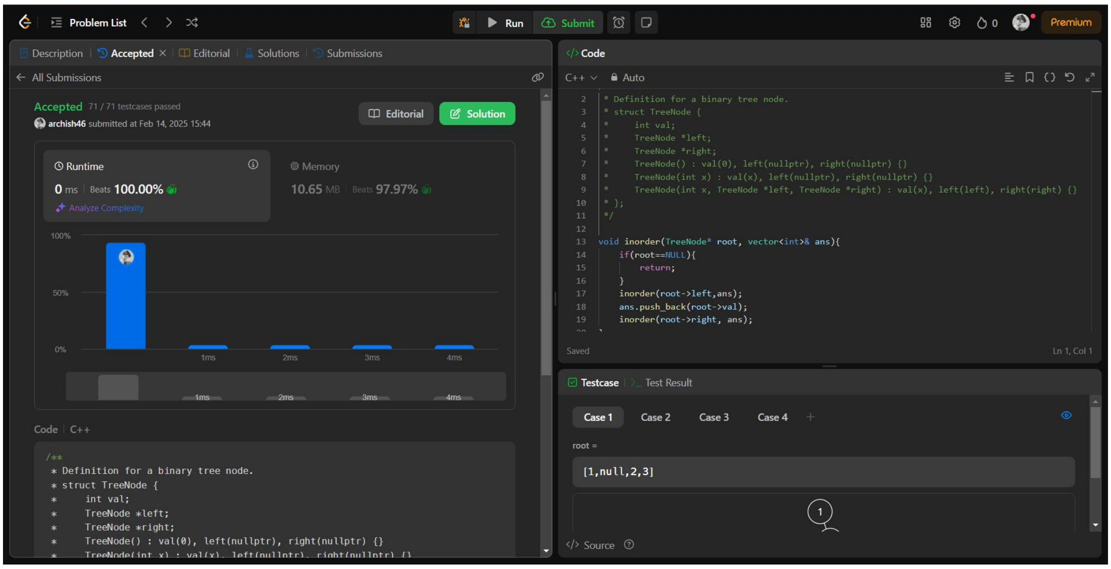
NAME: Archishman Mukherjee uid: 22BCS11671 Sec- 605(B)

1. Binary Tree Inorder Traversal



Accepted 71 / 71 testcases passed

archish46 submitted at Feb 14, 2025 15:44

Runtime 0 ms | Beats 100.00%

Memory 10.65 MB | Beats 97.97%

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) {}
```

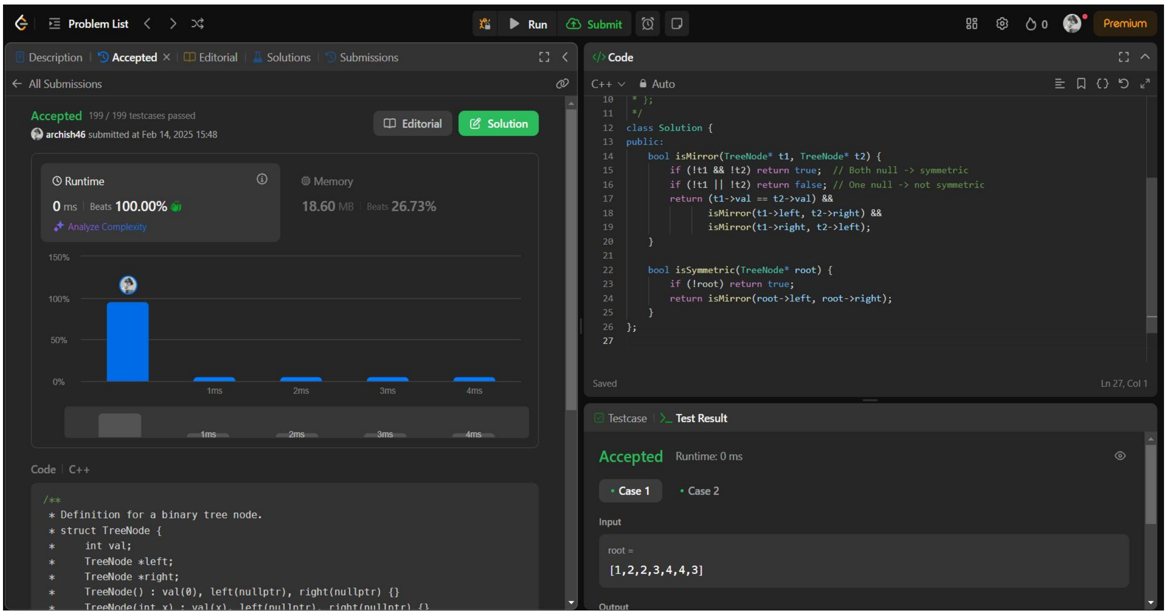
```
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode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

void inorder(TreeNode* root, vector<int>& ans){
    if(root==NULL){
        return;
    }
    inorder(root->left,ans);
    ans.push_back(root->val);
    inorder(root->right, ans);
```

Saved   Ln 1, Col 1

root =

```
[1,null,2,3]
```

2. Symmetric Tree



Accepted 199 / 199 testcases passed

archish46 submitted at Feb 14, 2025 15:48

Runtime 0 ms | Beats 100.00%

Memory 18.60 MB | Beats 26.73%

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) {}
```

```
 * };
 */
class Solution {
public:
    bool isMirror(TreeNode* t1, TreeNode* t2) {
        if (!t1 && !t2) return true;  // Both null -> symmetric
        if (!t1 || !t2) return false; // One null -> not symmetric
        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);
    }
};
```

Saved   Ln 27, Col 1

Accepted Runtime: 0 ms

• Case 1 • Case 2

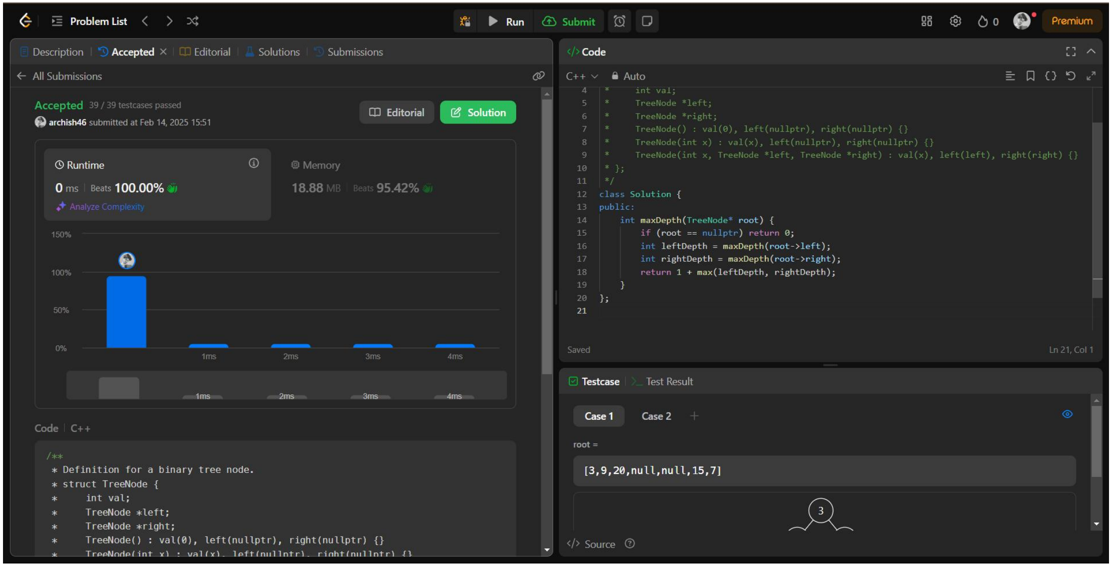
Input

root =

```
[1,2,2,3,4,4,3]
```

Output

3. Maximum Depth of binary tree



Accepted 39 / 39 testcases passed

archish46 submitted at Feb 14, 2025 15:51

Runtime 0 ms | Beats 100.00%

Memory 18.88 MB | Beats 95.42%

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) {}
```

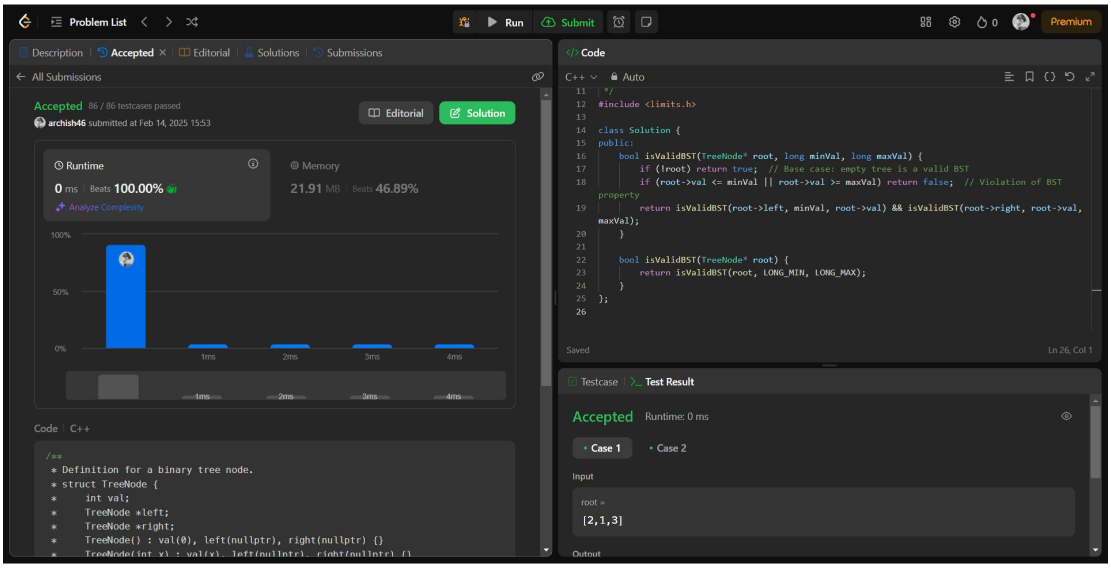
```
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if (root == nullptr) return 0;
        int leftDepth = maxDepth(root->left);
        int rightDepth = maxDepth(root->right);
        return 1 + max(leftDepth, rightDepth);
    }
};
```

Saved   Ln 21, Col 1

root =

```
[3,9,20,null,null,15,7]
```

4. Validate binary search tree



Accepted 86 / 86 testcases passed
archish46 submitted at Feb 14, 2025 15:53

Runtime
0 ms | Beats 100.00%

Memory
21.91 MB | Beats 46.89%

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) {}
```

```
 */
#include <limits.h>

class Solution {
public:
    bool isValidBST(TreeNode* root, long minVal, long maxVal) {
        if (!root) return true;  // Base case: empty tree is a valid BST
        if (root->val <= minVal || root->val >= maxVal) return false;  // Violation of BST property
        return isValidBST(root->left, minVal, root->val) && isValidBST(root->right, root->val, maxVal);
    }

    bool isValidBST(TreeNode* root) {
        return isValidBST(root, LONG_MIN, LONG_MAX);
    }
};
```

Accepted Runtime: 0 ms

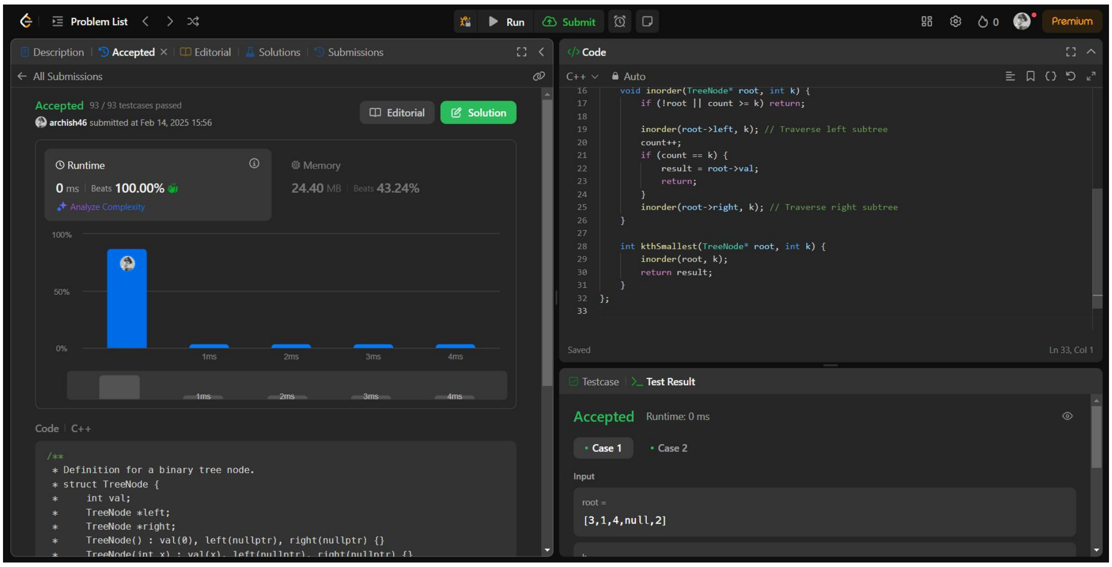
• Case 1 • Case 2

Input

root =
[2,1,3]

Output

5. Kth Smallest Element in a BST



Accepted 93 / 93 testcases passed
archish46 submitted at Feb 14, 2025 15:56

Runtime
0 ms | Beats 100.00%

Memory
24.40 MB | Beats 43.24%

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) {}
```

```
    void inorder(TreeNode* root, int k) {
        if (!root || count >= k) return;

        inorder(root->left, k); // Traverse left subtree
        count++;
        if (count == k) {
            result = root->val;
            return;
        }
        inorder(root->right, k); // Traverse right subtree
    }

    int kthSmallest(TreeNode* root, int k) {
        inorder(root, k);
        return result;
    }
};
```

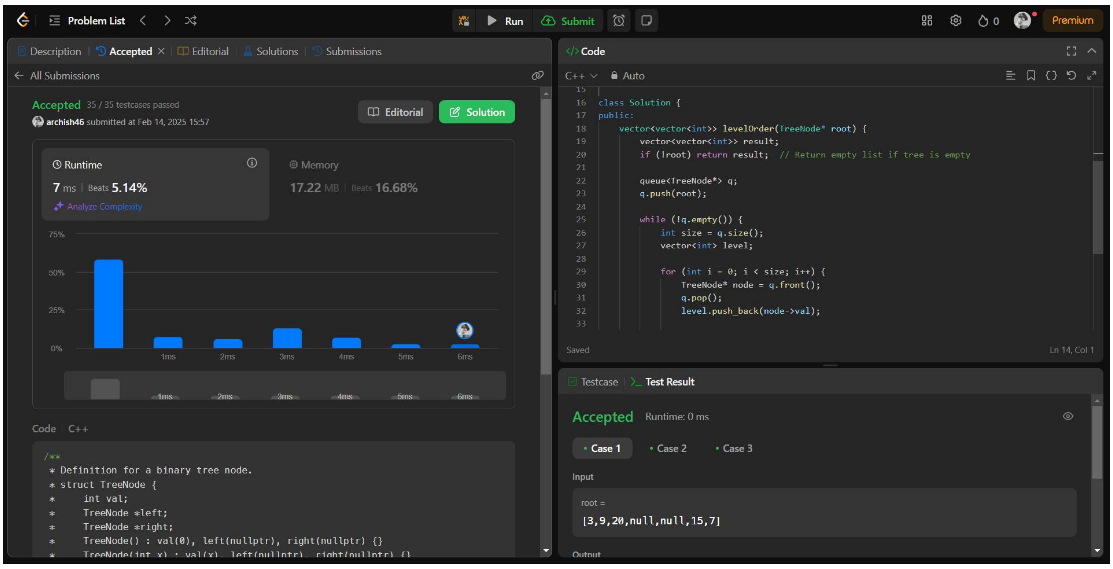
Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

6. Binary Tree Level Order Traversal



Accepted 35 / 35 testcases passed
archish46 submitted at Feb 14, 2025 15:57

Runtime
7 ms | Beats 5.14%

Memory
17.22 MB | Beats 16.68%

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) {}
```

```

class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> result;
        if (!root) return result;  // Return empty list if tree is empty

        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();
                level.push_back(node->val);

```

Accepted Runtime: 0 ms

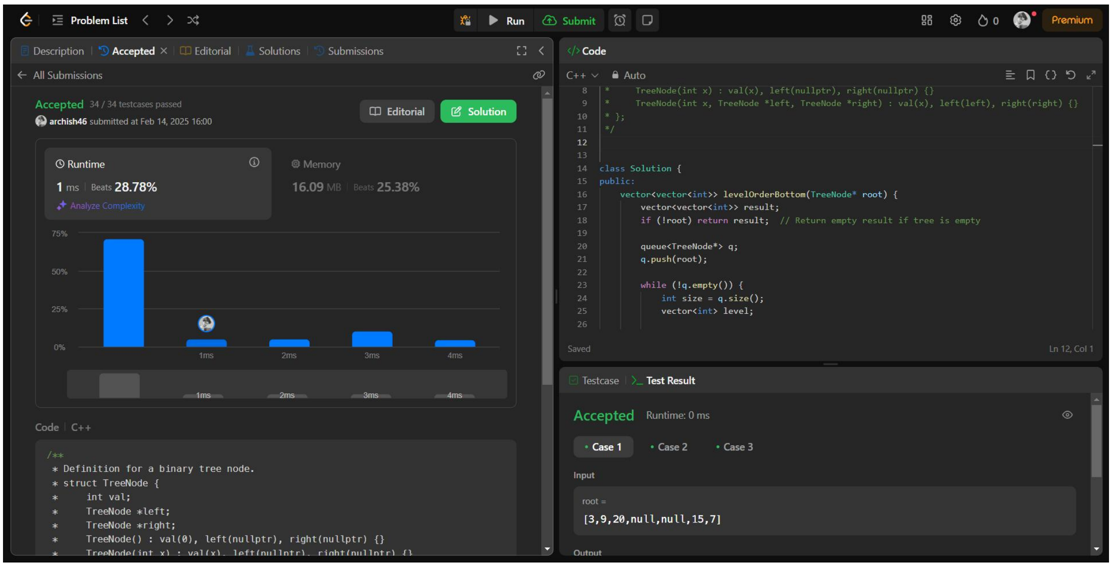
• Case 1 • Case 2 • Case 3

Input

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

Output

7. Binary Tree Level Order Traversal II



Accepted 34 / 34 testcases passed

archish46 submitted at Feb 14, 2025 16:00

Runtime

1 ms | Beats 28.78%

Memory

16.09 MB | Beats 25.38%

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) : 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>> levelOrderBottom(TreeNode* root) {
        vector<vector<int>> result;
        if (!root) return result;  // Return empty result if tree is empty

        queue<TreeNode*> q;
        q.push(root);

        while (!q.empty()) {
            int size = q.size();
            vector<int> level;

```

Accepted Runtime: 0 ms

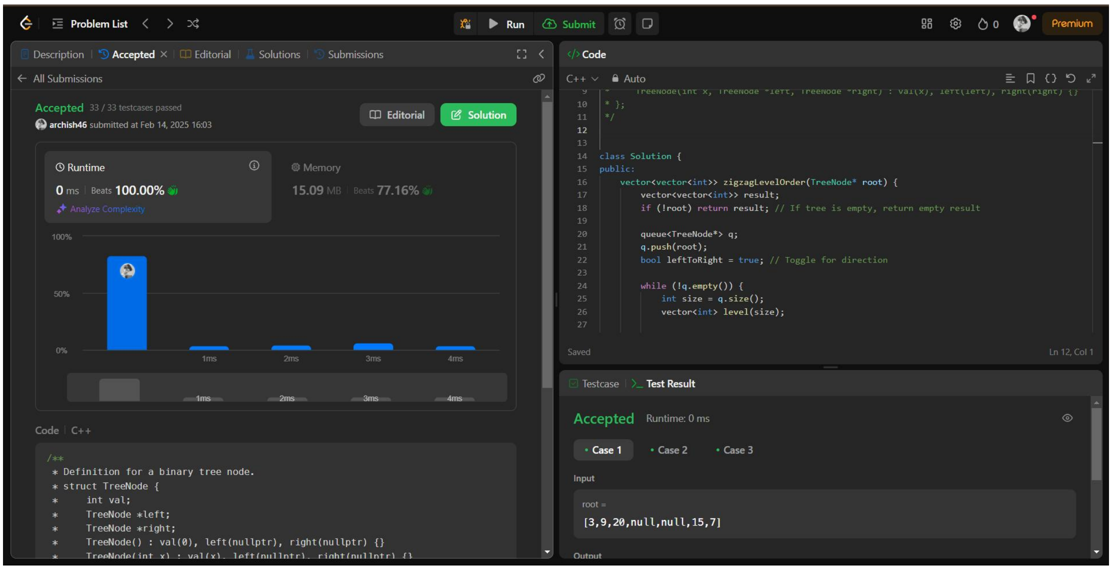
Input

root =

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

Output

8. Binary tree zigzag level order traversal



Accepted 33 / 33 testcases passed

archish46 submitted at Feb 14, 2025 16:03

Runtime

0 ms | Beats 100.00%

Memory

15.09 MB | Beats 77.16%

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>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>> result;
        if (!root) return result; // If tree is empty, return empty result

        queue<TreeNode*> q;
        q.push(root);
        bool leftToRight = true; // Toggle for direction

        while (!q.empty()) {
            int size = q.size();
            vector<int> level(size);

```

Accepted Runtime: 0 ms

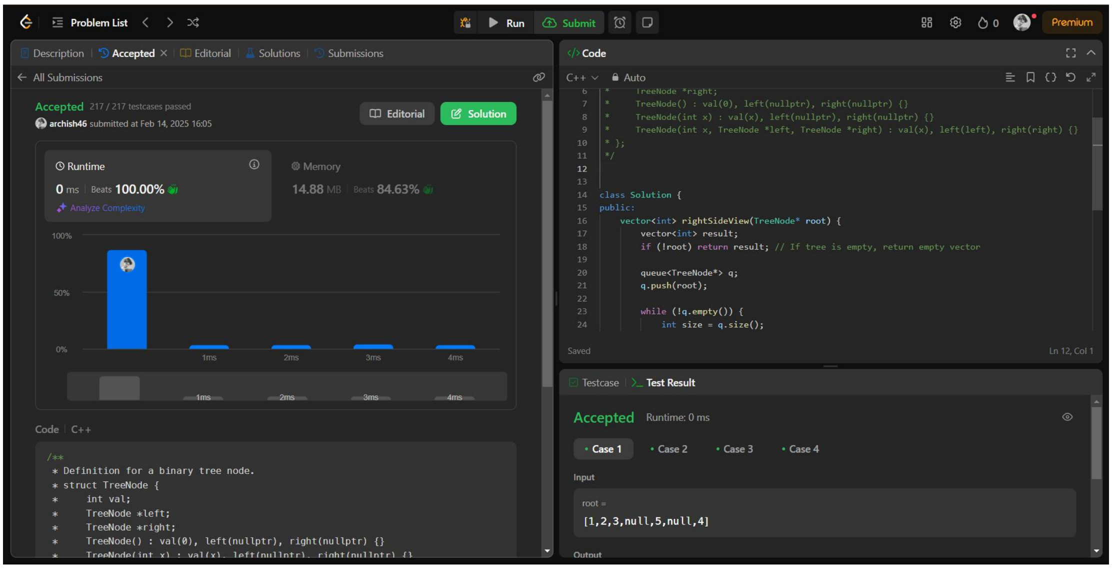
Input

root =

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

Output

9. Binary Tree Right Side View



Accepted 217 / 217 testcases passed

archish46 submitted at Feb 14, 2025 16:05

Runtime

0 ms | Beats 100.00%

Memory

14.88 MB | Beats 84.63%

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 *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<int> rightSideView(TreeNode* root) {
        vector<int> result;
        if (!root) return result; // If tree is empty, return empty vector

        queue<TreeNode*> q;
        q.push(root);

        while (!q.empty()) {
            int size = q.size();
```

Accepted Runtime: 0 ms

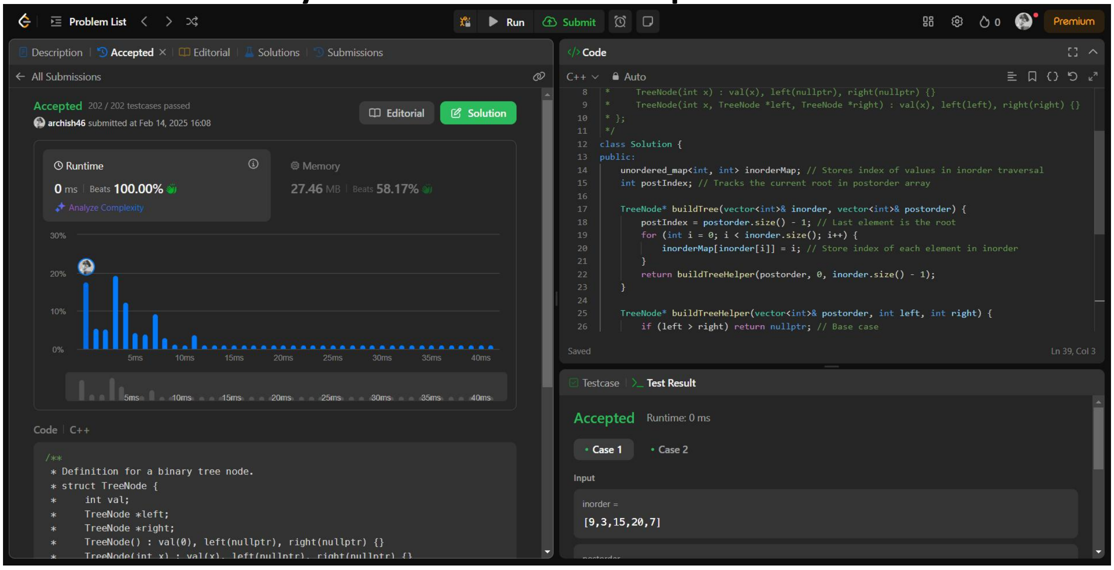
Input

root =

[1,2,3,null,5,null,4]

Output

10. Construct binary tree from inorder and post order traversal



Accepted 202 / 202 testcases passed

archish46 submitted at Feb 14, 2025 16:08

Runtime 0 ms | Beats 100.00%

Memory 27.46 MB | Beats 58.17%

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) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    unordered_map<int, int> inorderMap; // Stores index of values in inorder traversal
    int postIndex; // Tracks the current root in postorder array

    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {
        postIndex = postorder.size() - 1; // Last element is the root
        for (int i = 0; i < inorder.size(); i++) {
            inorderMap[inorder[i]] = i; // Store index of each element in inorder
        }
        return buildTreeHelper(postorder, 0, inorder.size() - 1);
    }

    TreeNode* buildTreeHelper(vector<int>& postorder, int left, int right) {
        if (left > right) return nullptr; // Base case
```

Ln 39, Col 3

Accepted Runtime: 0 ms

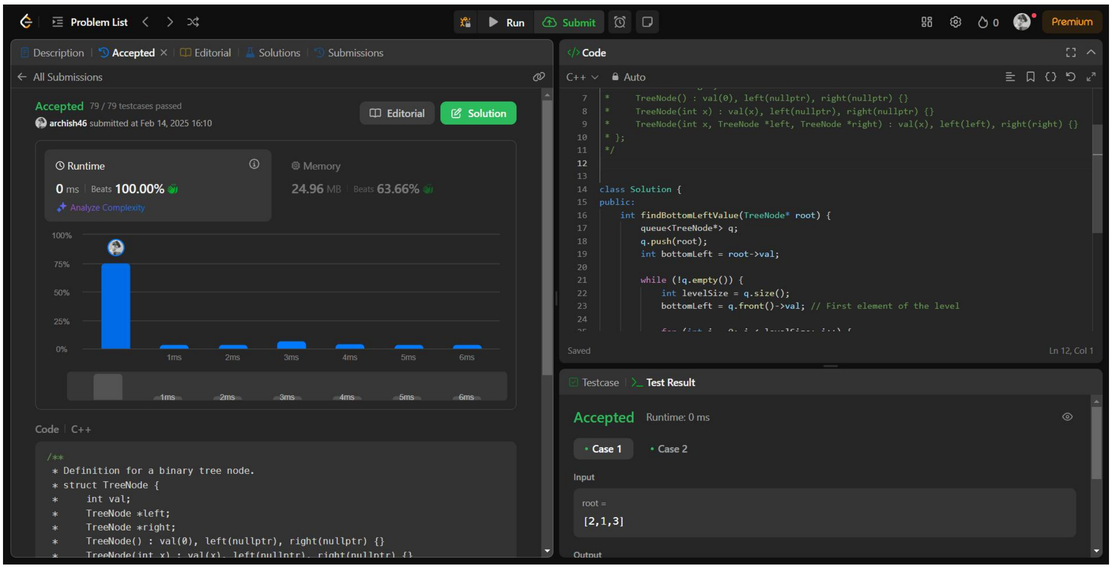
Input

inorder =

[9,3,15,20,7]

postorder

11. Find Bottom Left Tree Value



Accepted 79 / 79 testcases passed

archish46 submitted at Feb 14, 2025 16:10

Runtime 0 ms | Beats 100.00%

Memory 24.96 MB | Beats 63.66%

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() : 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 findBottomLeftValue(TreeNode* root) {
        queue<TreeNode*> q;
        q.push(root);
        int bottomLeft = root->val;

        while (!q.empty()) {
            int levelSize = q.size();
            bottomLeft = q.front()->val; // First element of the level

            for (int i = 0; i < levelSize; i++) {
```

Ln 12, Col 1

Accepted Runtime: 0 ms

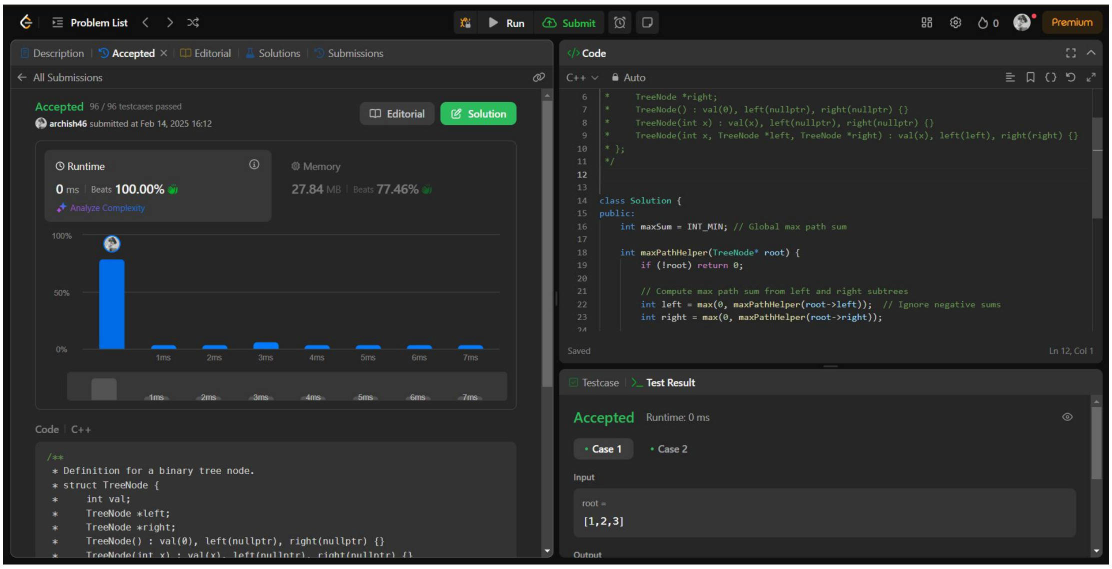
Input

root =

[2,1,3]

Output

12. Binary Tree Maximum Path Sum



Accepted 96 / 96 testcases passed

archish46 submitted at Feb 14, 2025 16:12

Runtime 0 ms | Beats 100.00%

Memory 27.84 MB | Beats 77.46%

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 *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 maxSum = INT_MIN; // Global max path sum

    int maxPathHelper(TreeNode* root) {
        if (!root) return 0;

        // Compute max path sum from left and right subtrees
        int left = max(0, maxPathHelper(root->left));  // Ignore negative sums
        int right = max(0, maxPathHelper(root->right));
```

Ln 12, Col 1

Accepted Runtime: 0 ms

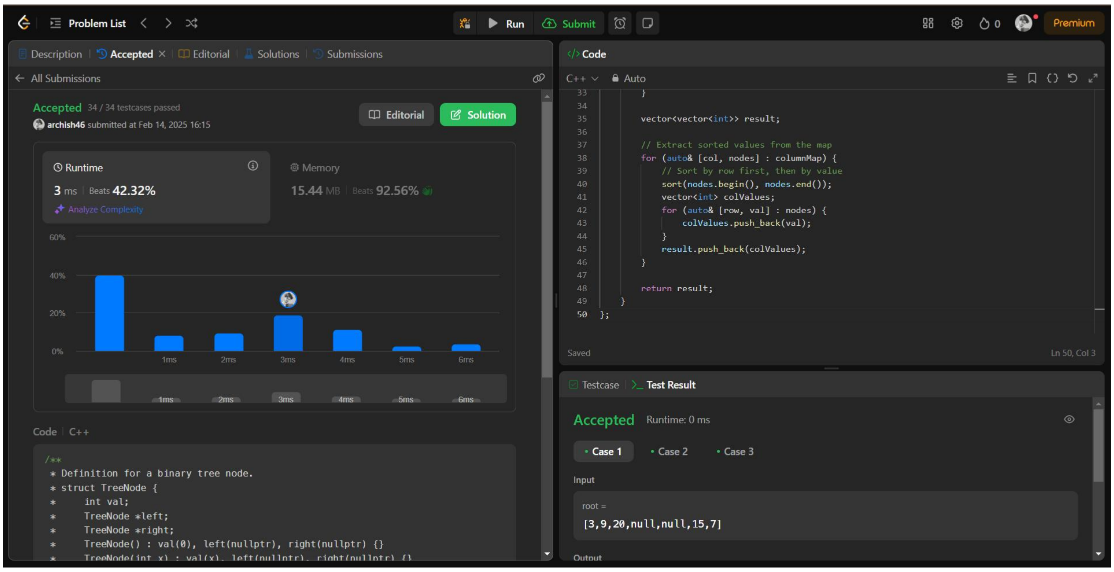
Input

root =

[1,2,3]

Output

13. Vertical Order Traversal of a Binary Tree



Accepted 34 / 34 testcases passed

archish46 submitted at Feb 14, 2025 16:15

Runtime: 3 ms | Beats 42.32%

Memory: 15.44 MB | Beats 92.56%

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) {}
```

C++

```
        }

        vector<vector<int>> result;

        // Extract sorted values from the map
        for (auto& [col, nodes] : columnMap) {
            // Sort by row first, then by value
            sort(nodes.begin(), nodes.end());
            vector<int> colValues;
            for (auto& [row, val] : nodes) {
                colValues.push_back(val);
            }
            result.push_back(colValues);
        }

        return result;
    }
};
```

Saved

Ln 50, Col 3

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

root =

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

Output