Name: Anish Patial

UID: 22BCS15029

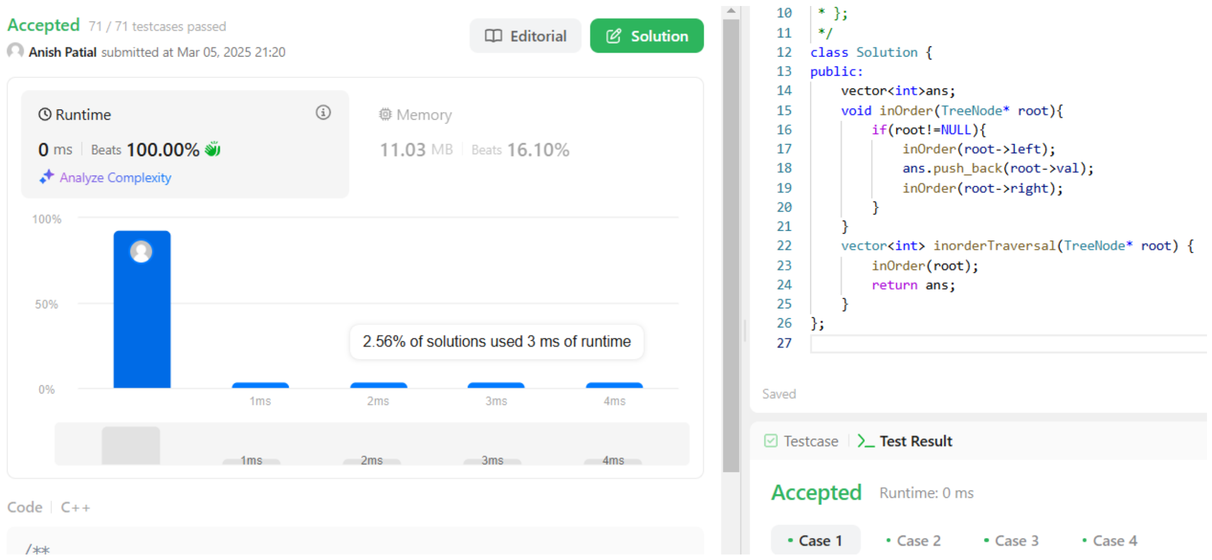
Section: FL_IOT_601 - A

Assignment – 3 Solutions:-

1. **Binary Tree Inorder Traversal:-**

```
/**
 * 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<int>ans;
    void inOrder(TreeNode* root){
        if(root!=NULL){
            inOrder(root->left);
            ans.push_back(root->val);
            inOrder(root->right);
        }
    }
    vector<int> inorderTraversal(TreeNode* root) {
        inOrder(root);
        return ans;
    }
};
```

Result:-

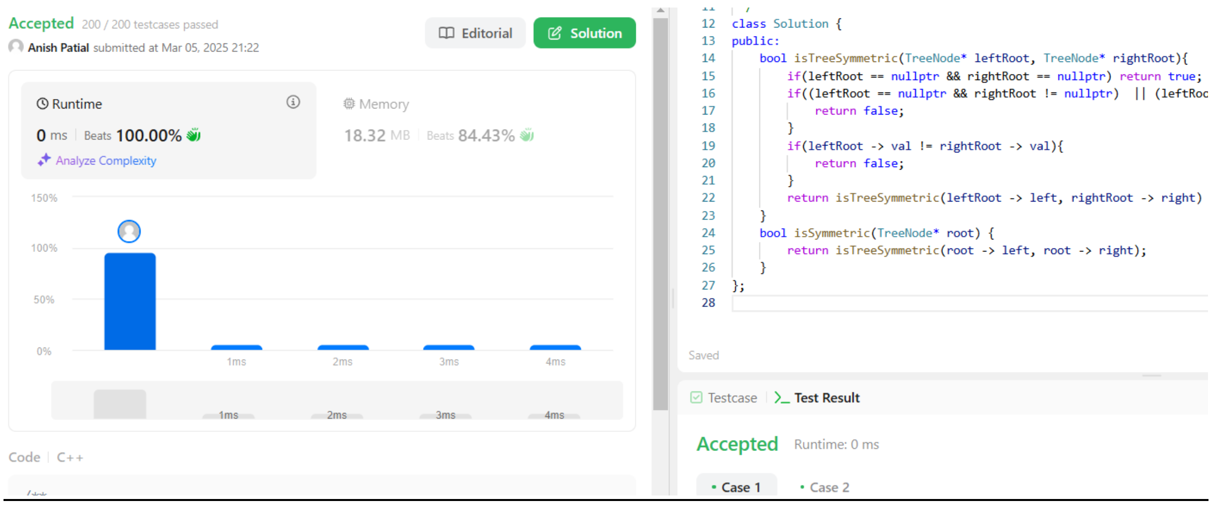


```
 * };
 */
class Solution {
public:
    vector<int>ans;
    void inOrder(TreeNode* root){
        if(root!=NULL){
            inOrder(root->left);
            ans.push_back(root->val);
            inOrder(root->right);
        }
    }
    vector<int> inorderTraversal(TreeNode* root) {
        inOrder(root);
        return ans;
    }
};
```

2. **Symmetric Tree:**

```
/* 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 isTreeSymmetric(TreeNode* leftRoot, TreeNode* rightRoot){
        if(leftRoot == nullptr && rightRoot == nullptr) return true;
        if((leftRoot == nullptr && rightRoot != nullptr)  || (leftRoot != nullptr
&& rightRoot == nullptr)){
            return false;
        }
        if(leftRoot -> val != rightRoot -> val){
            return false;
        }
        return isTreeSymmetric(leftRoot -> left, rightRoot -> right) &&
isTreeSymmetric(leftRoot -> right, rightRoot -> left);
    }
    bool isSymmetric(TreeNode* root) {
        return isTreeSymmetric(root -> left, root -> right);
    }
};
```

Result:



Accepted 200 / 200 testcases passed
Anish Patial submitted at Mar 05, 2025 21:22

Runtime
0 ms | Beats 100.00%

Memory
18.32 MB | Beats 84.43%

Code | C++

```
class Solution {
public:
    bool isTreeSymmetric(TreeNode* leftRoot, TreeNode* rightRoot){
        if(leftRoot == nullptr && rightRoot == nullptr) return true;
        if((leftRoot == nullptr && rightRoot != nullptr)  || (leftRoc
            return false;
        }
        if(leftRoot -> val != rightRoot -> val){
            return false;
        }
        return isTreeSymmetric(leftRoot -> left, rightRoot -> right)
    }
    bool isSymmetric(TreeNode* root) {
        return isTreeSymmetric(root -> left, root -> right);
    }
};
```

Saved

Test Result

Accepted Runtime: 0 ms

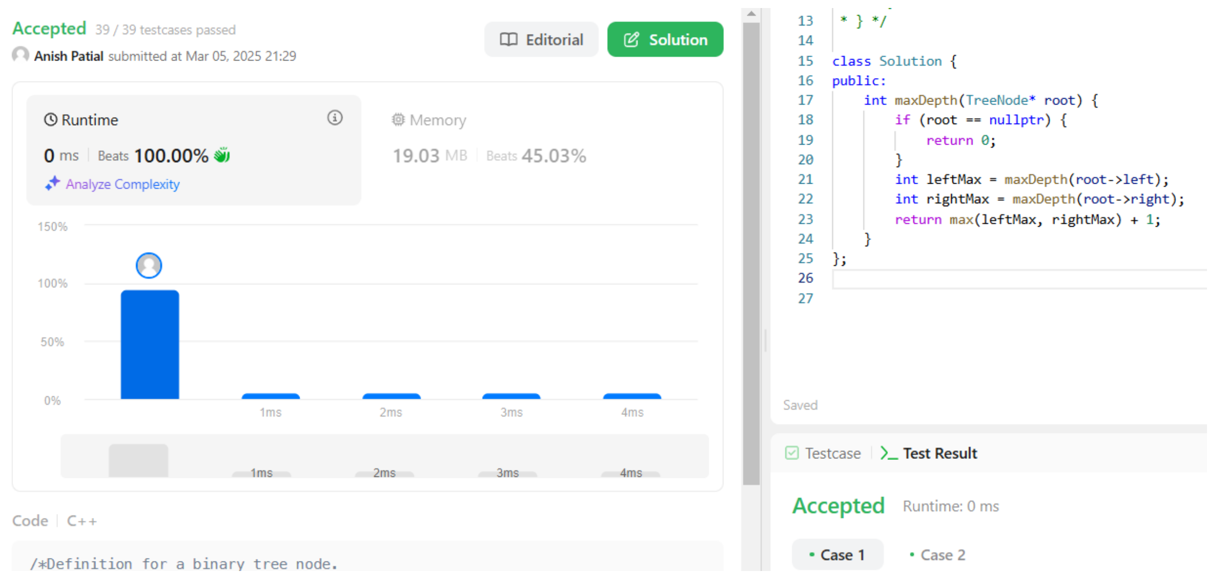
Case 1 Case 2

3. **Maximum Depth of Binary Tree:**

```
/*Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * } */

class Solution {
public:
```

```
int maxDepth(TreeNode* root) {
if (root == nullptr) {
return 0;
}
int leftMax = maxDepth(root->left);
int rightMax = maxDepth(root->right);
return max(leftMax, rightMax) + 1;
}
};
```

Result:



Accepted 39 / 39 testcases passed

Anish Patial submitted at Mar 05, 2025 21:29

```
/*Definition for a binary tree node.
```

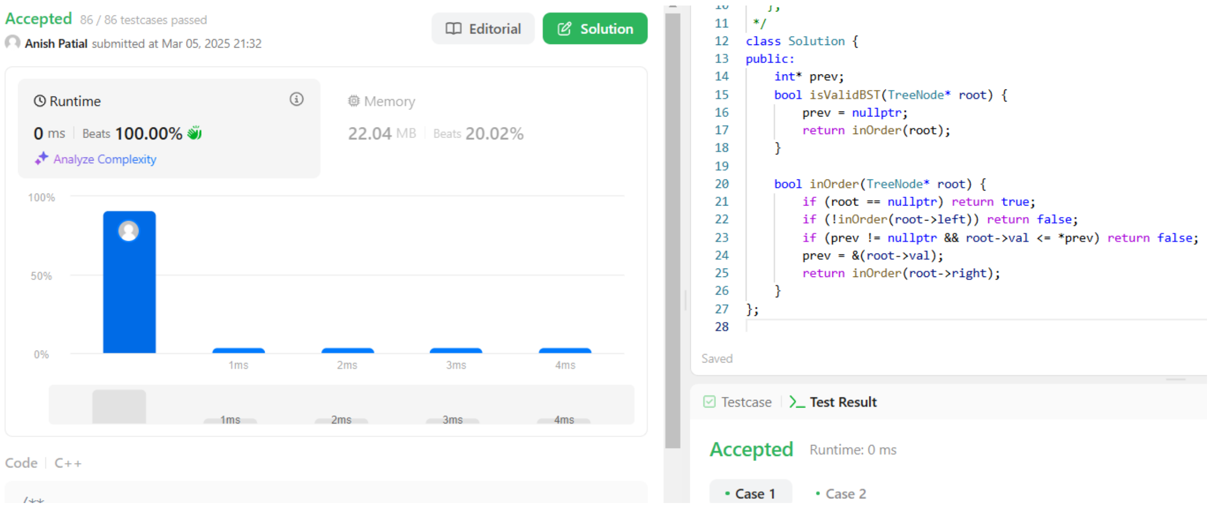
```
 * } */

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

4. Validate Binary Search Tree:

```
/*Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }*/
```

```
class Solution {
public:
    int* prev;
    bool isValidBST(TreeNode* root) {
        prev = nullptr;
        return inOrder(root);
    }

    bool inOrder(TreeNode* root) {
        if (root == nullptr) return true;
        if (!inOrder(root->left)) return false;
        if (prev != nullptr && root->val <= *prev) return false;
        prev = &(root->val);
        return inOrder(root->right);
    }
};
```

<u>Result:</u>



Accepted 86 / 86 testcases passed

Anish Patial submitted at Mar 05, 2025 21:32

0 ms | Beats 100.00%

22.04 MB | Beats 20.02%

```
/**
```

```
 */
class Solution {
public:
    int* prev;
    bool isValidBST(TreeNode* root) {
        prev = nullptr;
        return inOrder(root);
    }

    bool inOrder(TreeNode* root) {
        if (root == nullptr) return true;
        if (!inOrder(root->left)) return false;
        if (prev != nullptr && root->val <= *prev) return false;
        prev = &(root->val);
        return inOrder(root->right);
    }
};
```

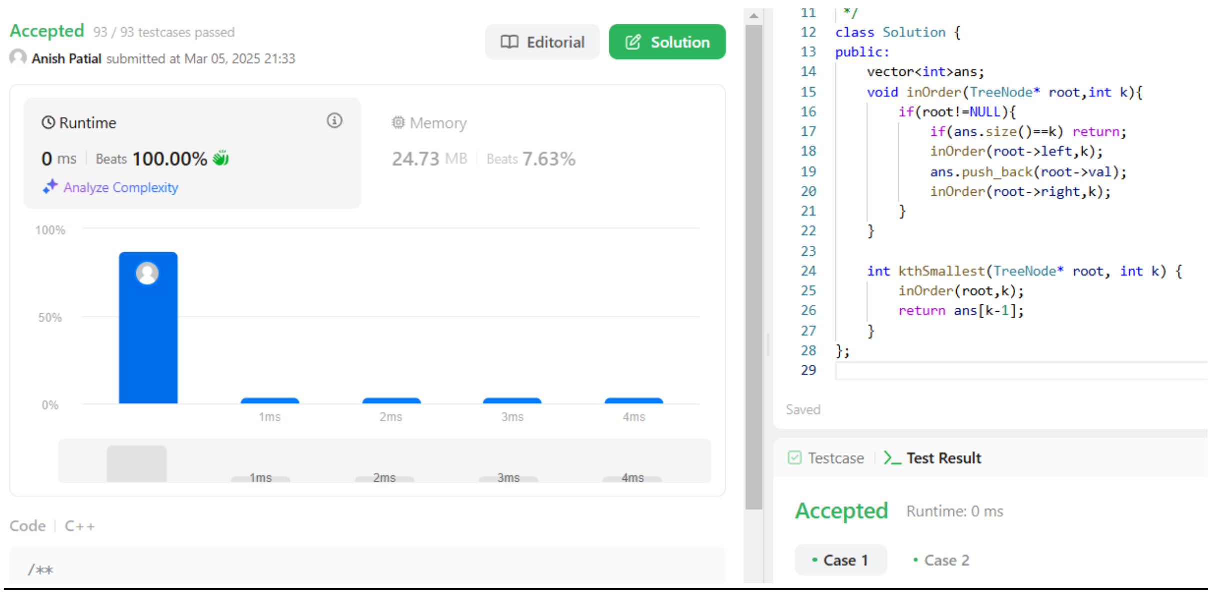
Accepted Runtime: 0 ms

• Case 1 • Case 2

5. <u>Kth Smallest Element in a BST:</u>

```
/*
 * 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<int>ans;
    void inOrder(TreeNode* root,int k){
        if(root!=NULL){
            if(ans.size()==k) return;
            inOrder(root->left,k);
            ans.push_back(root->val);
            inOrder(root->right,k);
        }
    }

    int kthSmallest(TreeNode* root, int k) {
        inOrder(root,k);
        return ans[k-1];
    }
};
```

Result:


Accepted 93 / 93 testcases passed

Anish Patial submitted at Mar 05, 2025 21:33

Runtime: 0 ms | Beats 100.00%

Memory: 24.73 MB | Beats 7.63%

Code | C++

```
 */
class Solution {
public:
    vector<int>ans;
    void inOrder(TreeNode* root,int k){
        if(root!=NULL){
            if(ans.size()==k) return;
            inOrder(root->left,k);
            ans.push_back(root->val);
            inOrder(root->right,k);
        }
    }

    int kthSmallest(TreeNode* root, int k) {
        inOrder(root,k);
        return ans[k-1];
    }
};
```

Saved

Test Result

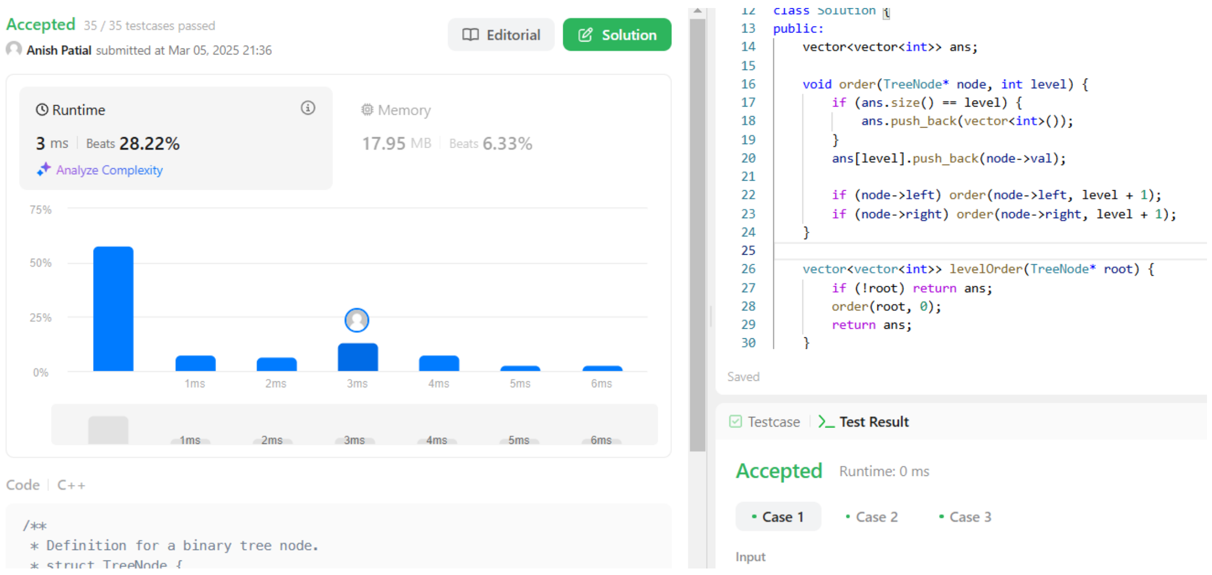
Accepted Runtime: 0 ms

Case 1 · Case 2

6. Binary Tree Level Order Traversal:

```
/* Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * } */

class Solution {
public:
    vector<vector<int>> ans;

    void order(TreeNode* node, int level) {
        if (ans.size() == level) {
            ans.push_back(vector<int>());
        }
        ans[level].push_back(node->val);

        if (node->left) order(node->left, level + 1);
        if (node->right) order(node->right, level + 1);
    }

    vector<vector<int>> levelOrder(TreeNode* root) {
        if (!root) return ans;
        order(root, 0);
        return ans;
    }
};
```

<u>Result:</u>


Accepted 35 / 35 testcases passed
Anish Patial submitted at Mar 05, 2025 21:36

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
```

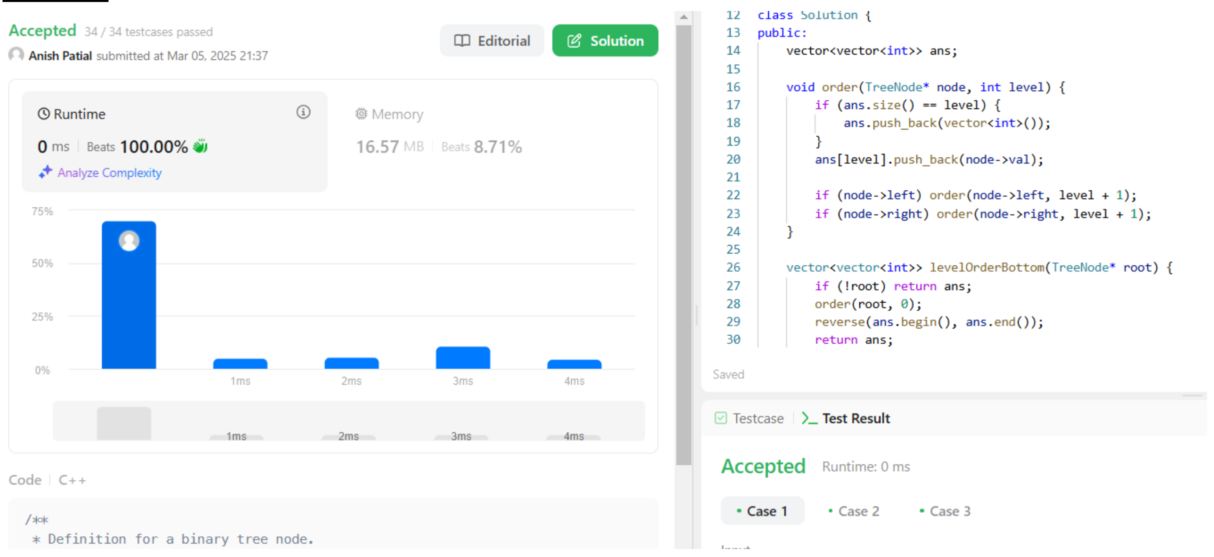
```
class Solution {
public:
    vector<vector<int>> ans;

    void order(TreeNode* node, int level) {
        if (ans.size() == level) {
            ans.push_back(vector<int>());
        }
        ans[level].push_back(node->val);

        if (node->left) order(node->left, level + 1);
        if (node->right) order(node->right, level + 1);
    }

    vector<vector<int>> levelOrder(TreeNode* root) {
        if (!root) return ans;
        order(root, 0);
        return ans;
    }
```

Accepted Runtime: 0 ms

7. <u>**Binary Tree Level Order Traversal II:**</u>

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
public:
    vector<vector<int>> ans;

    void order(TreeNode* node, int level) {
        if (ans.size() == level) {
            ans.push_back(vector<int>());
        }
        ans[level].push_back(node->val);

        if (node->left) order(node->left, level + 1);
        if (node->right) order(node->right, level + 1);
    }

    vector<vector<int>> levelOrderBottom(TreeNode* root) {
        if (!root) return ans;
        order(root, 0);
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

Result:



Accepted 34 / 34 testcases passed
Anish Patial submitted at Mar 05, 2025 21:37

Runtime
0 ms | Beats 100.00%

Memory
16.57 MB | Beats 8.71%

Code | C++

```
/**
 * Definition for a binary tree node.
```

```
class Solution {
public:
    vector<vector<int>> ans;

    void order(TreeNode* node, int level) {
        if (ans.size() == level) {
            ans.push_back(vector<int>());
        }
        ans[level].push_back(node->val);

        if (node->left) order(node->left, level + 1);
        if (node->right) order(node->right, level + 1);
    }

    vector<vector<int>> levelOrderBottom(TreeNode* root) {
        if (!root) return ans;
        order(root, 0);
        reverse(ans.begin(), ans.end());
        return ans;
```

Saved

Testcase | Test Result

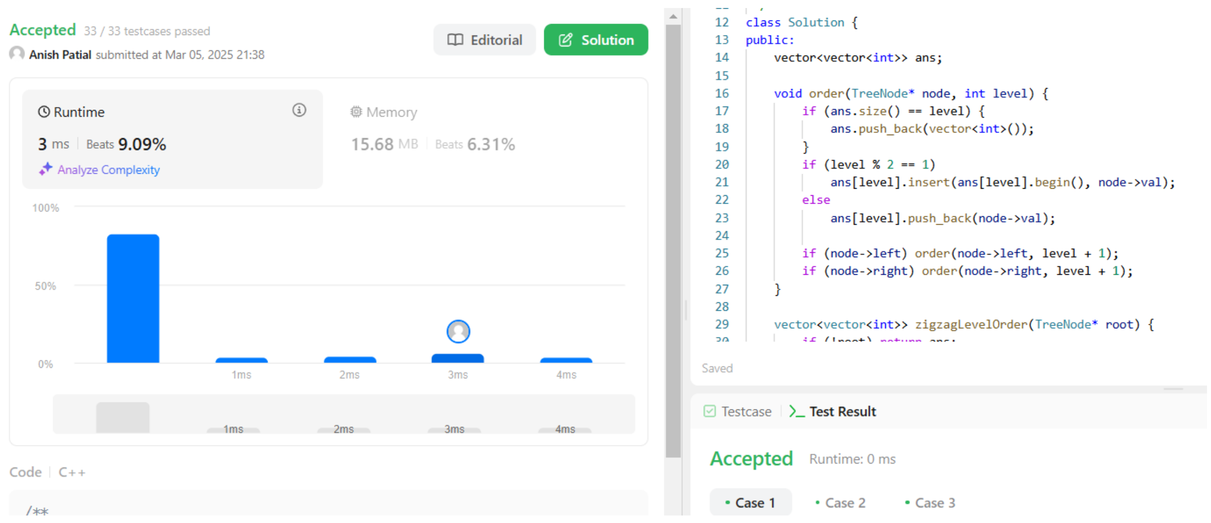
Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

8. Binary Tree Zigzag Level Order Traversal:

```
/* Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * } */
class Solution {
public:
    vector<vector<int>> ans;

    void order(TreeNode* node, int level) {
        if (ans.size() == level) {
            ans.push_back(vector<int>());
        }
        if (level % 2 == 1)
            ans[level].insert(ans[level].begin(), node->val);
        else
            ans[level].push_back(node->val);

        if (node->left) order(node->left, level + 1);
        if (node->right) order(node->right, level + 1);
    }

    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        if (!root) return ans;
        order(root, 0);
        return ans;
    }
};
```

Result:-



Accepted 33 / 33 testcases passed
Anish Patial submitted at Mar 05, 2025 21:38

Runtime: 3 ms | Beats 9.09%
Memory: 15.68 MB | Beats 6.31%

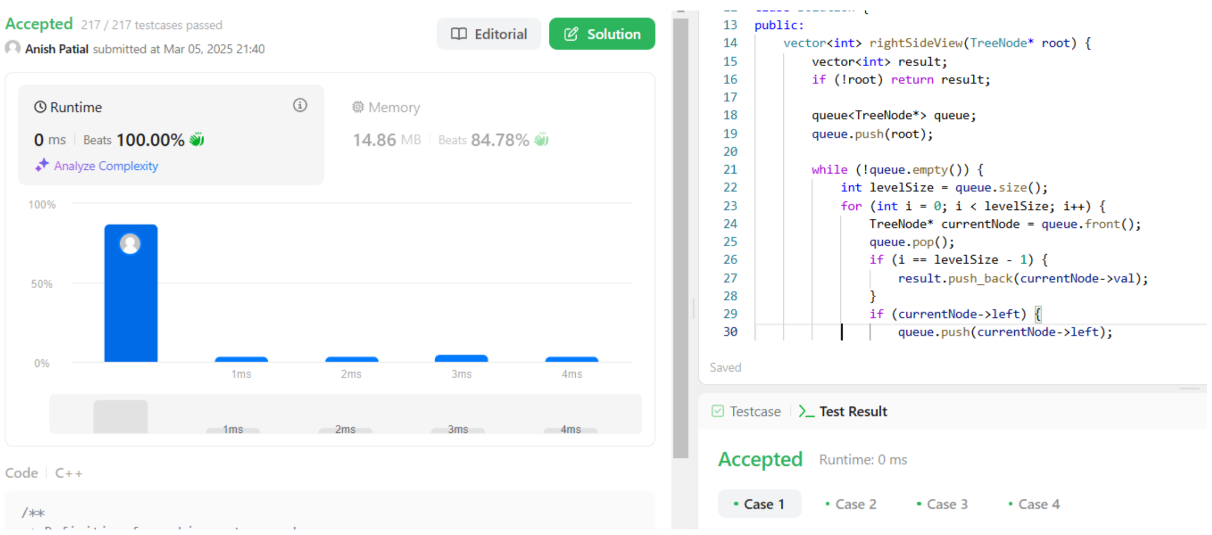
```
class Solution {
public:
    vector<vector<int>> ans;

    void order(TreeNode* node, int level) {
        if (ans.size() == level) {
            ans.push_back(vector<int>());
        }
        if (level % 2 == 1)
            ans[level].insert(ans[level].begin(), node->val);
        else
            ans[level].push_back(node->val);

        if (node->left) order(node->left, level + 1);
        if (node->right) order(node->right, level + 1);
    }

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

Accepted Runtime: 0 ms

9. Binary Tree Right Side View:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *        this.right = right;
 *    }
 * }
 */
class Solution {
public:
    vector<int> rightSideView(TreeNode* root) {
        vector<int> result;
        if (!root) return result;

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

        while (!queue.empty()) {
            int levelSize = queue.size();
            for (int i = 0; i < levelSize; i++) {
                TreeNode* currentNode = queue.front();
                queue.pop();
                if (i == levelSize - 1) {
                    result.push_back(currentNode->val);
                }
                if (currentNode->left) {
                    queue.push(currentNode->left);
                }
                if (currentNode->right) {
                    queue.push(currentNode->right);
                }
            }
        }
        return result;
    }
};
```

Result:


Accepted 217 / 217 testcases passed

Anish Patial submitted at Mar 05, 2025 21:40

Runtime
0 ms | Beats 100.00%

Memory
14.86 MB | Beats 84.78%

Code | C++

```
/**
```

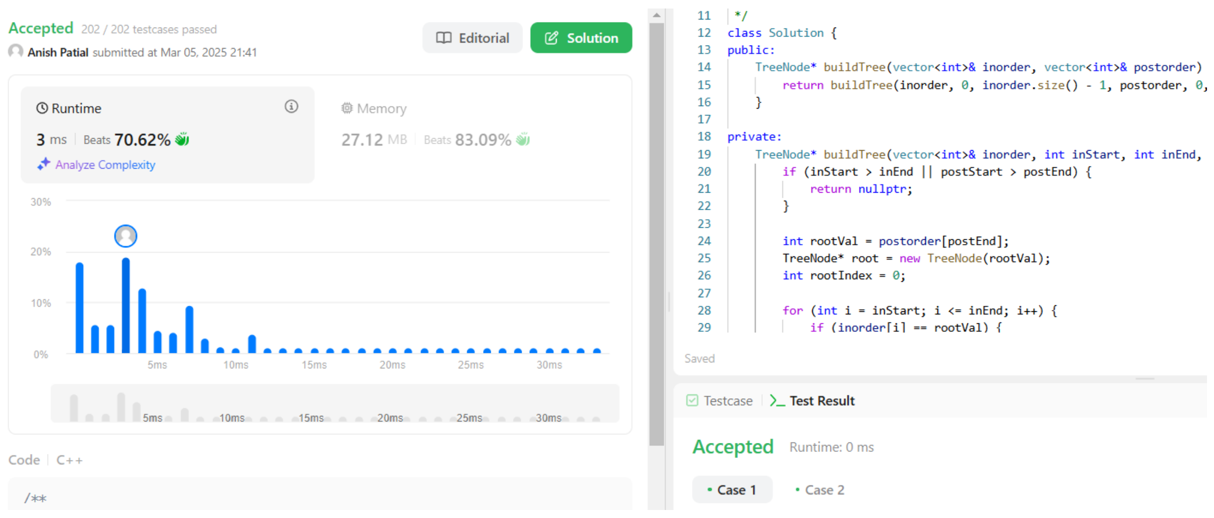
```
public:
    vector<int> rightSideView(TreeNode* root) {
        vector<int> result;
        if (!root) return result;

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

        while (!queue.empty()) {
            int levelSize = queue.size();
            for (int i = 0; i < levelSize; i++) {
                TreeNode* currentNode = queue.front();
                queue.pop();
                if (i == levelSize - 1) {
                    result.push_back(currentNode->val);
                }
                if (currentNode->left) {
                    queue.push(currentNode->left);
```

Accepted Runtime: 0 ms

10. Construct Binary Tree from Inorder and Post order Traversal:

```
/* Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * } */
class Solution {
public:
    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {
        return buildTree(inorder, 0, inorder.size() - 1, postorder, 0,
postorder.size() - 1);
    }

private:
    TreeNode* buildTree(vector<int>& inorder, int inStart, int inEnd,
vector<int>& postorder, int postStart, int postEnd) {
        if (inStart > inEnd || postStart > postEnd) {
            return nullptr;
        }

        int rootVal = postorder[postEnd];
        TreeNode* root = new TreeNode(rootVal);
        int rootIndex = 0;

        for (int i = inStart; i <= inEnd; i++) {
            if (inorder[i] == rootVal) {
                rootIndex = i;
                break;
            }
        }

        int leftSize = rootIndex - inStart;
        int rightSize = inEnd - rootIndex;

        root->left = buildTree(inorder, inStart, rootIndex - 1, postorder,
postStart, postStart + leftSize - 1);
        root->right = buildTree(inorder, rootIndex + 1, inEnd, postorder,
postEnd - rightSize, postEnd - 1);

        return root;
    }
};
```

Result:



Accepted 202 / 202 testcases passed

Anish Patial submitted at Mar 05, 2025 21:41

Runtime

3 ms | Beats 70.62%

Memory

27.12 MB | Beats 83.09%

Code | C++

```
/**
```

```
 */
class Solution {
public:
    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder)
        return buildTree(inorder, 0, inorder.size() - 1, postorder, 0,
    }

private:
    TreeNode* buildTree(vector<int>& inorder, int inStart, int inEnd,
        if (inStart > inEnd || postStart > postEnd) {
            return nullptr;
        }

        int rootVal = postorder[postEnd];
        TreeNode* root = new TreeNode(rootVal);
        int rootIndex = 0;

        for (int i = inStart; i <= inEnd; i++) {
            if (inorder[i] == rootVal) {
```

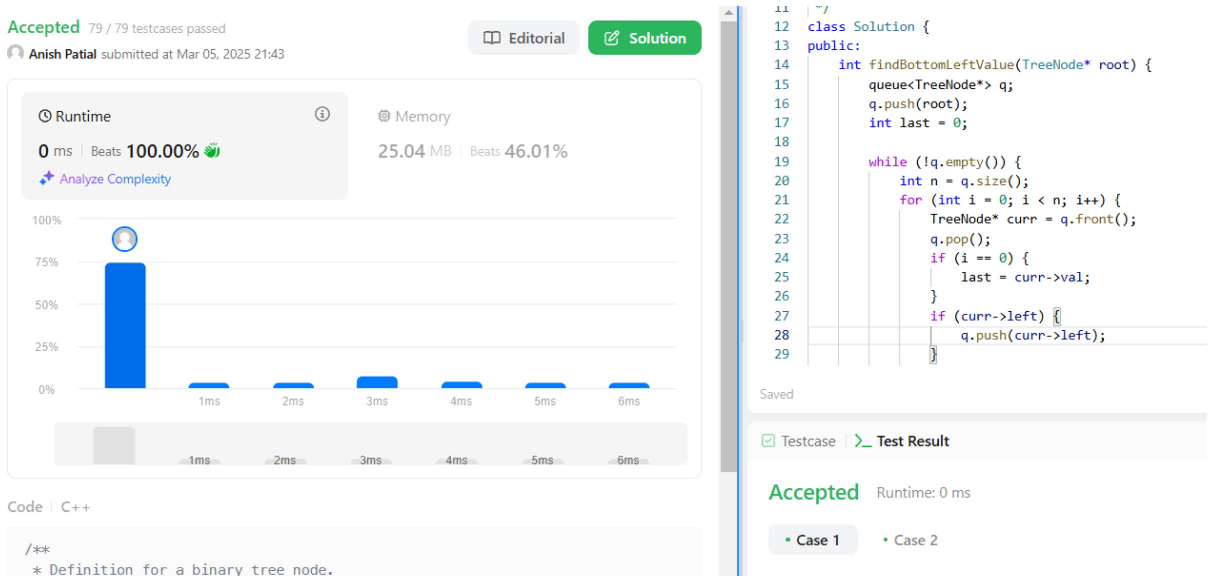
Saved

Accepted Runtime: 0 ms

11. Find Bottom Left Tree Value:

```
/* Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }  */

class Solution {
public:
    int findBottomLeftValue(TreeNode* root) {
        queue<TreeNode*> q;
        q.push(root);
        int last = 0;

        while (!q.empty()) {
            int n = q.size();
            for (int i = 0; i < n; i++) {
                TreeNode* curr = q.front();
                q.pop();
                if (i == 0) {
                    last = curr->val;
                }
                if (curr->left) {
                    q.push(curr->left);
                }
                if (curr->right) {
                    q.push(curr->right);
                }
            }
        }
        return last;
    }
};
```

Result:



Accepted 79 / 79 testcases passed

Anish Patial submitted at Mar 05, 2025 21:43

Runtime: 0 ms | Beats 100.00%

Memory: 25.04 MB | Beats 46.01%

Code | C++

```
/**
 * Definition for a binary tree node.
```

```
 */
class Solution {
public:
    int findBottomLeftValue(TreeNode* root) {
        queue<TreeNode*> q;
        q.push(root);
        int last = 0;

        while (!q.empty()) {
            int n = q.size();
            for (int i = 0; i < n; i++) {
                TreeNode* curr = q.front();
                q.pop();
                if (i == 0) {
                    last = curr->val;
                }
                if (curr->left) {
                    q.push(curr->left);
                }
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

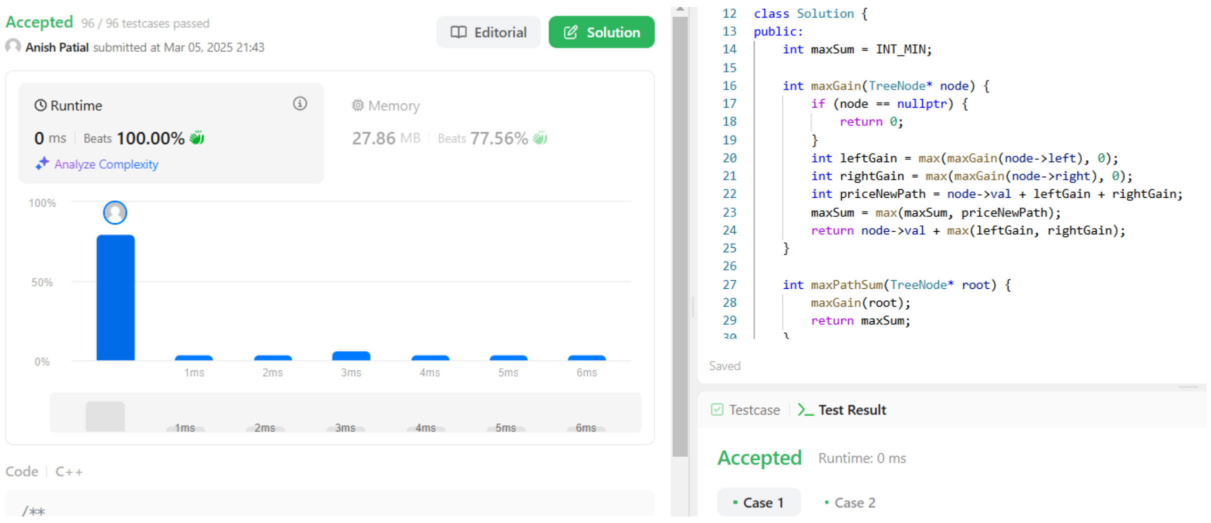
• Case 1 • Case 2

12. **Binary Tree Maximum Path Sum:**

```
/*Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * } */

class Solution {
```

```
public:
    int maxSum = INT_MIN;

    int maxGain(TreeNode* node) {
        if (node == nullptr) {
            return 0;
        }
        int leftGain = max(maxGain(node->left), 0);
        int rightGain = max(maxGain(node->right), 0);
        int priceNewPath = node->val + leftGain + rightGain;
        maxSum = max(maxSum, priceNewPath);
        return node->val + max(leftGain, rightGain);
    }

    int maxPathSum(TreeNode* root) {
        maxGain(root);
        return maxSum;
    }
};
```

<u>Result:</u>



Accepted 96 / 96 testcases passed

Anish Patial submitted at Mar 05, 2025 21:43

Runtime: 0 ms | Beats 100.00%

Memory: 27.86 MB | Beats 77.56%

Code | C++

```
/**
```

```
class Solution {
public:
    int maxSum = INT_MIN;

    int maxGain(TreeNode* node) {
        if (node == nullptr) {
            return 0;
        }
        int leftGain = max(maxGain(node->left), 0);
        int rightGain = max(maxGain(node->right), 0);
        int priceNewPath = node->val + leftGain + rightGain;
        maxSum = max(maxSum, priceNewPath);
        return node->val + max(leftGain, rightGain);
    }

    int maxPathSum(TreeNode* root) {
        maxGain(root);
        return maxSum;
```

Saved

Testcase | Test Result

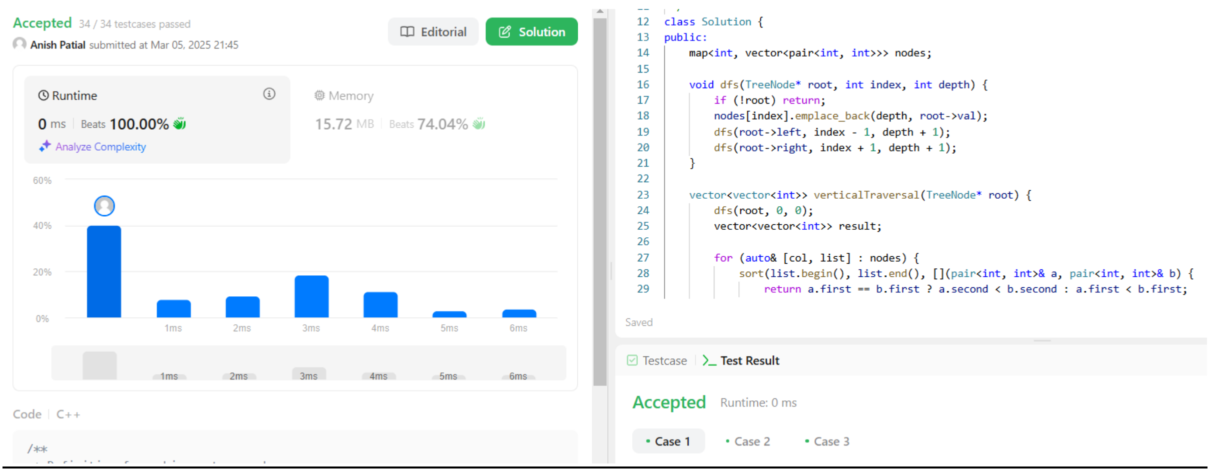
Accepted Runtime: 0 ms

Case 1 Case 2

13. <u>Vertical Order Traversal of a Binary Tree:</u>

```
/* Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * } */

class Solution {
public:
    map<int, vector<pair<int, int>>> nodes;

    void dfs(TreeNode* root, int index, int depth) {
        if (!root) return;
        nodes[index].emplace_back(depth, root->val);
        dfs(root->left, index - 1, depth + 1);
        dfs(root->right, index + 1, depth + 1);
    }

    vector<vector<int>> verticalTraversal(TreeNode* root) {
        dfs(root, 0, 0);
        vector<vector<int>> result;

        for (auto& [col, list] : nodes) {
            sort(list.begin(), list.end(), [](pair<int, int>& a, pair<int, int>& b)
{
                return a.first == b.first ? a.second < b.second : a.first < b.first;
            });
            vector<int> current;
            for (auto& num : list) {
                current.push_back(num.second);
            }
            result.push_back(current);
        }
        return result;
    }
};
```

Result:


Accepted 34 / 34 testcases passed

Anish Patial submitted at Mar 05, 2025 21:45

```
class Solution {
public:
    map<int, vector<pair<int, int>>> nodes;

    void dfs(TreeNode* root, int index, int depth) {
        if (!root) return;
        nodes[index].emplace_back(depth, root->val);
        dfs(root->left, index - 1, depth + 1);
        dfs(root->right, index + 1, depth + 1);
    }

    vector<vector<int>> verticalTraversal(TreeNode* root) {
        dfs(root, 0, 0);
        vector<vector<int>> result;

        for (auto& [col, list] : nodes) {
            sort(list.begin(), list.end(), [](pair<int, int>& a, pair<int, int>& b) {
                return a.first == b.first ? a.second < b.second : a.first < b.first;
```

Saved

Accepted Runtime: 0 ms