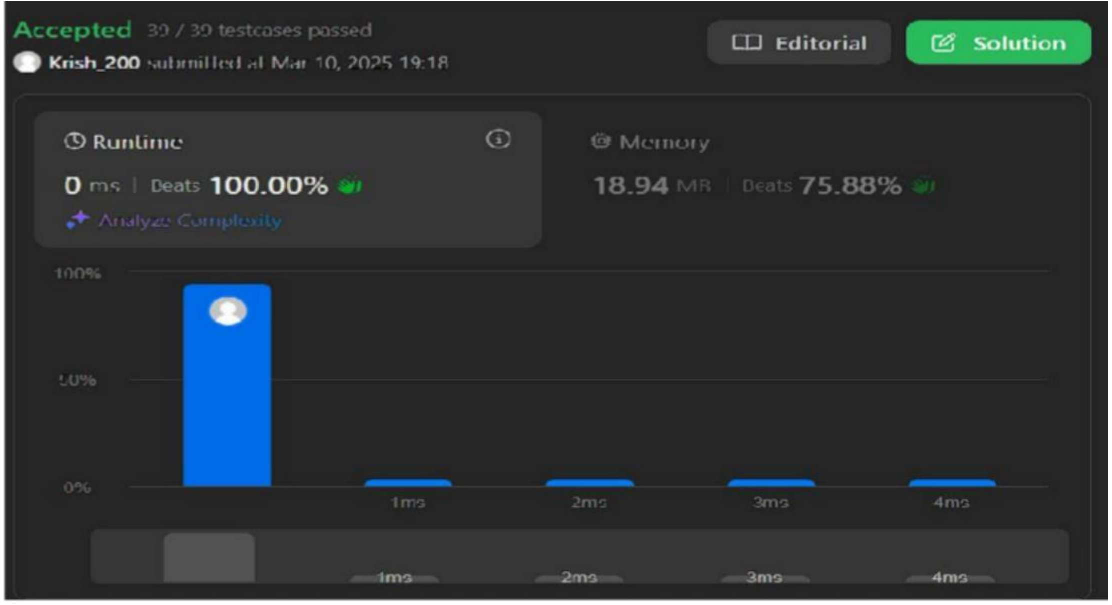
Name - Aditya Chaurasia, UID - 22BCS11655
Stream - BE-CSE, Subject - Advance Programming

1. Maximum Depth of Binary Tree

```
 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
Krish_200 submitted at Mar 10, 2025 19:18

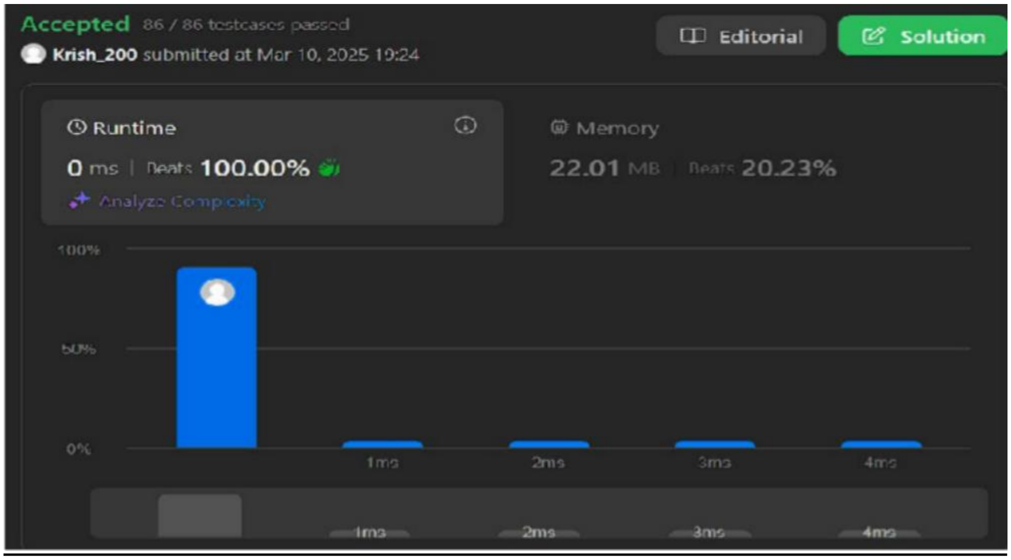
Runtime
0 ms | Beats 100.00%

Memory
18.94 MB | Beats 75.88%

2. Validate Binary Search Tree

```
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

Krish_200 submitted at Mar 10, 2025 19:24

Runtime

0 ms | Beats 100.00%

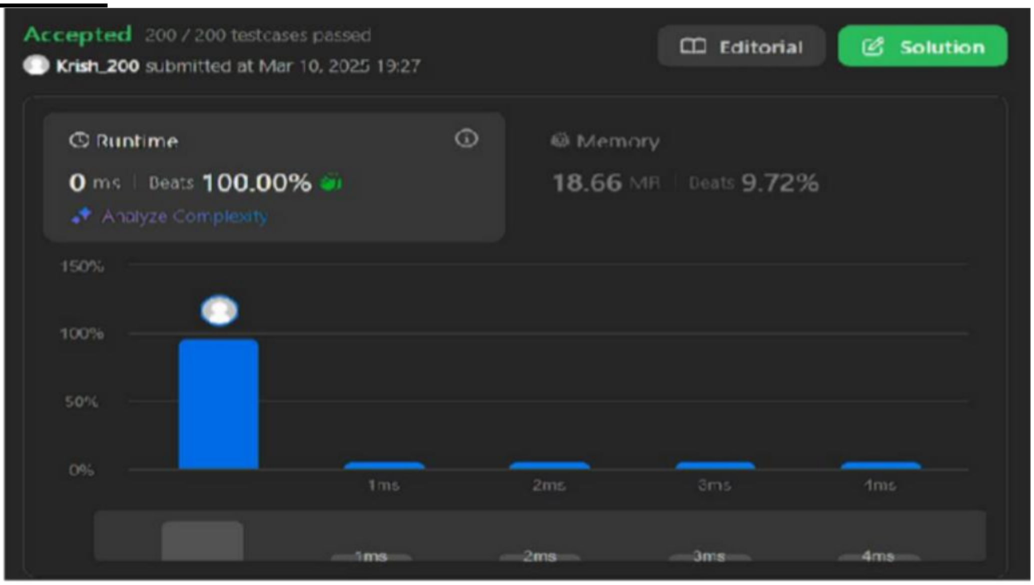
Memory

22.01 MB | Beats 20.23%

3. Symmetric Tree

```
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

Krish_200 submitted at Mar 10, 2025 19:27

Runtime

0 ms | Beats 100.00%

Memory

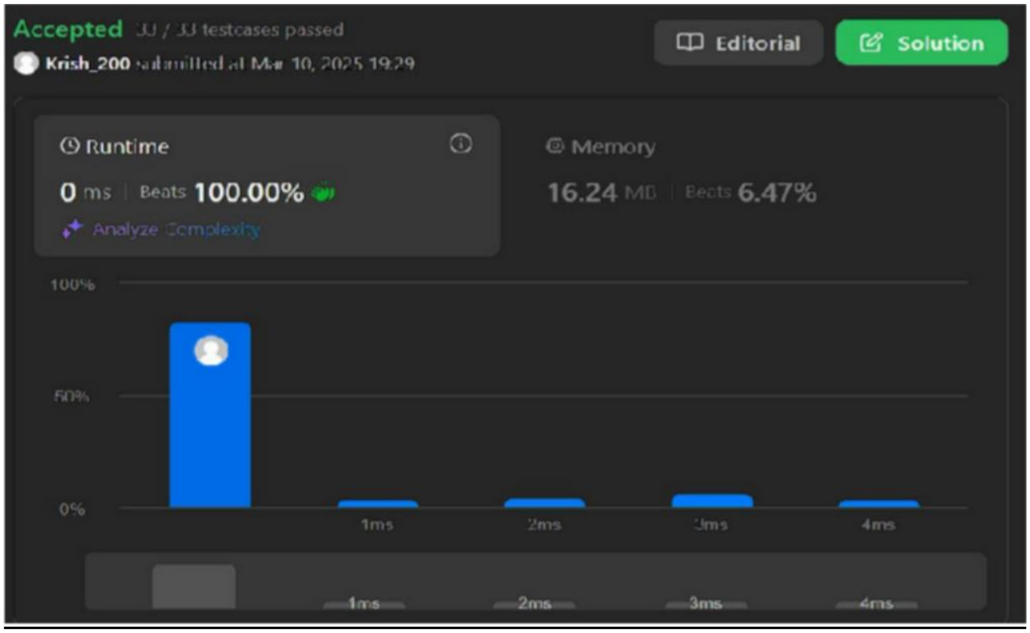
18.66 MB | Beats 9.72%

4. Binary Tree Zigzag Level Order Traversal

```
class Solution {

public:
```

```
vector<vector<int>> res;

void doLevelOrderTraversal(queue<TreeNode*> qu, bool alternate) {

    if (qu.empty()) {

        return;

    }

    queue<TreeNode*> newQu;

    vector<int> v;

    while (!qu.empty()) {

        TreeNode* ptr = qu.front();

        qu.pop();

        if (ptr->left) {

            newQu.push(ptr->left);

        }

        if (ptr->right) {

            newQu.push(ptr->right);

        }

        v.push_back(ptr->val);

    }

    if (alternate) {

        reverse(v.begin(), v.end());

    }

    res.push_back(v);

    doLevelOrderTraversal(newQu, !alternate);

}

vector<vector<int>> zigzagLevelOrder(TreeNode* root) {

    if (root == NULL) {

        return res;
        }
        queue<TreeNode*> qu;
        qu.push(root);
        doLevelOrderTraversal(qu, false);
        return res;
    }
};
```

OUTPUT:



Accepted 33 / 33 testcases passed

Krish_200 submitted at Mar 10, 2025 19:29

Runtime

0 ms | Beats 100.00%

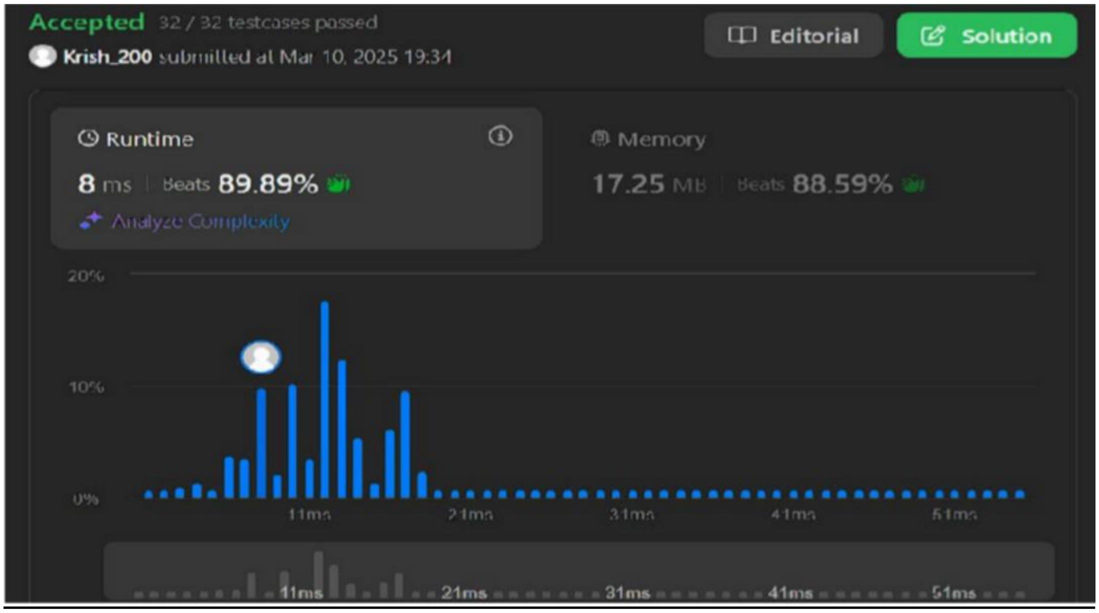
Memory

16.24 MB | Beats 6.47%

5. Lowest Common Ancestor of a Binary Tree

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

        if (left != nullptr && right != nullptr) {

            return root;

        }

        return left != nullptr ? left : right;

    }

};
```

OUTPUT:



Accepted 32 / 32 testcases passed

Krish_200 submitted at Mar 10, 2025 19:34

Runtime

8 ms | Beats 89.89%

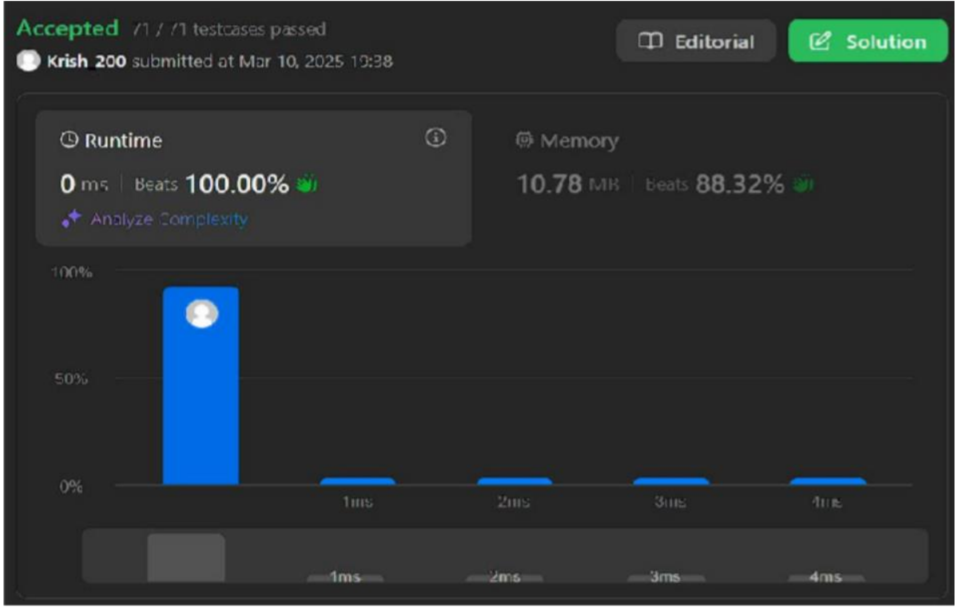
Memory

17.25 MB | Beats 88.59%

6. Binary Tree Inorder Traversal

```
class Solution {

public:

    vector<int> inorderTraversal(TreeNode* root) {

        vector<int> res;

        inorder(root, res);

        return res;

    }

private:
    void inorder(TreeNode* node, vector<int>& res) {
        if (!node) {
            return;
        }
        inorder(node->left, res);
        res.push_back(node->val);
        inorder(node->right, res);
    }
};
```

OUTPUT:



Accepted 71 / 71 testcases passed

Krish_200 submitted at Mar 10, 2025 19:38

Runtime

0 ms | Beats 100.00%

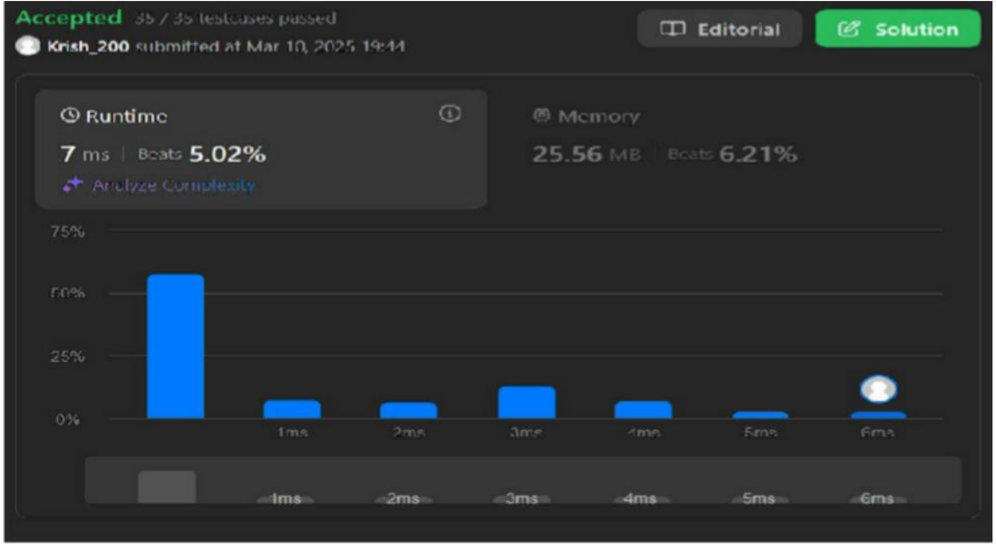
Memory

10.78 MB | Beats 88.32%

7. Binary Tree Level Order Traversal

```
class Solution {
public:
    vector<vector<int>> res;
    void doLevelOrderTraversal(queue<TreeNode*> qu) {
        if (qu.empty()) {
            return;
        }
        queue<TreeNode*> newQu;
        vector<int> v;
        while (!qu.empty()) {
            TreeNode* ptr = qu.front();
            qu.pop();
            if (ptr->left)
                newQu.push(ptr->left);
            if (ptr->right)
                newQu.push(ptr->right);
            v.push_back(ptr->val);
        }
        res.push_back(v);
        doLevelOrderTraversal(newQu);
    }
    vector<vector<int>> levelOrder(TreeNode* root) {
        if (root == NULL) {
            return res;
        }
        queue<TreeNode*> qu;
        qu.push(root);
        doLevelOrderTraversal(qu);
        return res;
    }
};
```

OUTPUT:



Accepted 35 / 35 testcases passed

Krish_200 submitted at Mar 10, 2025 19:44

Runtime

7 ms | Beats 5.02%

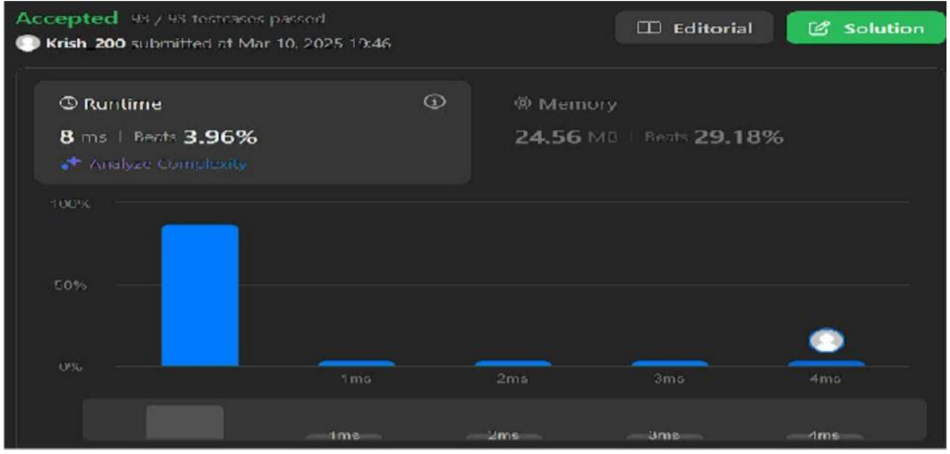
Memory

25.56 MB | Beats 6.21%

8. Kth Smallest Element in a BST

```
class Solution {
public:
    void preOrderTraversal(TreeNode* root, vector<int> &v){
        if(root == NULL)   return;
        //root, left, right
        v.push_back(root->val);
        preOrderTraversal(root->left, v);
        preOrderTraversal(root->right,  v);
    }
    int kthSmallest(TreeNode* root, int k) {
        vector<int> v;
        preOrderTraversal(root, v);
        sort(v.begin(), v.end());
        return v[k-1];
    }
};
```

OUTPUT:



Accepted 93 / 93 testcases passed

Krish_200 submitted at Mar 10, 2025 10:46

Runtime

8 ms | Beats 3.96%

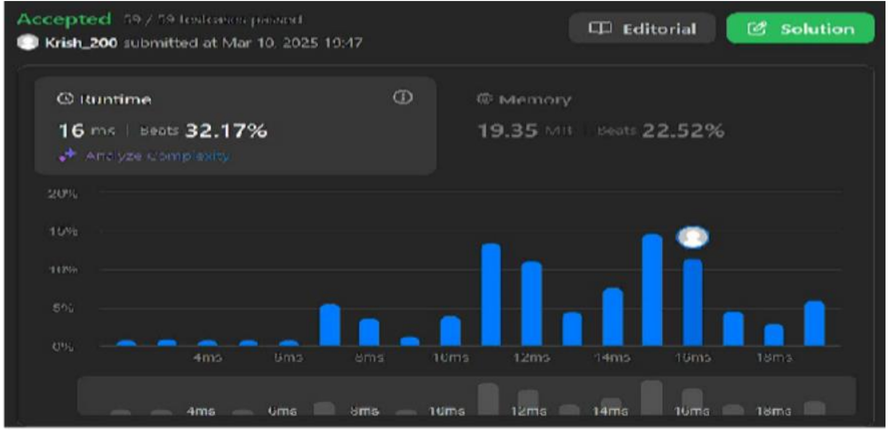
Memory

24.56 MB | Beats 29.18%

9. Populating Next Right Pointers in Each Node

```
class Solution {
public:
    Node* connect(Node* root) {
        if(!root) return nullptr;
        queue<Node*> q;
        q.push(root);
        while(size(q)) {
            Node* rightNode = nullptr;
            for(int i = size(q); i; i--) {
                auto cur = q.front(); q.pop();
                cur -> next = rightNode;
                rightNode = cur;
                if(cur -> right)
                    q.push(cur -> right),
                    q.push(cur -> left);
            }
        }
        return root;
    }
};
```

OUTPUT:



Accepted 59 / 59 testcases passed

Krish_200 submitted at Mar 10, 2025 10:47

Runtime

16 ms | Beats 32.17%

Memory

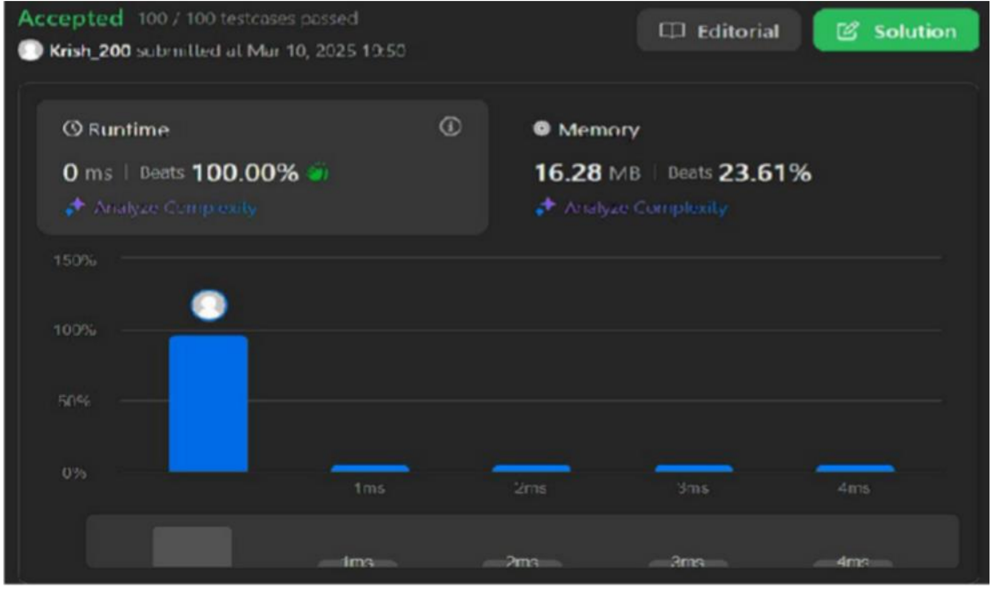
19.35 MB | Beats 22.52%

10. Sum of Left Leaves

```
class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {
        if (!root) {
            return 0;
        }
        queue<pair<TreeNode*, bool>> q; // (node, is_left)
        q.push({root, false});
        int totalSum = 0;
        while (!q.empty()) {
            auto [node, isLeft] = q.front();
            q.pop();
            if (isLeft && !node->left && !node->right) {
                totalSum += node->val;
            }
            if (node->left) {
                q.push({node->left, true});
            }
```

```
                if (node->right) {
                    q.push({node->right, false});
                }
            }
            return totalSum;
        }
    };
```

OUTPUT:



Accepted 100 / 100 testcases passed

Krish_200 submitted at Mar 10, 2025 19:50

Runtime

0 ms | Beats 100.00%

Memory

16.28 MB | Beats 23.61%