**ASSIGNMENT -3 (ADVANCED PROGRAMMING)**

Profile: <https://leetcode.com/u/AnkitSingh101/>

1. **Problem 1:** [Binary Tree Inorder Traversal](https://leetcode.com/problems/binary-tree-inorder-traversal/) (94)
2. **Code:**

class Solution {

public:

    vector<int> result;

    void inorder(TreeNode\* root){

        if(root!=NULL){

            inorder(root->left);

            result.push\_back(root->val);

            inorder(root->right);

        }

    }

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

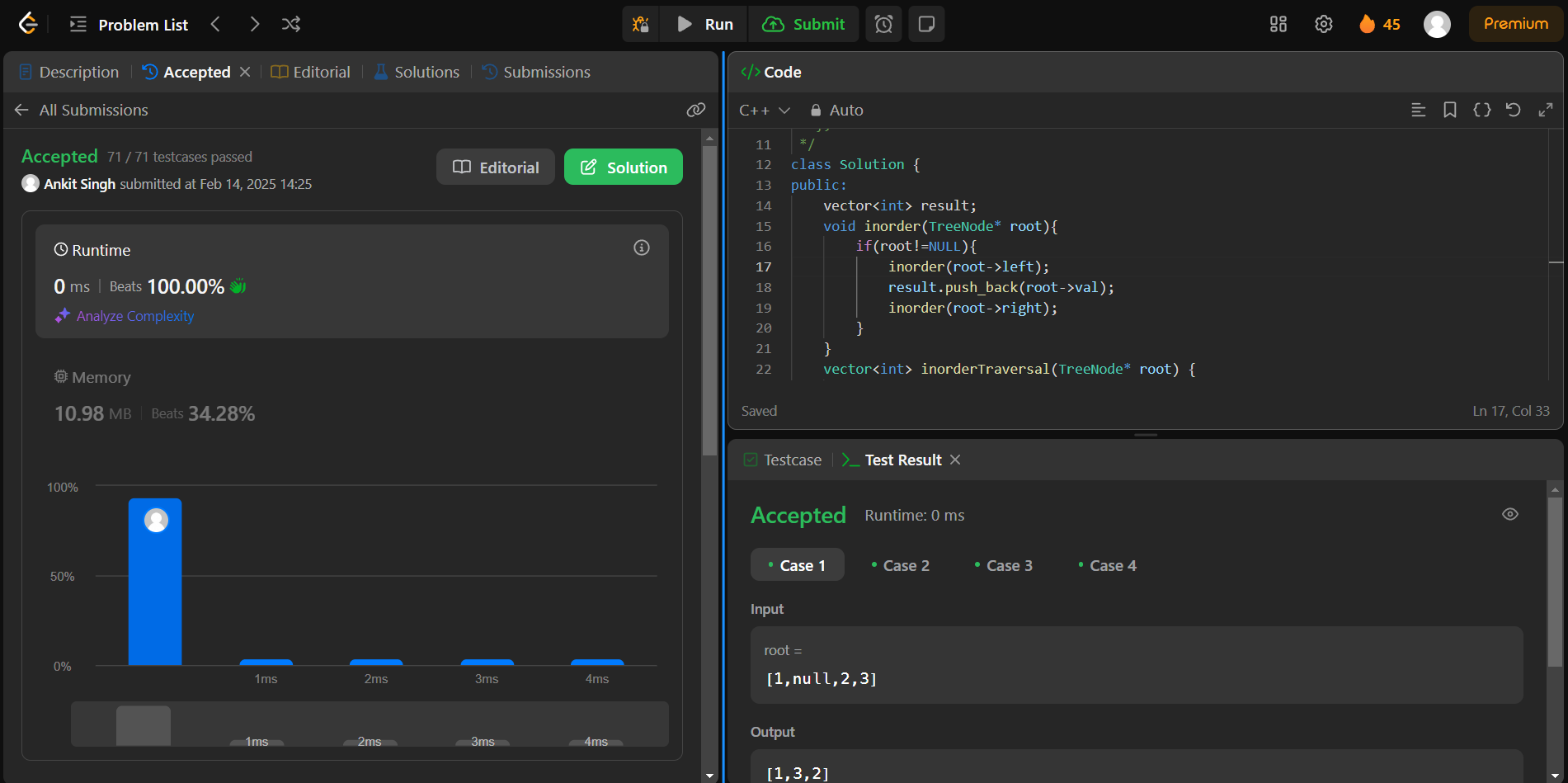
        inorder(root);

        return result;

    }

};

1. **Output:**

****

1. **Problem 2:** [Symmetric Tree](https://leetcode.com/problems/symmetric-tree/) (101)
2. **Code:**

Class Solution {

public:

    bool check(TreeNode\* left,TreeNode\* right){

        if(!left && !right) return true;

        if(!left || !right) return false;

        return (left->val==right->val) && check(left->left,right->right) && check(left->right,right->left );

    }

    bool isSymmetric(TreeNode\* root) {

        if (root == NULL) {

            return true;

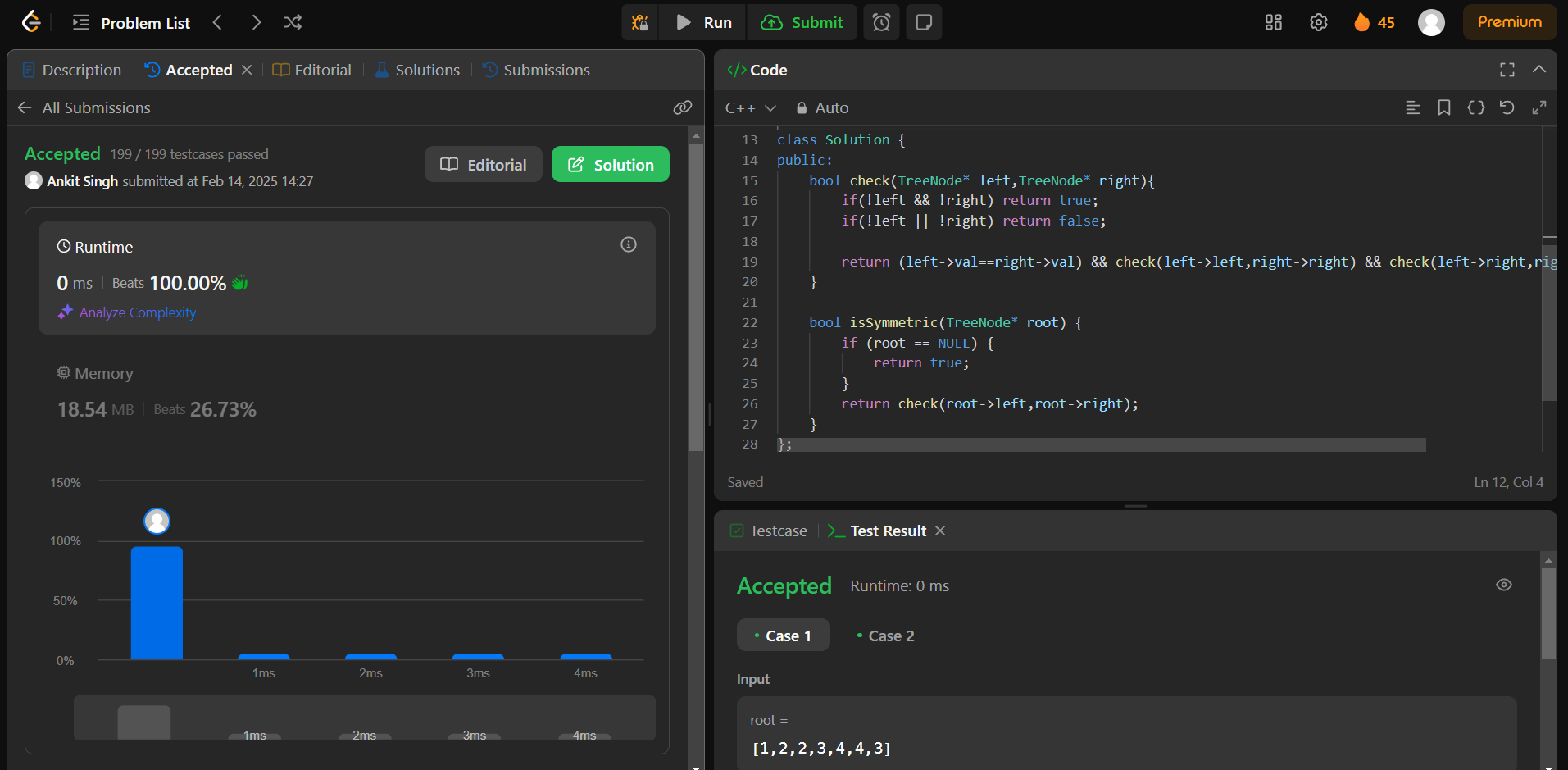
        }

        return check(root->left,root->right);

    }

};

1. **Output:**

****

1. **Problem 3:** [Maximum Depth of Binary Tree](https://leetcode.com/problems/maximum-depth-of-binary-tree/) (104)
2. **Code:**

class Solution {

public:

    int maxDepth(TreeNode\* root) {

        if(root==NULL){

            return 0;

        }

        int lh=maxDepth(root->left);

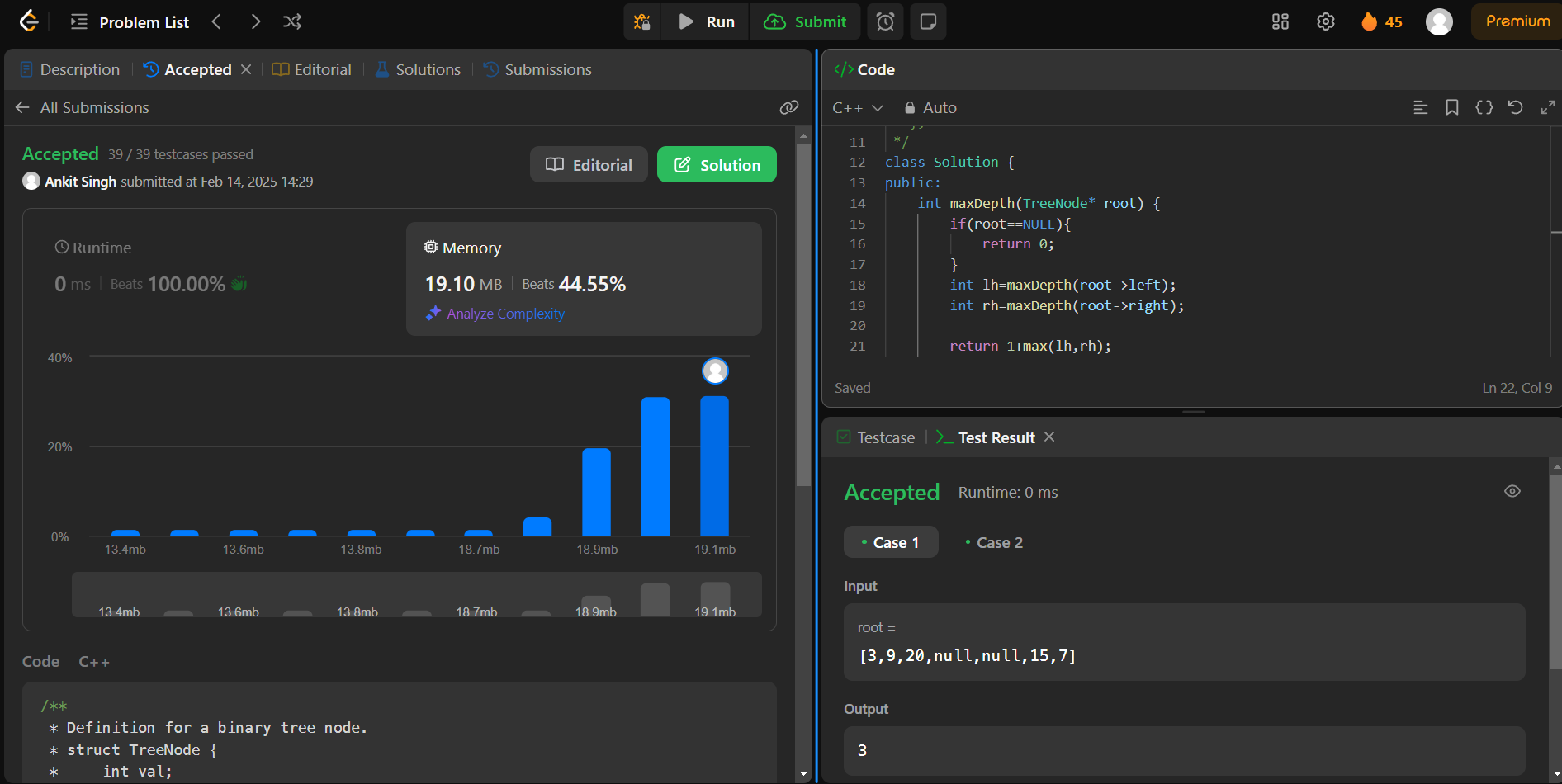
        int rh=maxDepth(root->right);

        return 1+max(lh,rh);

    }

}

1. **Output:**

****

1. **Problem 4:** [Validate Binary Search Tree](https://leetcode.com/problems/validate-binary-search-tree/) (98)
2. **Code:**

class Solution {

    bool isPossible(TreeNode\* root, long long l, long long r) {

        if (root == nullptr)

            return true;

        if (root->val < r and root->val > l)

            return isPossible(root->left, l, root->val) and

                   isPossible(root->right, root->val, r);

        return false;

    }

public:

    bool isValidBST(TreeNode\* root) {

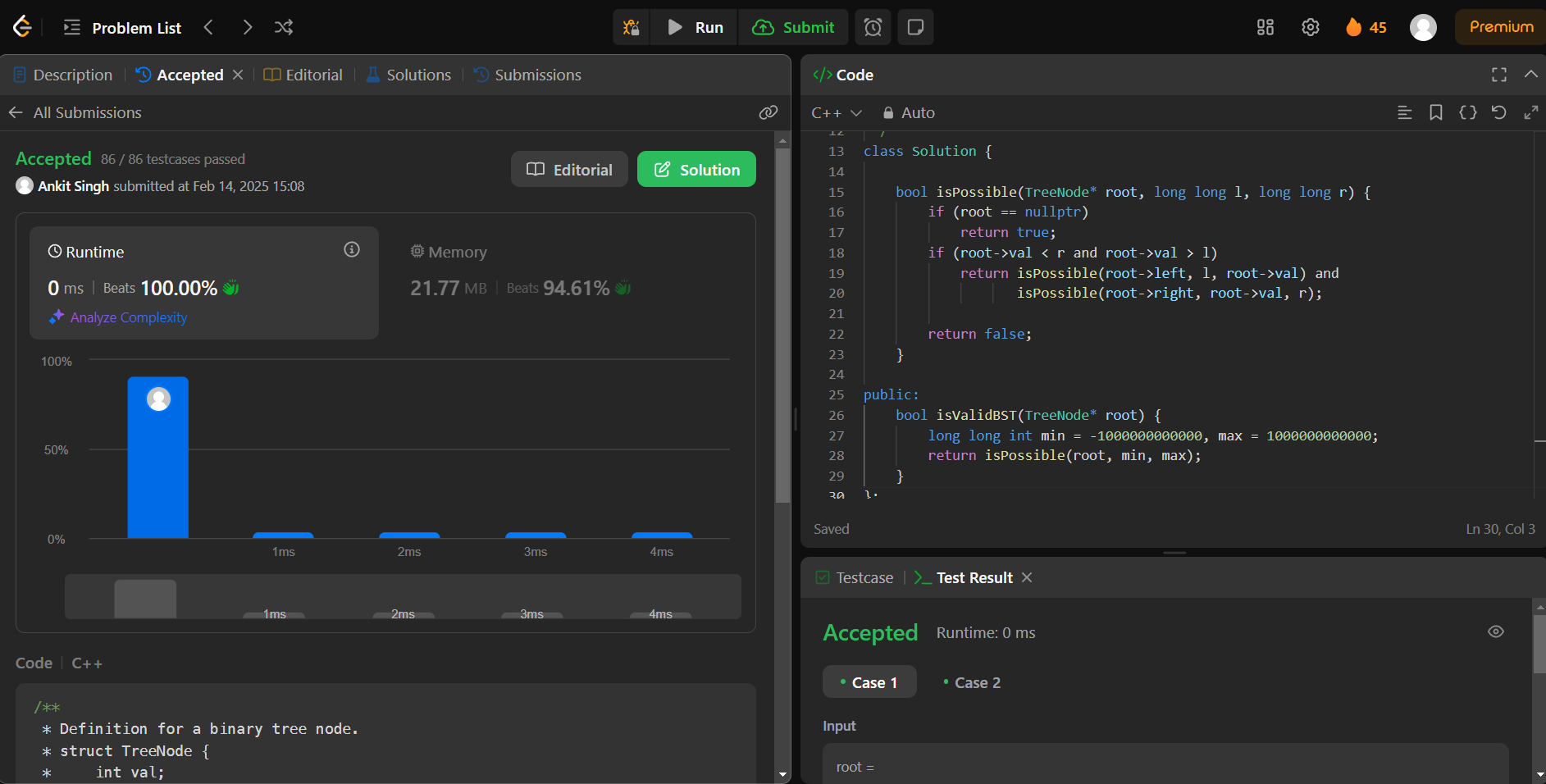
        long long int min = -1000000000000, max = 1000000000000;

        return isPossible(root, min, max);

    }

};

1. **Output:**

****

**1.Problem 5:** [Kth Smallest Element in a BST](https://leetcode.com/problems/kth-smallest-element-in-a-bst/) (230)

1. **Code:**

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);

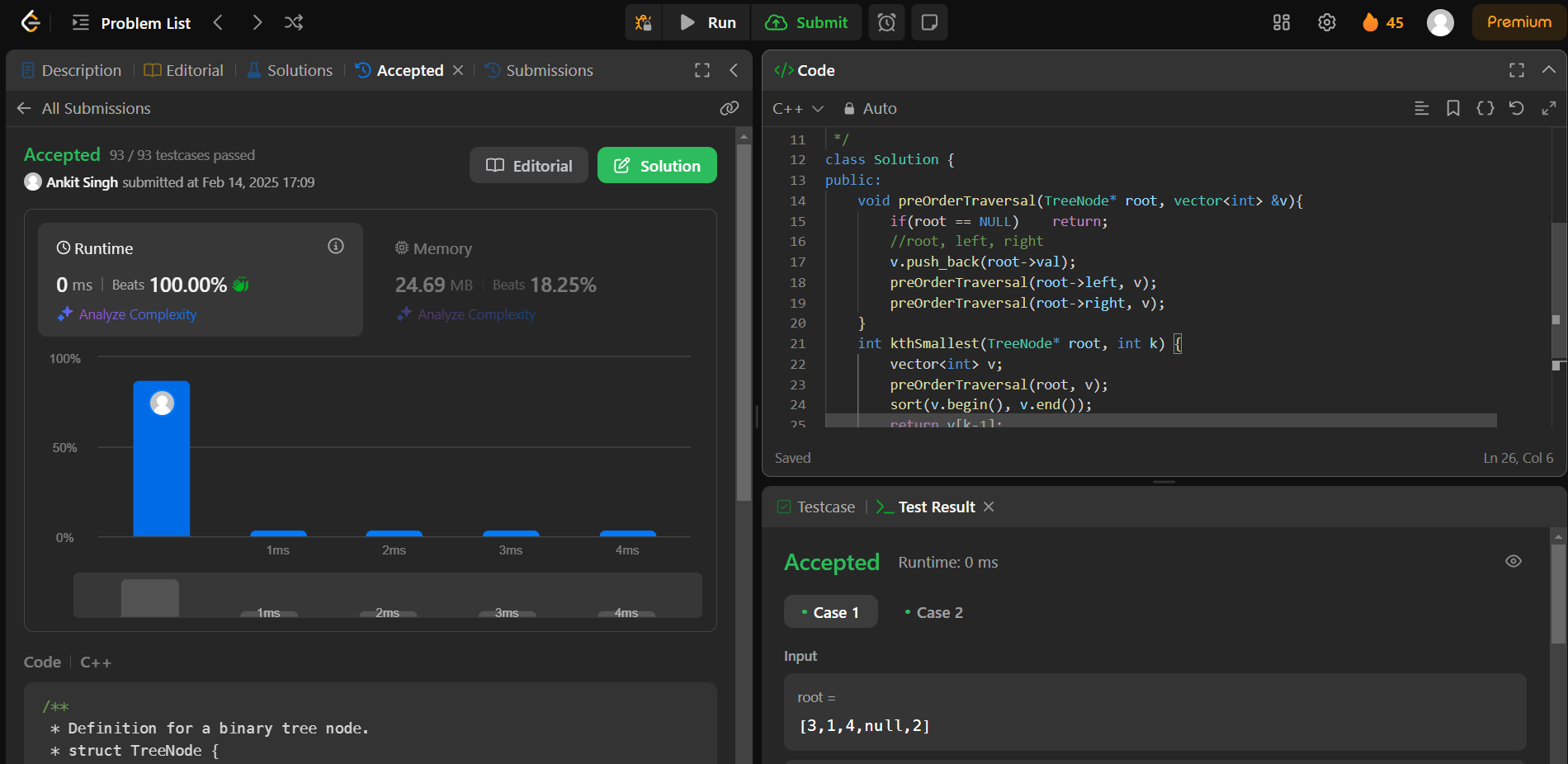
        sort(v.begin(), v.end());

        return v[k-1];

    }

};

1. **Output:**

****

1. **Problem 6:** [Binary Tree Level Order Traversal](https://leetcode.com/problems/binary-tree-level-order-traversal/) (102)
2. **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 size=q.size();

            vector<int> level;

            for(int i=0;i<size;i++){

                TreeNode\* node=q.front();

                q.pop();

                if(node->left!=NULL){q.push(node->left);}

                if(node->right!=NULL){q.push(node->right);}

                level.push\_back(node->val);

            }

            ans.push\_back(level);

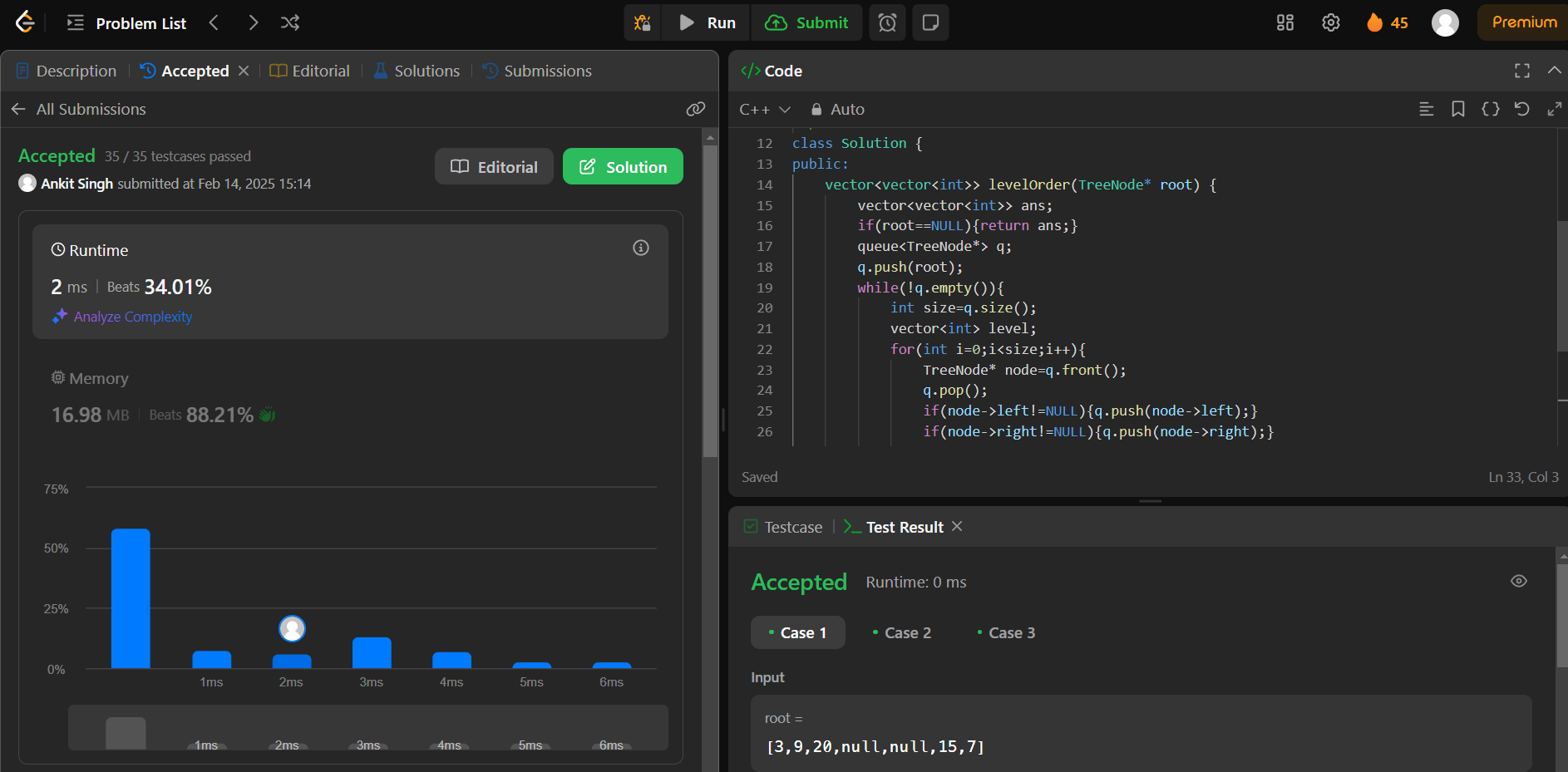
        }

    return ans;

    }

};

1. **Output:**

****

1. **Problem 7:** [Binary Tree Level Order Traversal II](https://leetcode.com/problems/binary-tree-level-order-traversal-ii/) (107)
2. **Code:**

class Solution {

public:

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

         vector<vector<int>> ans;

        if(!root) return ans;

        queue<TreeNode\*> q;

        q.push(root);

        int level=0;

        TreeNode\* curr;

        while(!q.empty()){

            int n=q.size();

            level++;

            vector<int> v;

            while(n--){

                curr=q.front();

                v.push\_back(curr->val);

                q.pop();

                if(curr->left) {

                    q.push(curr->left);

                }

                if(curr->right) {

                    q.push(curr->right);

                }

            }

            ans.push\_back(v);

        }

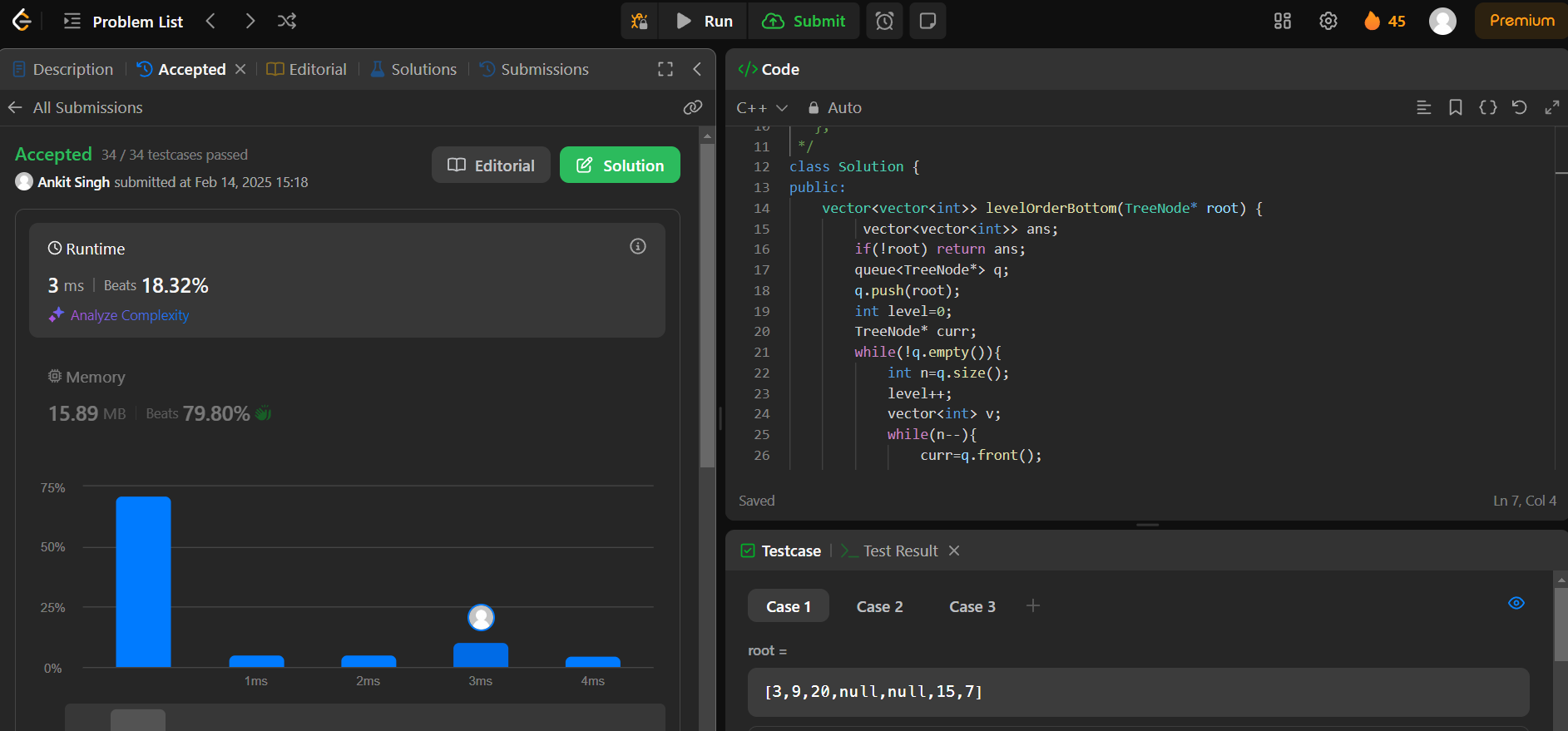
        reverse(ans.begin(), ans.end());

        return ans;

    }

};

1. **Output:**

****

1. **Problem 8:** [Binary Tree Zigzag Level Order Traversal](https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/) (103)
2. **Code:**

class Solution {

public:

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

        vector<vector<int>> ans;

        if(!root) return ans;

        queue<TreeNode\*> q;

        q.push(root);

        int level=0;

        TreeNode\* curr;

        while(!q.empty()){

            int n=q.size();

            level++;

            vector<int> v;

            while(n--){

                curr=q.front();

                v.push\_back(curr->val);

                q.pop();

                if(curr->left) {

                    q.push(curr->left);

                }

                if(curr->right) {

                    q.push(curr->right);

                }

            }

            if(level%2==0) reverse(v.begin(), v.end());

            ans.push\_back(v);

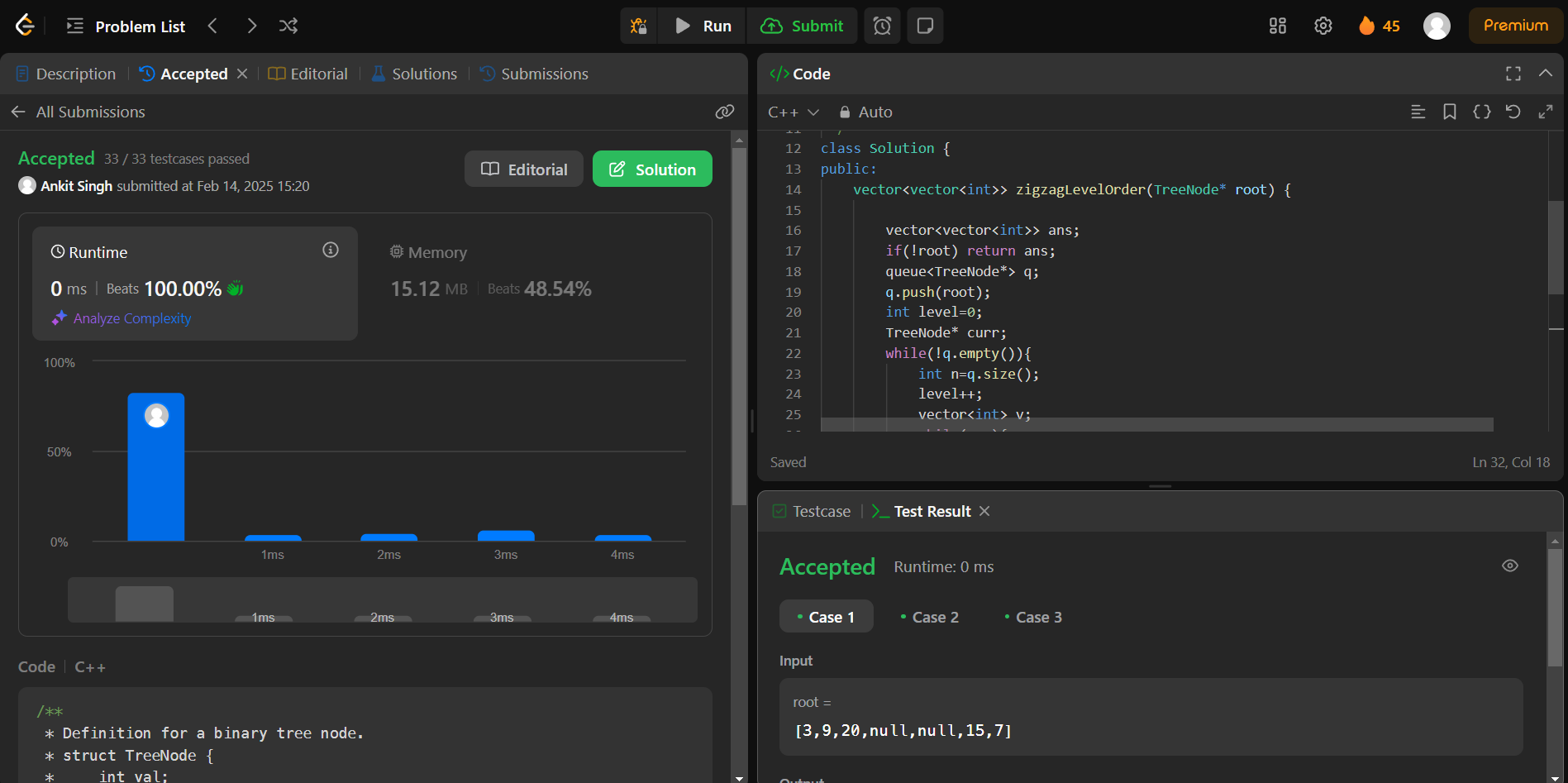
        }

        return ans;

    }

};

1. **Output:**

****

1. **Problem 9:** [Binary Tree Right Side View](https://leetcode.com/problems/binary-tree-right-side-view/) (199)
2. **Code:**

class Solution {

public:

    vector<int> res;

    unordered\_map<int,int> mp;

    void check(TreeNode\* root,int n){

        if(!root){

            return;

        }

        if(!(mp.find(n) != mp.end())){

            res.push\_back(root->val);

            mp[n]++;

        }

        check(root->right,n+1);

        check(root->left,n+1);

    }

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

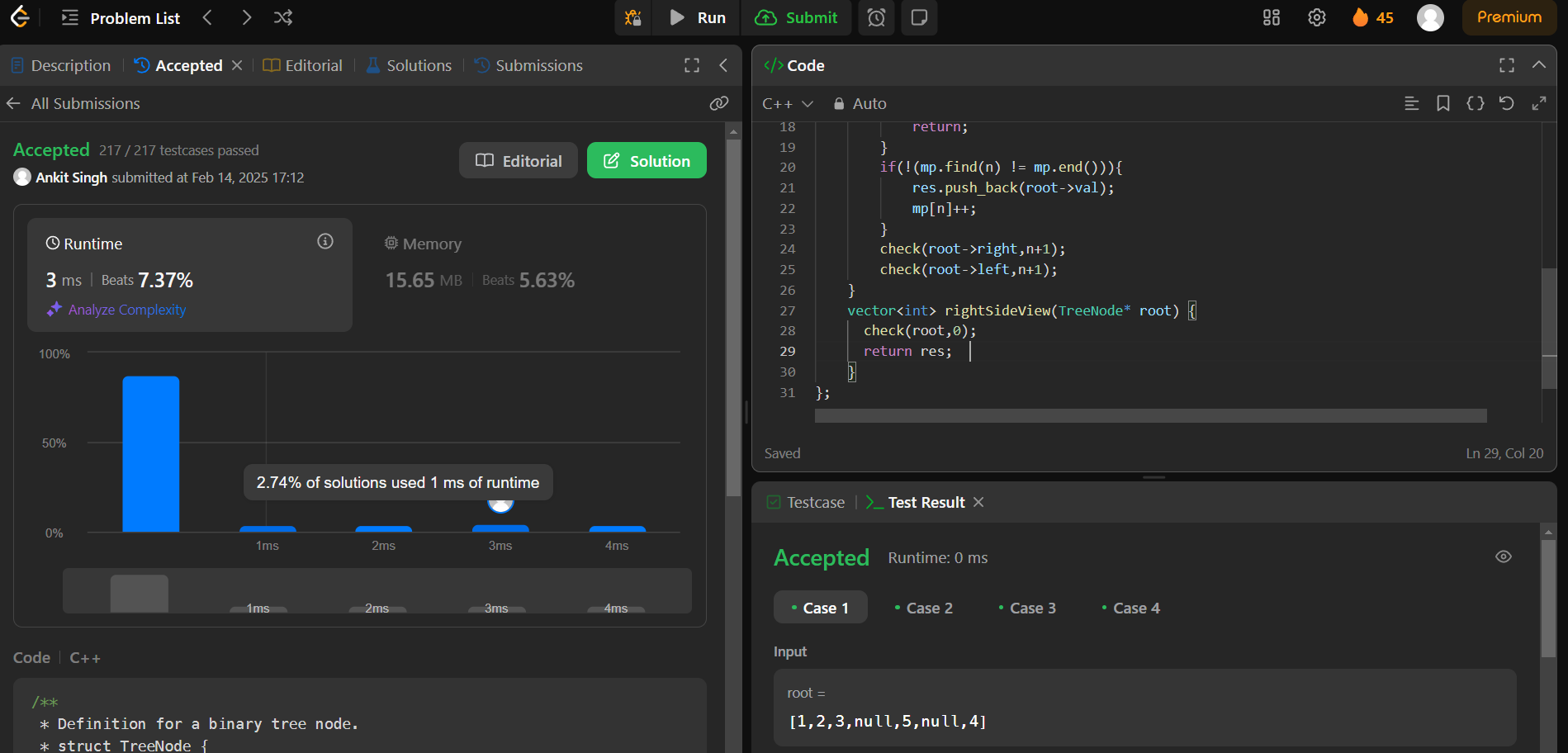
      check(root,0);

      return res;

    }

};

1. **Output:**

****

1. **Problem 10**: [Construct Binary Tree from Inorder and Postorder Traversal](https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/) (106)
2. **Code:\*\***

class Solution {

public:

    TreeNode\* buildTree(vector<int>& inorder, vector<int>& postorder) {

        unordered\_map<int, int> inorderIndexMap;

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

            inorderIndexMap[inorder[i]] = i;

        }

        int postIndex = postorder.size() - 1;

        return constructTree(inorder, postorder, inorderIndexMap, postIndex, 0, inorder.size() - 1);

    }

    TreeNode\* constructTree(vector<int>& inorder, vector<int>& postorder, unordered\_map<int, int>& inorderIndexMap, int& postIndex, int inStart, int inEnd) {

        if (inStart > inEnd) return nullptr;

        int rootVal = postorder[postIndex--];

        TreeNode\* root = new TreeNode(rootVal);

        int rootIndex = inorderIndexMap[rootVal];

        root->right = constructTree(inorder, postorder, inorderIndexMap, postIndex, rootIndex + 1, inEnd);

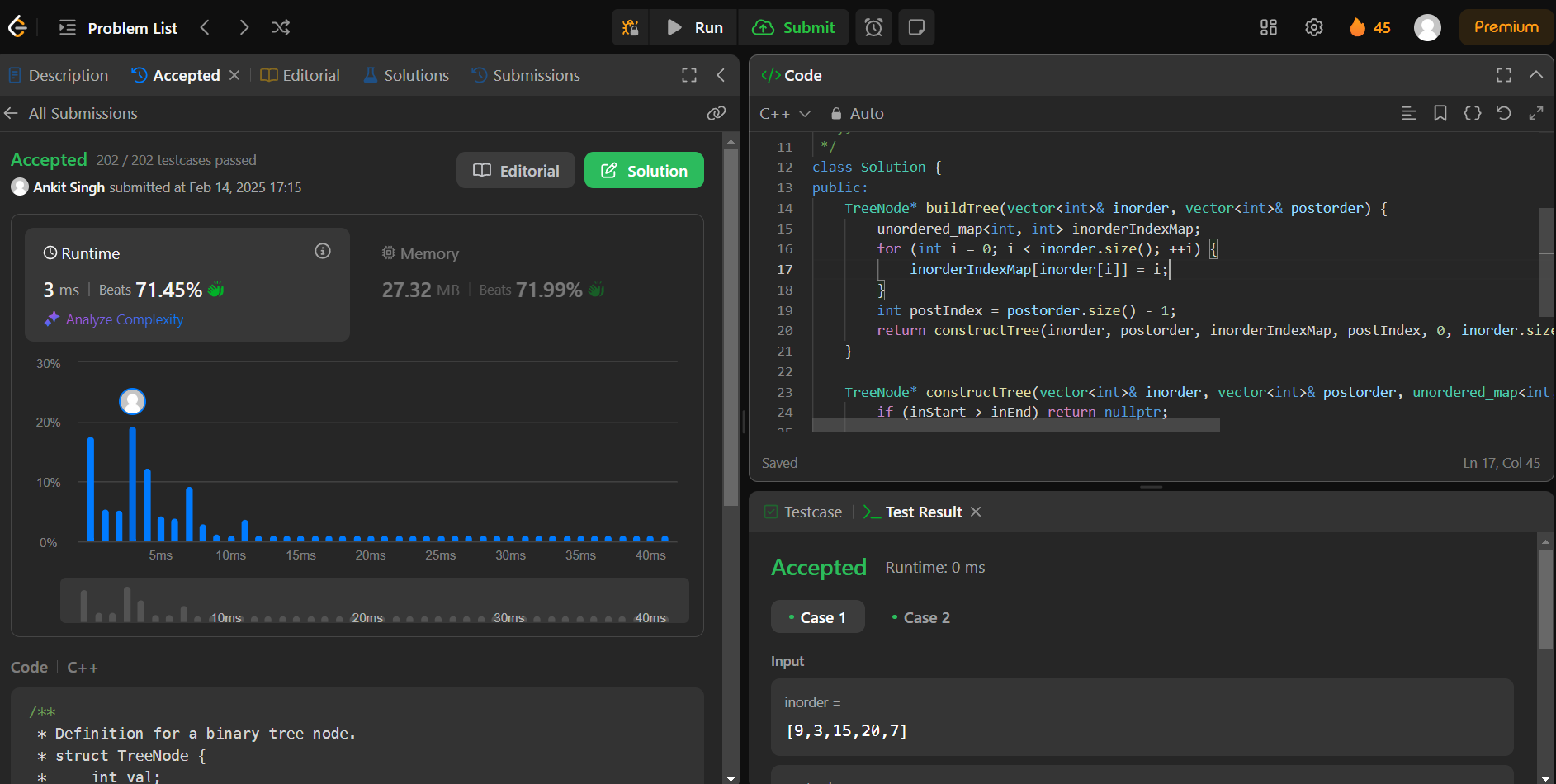
        root->left = constructTree(inorder, postorder, inorderIndexMap, postIndex, inStart, rootIndex - 1);

        return root;

    }

};

1. **Output:**

****

1. **Problem 11:** [Find Bottom Left Tree Value](https://leetcode.com/problems/find-bottom-left-tree-value/) (513)
2. **Code:**

class Solution {

public:

    void tt(TreeNode\* root, int level, vector<vector<int>>&nums){

        if(root==NULL){

            return;

        }

        if(nums.size()<=level){

            nums.push\_back({});

        }

        nums[level].push\_back(root->val);

        tt(root->right,level+1,nums);

        tt(root->left,level+1,nums);

    }

    int findBottomLeftValue(TreeNode\* root) {

       vector<vector<int>>nums;

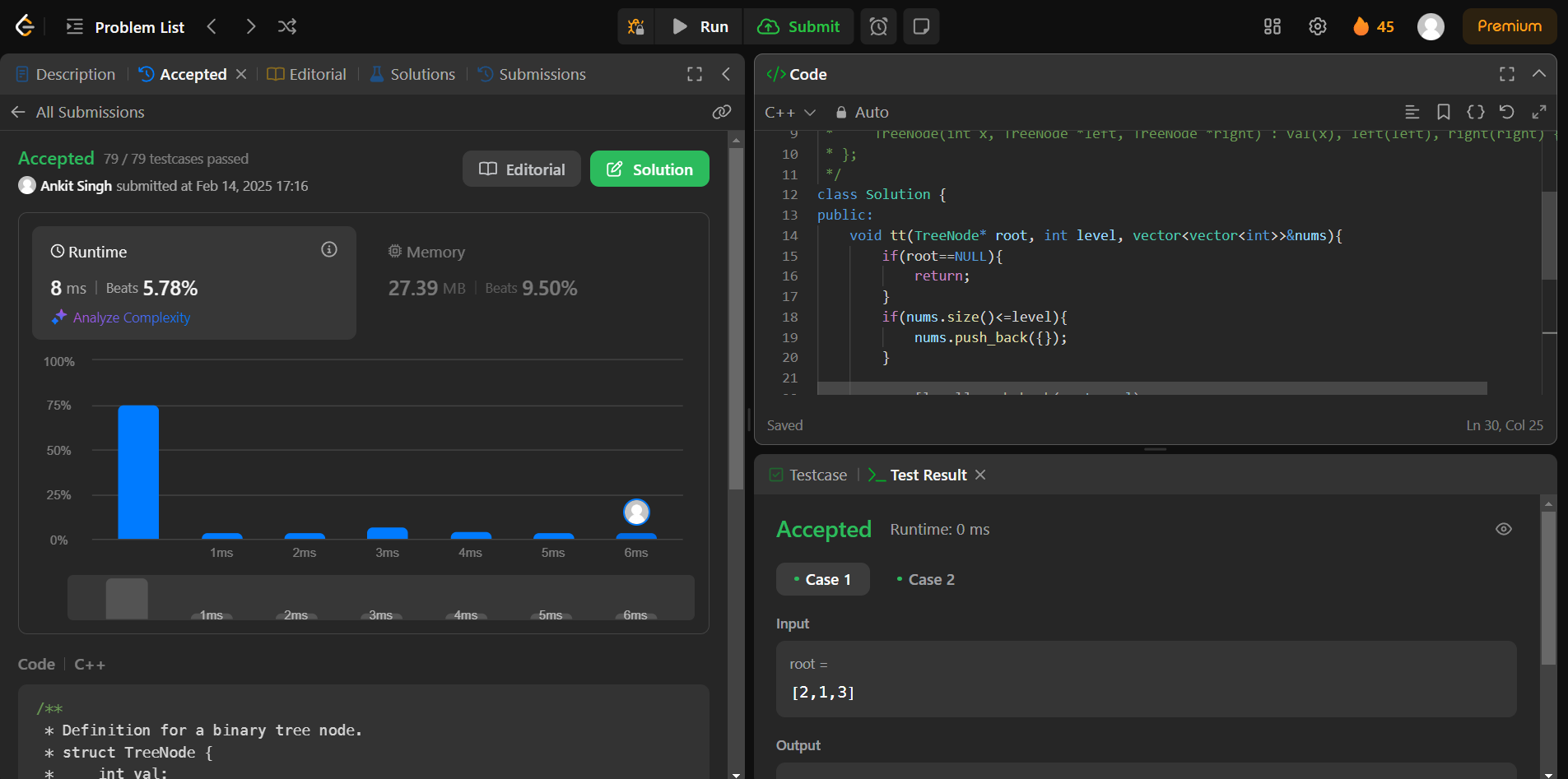
        tt(root,0,nums);

        return nums.back().back();

    }

};

1. **Output:**

****

1. **Problem 12:** [Binary Tree Maximum Path Sum](https://leetcode.com/problems/binary-tree-maximum-path-sum/) (124)
2. **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 size=q.size();

            vector<int> level;

            for(int i=0;i<size;i++){

                TreeNode\* node=q.front();

                q.pop();

                if(node->left!=NULL){q.push(node->left);}

                if(node->right!=NULL){q.push(node->right);}

                level.push\_back(node->val);

            }

            ans.push\_back(level);

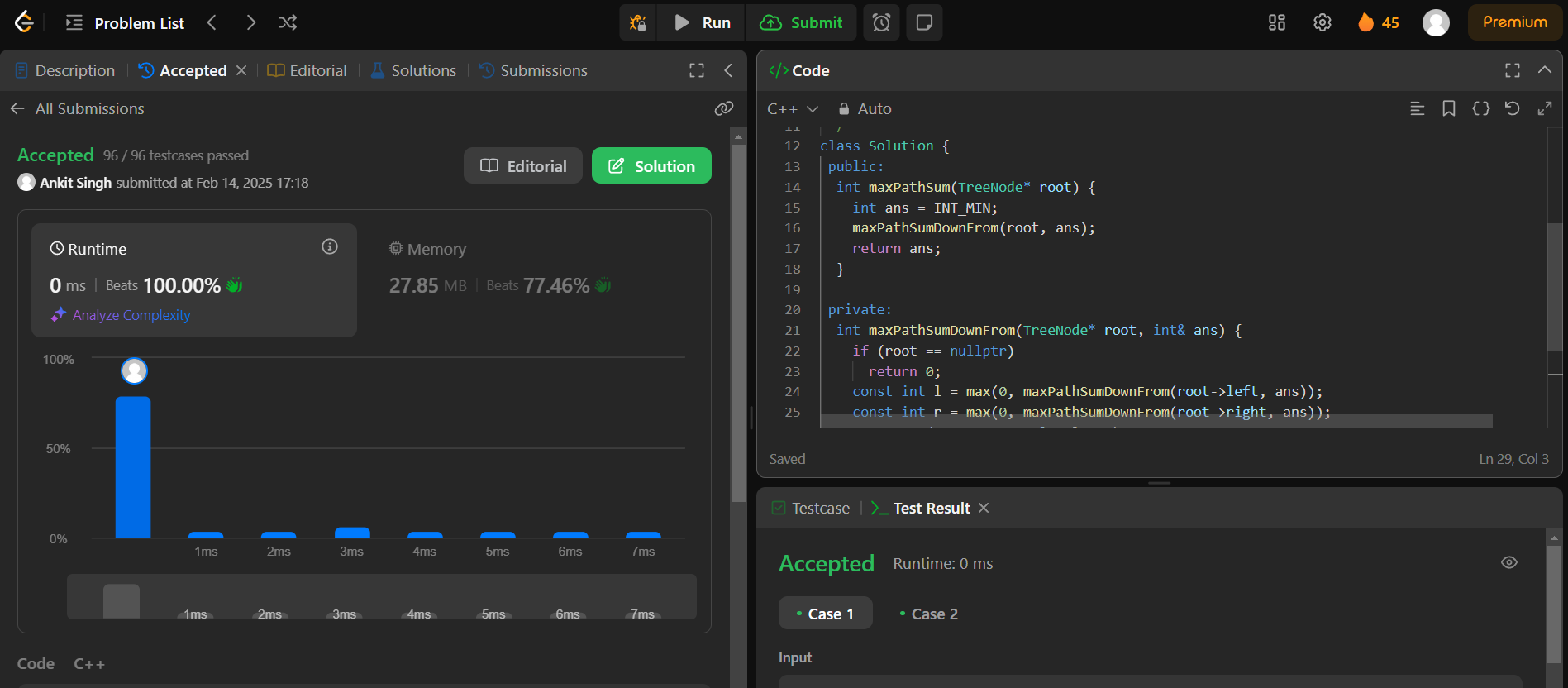
        }

    return ans;

    }

};

1. **Output:**

****

1. **Problem 13:** [Vertical Order Traversal of a Binary Tree](https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/) (987)
2. **Code:**

class Solution {

public:

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

        vector<vector<int>> ans;

        queue<pair<TreeNode\*,int>> Q; // node and col

        Q.push({root,0});

        int depth=0;

        while(!Q.empty()){

            int s=Q.size();

            while(s--){

                auto [node,col]=Q.front();

                Q.pop();

                ans.push\_back({col,depth,node->val});

                if(node->left!=nullptr) Q.push({node->left,col-1});

                if(node->right!=nullptr) Q.push({node->right,col+1});

            }

            depth++;

        }

        sort(ans.begin(),ans.end());

        vector<vector<int>> final;

        vector<int> temp;

        int curr=ans[0][0];

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

            if(ans[i][0]==curr) temp.push\_back(ans[i][2]);

            else{

                final.push\_back(temp);

                temp.clear();

                curr=ans[i][0];

                temp.push\_back(ans[i][2]);

            }

        }

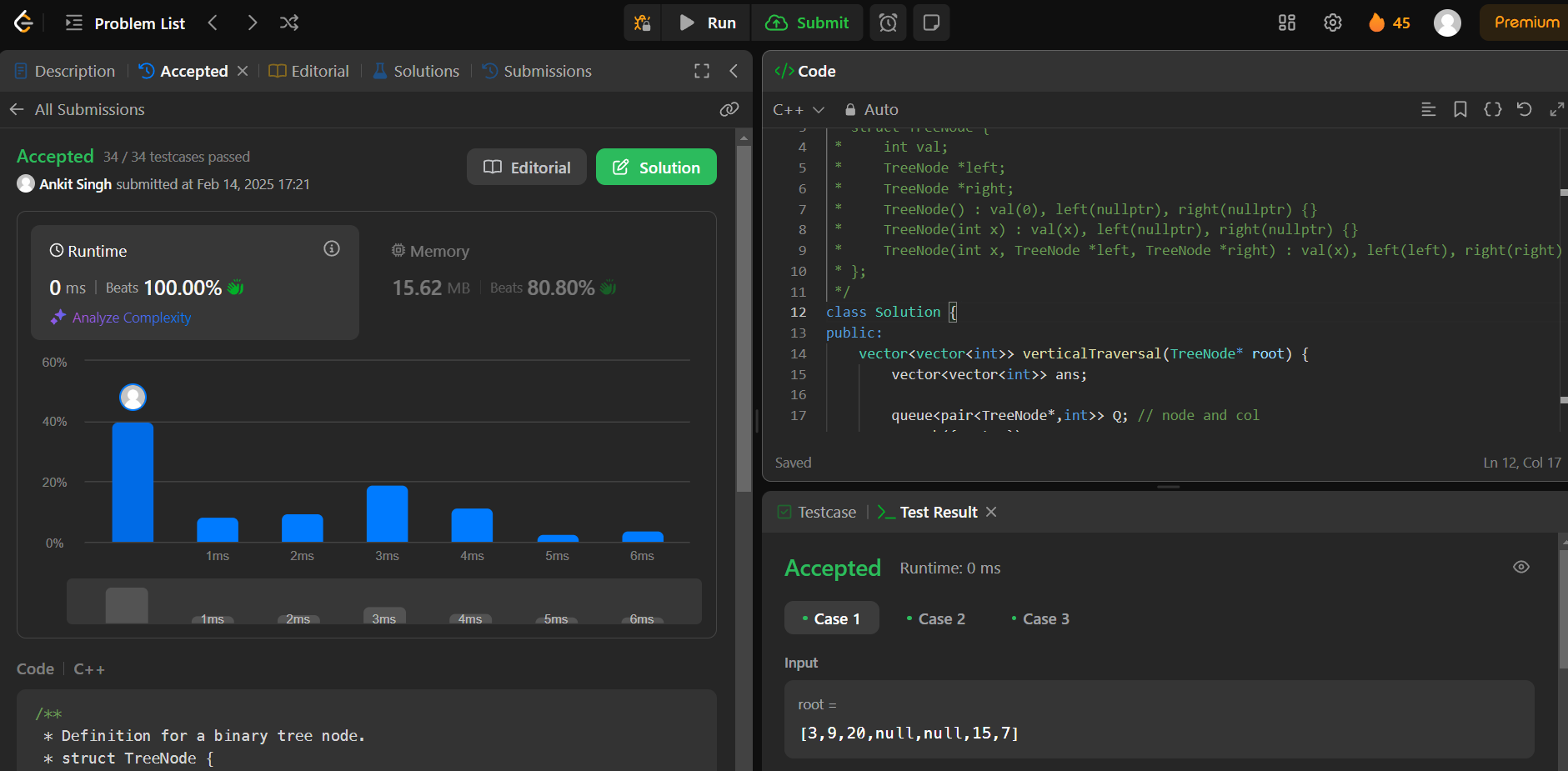
        if(!temp.empty()) final.push\_back(temp);

        return final;

    }

};

1. **Output:**

****