**104. Maximum Depth of Binary Tree**

class Solution {

public:

int maxDepth(TreeNode\* root) {

if(root==NULL){

return 0;

}

int left=maxDepth(root->left);

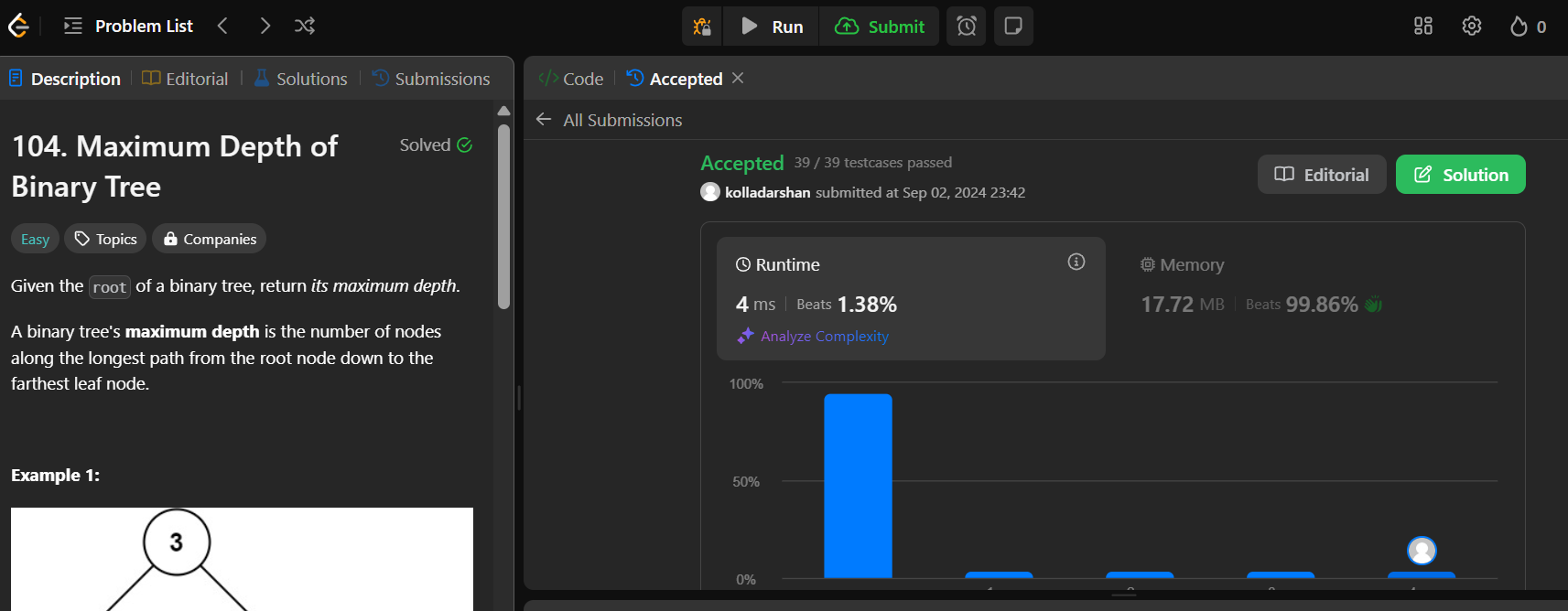
int right=maxDepth(root->right);

int ans=max(left,right)+1;

return ans;

}

};



**98. Validate Binary Search Tree**

class Solution {

public:

    bool validate(TreeNode\* root, long long minVal, long long maxVal) {

        if (!root) return true;

        if (root->val <= minVal || root->val >= maxVal) return false;

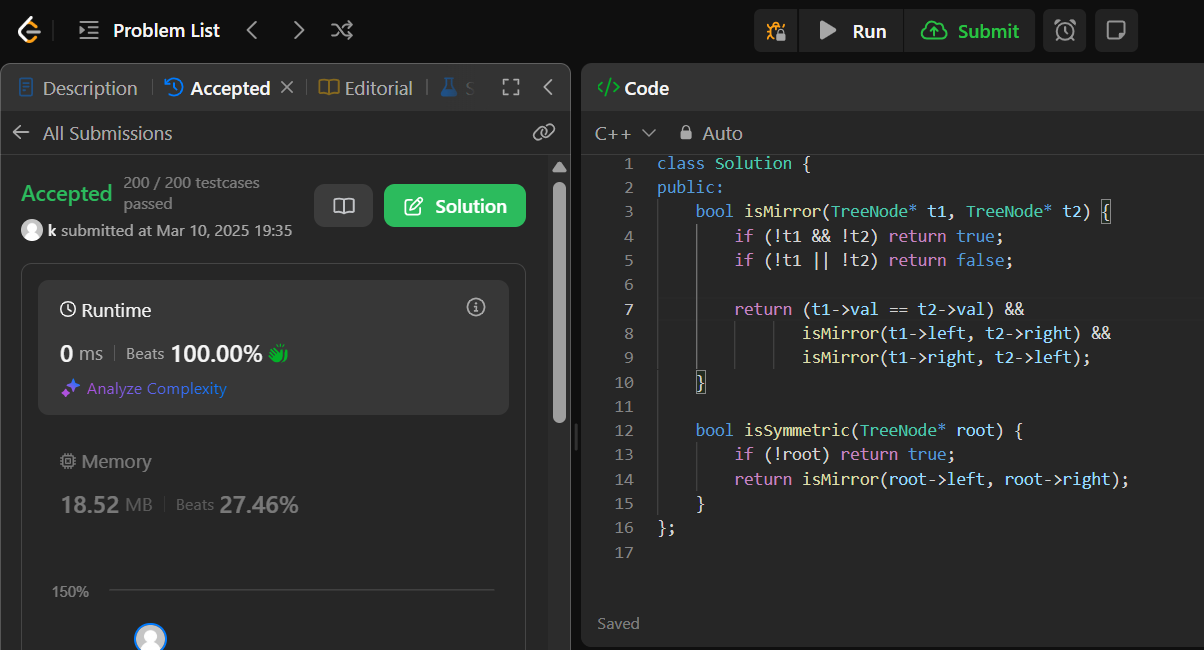
        return validate(root->left, minVal, root->val) &&

               validate(root->right, root->val, maxVal);

    }

    bool isValidBST(TreeNode\* root) {

        return validate(root, LLONG\_MIN, LLONG\_MAX);}

****

**101. Symmetric Tree**

class Solution {

public:

    bool isMirror(TreeNode\* t1, TreeNode\* t2) {

        if (!t1 && !t2) return true;

        if (!t1 || !t2) return false;

        return (t1->val == t2->val) &&

               isMirror(t1->left, t2->right) &&

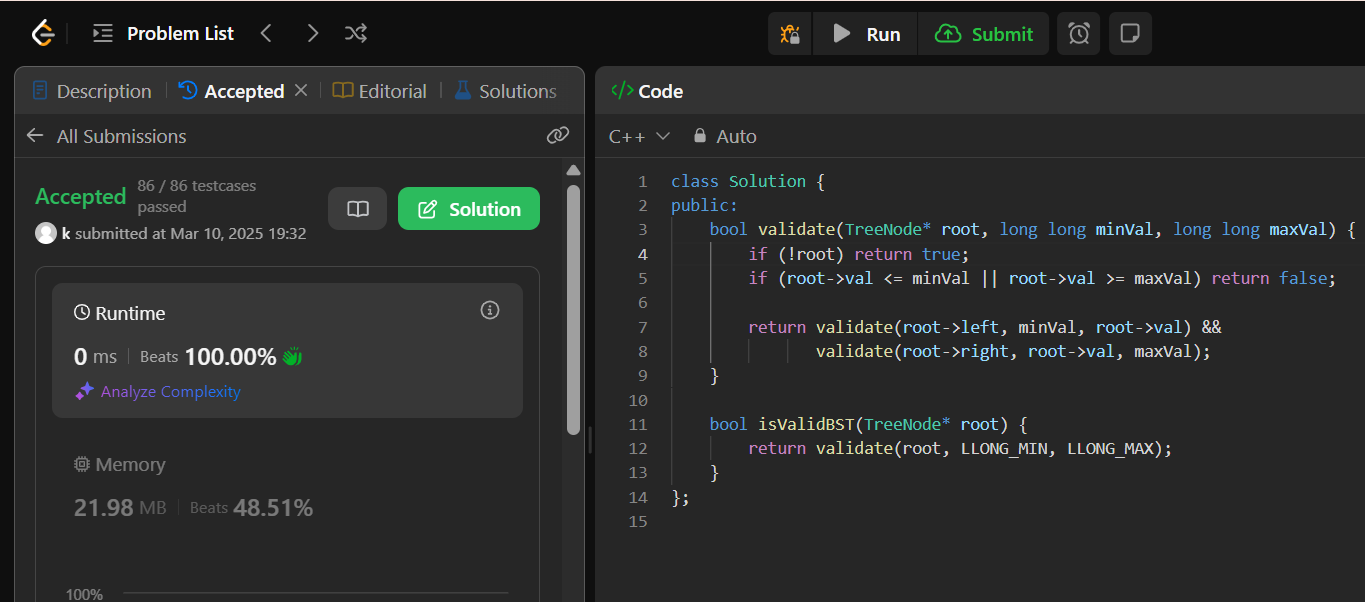
               isMirror(t1->right, t2->left);

    }

    bool isSymmetric(TreeNode\* root) {

        if (!root) return true;

        return isMirror(root->left, root->right); }};

****

**103. Binary Tree Zigzag Level Order Traversal**

 class Solution {

public:

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

        vector<vector<int>> result;

        if (!root) return result;

        queue<TreeNode\*> q;

        q.push(root);

        bool leftToRight = true;

        while (!q.empty()) {

            int size = q.size();

            vector<int> level(size);

             for (int i = 0; i < size; i++) {

                TreeNode\* node = q.front();

                q.pop();

             int index = leftToRight ? i : (size - 1 - i);

                level[index] = node->val;

                if (node->left) q.push(node->left);

                if (node->right) q.push(node->right);

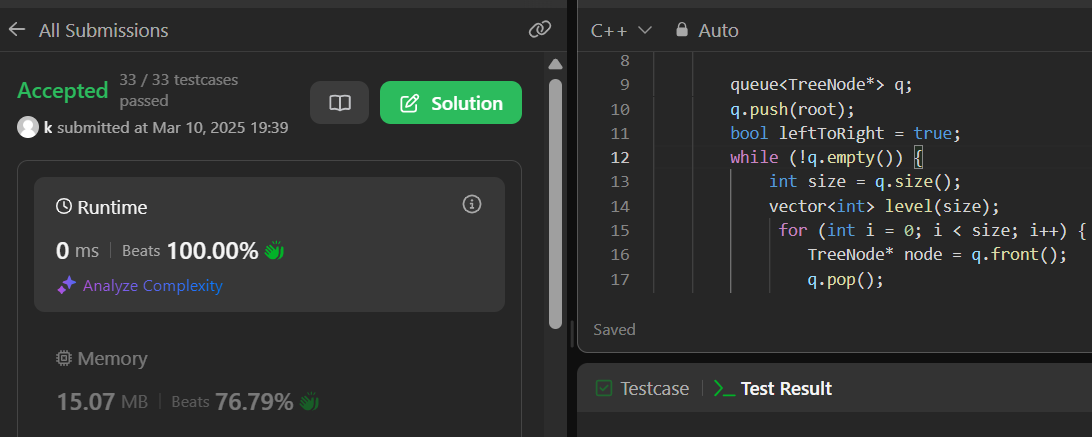
            }

            result.push\_back(level);

            leftToRight = !leftToRight;

        }

        return result; }};

****

[**94. Binary Tree Inorder Traversal**](https://leetcode.com/problems/binary-tree-inorder-traversal/)

class Solution {

public:

    void inorder(TreeNode\* root, vector<int>& result) {

        if (!root) return;

        inorder(root->left, result);

        result.push\_back(root->val);

        inorder(root->right, result);

    }

    vector<int> inorderTraversal(TreeNode\* root) {

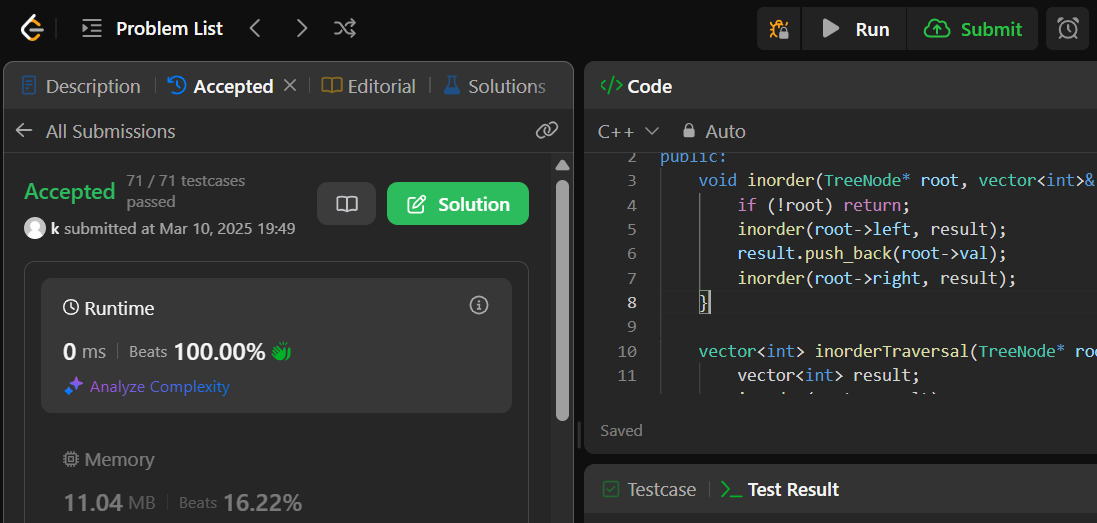
        vector<int> result;

        inorder(root, result);

        return result;

    }

};

****

**404. Sum of Left Leaves**

class Solution {

public:

    int sumOfLeftLeaves(TreeNode\* root) {

        if (!root) return 0;

           int sum = 0;

        if (root->left && !root->left->left && !root->left->right) {

            sum += root->left->val;

        }

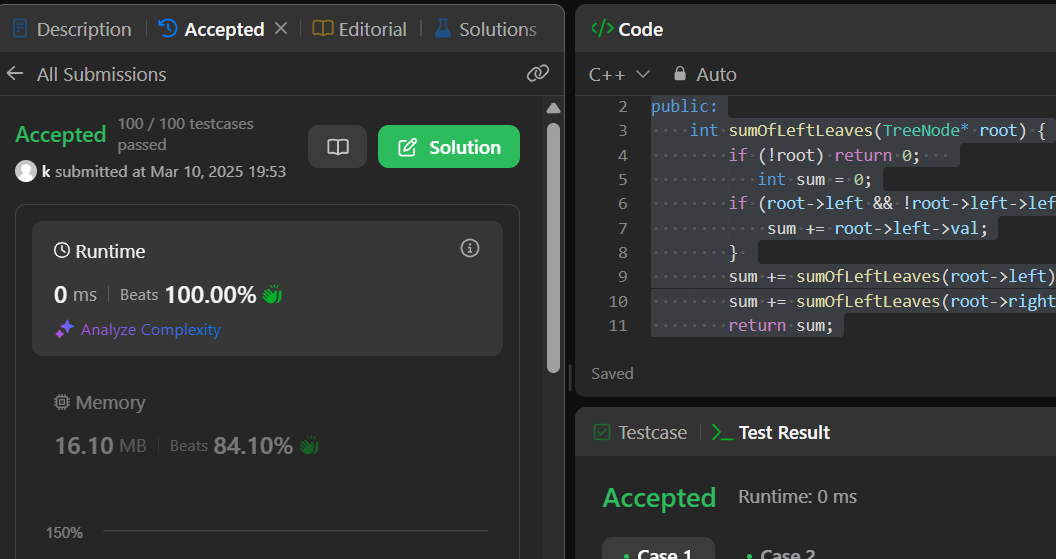
        sum += sumOfLeftLeaves(root->left);

        sum += sumOfLeftLeaves(root->right);

        return sum;

    }

};

****