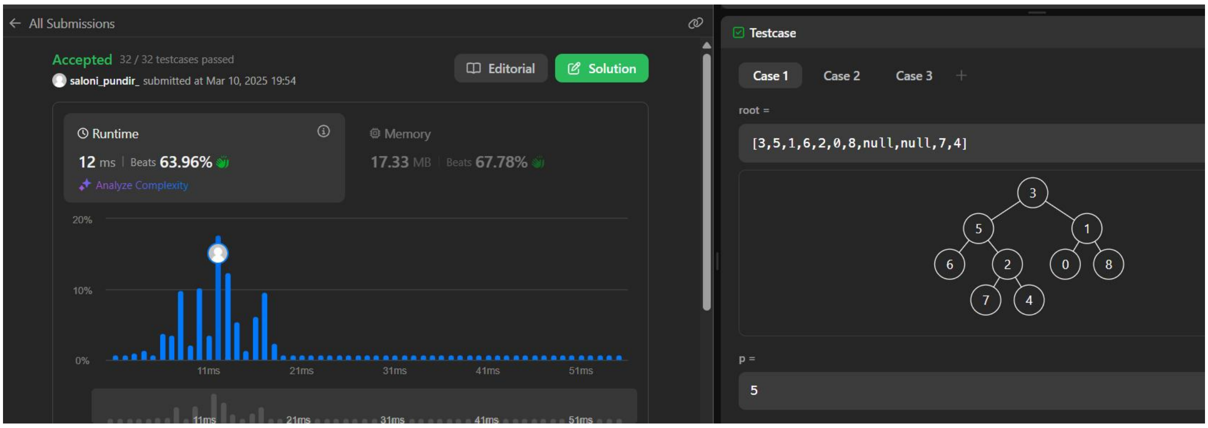
Q5. Lowest Common Ancestor of a Binary Tree

Code:



```
 * };
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if (!root || root == p || root == q) return root;
    TreeNode* left = lowestCommonAncestor(root->left, p, q);
    TreeNode* right = lowestCommonAncestor(root->right, p, q);
    return !left ? right : !right ? left : root;

    }
};
```

Output:



Accepted 32 / 32 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:54

Runtime

12 ms | Beats 63.96%

Memory

17.33 MB | Beats 67.78%

root =

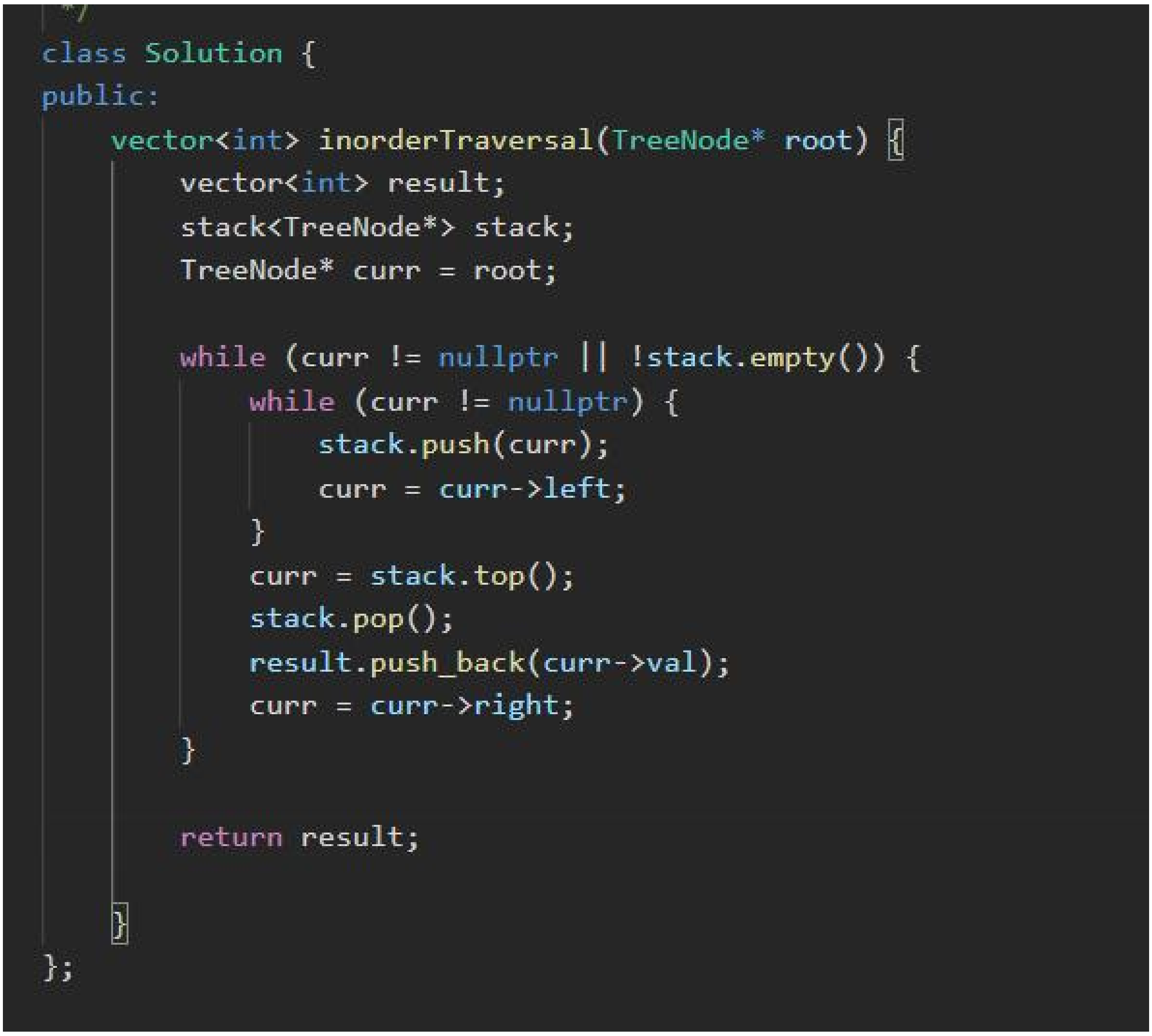
```
[3,5,1,6,2,0,8,null,null,7,4]
```

p =

```
5
```

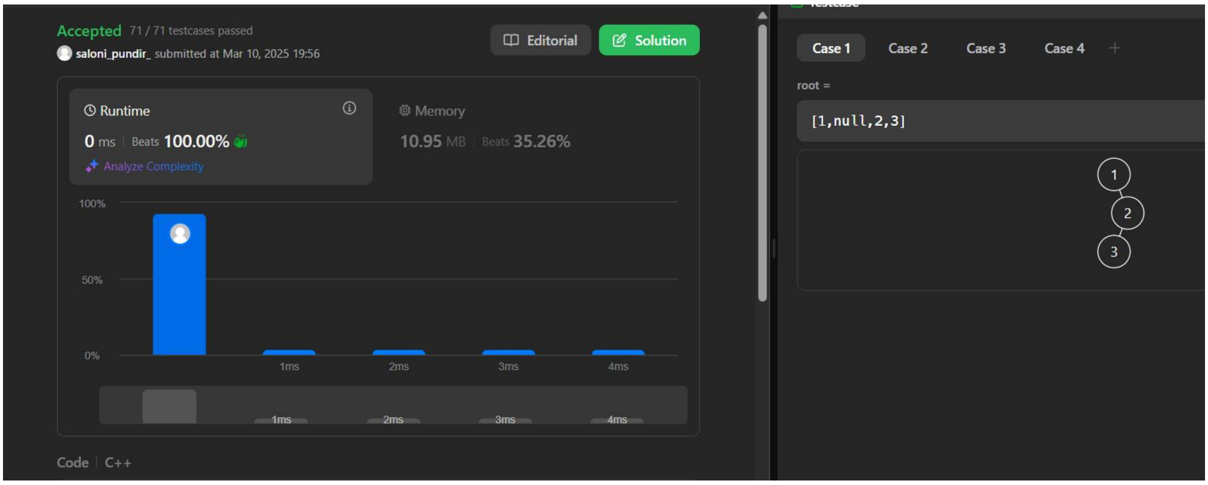
Q6. Binary Tree Inorder Traversal

Code:



```
 */
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> result;
        stack<TreeNode*> stack;
        TreeNode* curr = root;

        while (curr != nullptr || !stack.empty()) {
            while (curr != nullptr) {
                stack.push(curr);
                curr = curr->left;
            }
            curr = stack.top();
            stack.pop();
            result.push_back(curr->val);
            curr = curr->right;
        }

        return result;

    }
};
```

Output:



Accepted 71 / 71 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:56

Runtime

0 ms | Beats 100.00%

Memory

10.95 MB | Beats 35.26%

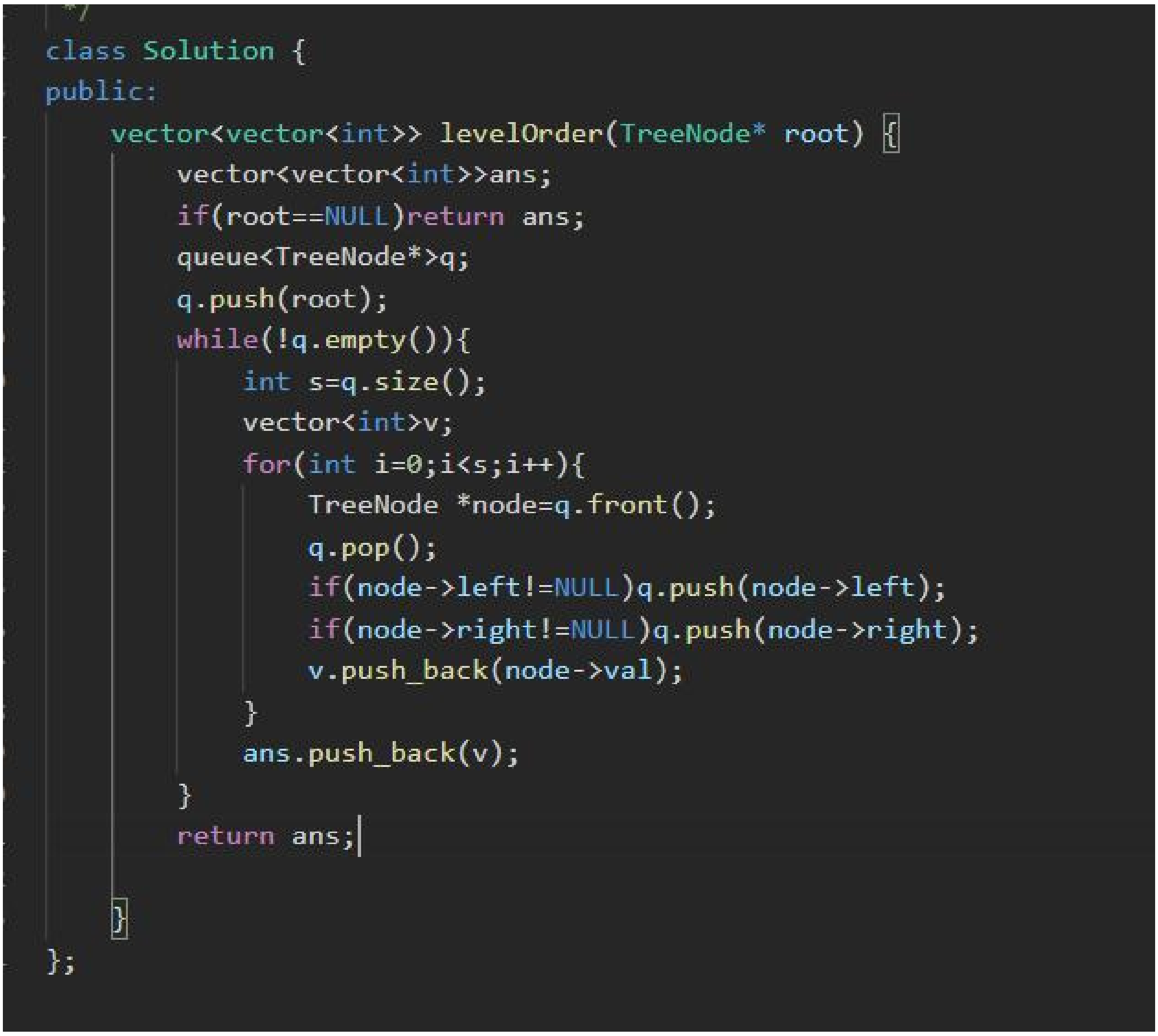
Code | C++

root =

[1,null,2,3]

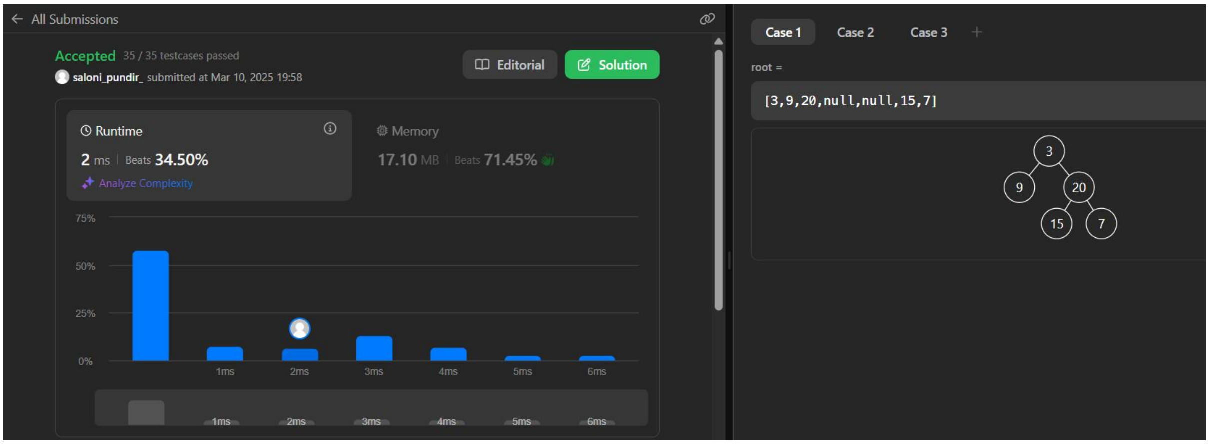
Q7. Binary Tree Level Order Traversal

Code:



```
 */
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>>ans;
        if(root==NULL)return ans;
        queue<TreeNode*>q;
        q.push(root);
        while(!q.empty()){
            int s=q.size();
            vector<int>v;
            for(int i=0;i<s;i++){
                TreeNode *node=q.front();
                q.pop();
                if(node->left!=NULL)q.push(node->left);
                if(node->right!=NULL)q.push(node->right);
                v.push_back(node->val);
            }
            ans.push_back(v);
        }
        return ans;

    }
};
```

Output:



All Submissions

Accepted 35 / 35 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 19:58

Runtime

2 ms | Beats 34.50%

Memory

17.10 MB | Beats 71.45%

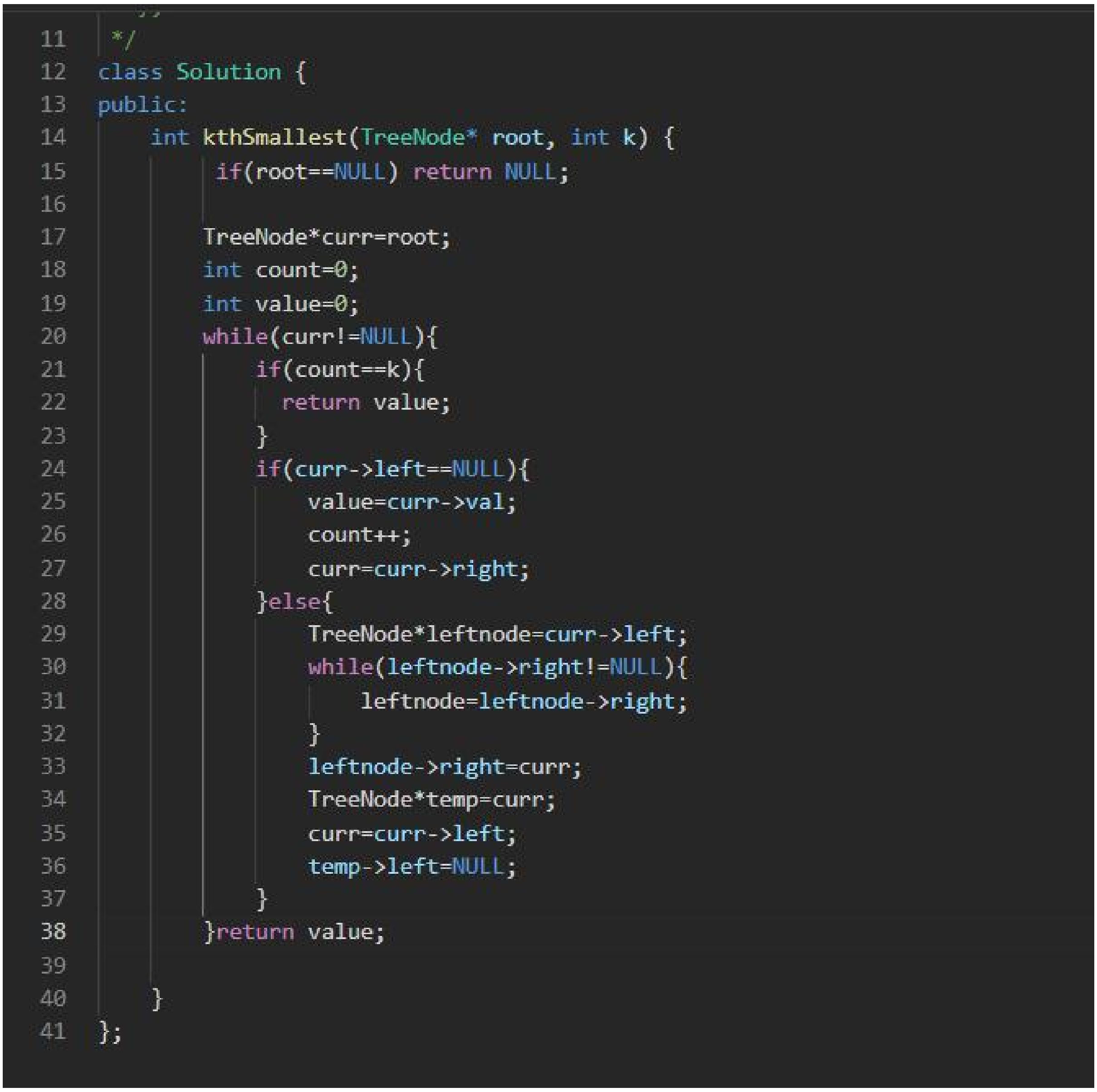
Case 1 Case 2 Case 3

root =

[3,9,20,null,null,15,7]

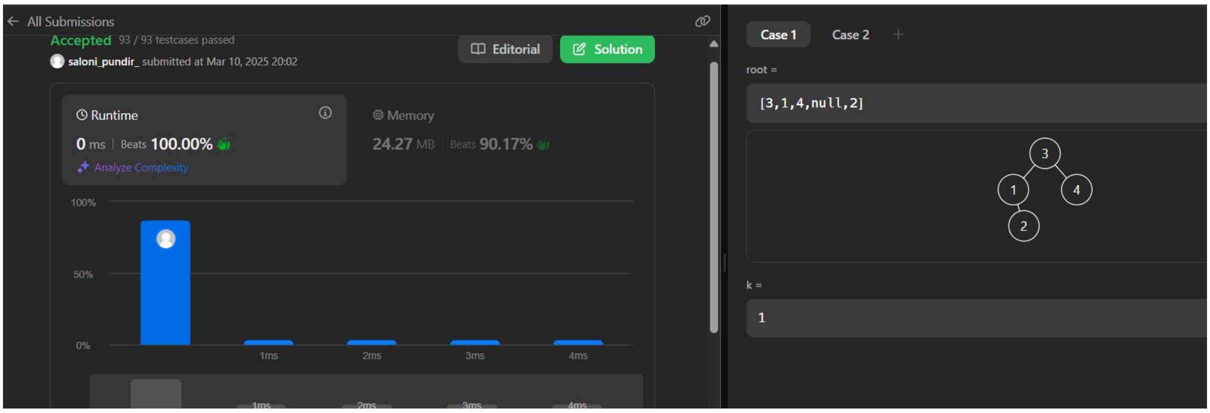
Q8. kth Sammest Element in a BST.

Code:



```cpp
 */
class Solution {
public:
    int kthSmallest(TreeNode* root, int k) {
         if(root==NULL) return NULL;

        TreeNode*curr=root;
        int count=0;
        int value=0;
        while(curr!=NULL){
            if(count==k){
              return value;
            }
            if(curr->left==NULL){
                value=curr->val;
                count++;
                curr=curr->right;
            }else{
                TreeNode*leftnode=curr->left;
                while(leftnode->right!=NULL){
                    leftnode=leftnode->right;
                }
                leftnode->right=curr;
                TreeNode*temp=curr;
                curr=curr->left;
                temp->left=NULL;
            }
        }return value;

    }
};
```

Output:



Accepted 93 / 93 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 20:02

Runtime
0 ms | Beats 100.00%

Memory
24.27 MB Beats 90.17%

root =

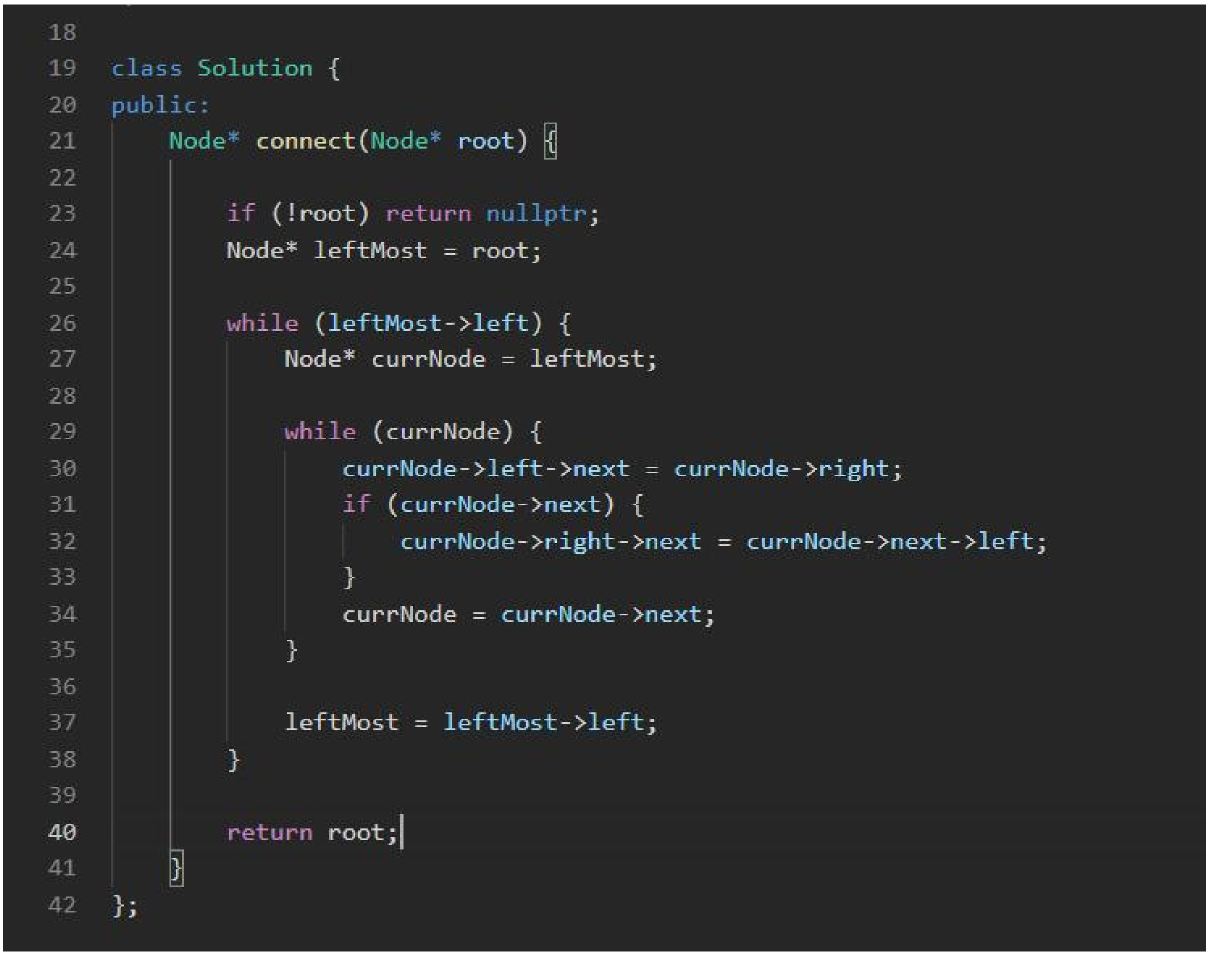
[3,1,4,null,2]

k =

1

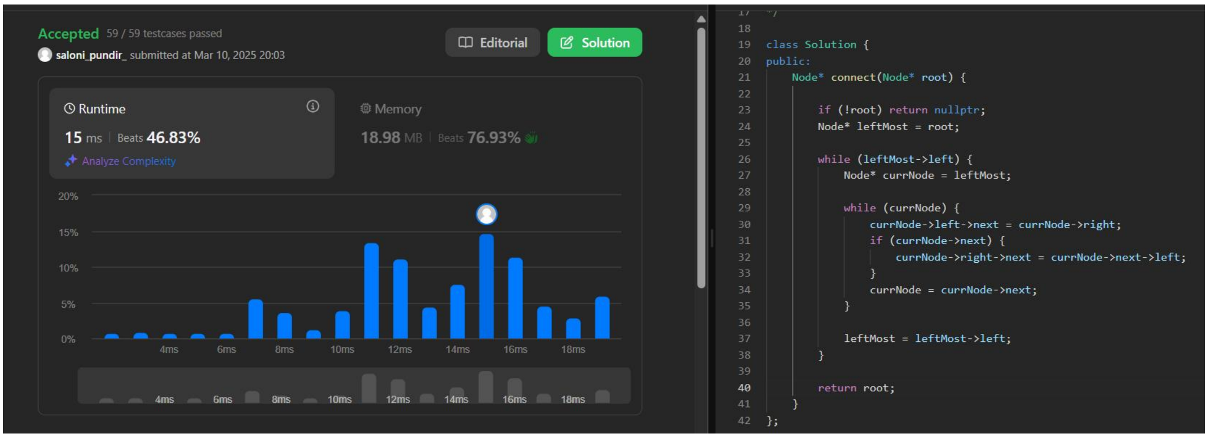
Q9. Populating Next Right Pointers in Each Node

Code:



```
class Solution {
public:
    Node* connect(Node* root) {

        if (!root) return nullptr;
        Node* leftMost = root;

        while (leftMost->left) {
            Node* currNode = leftMost;

            while (currNode) {
                currNode->left->next = currNode->right;
                if (currNode->next) {
                    currNode->right->next = currNode->next->left;
                }
                currNode = currNode->next;
            }

            leftMost = leftMost->left;
        }

        return root;
    }
};
```

Output:



Accepted 59 / 59 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 20:03

Runtime

15 ms | Beats 46.83%

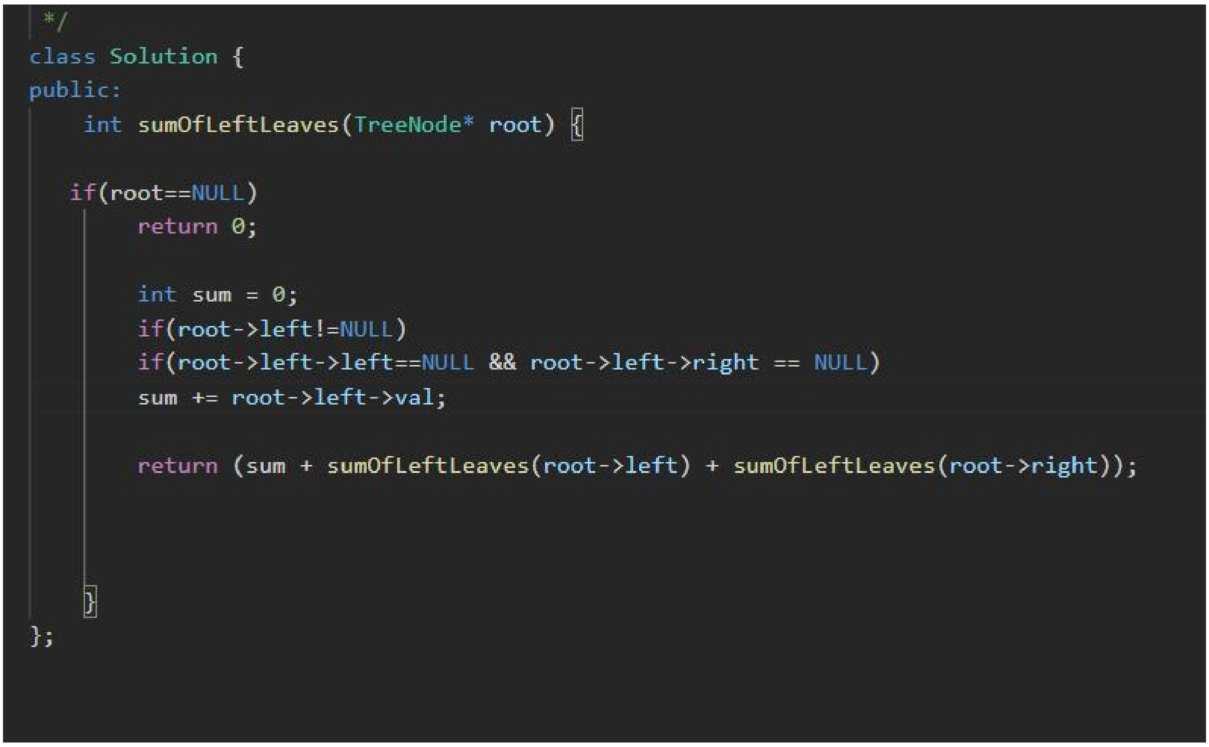
Memory

18.98 MB | Beats 76.93%

```
class Solution {
public:
    Node* connect(Node* root) {

        if (!root) return nullptr;
        Node* leftMost = root;

        while (leftMost->left) {
            Node* currNode = leftMost;

            while (currNode) {
                currNode->left->next = currNode->right;
                if (currNode->next) {
                    currNode->right->next = currNode->next->left;
                }
                currNode = currNode->next;
            }

            leftMost = leftMost->left;
        }

        return root;
    }
};
```

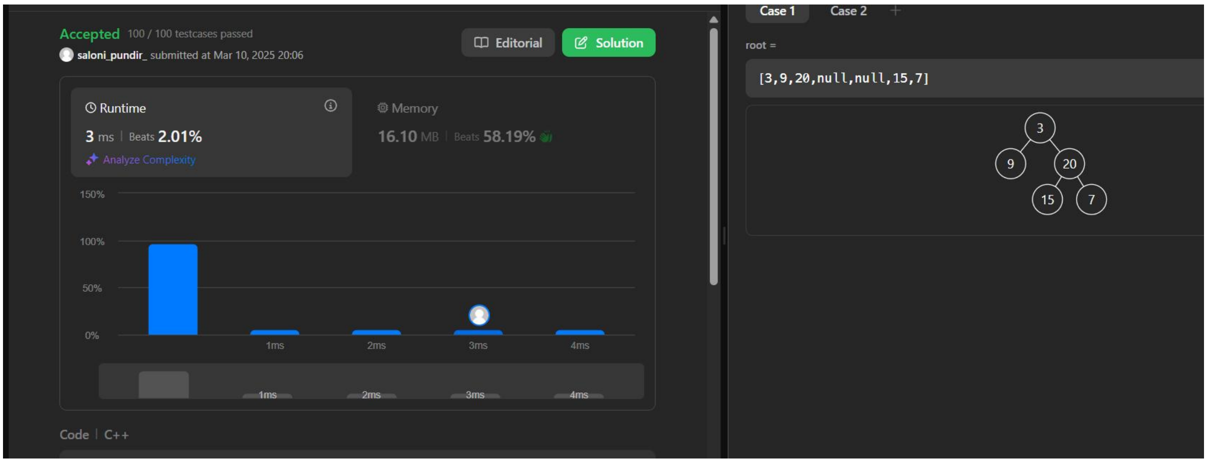
Q10. Sum of Left Leaves

Code:



```
 */
class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {

   if(root==NULL)
        return 0;

        int sum = 0;
        if(root->left!=NULL)
        if(root->left->left==NULL && root->left->right == NULL)
        sum += root->left->val;

        return (sum + sumOfLeftLeaves(root->left) + sumOfLeftLeaves(root->right));



    }
};
```

Output:



Accepted 100 / 100 testcases passed

saloni_pundir_ submitted at Mar 10, 2025 20:06

Runtime

3 ms | Beats 2.01%

Memory

16.10 MB | Beats 58.19%

Code | C++

Case 1 Case 2

root =

```
[3,9,20,null,null,15,7]
```