Name: Khushbu Chauhan
Uid:22BCS15830
SEC/GRP:FL_IOT_611/B

Experiment – 3(AP)

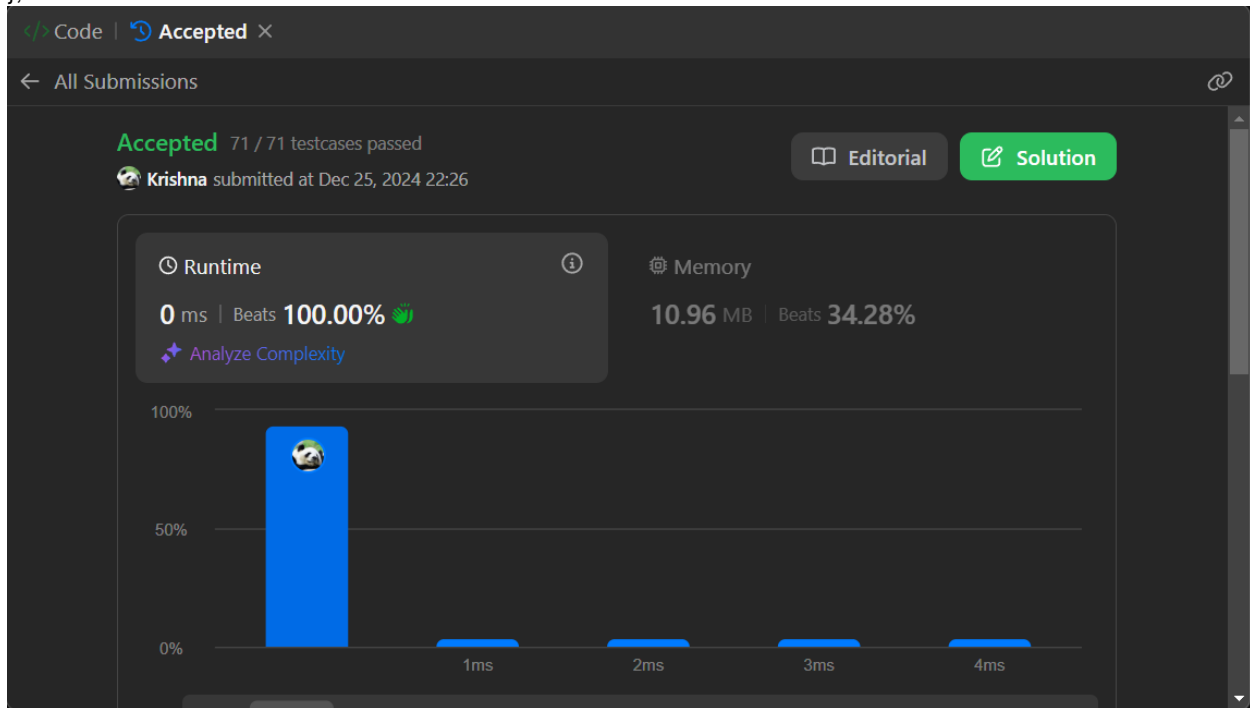94.[Binary Tree Inorder Traversal](#)

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

        vector<int>result;

        vector<int>lt=inorderTraversal(root->left);
        result.insert(result.end(),lt.begin(),lt.end());
        result.push_back(root->val);
        vector<int>rt=inorderTraversal(root->right);
        result.insert(result.end(),rt.begin(),rt.end());

        return result;
    }
};
```
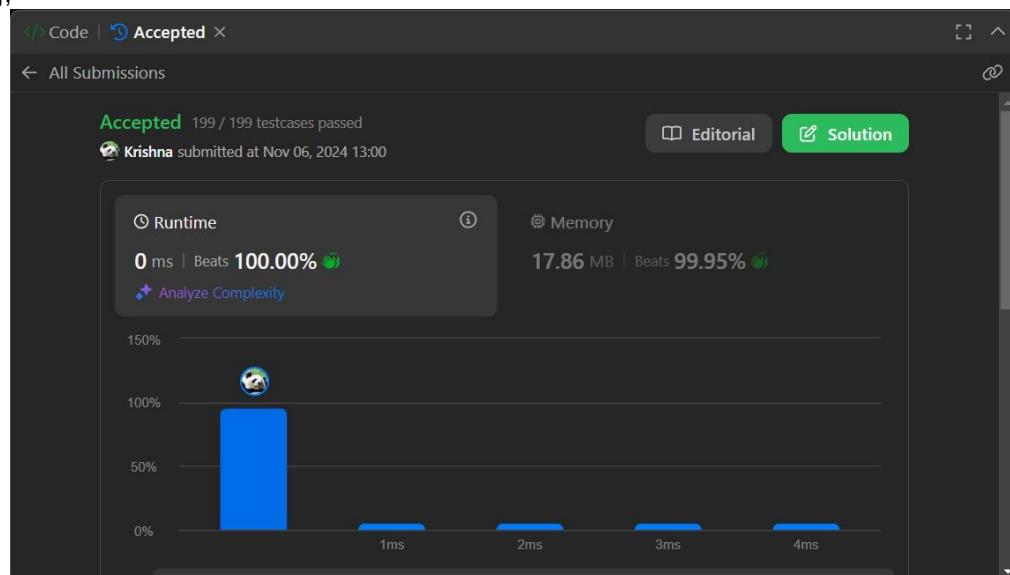
101. [Symmetric Tree](#)

```cpp
class Solution {
public:

bool ismirror(TreeNode* r,TreeNode* l)
{
    if(r==nullptr && l==nullptr)
    {
        return true;
    }
    if(r==nullptr || l==nullptr)
    {
        return false;
    }

    return (r->val==l->val) && ismirror(r->right,l->left) && ismirror(r->left,l->right);
}


    bool isSymmetric(TreeNode* root) {
        return ismirror(root->right,root->left);
    }
};
```
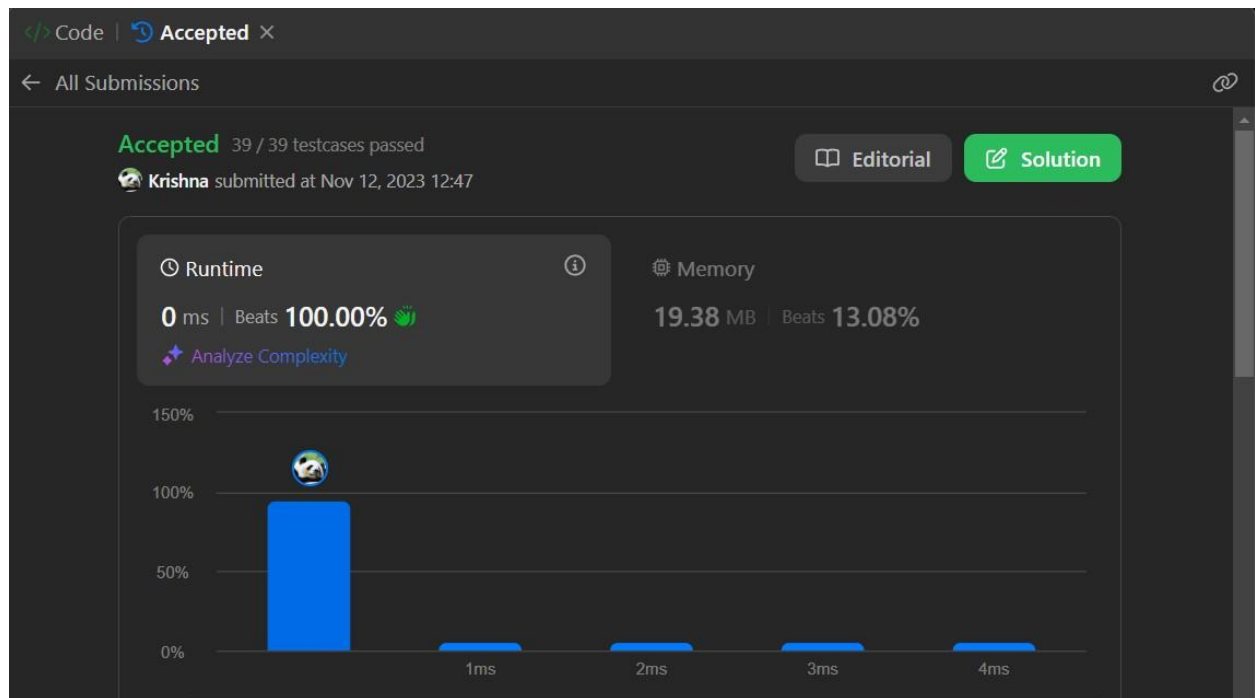
</> Code | ⟲ Accepted ×

← All Submissions

Accepted 199 / 199 testcases passed

Krishna submitted at Nov 06, 2024 13:00

📖 Editorial   ☑ Solution

🕐 Runtime   ⓘ   ⚙ Memory

0 ms | Beats **100.00%** 👋   **17.86** MB | Beats **99.95%** 👋

✦ Analyze Complexity

150%

100%

50%

0%

        1ms   2ms   3ms   4ms

104.[Maximum Depth of Binary Tree](#)

```cpp
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if(root==nullptr)
        {
            return 0;
        }
        int lh=maxDepth(root->left);
        int rh=maxDepth(root->right);

        return 1+max(lh,rh);
    }
};
```

98.[Validate Binary Search Tree](#)

```cpp
class Solution {
public:
    bool solve(TreeNode* root, long long minVal, long long maxVal) {
        if(!root) return true;
        if(root->val >= maxVal || root->val <= minVal) return false;
        return solve(root->left, minVal, root->val) && solve(root->right, root->val, maxVal);
    }
    bool isValidBST(TreeNode* root) {
        return solve(root, LLONG_MIN, LLONG_MAX);
    }
};
```
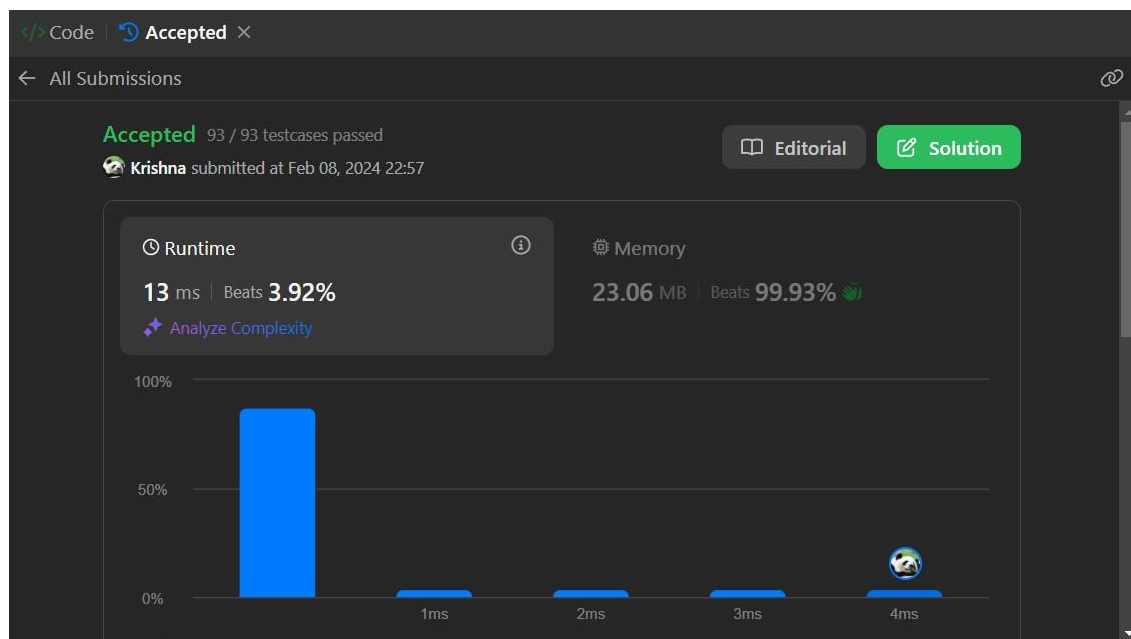
230.Kth Smallest Element in a BST

```cpp
class Solution {
public:
    int kthSmallest(TreeNode* root, int& k) {
        if(root==nullptr)
        {
            return INT_MAX;
        }
        int  leftans=kthSmallest(root->left,k);

        if(k==0)
        {
            return leftans;
        }

        if(--k==0)
        {
            return root->val;
        }

        return  kthSmallest(root->right,k);
    }
};
```
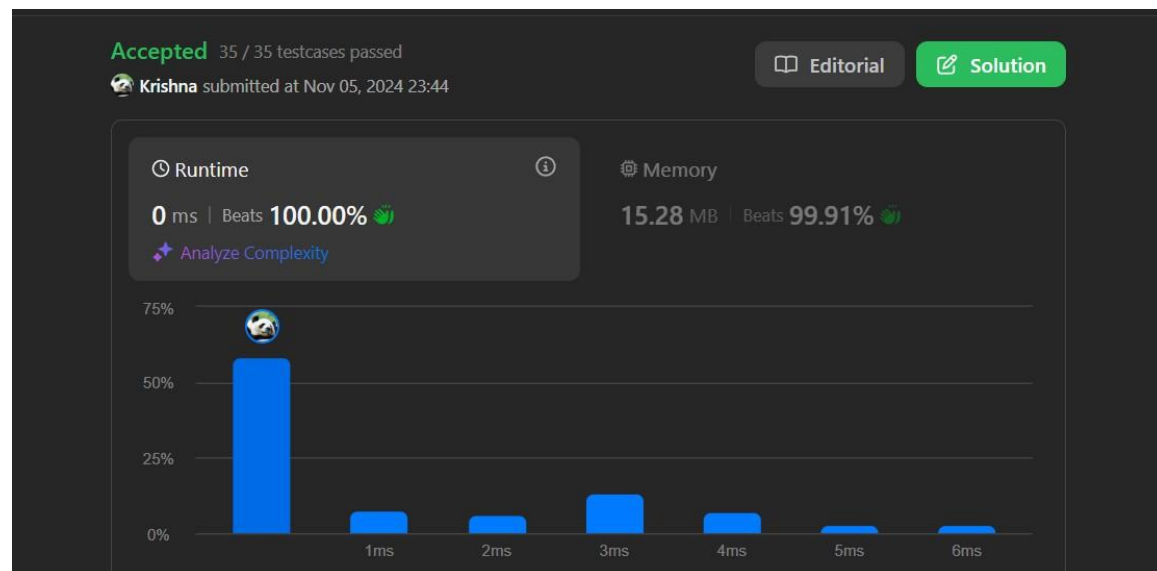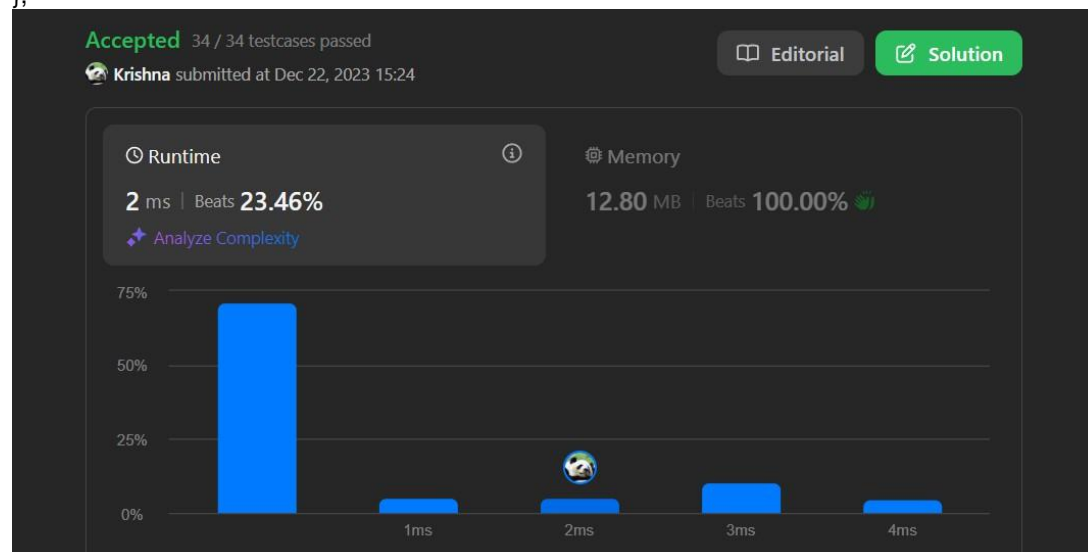
102. Binary Tree Level Order Traversal

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

107.<u>Binary Tree Level Order Traversal II</u>

```cpp
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<vector<int>> ans;
        multiset<int> pq{0};
        vector<pair<int, int>> points;
        for(auto b: buildings){
            points.push_back({b[0], -b[2]});
            points.push_back({b[1], b[2]});
        }
        sort(points.begin(), points.end());
        int ongoingHeight = 0;
        for(int i = 0; i < points.size(); i++){
            int currentPoint = points[i].first;
            int heightAtCurrentPoint = points[i].second;
            if(heightAtCurrentPoint < 0){
                pq.insert(-heightAtCurrentPoint);
            } else pq.erase(pq.find(heightAtCurrentPoint));
            auto pqTop = *pq.rbegin();
            if(ongoingHeight != pqTop){
                ongoingHeight = pqTop;
                ans.push_back({currentPoint, ongoingHeight});
            }
        }
 reverse(ans.begin(),ans.end());
return ans;
    }
};
```
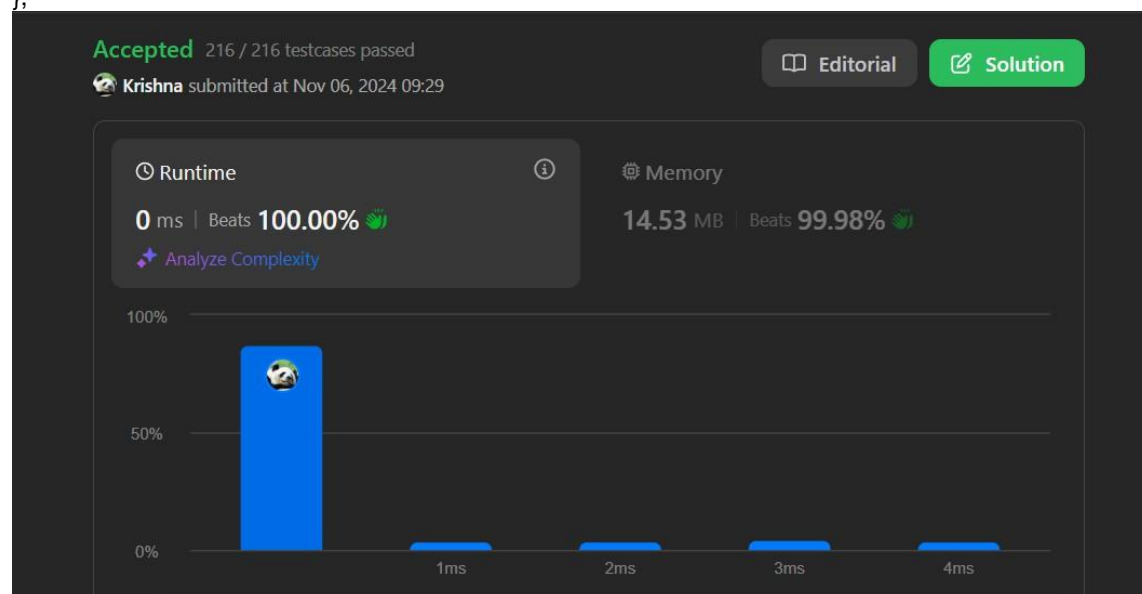
103. Binary Tree Zigzag Level Order Traversal

```cpp
class Solution {
public:
    vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
        vector<vector<int>>res;
        if(!root)
        {return res;}
        queue<TreeNode*>q;
        q.push(root);
        while(!q.empty()) {
        vector<int>ans;
         int n=q.size();
         int c=0;
         for(int i=0;i<n;i++) {
            TreeNode* temp=q.front();
            q.pop();
            ans.push_back(temp->val);
            if(temp->left) q.push(temp->left);
            if(temp->right) q.push(temp->right);
        }
        if(c%2!=0)  {
            reverse(ans.begin(),ans.end());
        }
         res.push_back(ans);
            c++;
        }
        return res;
    }
};
```
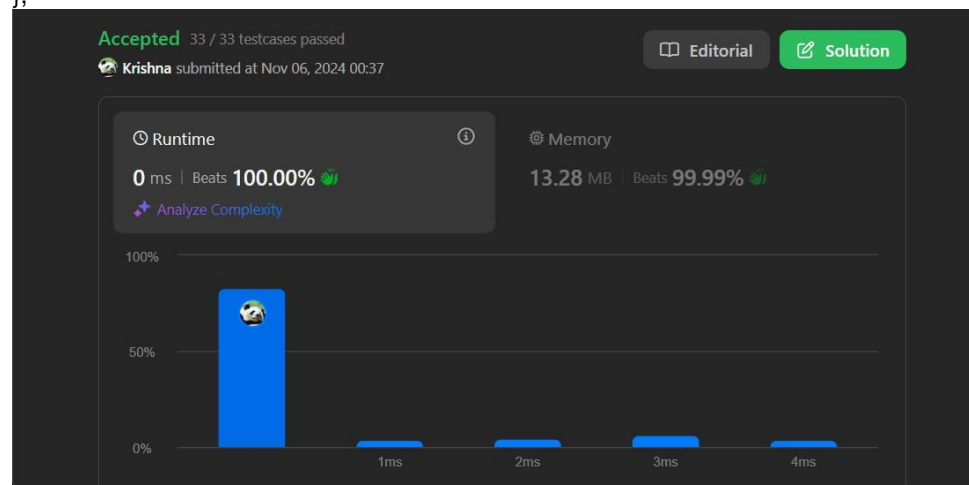
199.<u>Binary Tree Right Side View</u>

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


Accepted 216 / 216 testcases passed
Krishna submitted at Nov 06, 2024 09:29

Editorial   Solution

🕐 Runtime   ⓘ
0 ms | Beats **100.00%** 👏
✦ Analyze Complexity

⚙ Memory
14.53 MB | Beats **99.98%** 👏

100%

50%

0%
        1ms   2ms   3ms   4ms

106.Construct Binary Tree from Inorder and Postorder Traversal

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

private:
    TreeNode* buildTreeHelper(vector<int>& inorder, vector<int>& postorder, int inStart, int inEnd, int postIndex) {
        if (inStart > inEnd) return nullptr;
        TreeNode* root = new TreeNode(postorder[postIndex]);
        int inIndex = 0;
        for (int i = inStart; i <= inEnd; ++i) {
            if (inorder[i] == root->val) {
                inIndex = i;
                break;
            }
        }
        root->right = buildTreeHelper(inorder, postorder, inIndex + 1, inEnd, postIndex - 1);
        root->left = buildTreeHelper(inorder, postorder, inStart, inIndex - 1, postIndex - (inEnd - inIndex) - 1);
        return root;
    }
};
```
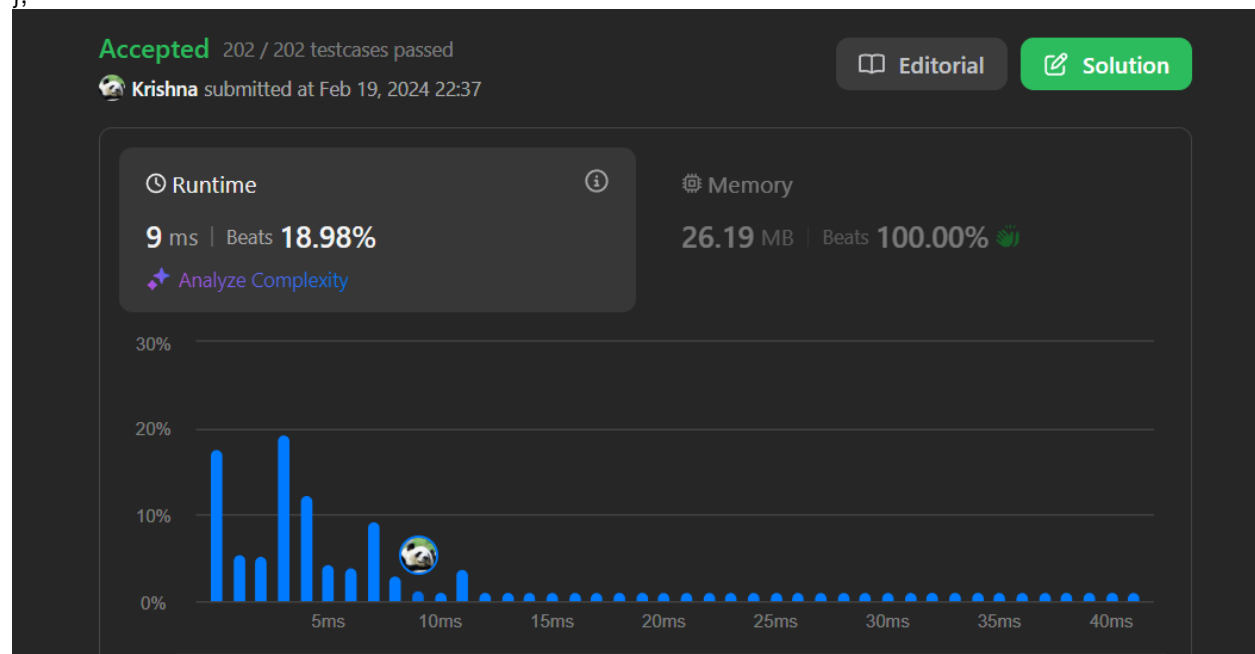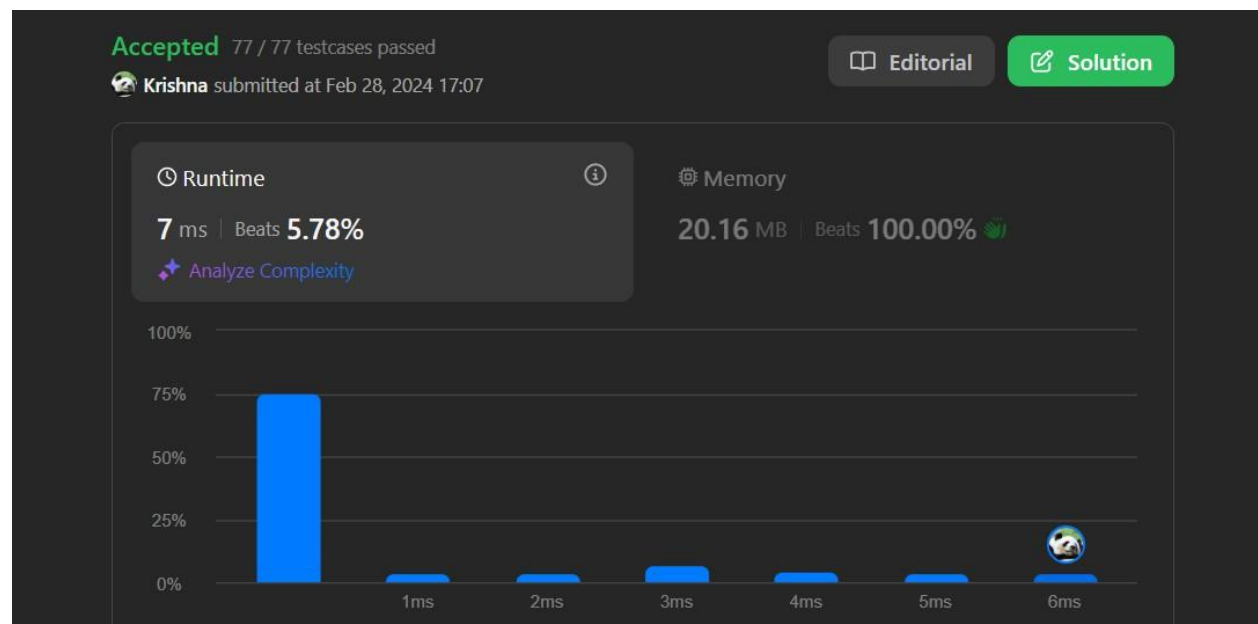
513.Find Bottom Left Tree Value

```cpp
class Solution {
public:
    int findBottomLeftValue(TreeNode* root) {
        if (!root) return -1;

        queue<TreeNode*> q;
        q.push(root);
        int result = -1;

        while (!q.empty()) {
            int size = q.size();
            result = q.front()->val;
            for (int i = 0; i < size; ++i) {
                TreeNode* node = q.front();
                q.pop();

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

        return result;
    }
};
```


Accepted 77 / 77 testcases passed
Krishna submitted at Feb 28, 2024 17:07

Runtime
7 ms | Beats 5.78%
Analyze Complexity

Memory
20.16 MB | Beats 100.00%

124. Binary Tree Maximum Path Sum

```cpp
class Solution {
public:
    int maxPathSum(TreeNode* root) {
        int maxSum = INT_MIN;
        maxGain(root, maxSum);
        return maxSum;
    }

private:
    int maxGain(TreeNode* node, int& maxSum) {
        if (!node) return 0;

        int leftGain = max(maxGain(node->left, maxSum), 0);
        int rightGain = max(maxGain(node->right, maxSum), 0);

        int priceNewpath = node->val + leftGain + rightGain;
        maxSum = max(maxSum, priceNewpath);

        return node->val + max(leftGain, rightGain);
    }
};
```
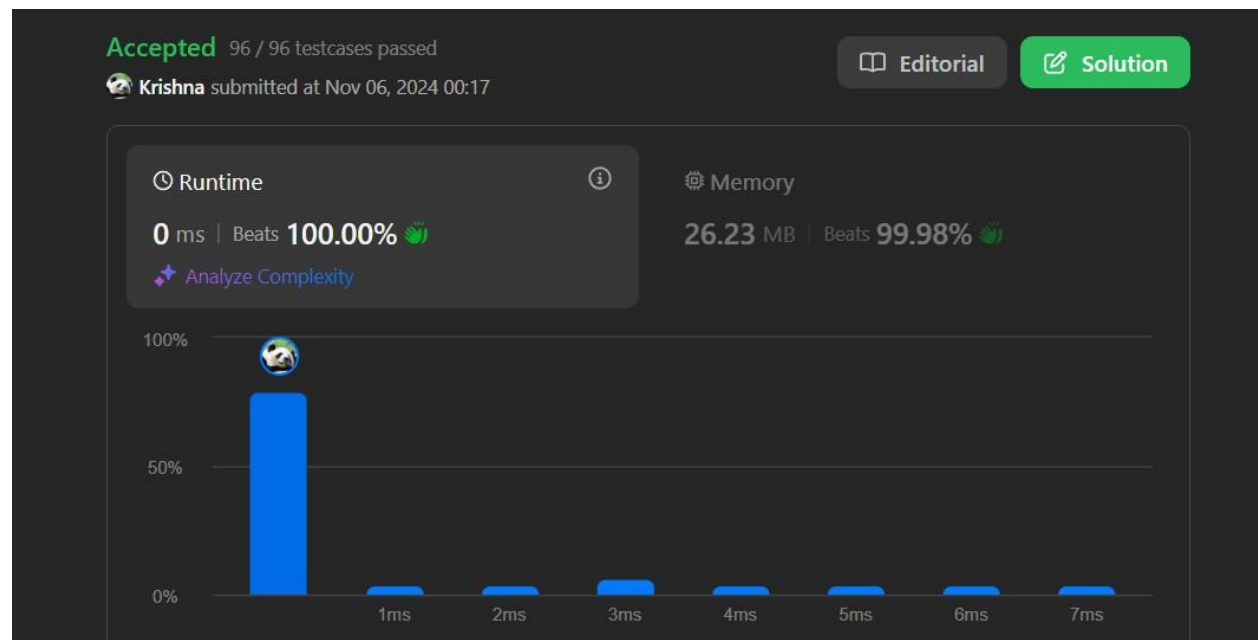
987.Vertical Order Traversal of a Binary Tree

```cpp
class Solution {
public:
    vector<vector<int>> verticalTraversal(TreeNode* root) {
        if (!root) return {};
        map<int, map<int, multiset<int>>> nodes;
        queue<pair<TreeNode*, pair<int, int>>> q;
        q.push({root, {0, 0}});
        while (!q.empty()) {
            auto p = q.front();
            q.pop();
            TreeNode* node = p.first;
            int x = p.second.first, y = p.second.second;
            nodes[x][y].insert(node->val)
            if (node->left) {
                q.push({node->left, {x - 1, y + 1}});
            }
            if (node->right) {
                q.push({node->right, {x + 1, y + 1}});
            }
        }
        vector<vector<int>> result;
        for (auto& p : nodes) {
            vector<int> col;
            for (auto& q : p.second) {
                col.insert(col.end(), q.second.begin(), q.second.end());
            }
            result.push_back(col);
        }
        return result;
    }
};
```


Accepted 34 / 34 testcases passed

Krishna submitted at Nov 06, 2024 09:18

Editorial   Solution

Runtime
1 ms | Beats 60.15%
Analyze Complexity

Memory
14.24 MB | Beats 100.00%