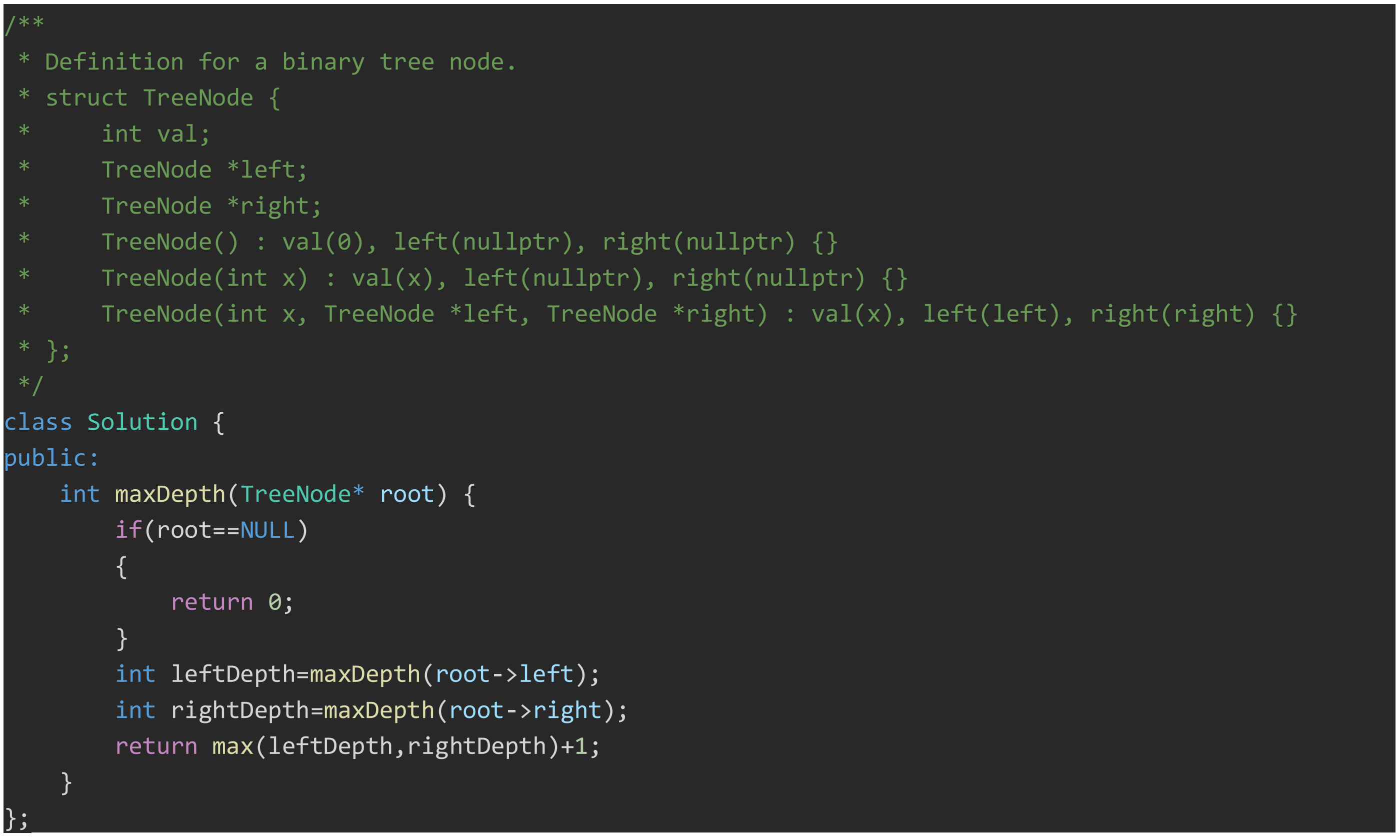
DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

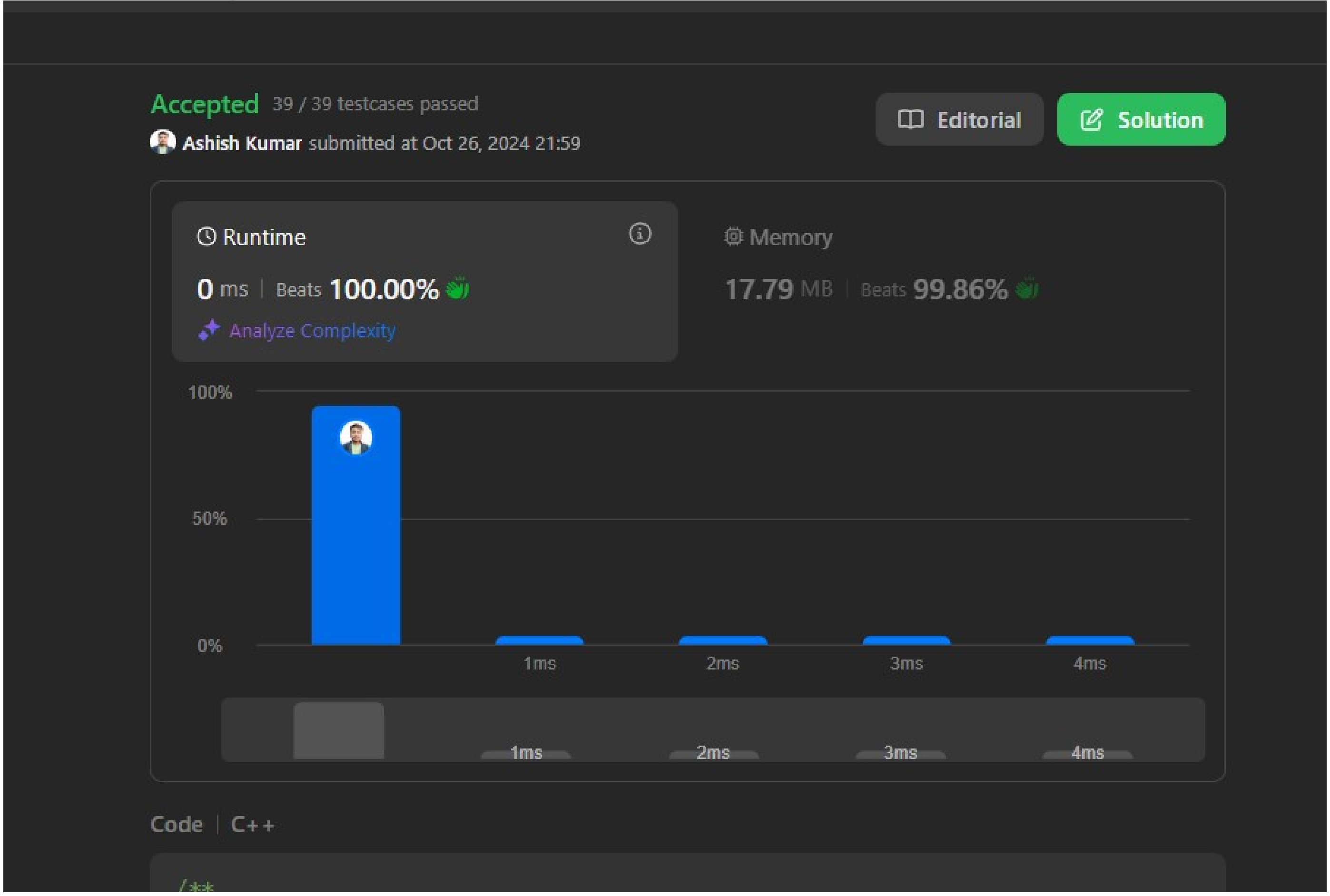
CU CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Experiment 5

Student Name: Ashish Kumar	**UID:22BCS11958**
Branch: CSE	**Section/Group:614(B)**
Semester: 6	**Date of Performance:10/03/25**
Subject Name: AP Lab	**Subject Code: 22CSP-351**



```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if(root==NULL)
        {
            return 0;
        }
        int leftDepth=maxDepth(root->left);
        int rightDepth=maxDepth(root->right);
        return max(leftDepth,rightDepth)+1;
    }
};
```



Accepted 39 / 39 testcases passed

Ashish Kumar submitted at Oct 26, 2024 21:59

Runtime

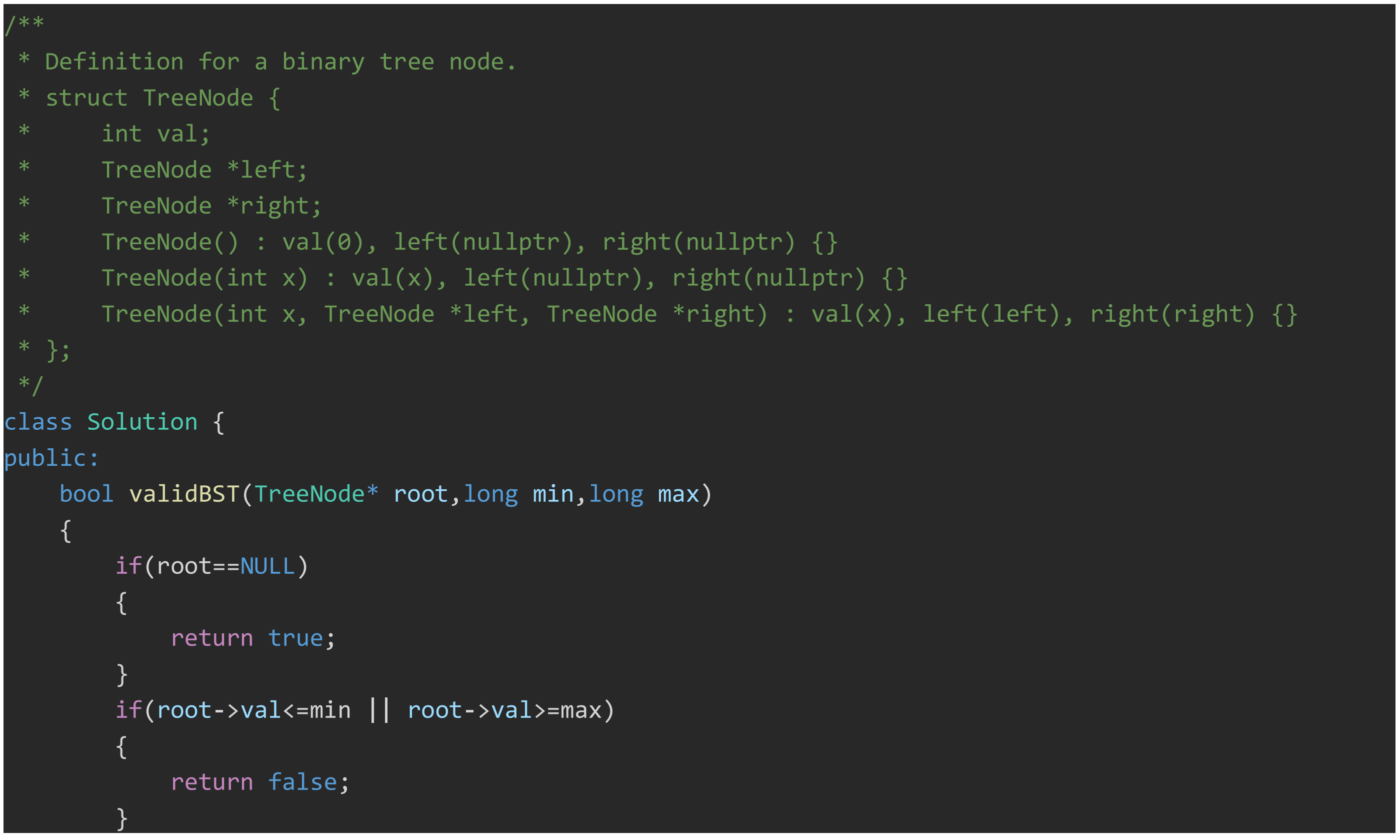
0 ms | Beats 100.00%

Memory

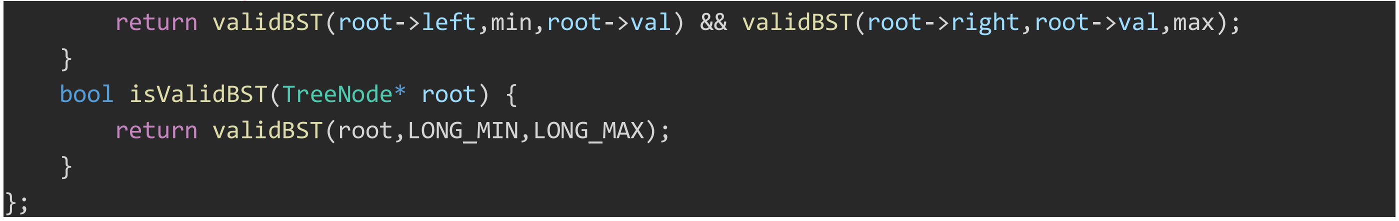
17.79 MB | Beats 99.86%

Code | C++

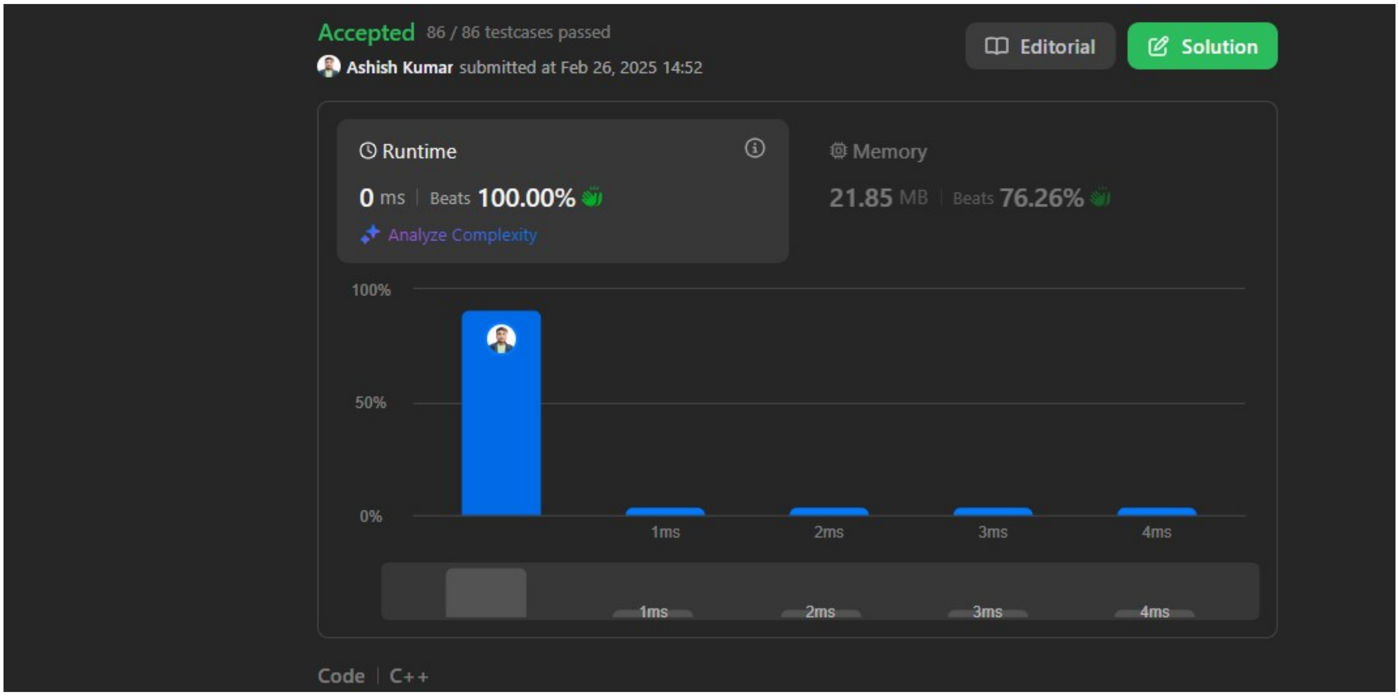
```
/**
```



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool validBST(TreeNode* root,long min,long max)
    {
        if(root==NULL)
        {
            return true;
        }
        if(root->val<=min || root->val>=max)
        {
            return false;
        }
```



```
        return validBST(root->left,min,root->val) && validBST(root->right,root->val,max);
    }
    bool isValidBST(TreeNode* root) {
        return validBST(root,LONG_MIN,LONG_MAX);
    }
};
```



Accepted 86 / 86 testcases passed

Ashish Kumar submitted at Feb 26, 2025 14:52

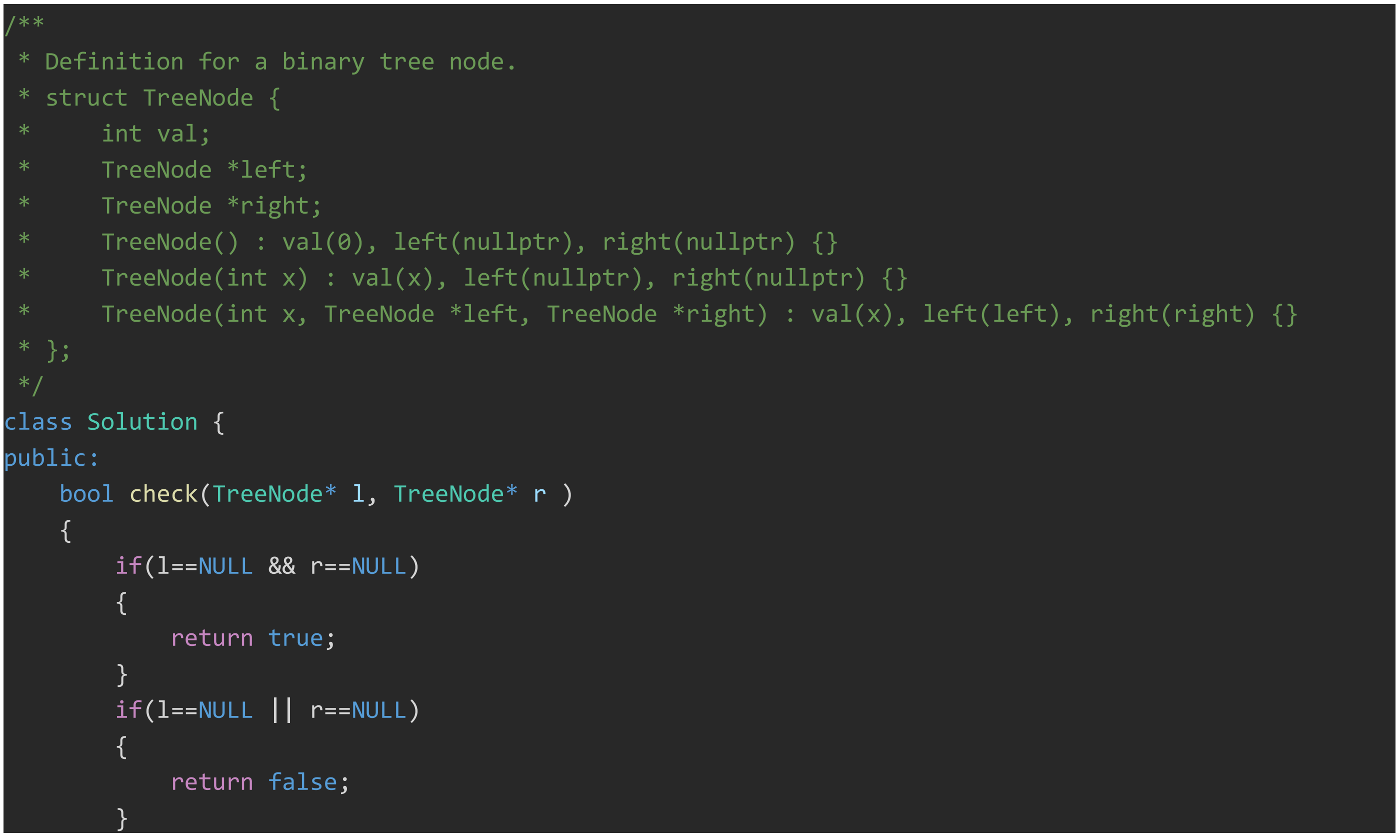
Runtime

0 ms | Beats 100.00%

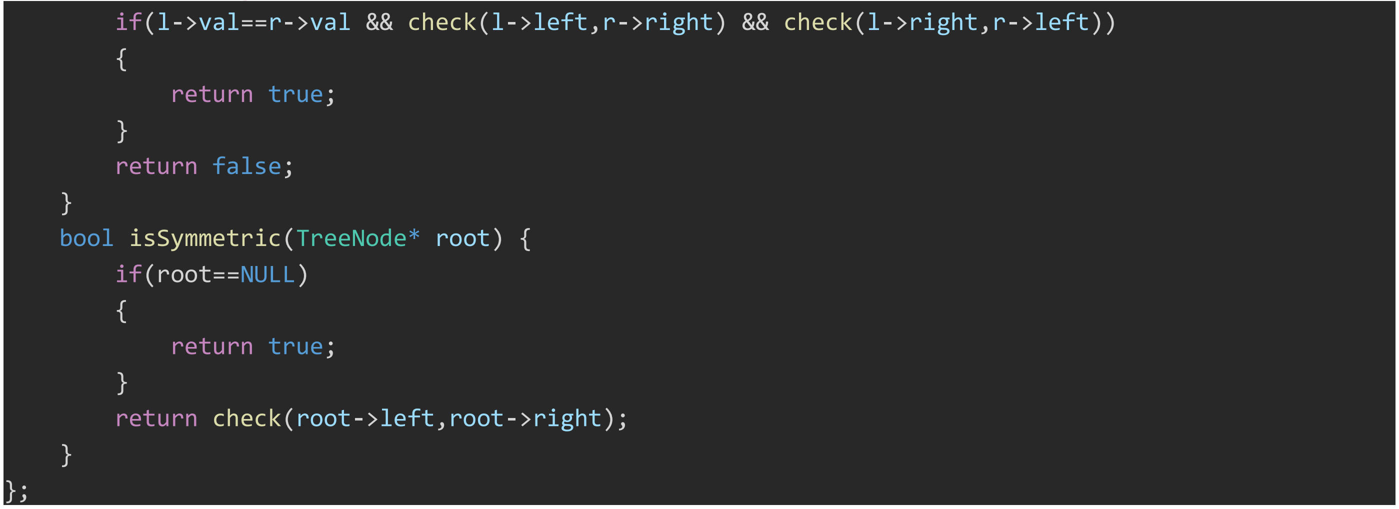
Memory

21.85 MB | Beats 76.26%

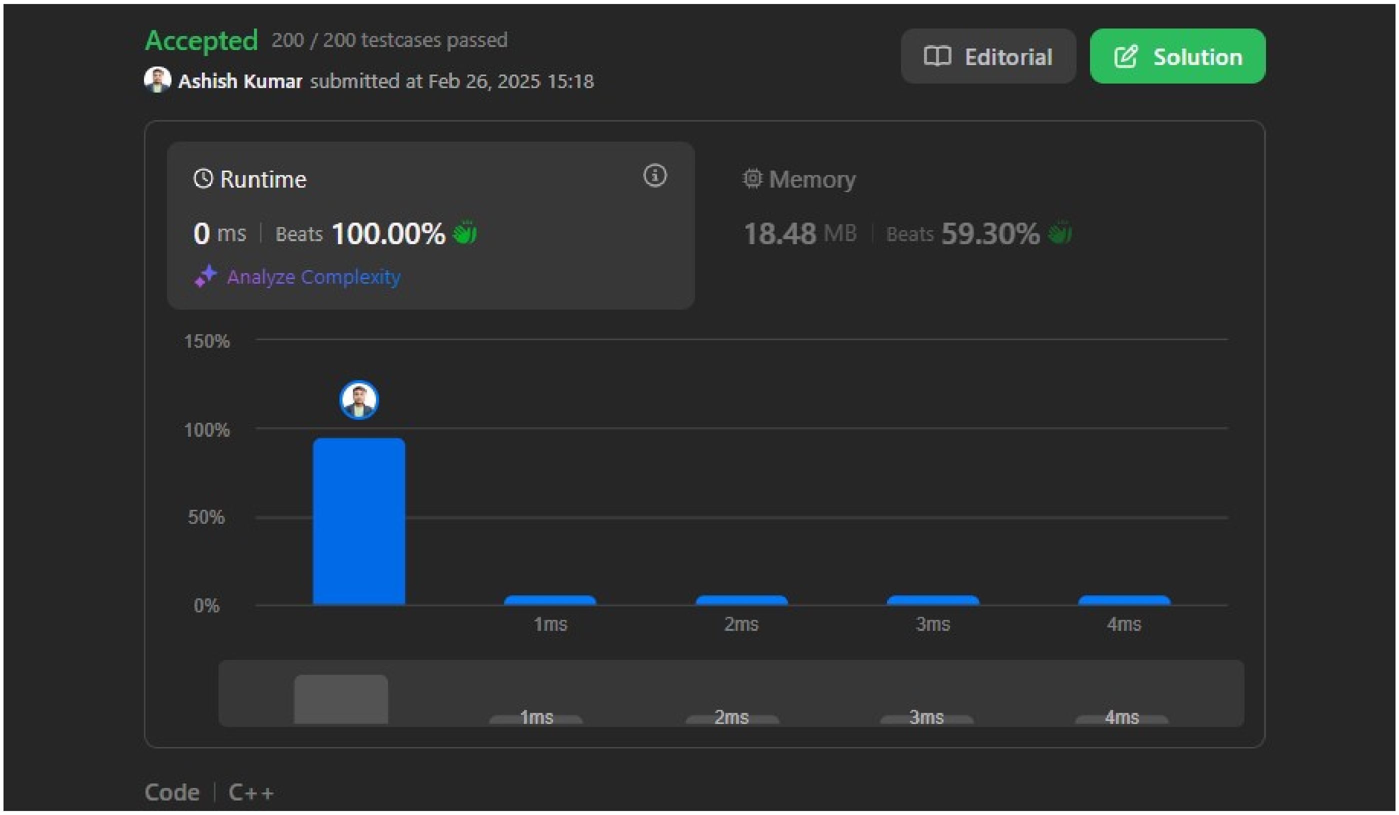
Code | C++



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool check(TreeNode* l, TreeNode* r )
    {
        if(l==NULL && r==NULL)
        {
            return true;
        }
        if(l==NULL || r==NULL)
        {
            return false;
        }
```



```
        if(l->val==r->val && check(l->left,r->right) && check(l->right,r->left))
        {
            return true;
        }
        return false;
    }
    bool isSymmetric(TreeNode* root) {
        if(root==NULL)
        {
            return true;
        }
        return check(root->left,root->right);
    }
};
```



Accepted 200 / 200 testcases passed

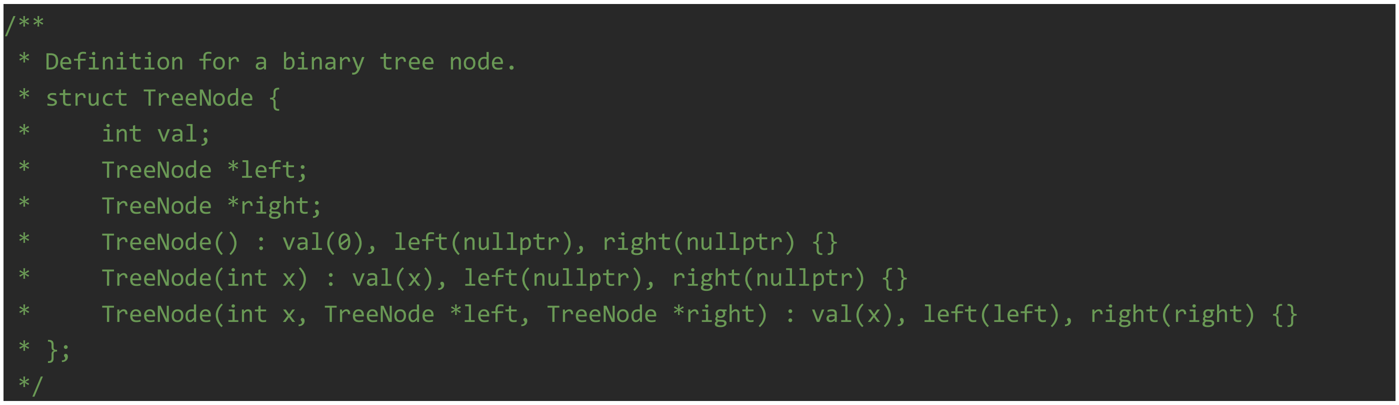
Ashish Kumar submitted at Feb 26, 2025 15:18

Runtime

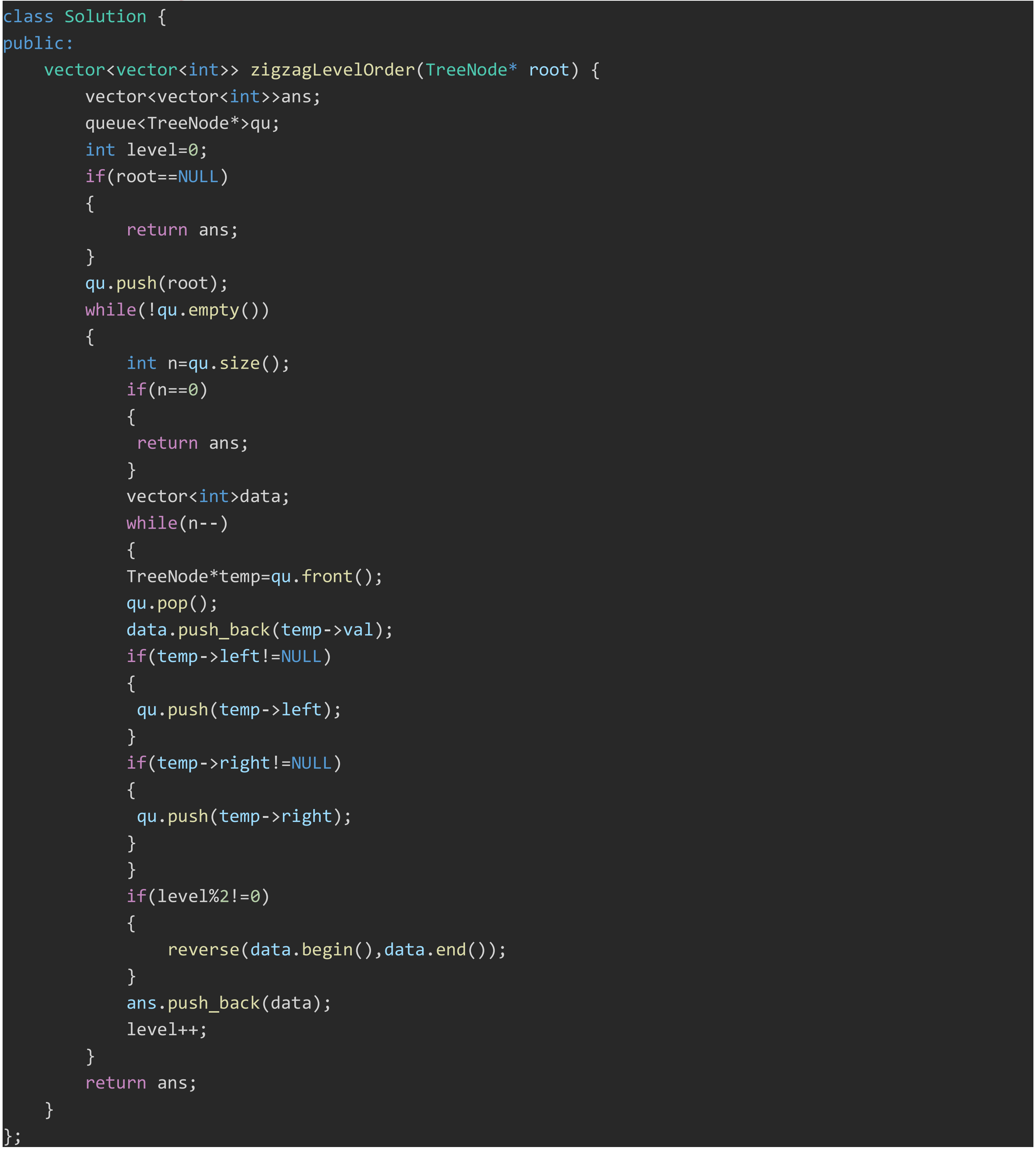
0 ms | Beats 100.00%

Memory

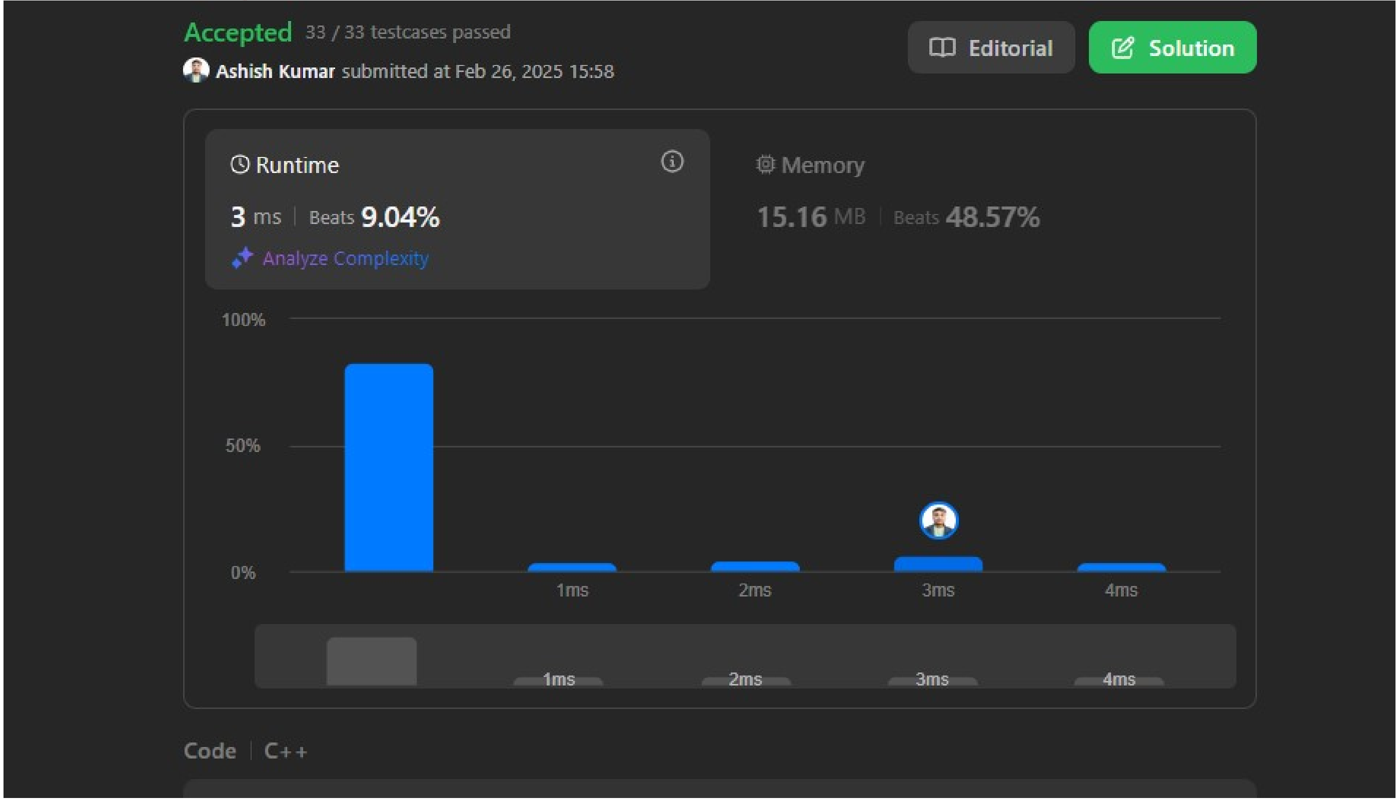
18.48 MB | Beats 59.30%



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
```

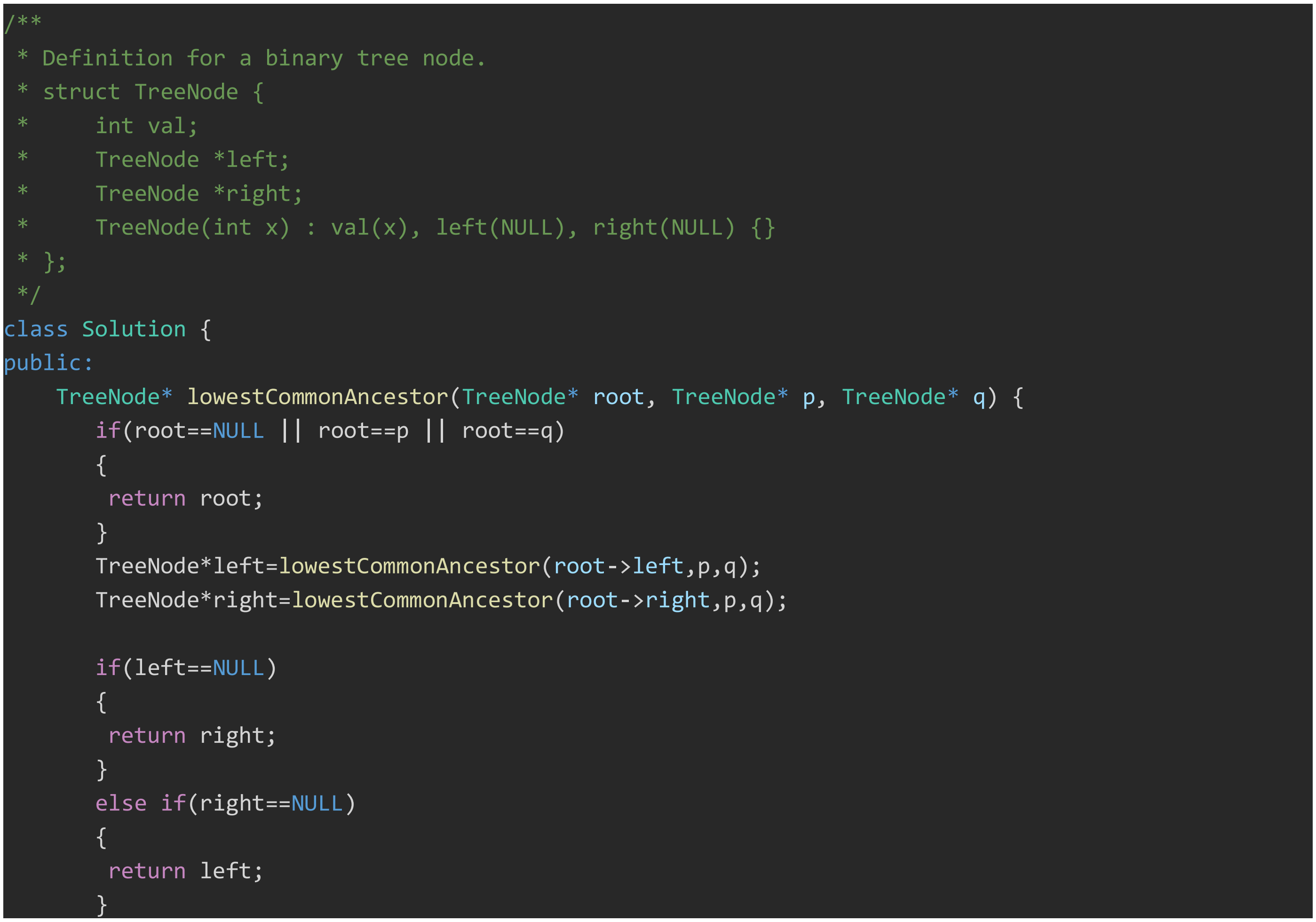


```
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>>ans;
        queue<TreeNode*>qu;
        int level=0;
        if(root==NULL)
        {
            return ans;
        }
        qu.push(root);
        while(!qu.empty())
        {
            int n=qu.size();
            if(n==0)
            {
             return ans;
            }
            vector<int>data;
            while(n--)
            {
            TreeNode*temp=qu.front();
            qu.pop();
            data.push_back(temp->val);
            if(temp->left!=NULL)
            {
             qu.push(temp->left);
            }
            if(temp->right!=NULL)
            {
             qu.push(temp->right);
            }
            }
            if(level%2!=0)
            {
                reverse(data.begin(),data.end());
            }
            ans.push_back(data);
            level++;
        }
        return ans;
    }
};
```

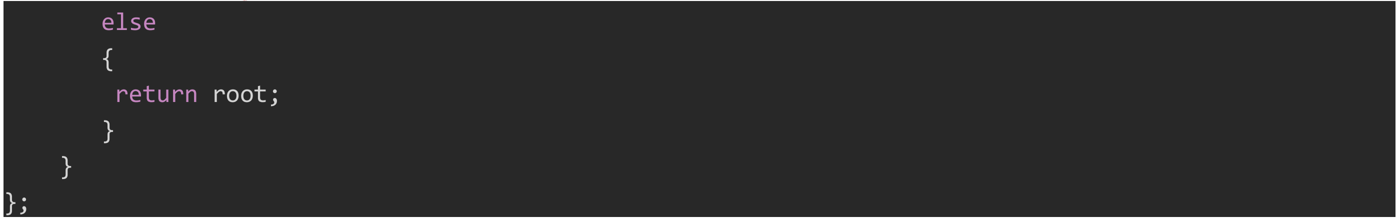


Accepted 33 / 33 testcases passed

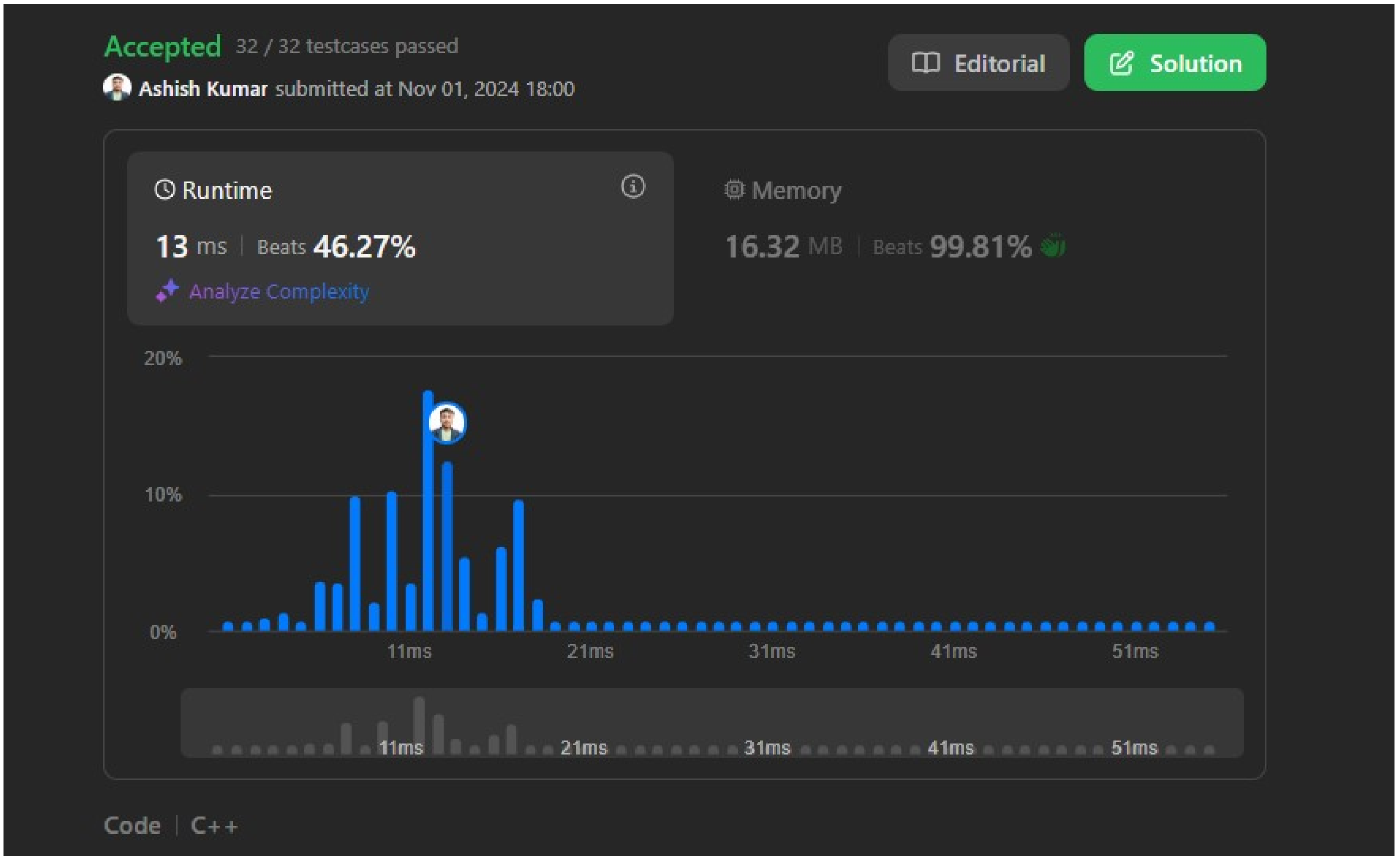
Ashish Kumar submitted at Feb 26, 2025 15:58



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
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)
       {
        return right;
       }
       else if(right==NULL)
       {
        return left;
       }
```



```
       else
       {
        return root;
       }
    }
};
```



Accepted 32 / 32 testcases passed

Ashish Kumar submitted at Nov 01, 2024 18:00

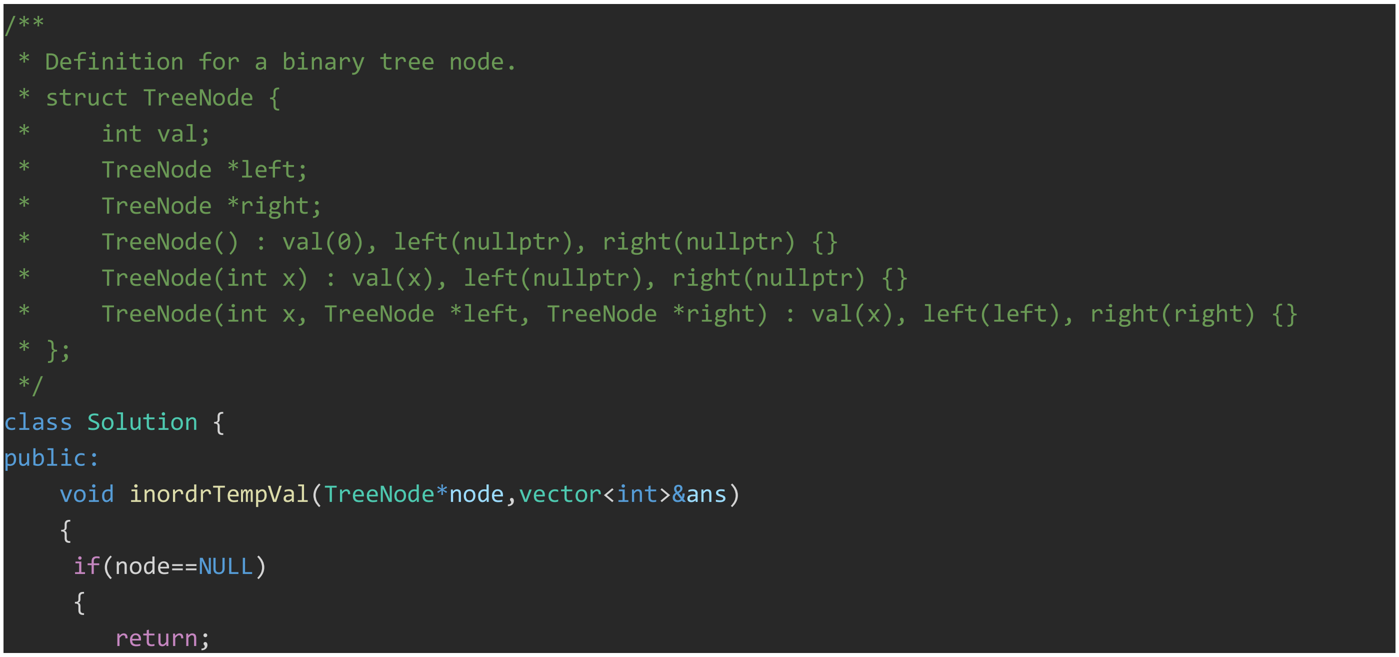
Runtime

13 ms | Beats 46.27%

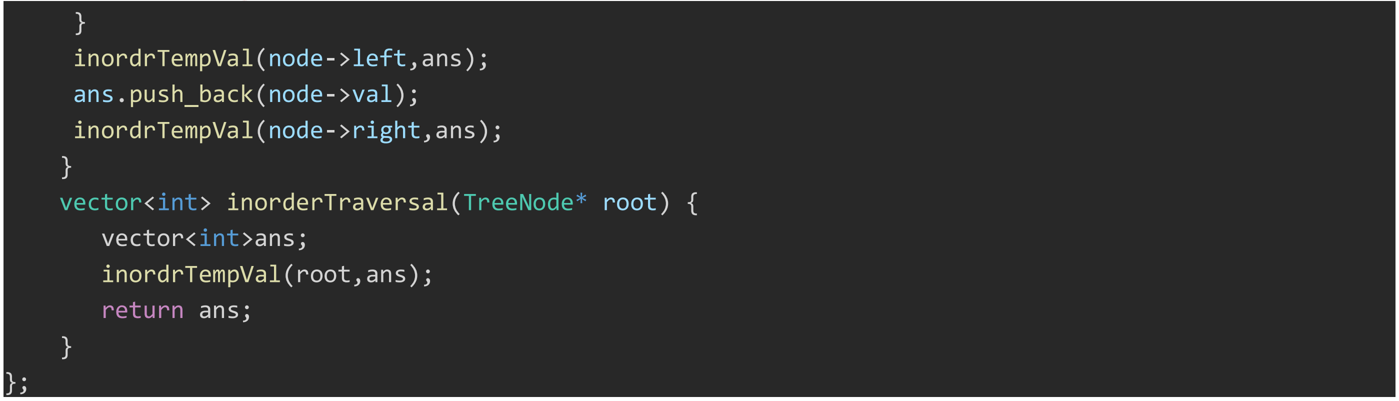
Memory

16.32 MB | Beats 99.81%

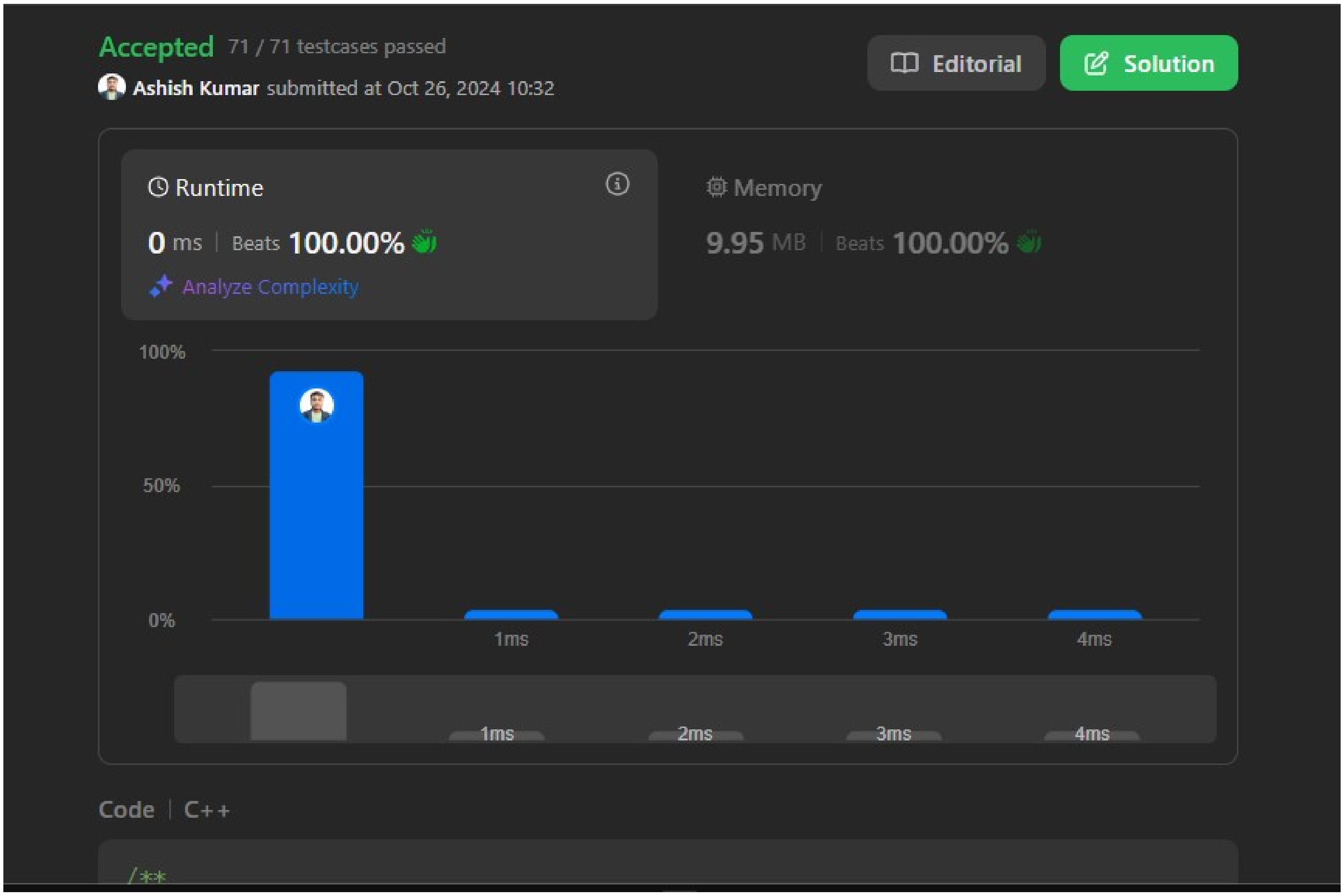
Code | C++



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    void inordrTempVal(TreeNode*node,vector<int>&ans)
    {
     if(node==NULL)
     {
        return;
```



```
     }
     inordrTempVal(node->left,ans);
     ans.push_back(node->val);
     inordrTempVal(node->right,ans);
    }
    vector<int> inorderTraversal(TreeNode* root) {
       vector<int>ans;
       inordrTempVal(root,ans);
       return ans;
    }
};
```



Accepted 71 / 71 testcases passed

Ashish Kumar submitted at Oct 26, 2024 10:32

Runtime

0 ms | Beats 100.00%

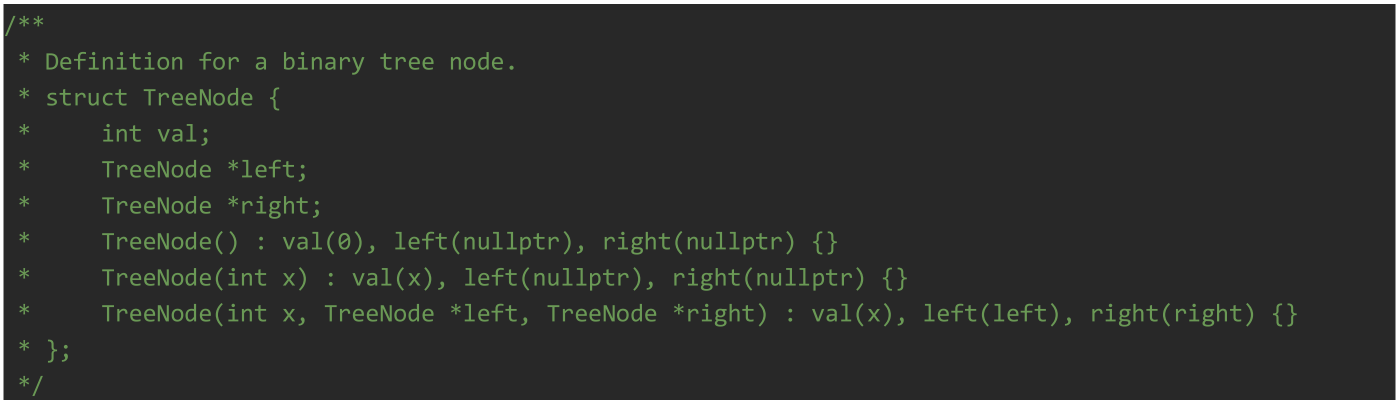
Analyze Complexity

Memory

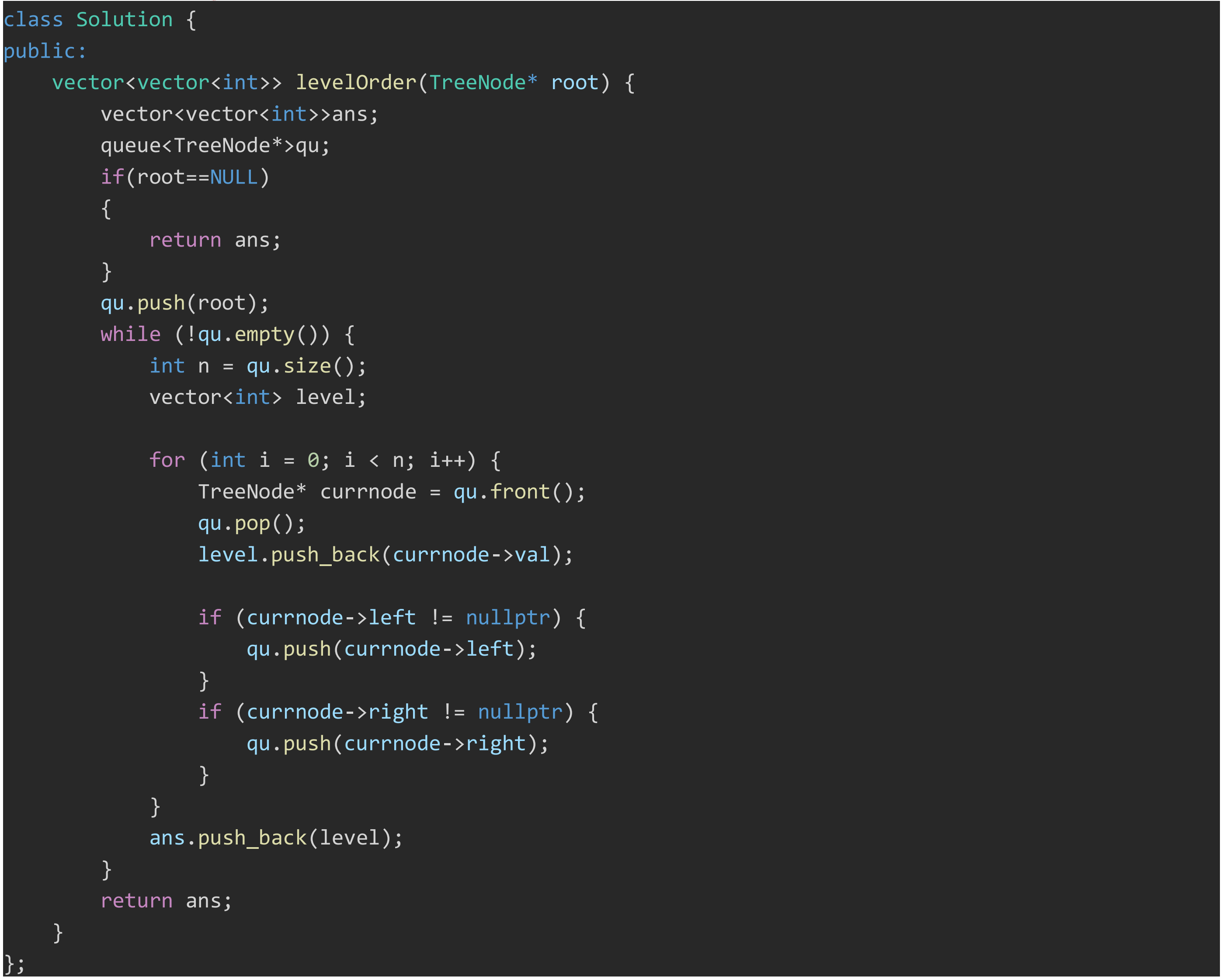
9.95 MB | Beats 100.00%

Code | C++

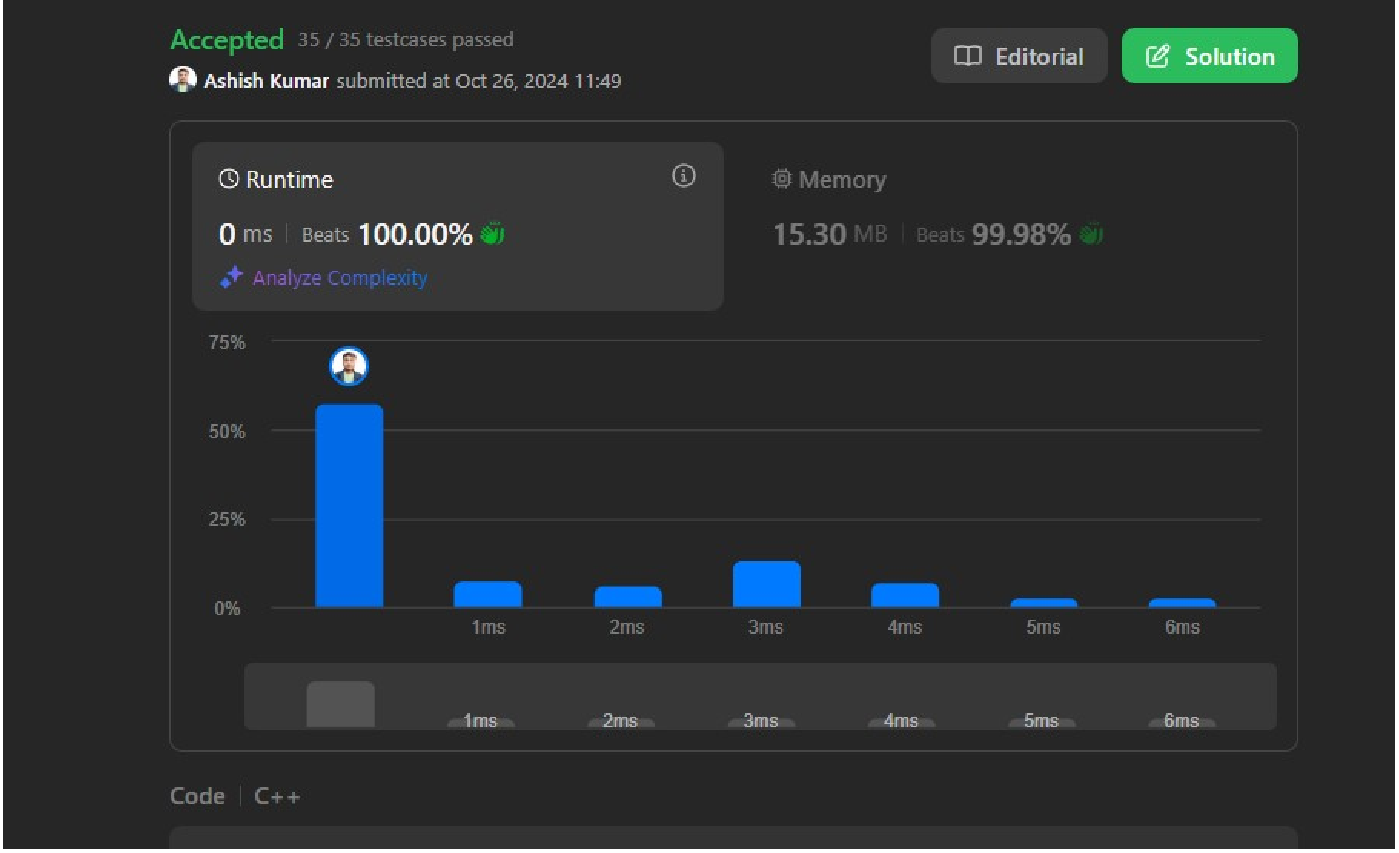
```
/**
```



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
```



```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>>ans;
        queue<TreeNode*>qu;
        if(root==NULL)
        {
            return ans;
        }
        qu.push(root);
        while (!qu.empty()) {
            int n = qu.size();
            vector<int> level;

            for (int i = 0; i < n; i++) {
                TreeNode* currnode = qu.front();
                qu.pop();
                level.push_back(currnode->val);

                if (currnode->left != nullptr) {
                    qu.push(currnode->left);
                }
                if (currnode->right != nullptr) {
                    qu.push(currnode->right);
                }
            }
            ans.push_back(level);
        }
        return ans;
    }
};
```



Accepted 35 / 35 testcases passed

Ashish Kumar submitted at Oct 26, 2024 11:49

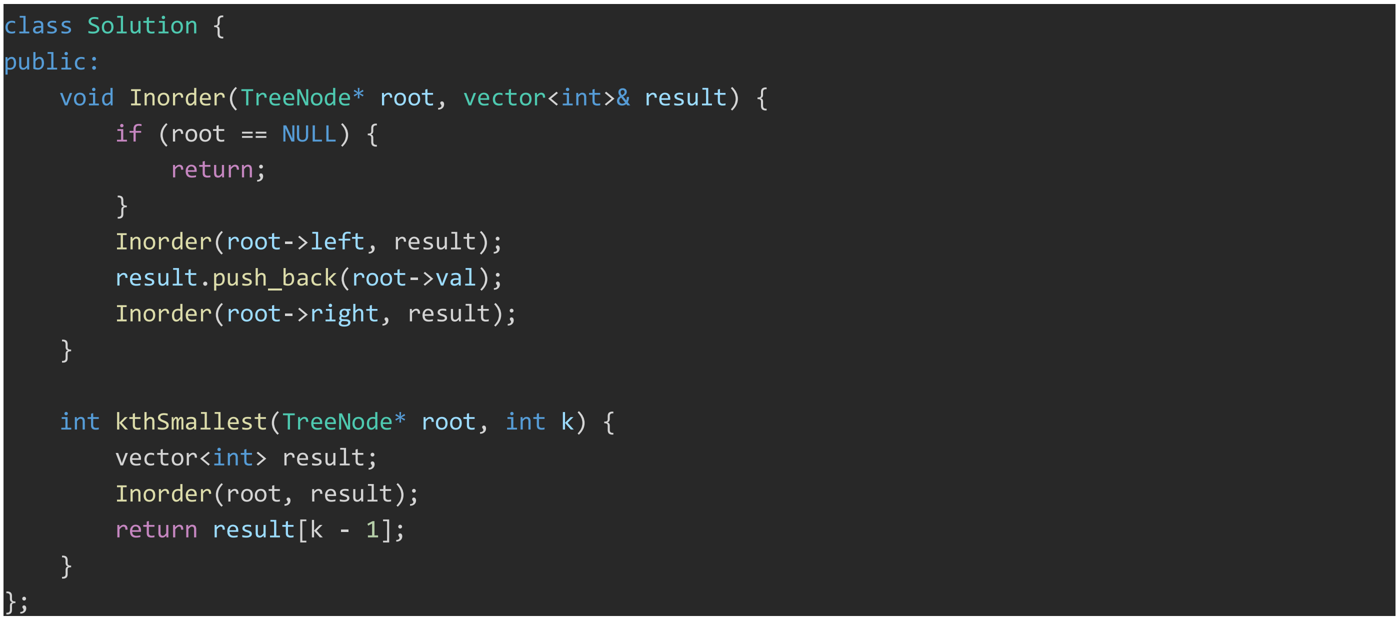
Runtime

0 ms | Beats 100.00%

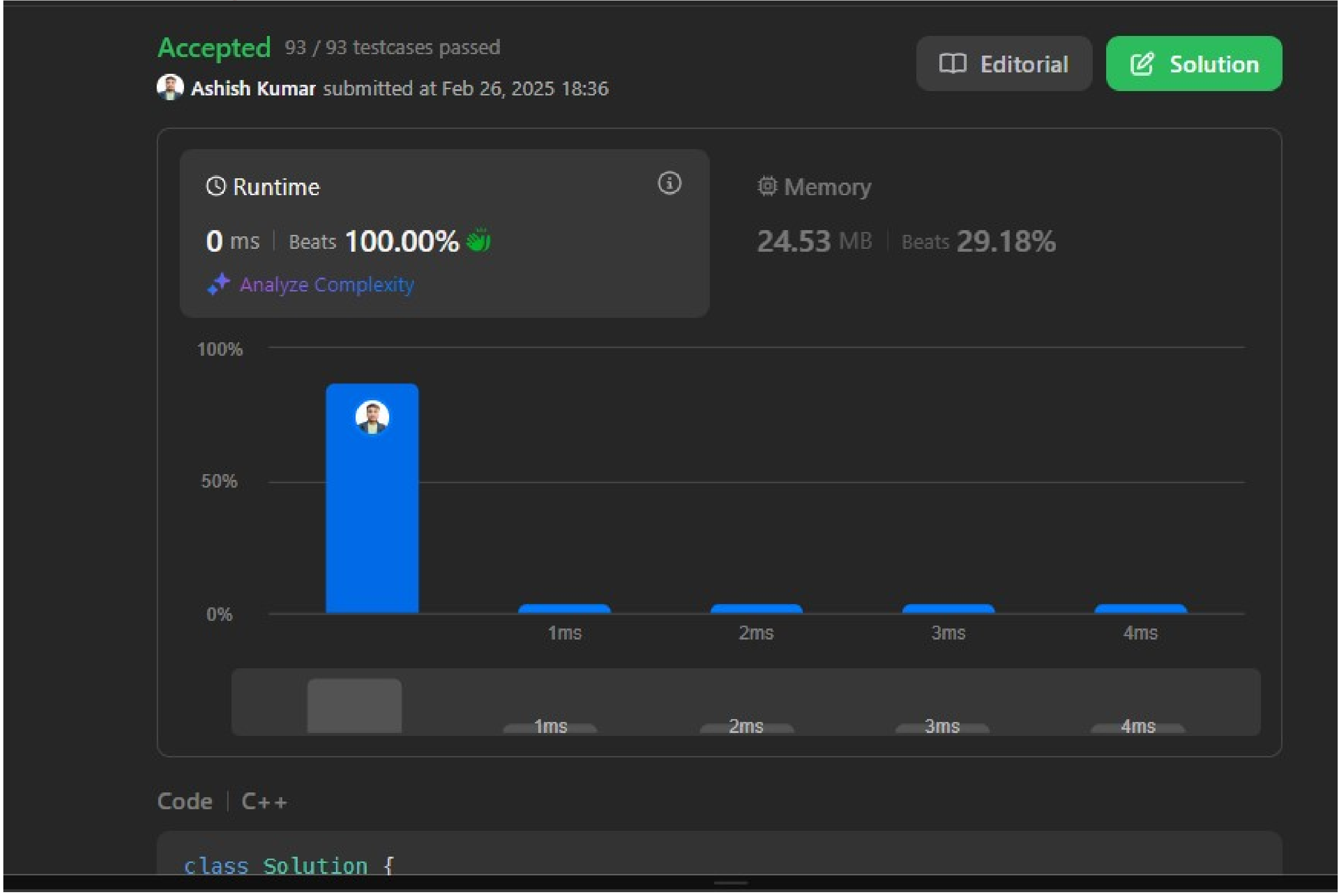
Memory

15.30 MB | Beats 99.98%

Code | C++



```
class Solution {
public:
    void Inorder(TreeNode* root, vector<int>& result) {
        if (root == NULL) {
            return;
        }
        Inorder(root->left, result);
        result.push_back(root->val);
        Inorder(root->right, result);
    }

    int kthSmallest(TreeNode* root, int k) {
        vector<int> result;
        Inorder(root, result);
        return result[k - 1];
    }
};
```



Accepted 93 / 93 testcases passed

Ashish Kumar submitted at Feb 26, 2025 18:36

Runtime

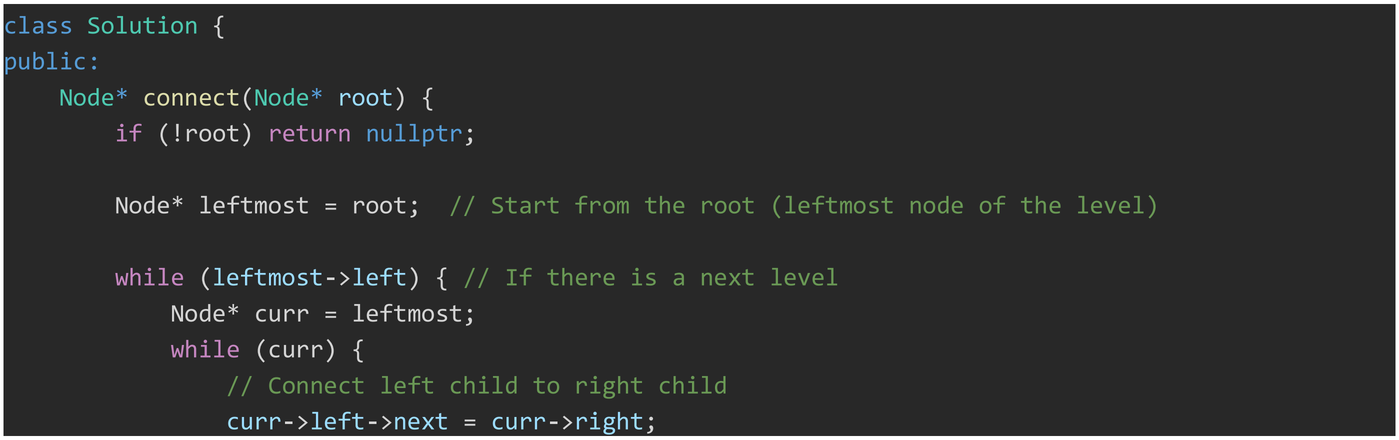
0 ms | Beats 100.00%

Memory

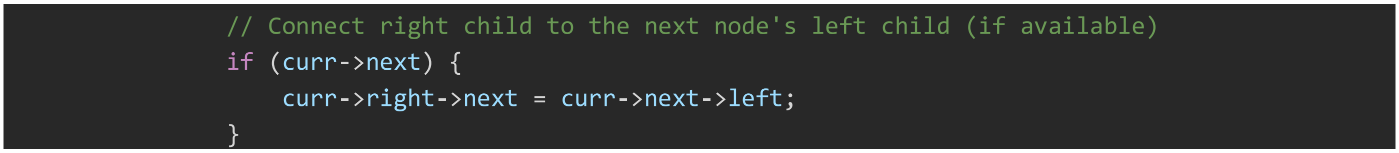
24.53 MB | Beats 29.18%

Code | C++

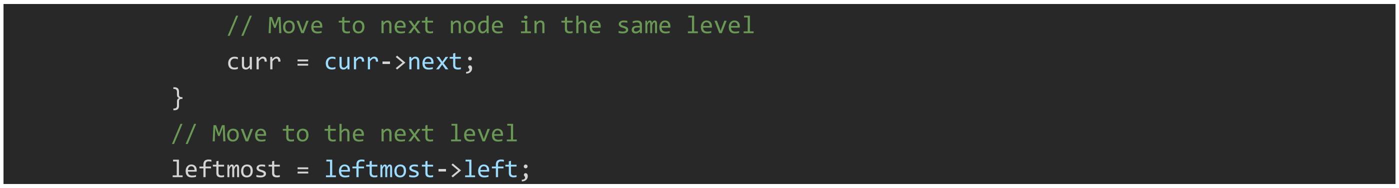
```
class Solution {
```



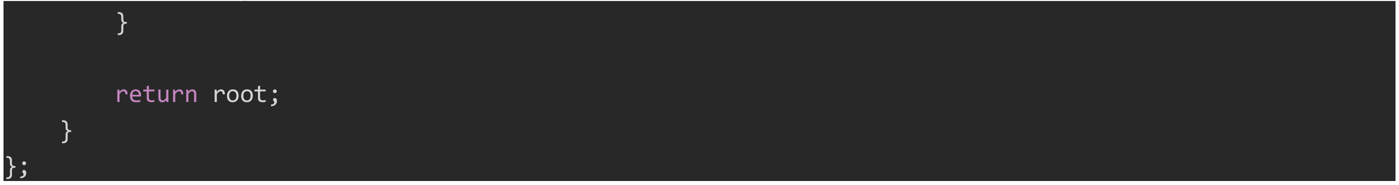
```
class Solution {
public:
    Node* connect(Node* root) {
        if (!root) return nullptr;

        Node* leftmost = root;  // Start from the root (leftmost node of the level)

        while (leftmost->left) { // If there is a next level
            Node* curr = leftmost;
            while (curr) {
                // Connect left child to right child
                curr->left->next = curr->right;
```



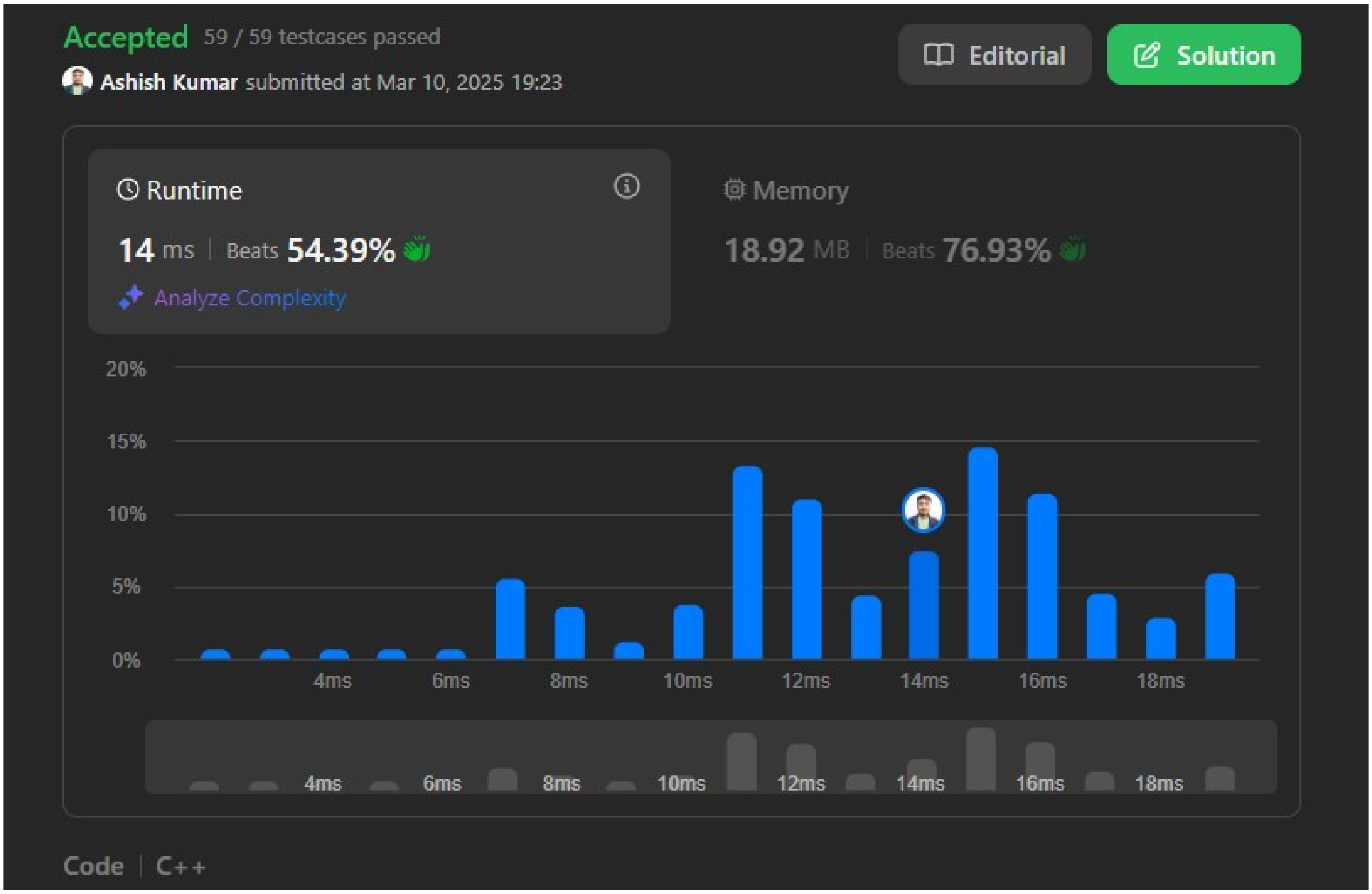
```
                // Connect right child to the next node's left child (if available)
                if (curr->next) {
                    curr->right->next = curr->next->left;
                }
```



```
                // Move to next node in the same level
                curr = curr->next;
            }
            // Move to the next level
            leftmost = leftmost->left;
```

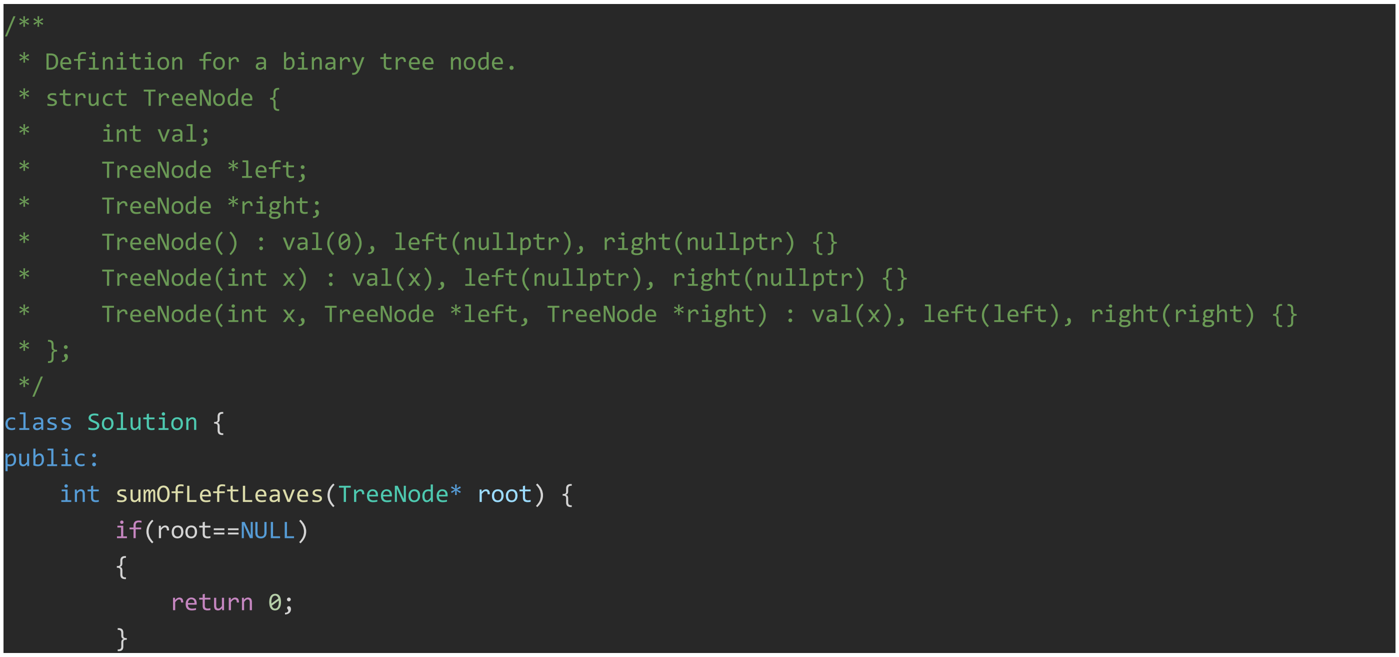


```
        }

        return root;
    }
};
```

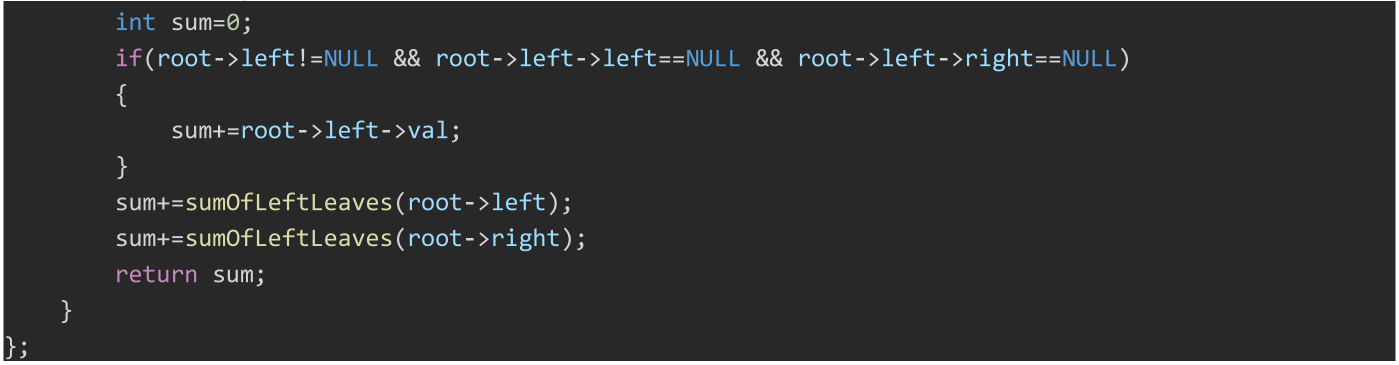


Accepted 59 / 59 testcases passed

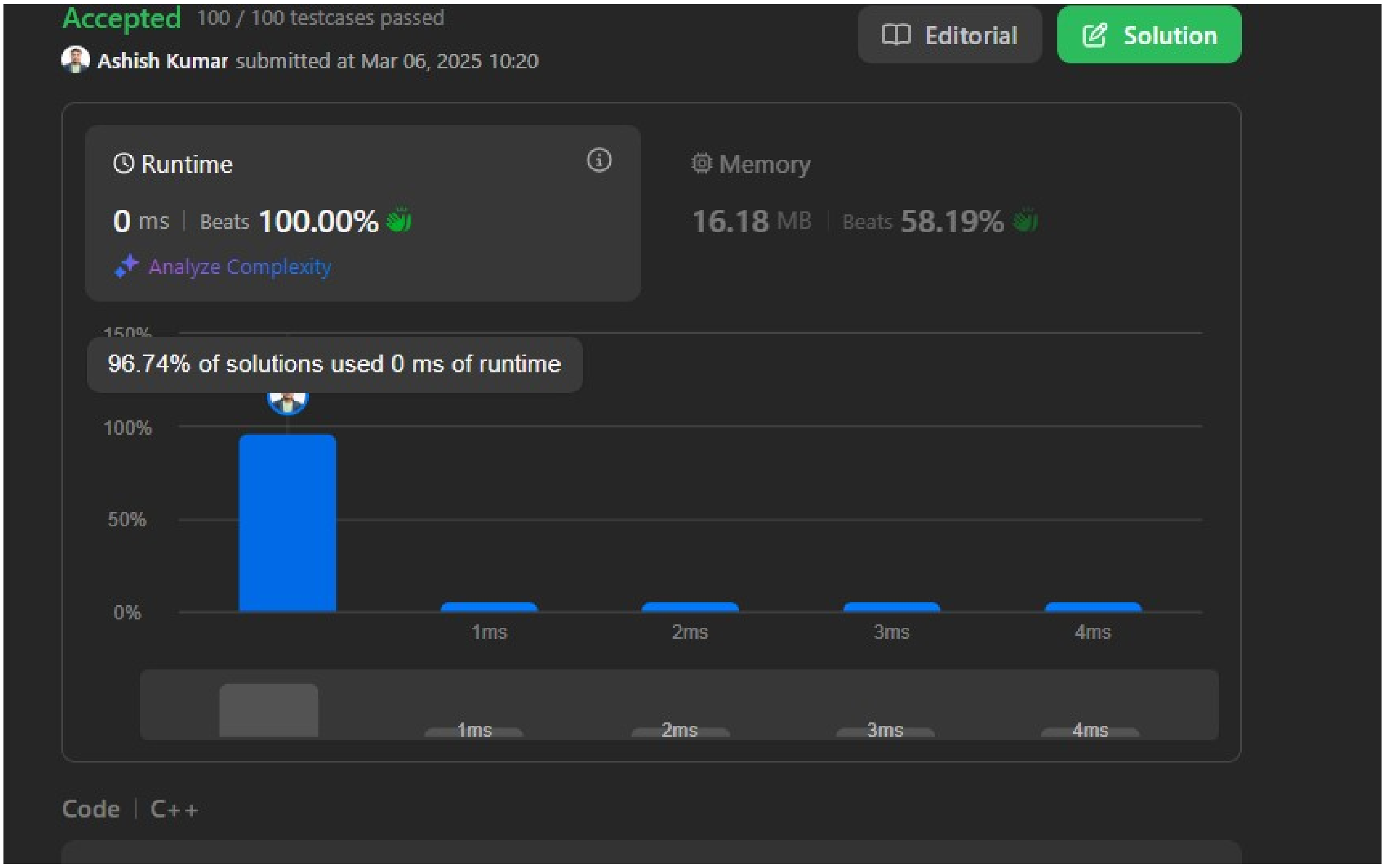
Ashish Kumar submitted at Mar 10, 2025 19:23



```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {
        if(root==NULL)
        {
            return 0;
        }
```



```
        int sum=0;
        if(root->left!=NULL && root->left->left==NULL && root->left->right==NULL)
        {
            sum+=root->left->val;
        }
        sum+=sumOfLeftLeaves(root->left);
        sum+=sumOfLeftLeaves(root->right);
        return sum;
    }
};
```



Accepted 100 / 100 testcases passed

Ashish Kumar submitted at Mar 06, 2025 10:20

Runtime

0 ms | Beats 100.00%

Memory

16.18 MB | Beats 58.19%

96.74% of solutions used 0 ms of runtime

Code | C++