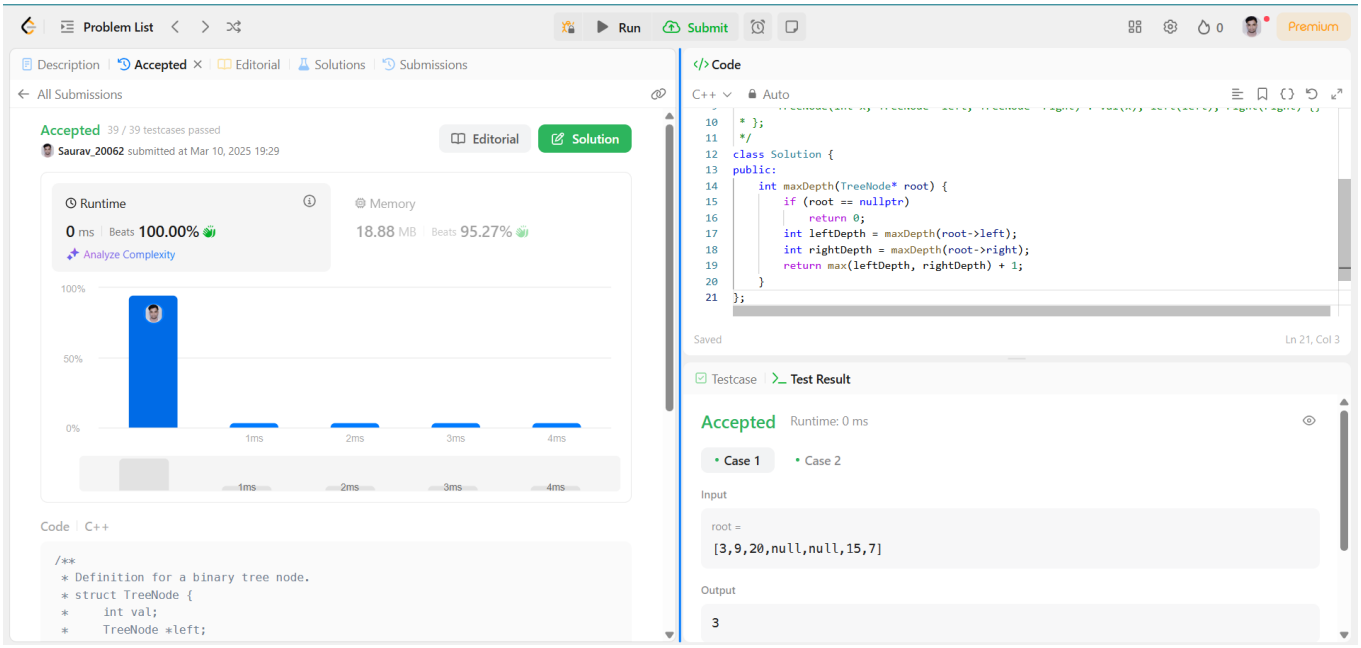
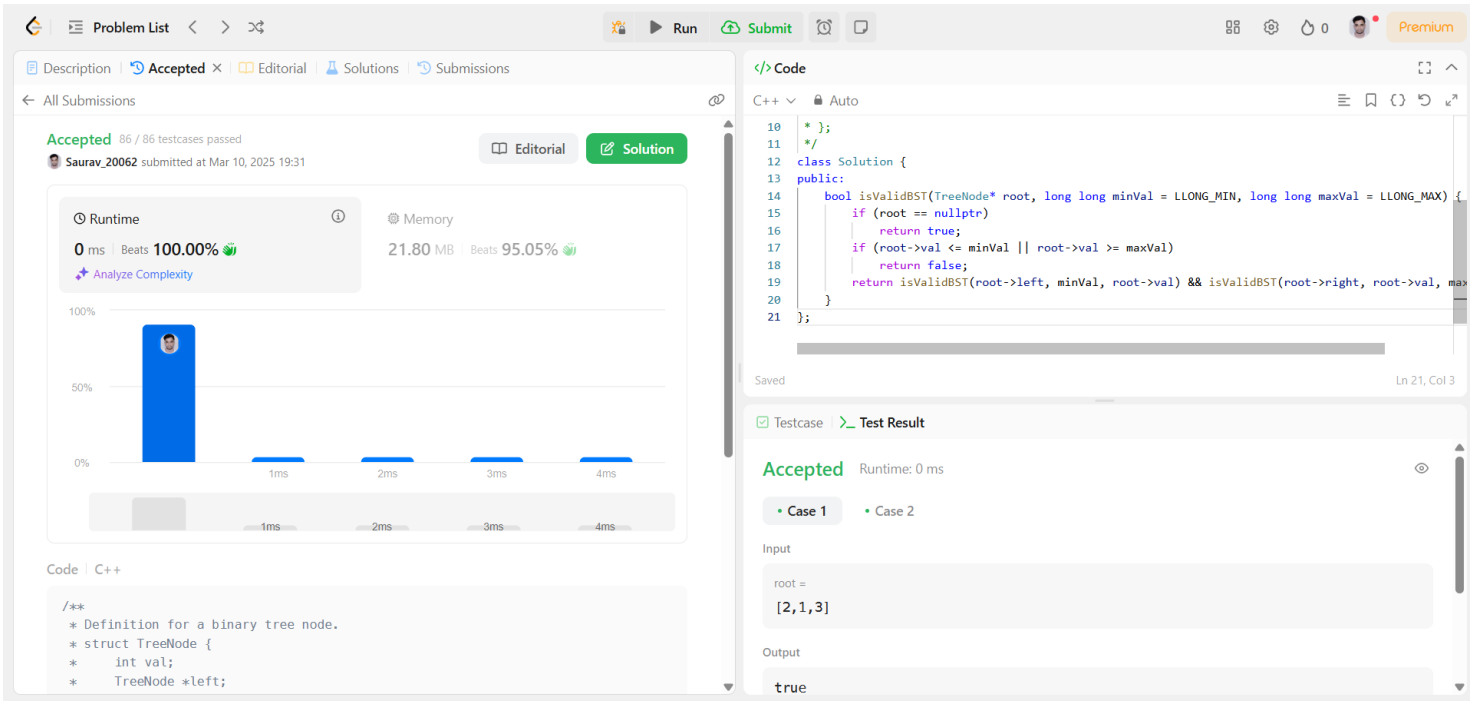
Experiment 5

Student Name: Saurav Ashiwal **UID: 22BCS13250**
Branch: CSE **Section/Group: 614/B**
Semester: 6 **Date of Performance: 9/03/2025**
Subject Name: AP lab-2 **Subject Code: 22CSP-351**

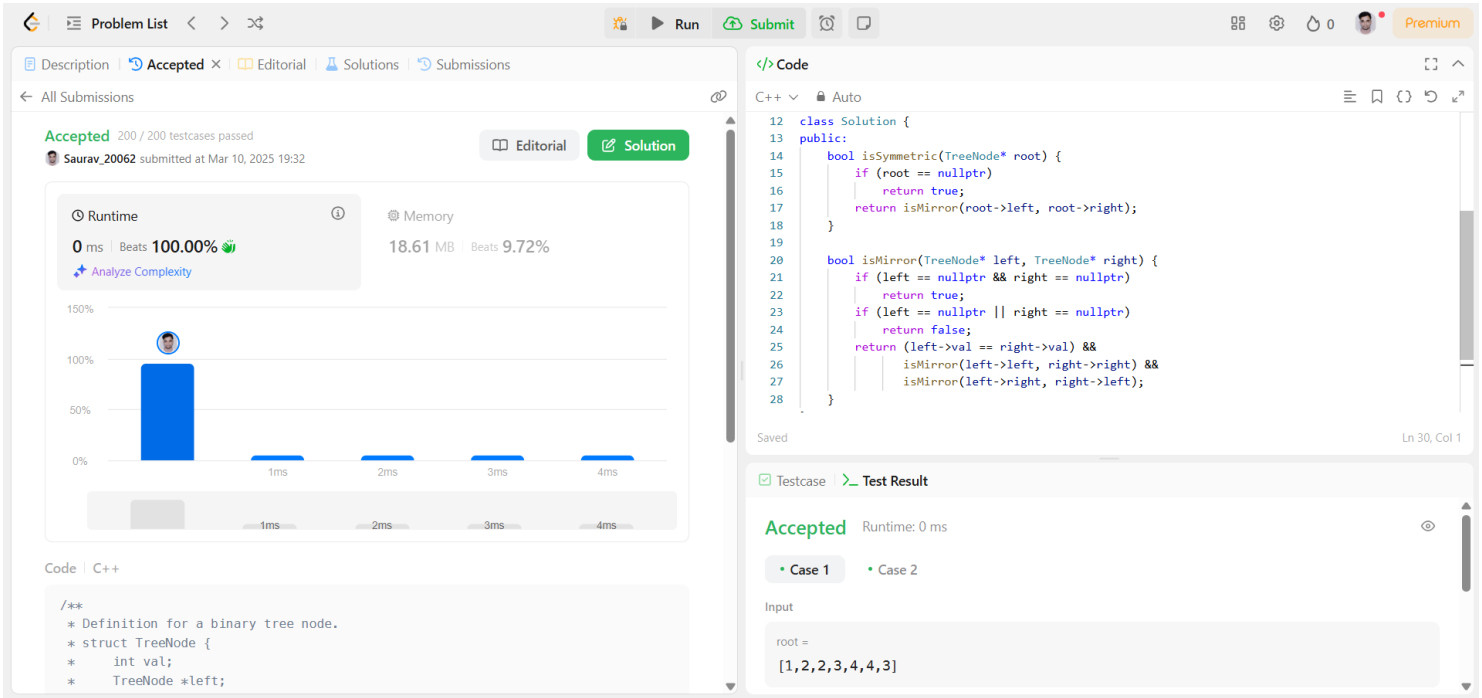
Q 1. .Maximum Depth of Binary Tree



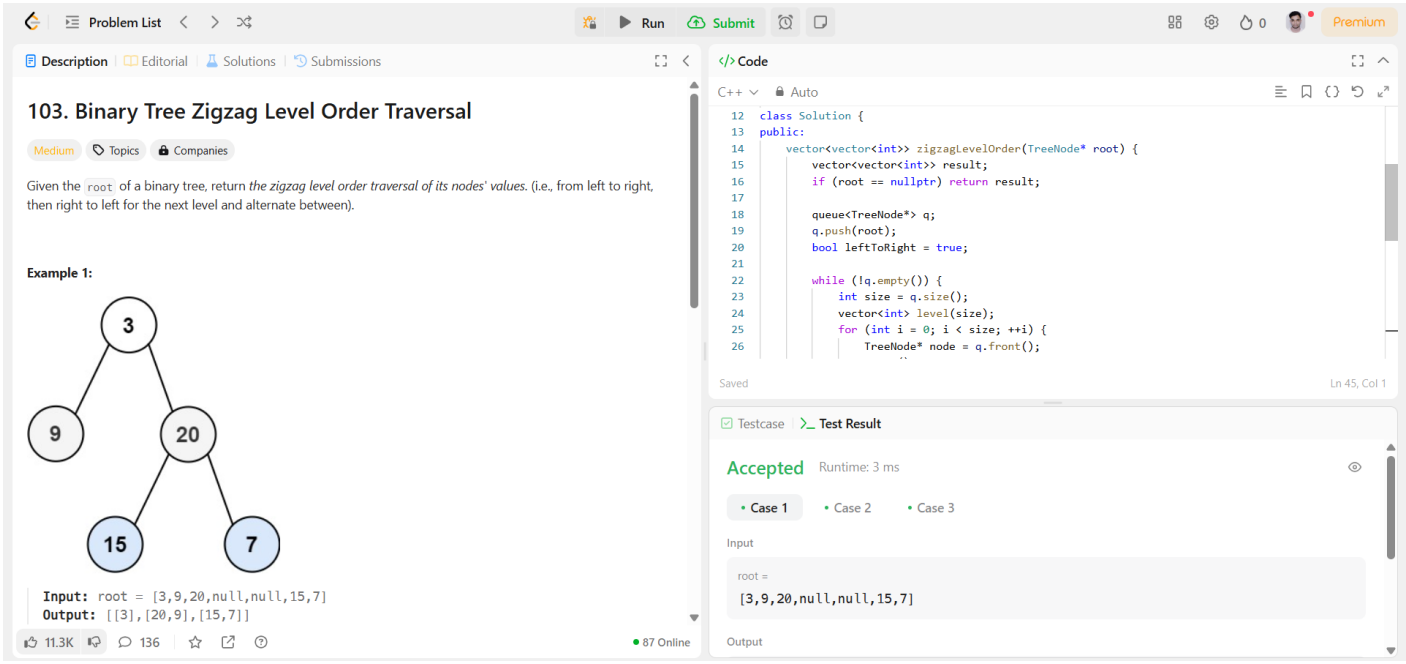
Q 2. Validate Binary Search Tree



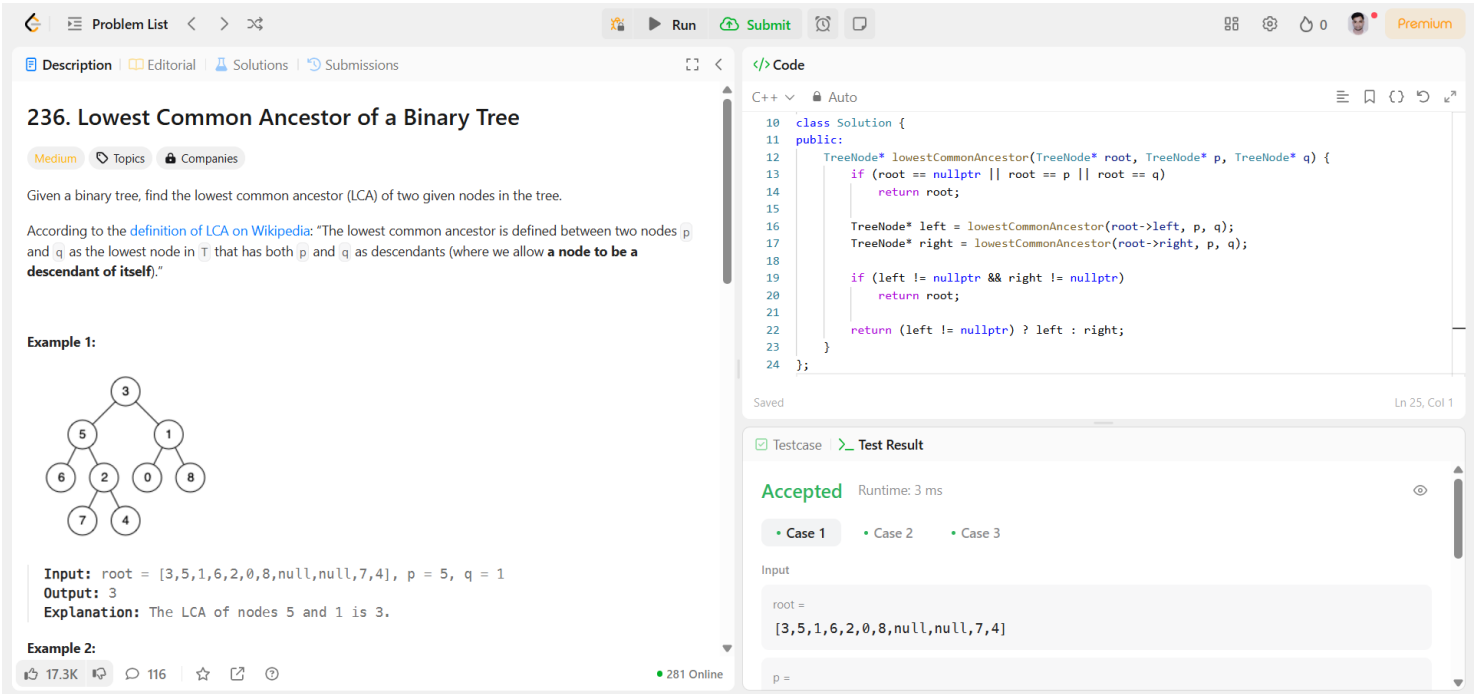
Q 3. Symmetric Tree



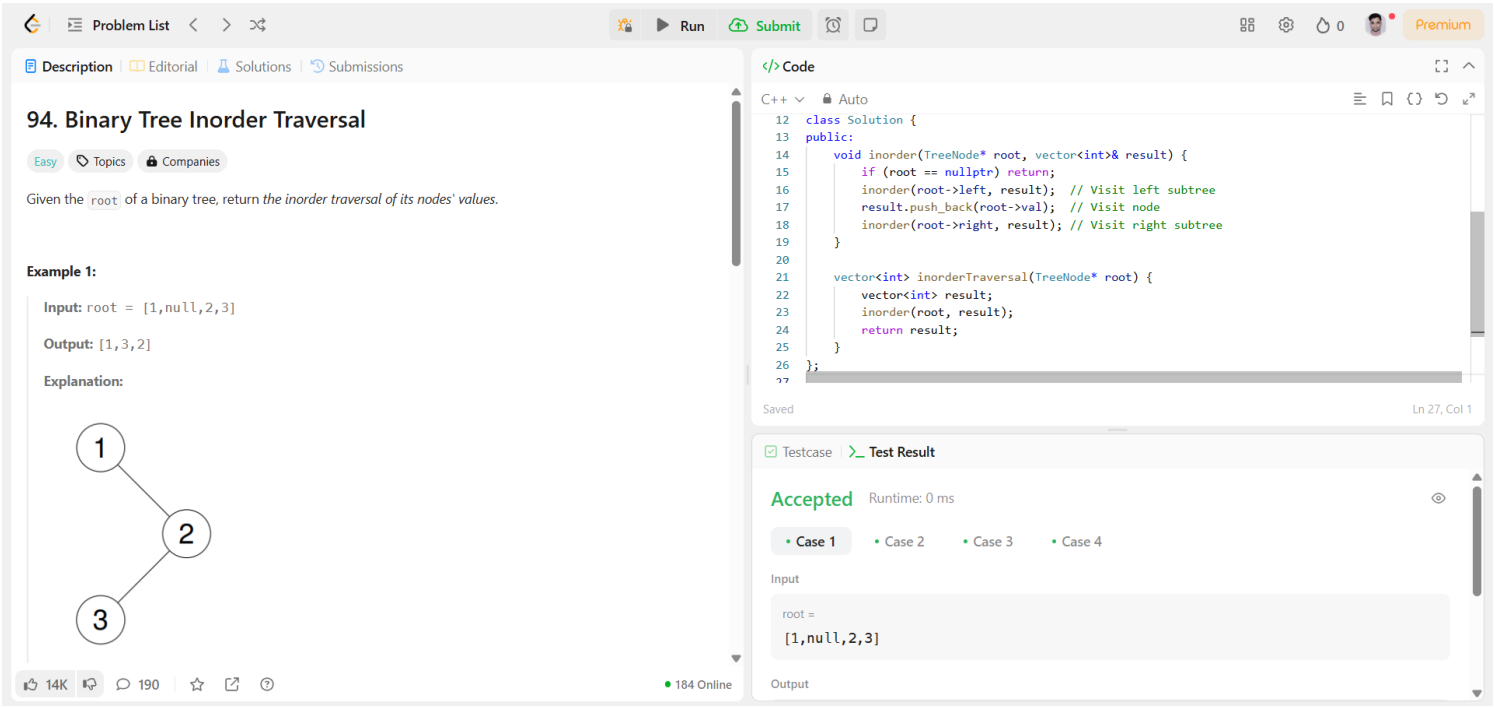
Q 4. Binary Tree Zigzag Level Order Traversal



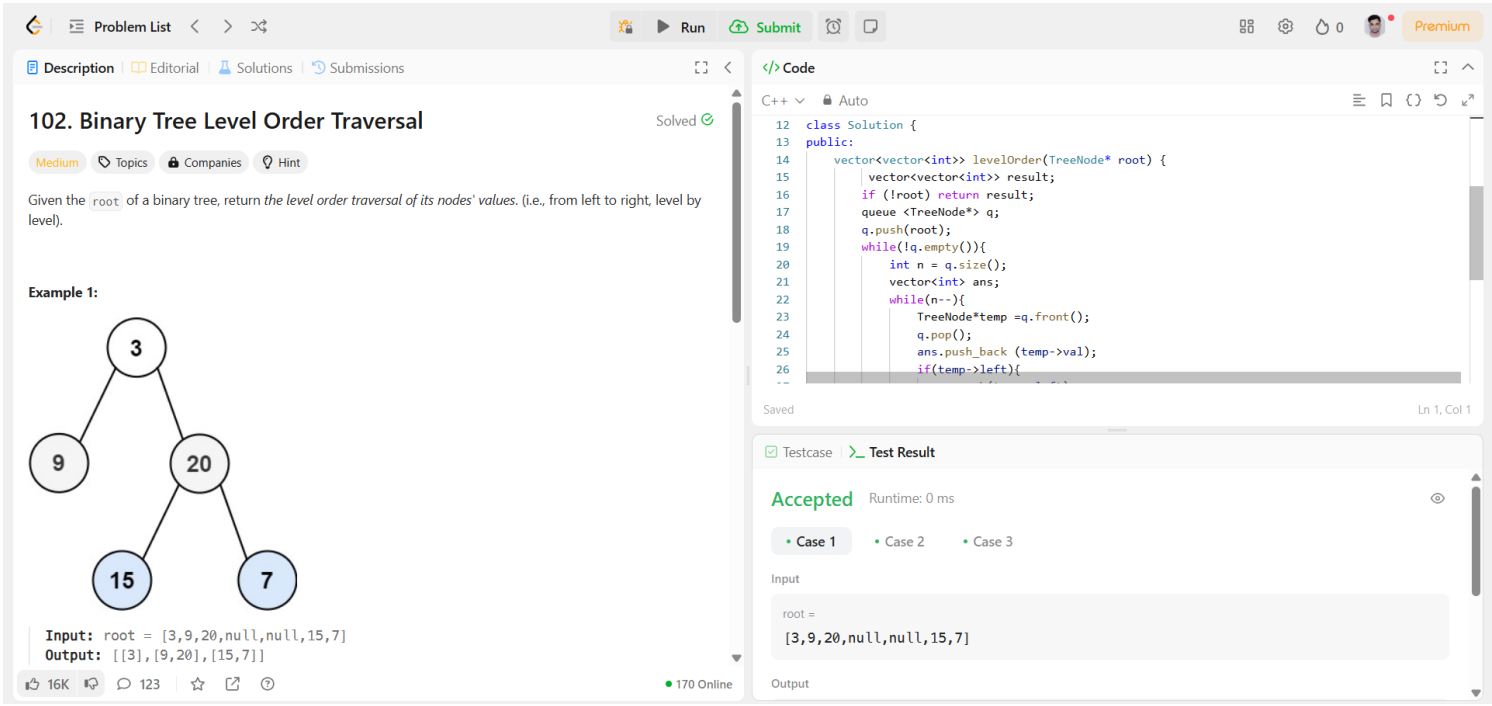
Q 5. Lowest Common Ancestor of a Binary Tree



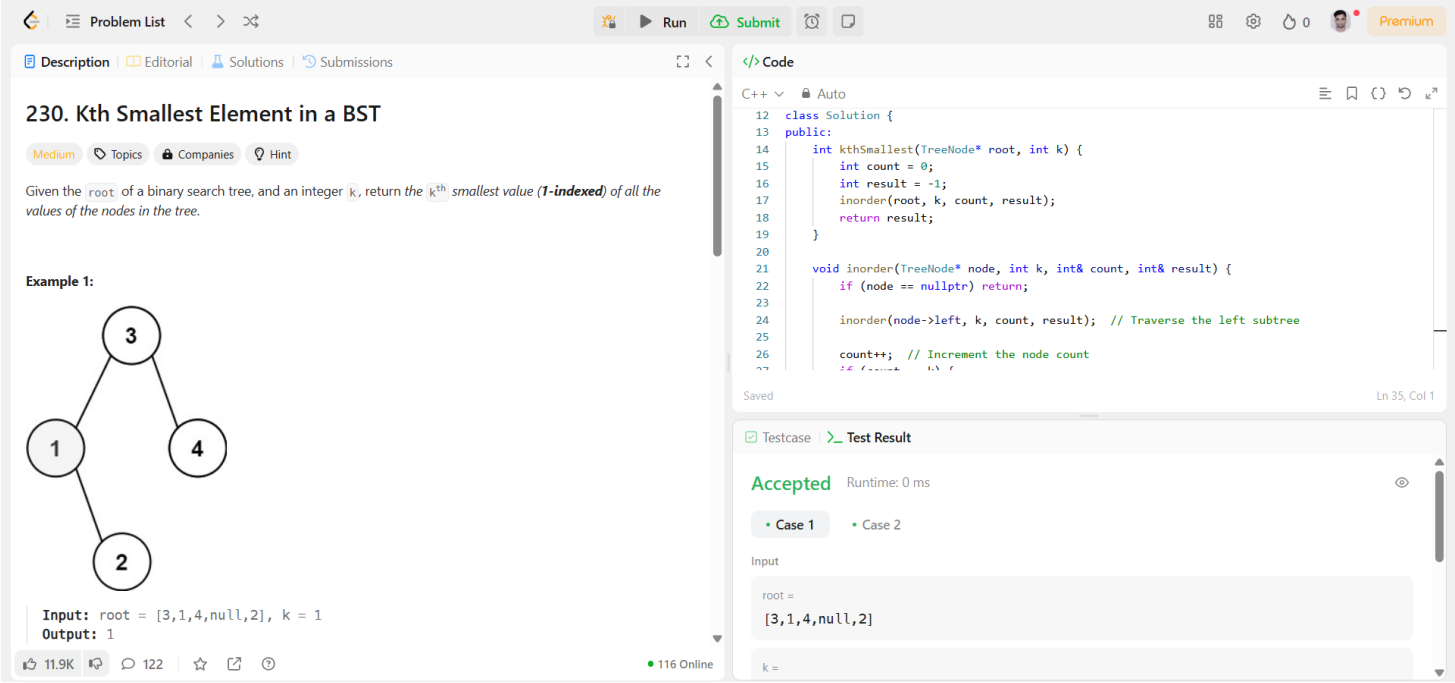
Q 6. Binary Tree Inorder Traversal



Q7. Binary Tree Level Order Traversal



Q 8. Kth Smallest Element in a BST



230. Kth Smallest Element in a BST

Medium | Topics | Companies | Hint

Given the `root` of a binary search tree, and an integer `k`, return the k^{th} *smallest value (**1-indexed**) of all the values of the nodes in the tree.*

Example 1:


```
Input: root = [3,1,4,null,2], k = 1
Output: 1
```

```cpp
class Solution {
public:
    int kthSmallest(TreeNode* root, int k) {
        int count = 0;
        int result = -1;
        inorder(root, k, count, result);
        return result;
    }

    void inorder(TreeNode* node, int k, int& count, int& result) {
        if (node == nullptr) return;

        inorder(node->left, k, count, result);  // Traverse the left subtree

        count++;  // Increment the node count
```

Testcase | Test Result

Accepted Runtime: 0 ms

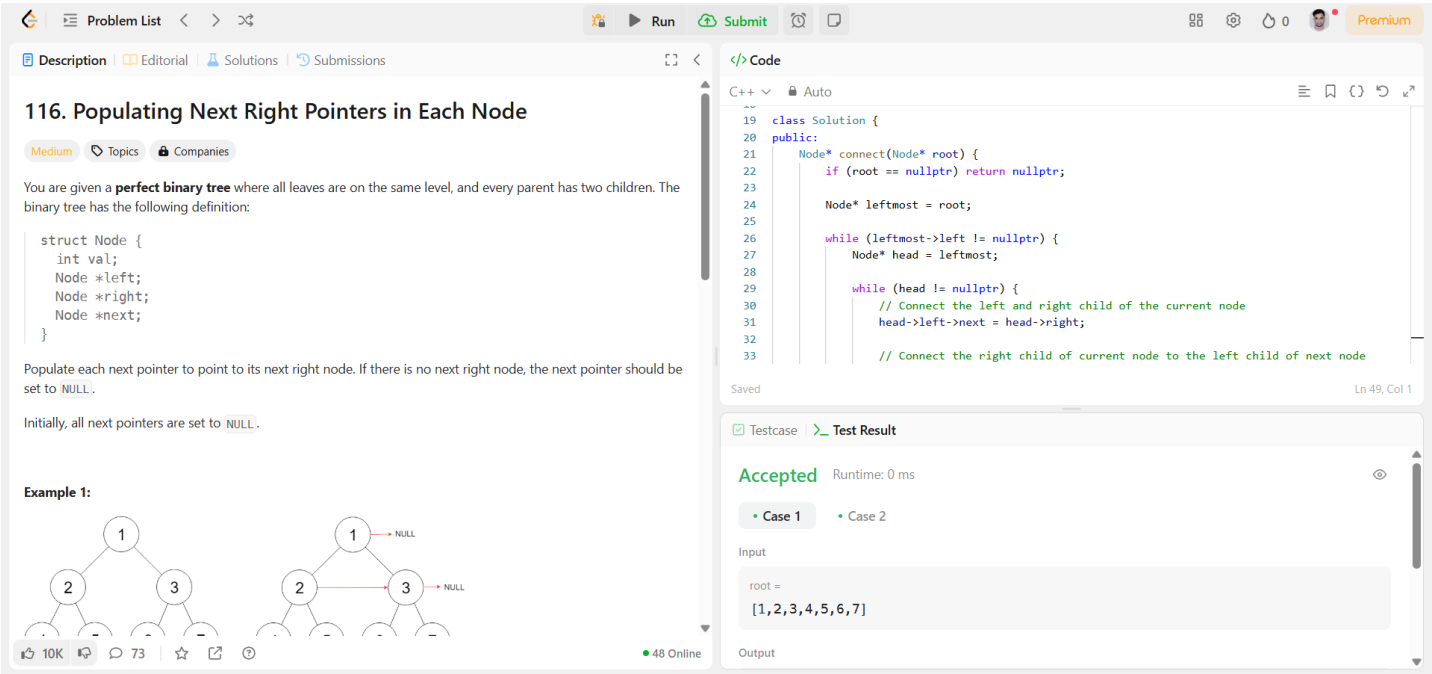
Case 1 | Case 2

Input

root =
[3,1,4,null,2]

k =

Q 9. Populating Next Right Pointers in Each Node



116. Populating Next Right Pointers in Each Node

Medium | Topics | Companies

You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to `NULL`.

Initially, all next pointers are set to `NULL`.

Example 1:


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

        Node* leftmost = root;

        while (leftmost->left != nullptr) {
            Node* head = leftmost;

            while (head != nullptr) {
                // Connect the left and right child of the current node
                head->left->next = head->right;

                // Connect the right child of current node to the left child of next node
```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[1,2,3,4,5,6,7]

Output

Q 10. Sum of Left Leaves

Problem List | Run | Submit | Premium

Description | Editorial | Solutions | Submissions

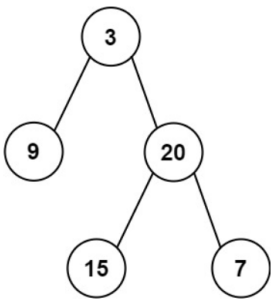
404. Sum of Left Leaves

Easy | Topics | Companies

Given the `root` of a binary tree, return *the sum of all left leaves.*

A **leaf** is a node with no children. A **left leaf** is a leaf that is the left child of another node.

Example 1:



Input: root = [3,9,20,null,null,15,7]

5.6K | 92 | 21 Online

Code

C++ | Auto

```
*/
class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {
        if (root == nullptr) return 0;

        int sum = 0;

        // Check if the left child is a leaf
        if (root->left && !root->left->left && !root->left->right) {
            sum += root->left->val;
        }

        // Recursively find the sum of left leaves in the left and right subtrees
        sum += sumOfLeftLeaves(root->left);
```

Saved | Ln 31, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

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

Output