Assignment

Student Name: Khushal | **UID:** 22CS13927

Branch:CSE | **Section/Group:**605-B

Semester: 5 | **Date of Performance:** 14/02/25

Subject Name: AP | **Subject Code: 22CSP-351**

Q.1 101. Symmetric Tree

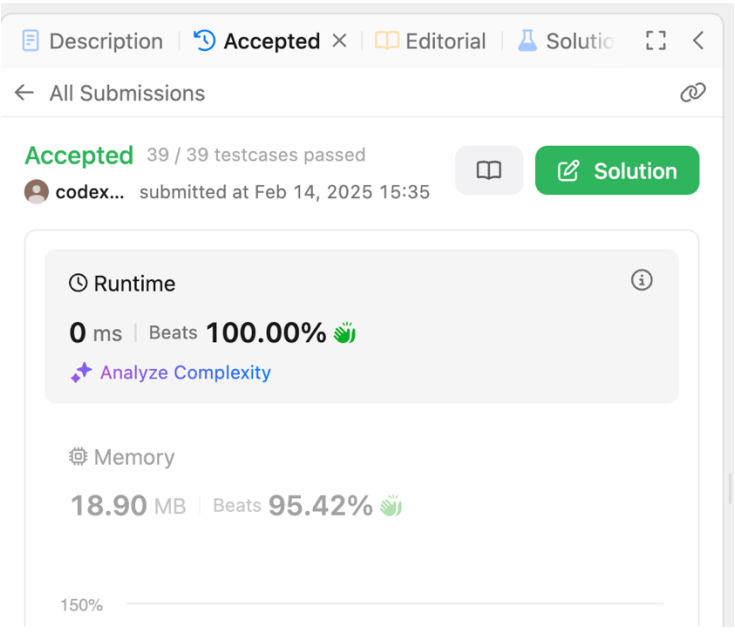
```
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        if (!root) return true;
        return isMirror(root->left, root->right);
    }

    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);
    }
};
```

Output :

Q.2 [104. Maximum Depth of Binary Tree]

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

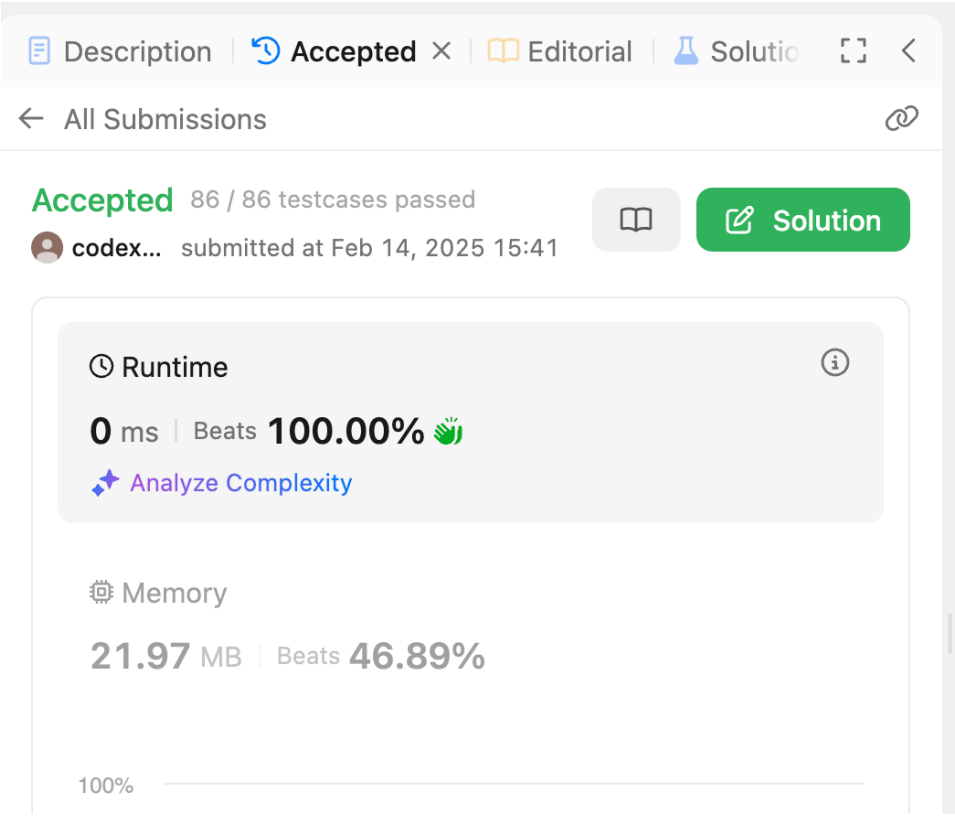
Output :



Q.3 [98. Validate Binary Search Tree]

```
class Solution {
public:
    bool isValidBST(TreeNode* root) {
        return isValidBSTHelper(root, LONG_MIN, LONG_MAX);
    }

private:
    bool isValidBSTHelper(TreeNode* root, long minVal, long maxVal) {
        if (!root) return true;
        if (root->val <= minVal || root->val >= maxVal) return false;
        return isValidBSTHelper(root->left, minVal, root->val) &&
               isValidBSTHelper(root->right, root->val, maxVal);
    }
};
```

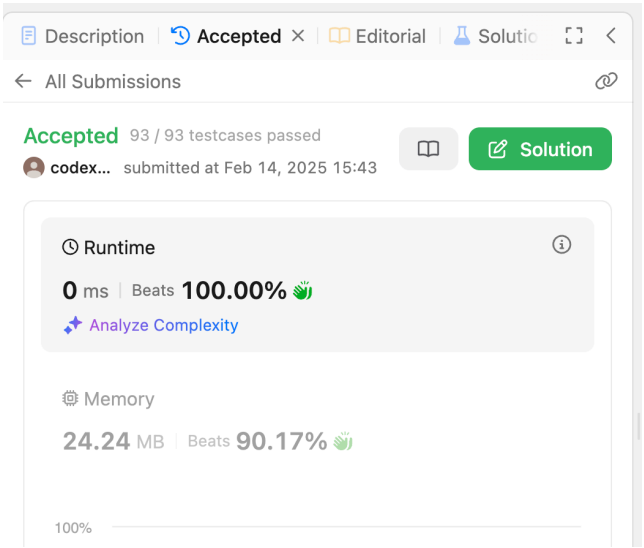
Output :



Q.4 [230. Kth Smallest Element in a BST]

```
class Solution {
public:
    pair<int,int> kthSmall(TreeNode* root,int k){
        if(root == NULL)    return make_pair(-1,0);
        pair<int,int> pr = kthSmall(root->left,k);
        if(pr.first!=-1)    return pr;
        else k -= pr.second;
        if(k==1)    {
            pr.first = root->val;
            return pr;
        }
        pair<int,int> p2 = kthSmall(root->right,k-1);
        p2.second += 1 + pr.second;
        return p2;
    }
    int kthSmallest(TreeNode* root, int k) {
        return kthSmall(root,k).first;
    }
};
```

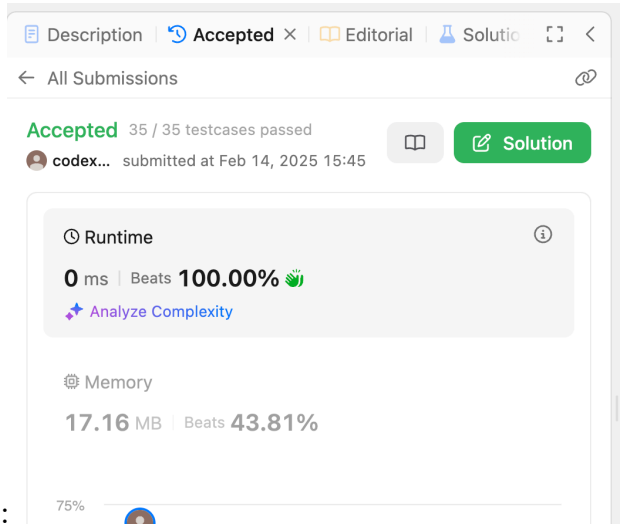
Output :



Q.5 [102. Binary Tree Level Order Traversal]

```
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 s=q.size();
            vector<int>v;
            for(int i=0;i<s;i++){
                TreeNode *node=q.front();
                q.pop();
                if(node->left!=NULL)q.push(node->left);
                if(node->right!=NULL)q.push(node->right);
                v.push_back(node->val);
            }
            ans.push_back(v);
        }
        return ans;
    }
};
```

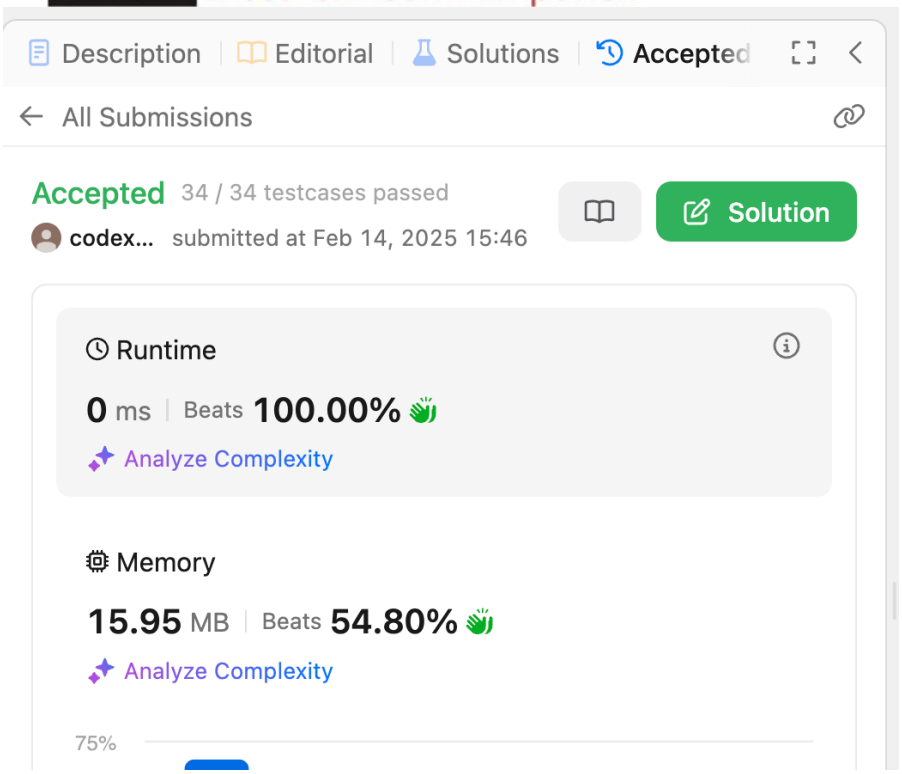
Output :



Q.6. **107. Binary Tree Level Order Traversal II**

```
class Solution {
public:
    vector<vector<int>> levelOrderBottom(TreeNode* root) {
        if (root == nullptr) return {};

        stack<vector<int>> s;
        queue<TreeNode*> q;
        q.push(root);

        while (!q.empty()) {
            int numberOfNodes = q.size();
            vector<int> array(numberOfNodes, 0);

            for (int i = 0; i < numberOfNodes; i++) {
                TreeNode* node = q.front();
                q.pop();
                array[i] = node->val;

                if (node->left != nullptr) q.push(node->left);
                if (node->right != nullptr) q.push(node->right);
            }

            s.push(array);
        }

        vector<vector<int>> ans;
        while (!s.empty()) {
            ans.push_back(s.top());
            s.pop();
        }

        return ans;
    }
};
```

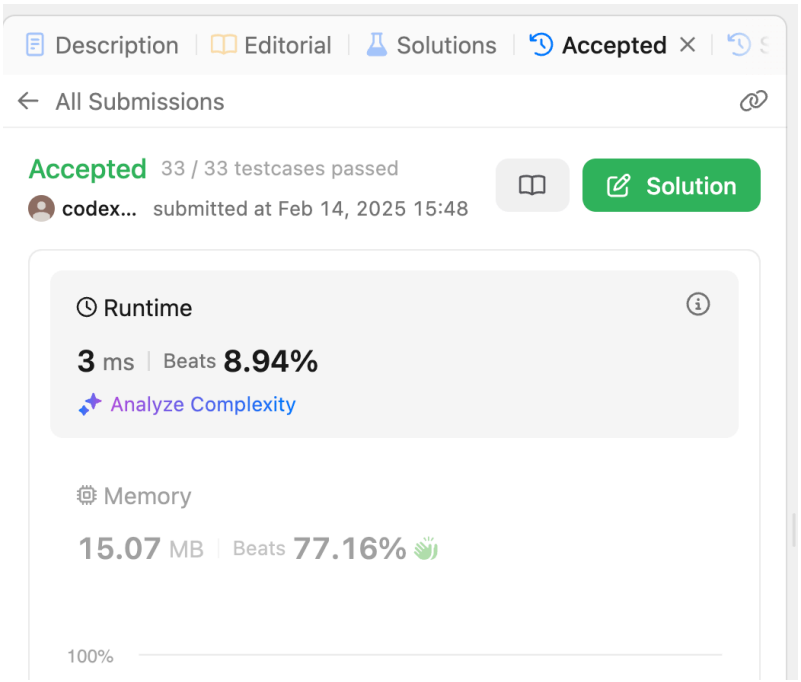
Output :



Q.7. **103. Binary Tree Zigzag Level Order Traversal**

```
class Solution {
public:
    void solve(vector<vector<int>>& ans, TreeNode* temp, int level) {
        if (temp == NULL) return;
        if (ans.size() <= level) ans.push_back({});
        if (level % 2 == 0) ans[level].push_back(temp->val);
        else ans[level].insert(ans[level].begin(), temp->val);
        solve(ans, temp->left, level + 1);
        solve(ans, temp->right, level + 1);
    }

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

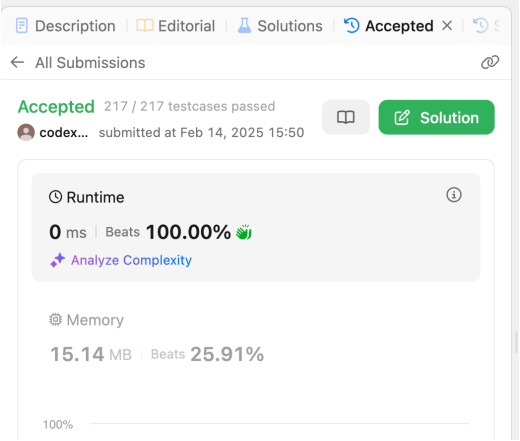
Output:


Description
Editorial
Solutions
Accepted
All Submissions
Accepted
33 / 33 testcases passed
codex...
submitted at Feb 14, 2025 15:48
Solution
Runtime
3 ms
Beats 8.94%
Analyze Complexity
Memory
15.07 MB
Beats 77.16%
100%

Q.8 **199. Binary Tree Right Side View**

```
class Solution {
 public:
  vector<int> rightSideView(TreeNode* root) {
    if (root == nullptr)
      return {};
    vector<int> ans;
    queue<TreeNode*> q{{root}};
    while (!q.empty()) {
      const int size = q.size();
      for (int i = 0; i < size; ++i) {
        TreeNode* node = q.front();
        q.pop();
        if (i == size - 1)
          ans.push_back(node->val);
        if (node->left)
          q.push(node->left);
        if (node->right)
          q.push(node->right);
      }
    }
    return ans;
  }
};
```

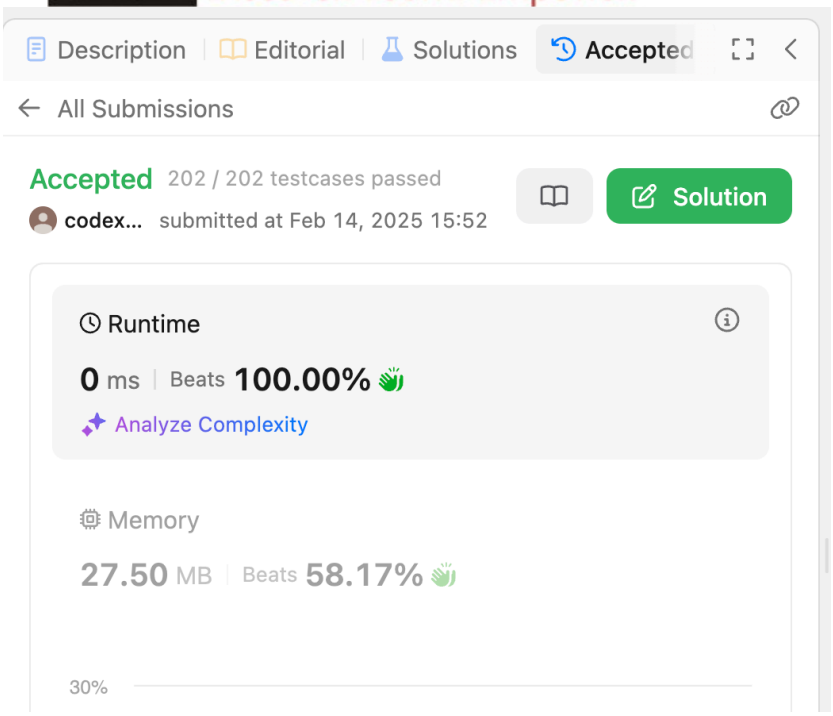
Output :



Q.9 **106. Construct Binary Tree from Inorder and Postorder Traversal**

```
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;
    }
};
```

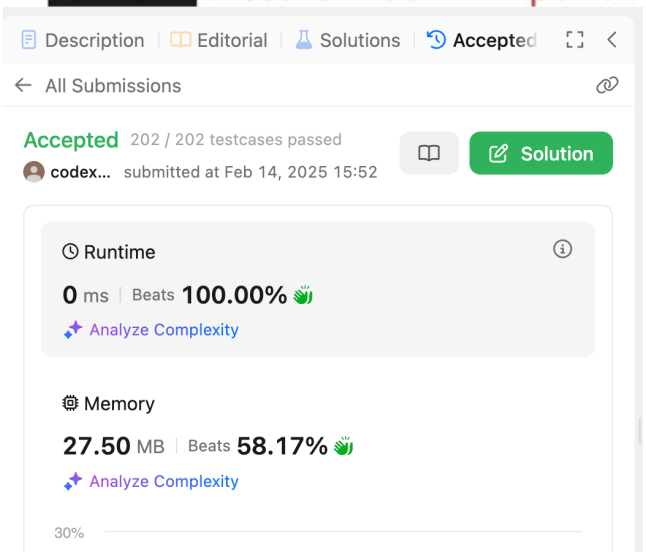
Output :


Description
Editorial
Solutions
Accepted
All Submissions
Accepted
202 / 202 testcases passed
codex...
submitted at Feb 14, 2025 15:52
Solution
Runtime
0 ms
Beats 100.00%
Analyze Complexity
Memory
27.50 MB
Beats 58.17%
30%

Q.10 106. Construct Binary Tree from Inorder and Postorder Traversal

```
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;
    }
};
```

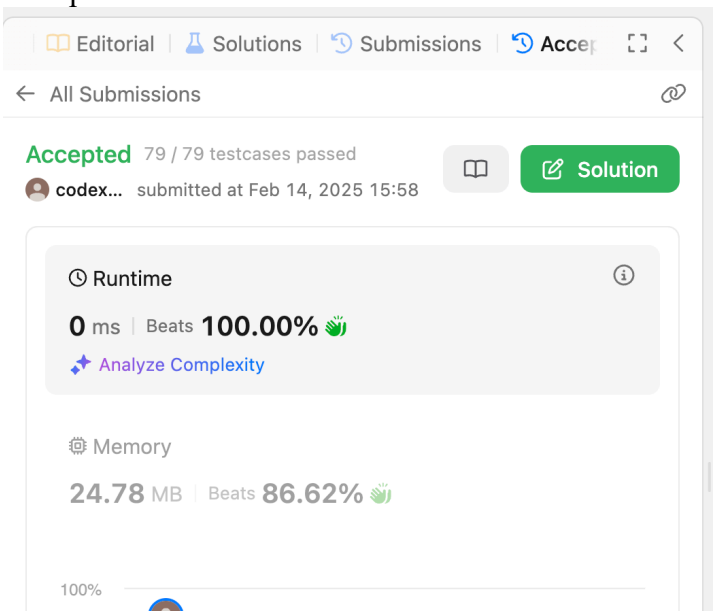
Output :



Q.11 **513. Find Bottom Left Tree Value**

```
class Solution {
public:
    int findBottomLeftValue(TreeNode* root) {
        maxDepth = -1;
        bottomLeftValue = 0;
        dfs(root, 0);
        return bottomLeftValue;
    }

private:
    int maxDepth;
    int bottomLeftValue;

    void dfs(TreeNode* current, int depth) {
        if (current == nullptr) {
            return;
        }
        if (depth > maxDepth) {  // If true, we discovered a new level
            maxDepth = depth;
            bottomLeftValue = current->val;
        }
        dfs(current->left, depth + 1);
        dfs(current->right, depth + 1);
    }
};
```

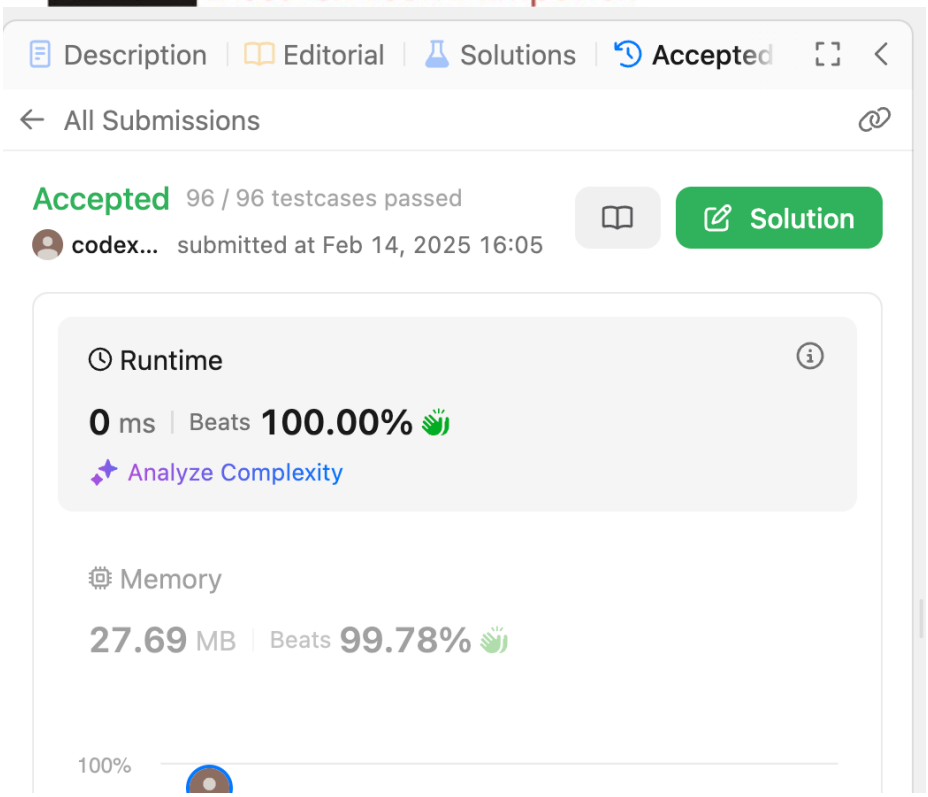
Output :



Q.12. **124. Binary Tree Maximum Path Sum**

```
class Solution {
 public:
  int maxPathSum(TreeNode* root) {
    int ans = INT_MIN;
    maxPathSumDownFrom(root, ans);
    return ans;
  }

 private:
  int maxPathSumDownFrom(TreeNode* root, int& ans) {
    if (root == nullptr)
      return 0;
    const int l = max(0, maxPathSumDownFrom(root->left, ans));
    const int r = max(0, maxPathSumDownFrom(root->right, ans));
    ans = max(ans, root->val + l + r);
    return root->val + max(l, r);
  }
};
```

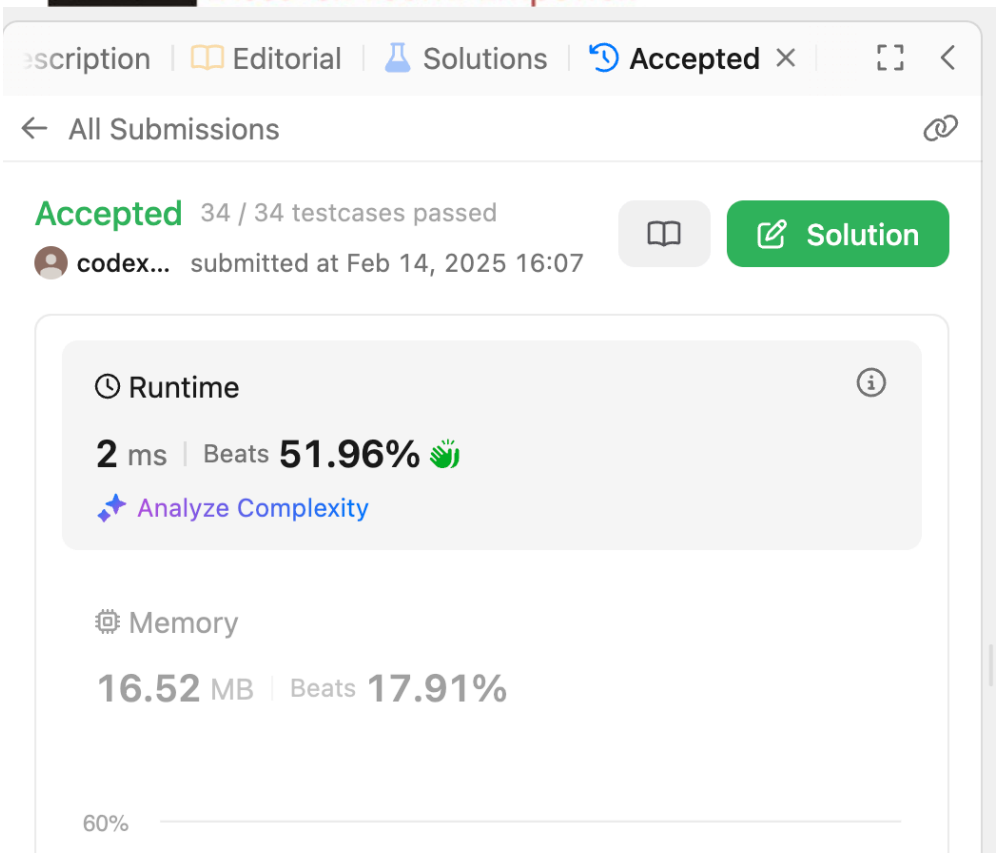
Output :


Description
Editorial
Solutions
Accepted
All Submissions
Accepted
96 / 96 testcases passed
codex...
submitted at Feb 14, 2025 16:05
Solution
Runtime
0 ms
Beats 100.00%
Analyze Complexity
Memory
27.69 MB
Beats 99.78%
100%

Q.13 [987. Vertical Order Traversal of a Binary Tree]

```
class Solution {
public:
    vector<vector<int>> verticalTraversal(TreeNode* root) {
        map<int,map<int,multiset<int>>>nodes;
        queue<pair<TreeNode*,pair<int,int>>>q;
        q.push({root,{0,0}});
        while(!q.empty()){
            auto t = q.front();
            q.pop();
            TreeNode* a = t.first;
            int x =t.second.first, y = t.second.second;
            nodes[x][y].insert(a->val);
            if(a->left){
                q.push({a->left,{x-1,y+1}});
            }
            if(a->right){
                q.push({a->right,{x+1,y+1}});
            }

        }
        vector<vector<int>>ans;
        for(auto p: nodes){
            vector<int>col;
            for(auto b:p.second){
                col.insert(col.end(),b.second.begin(),b.second.end());
            }
            ans.push_back(col);
        }
        return ans;
    }
};
```

Output :


scription
Editorial
Solutions
Accepted
All Submissions
Accepted
34 / 34 testcases passed
codex...
submitted at Feb 14, 2025 16:07
Solution
Runtime
2 ms
Beats 51.96%
Analyze Complexity
Memory
16.52 MB
Beats 17.91%
60%

Q.14 [94. Binary Tree Inorder Traversal]

```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int>res;
        DFS(root,res);
        return res;
    }
private:
    void DFS(TreeNode* r, vector<int>& res)
    {
        if(r==NULL)
        return;
        DFS(r->left,res);
        res.push_back(r->val);
        DFS(r->right,res);
    }
};
```

Output :