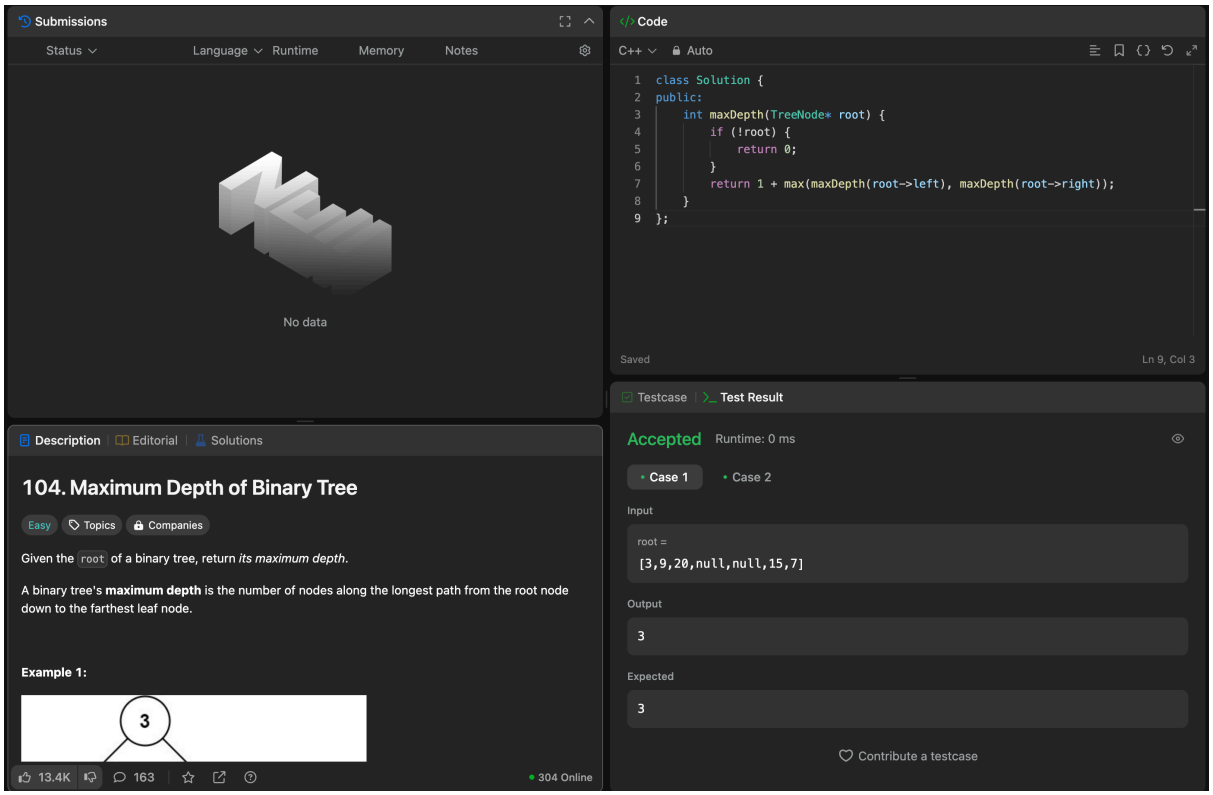
Assignment 7

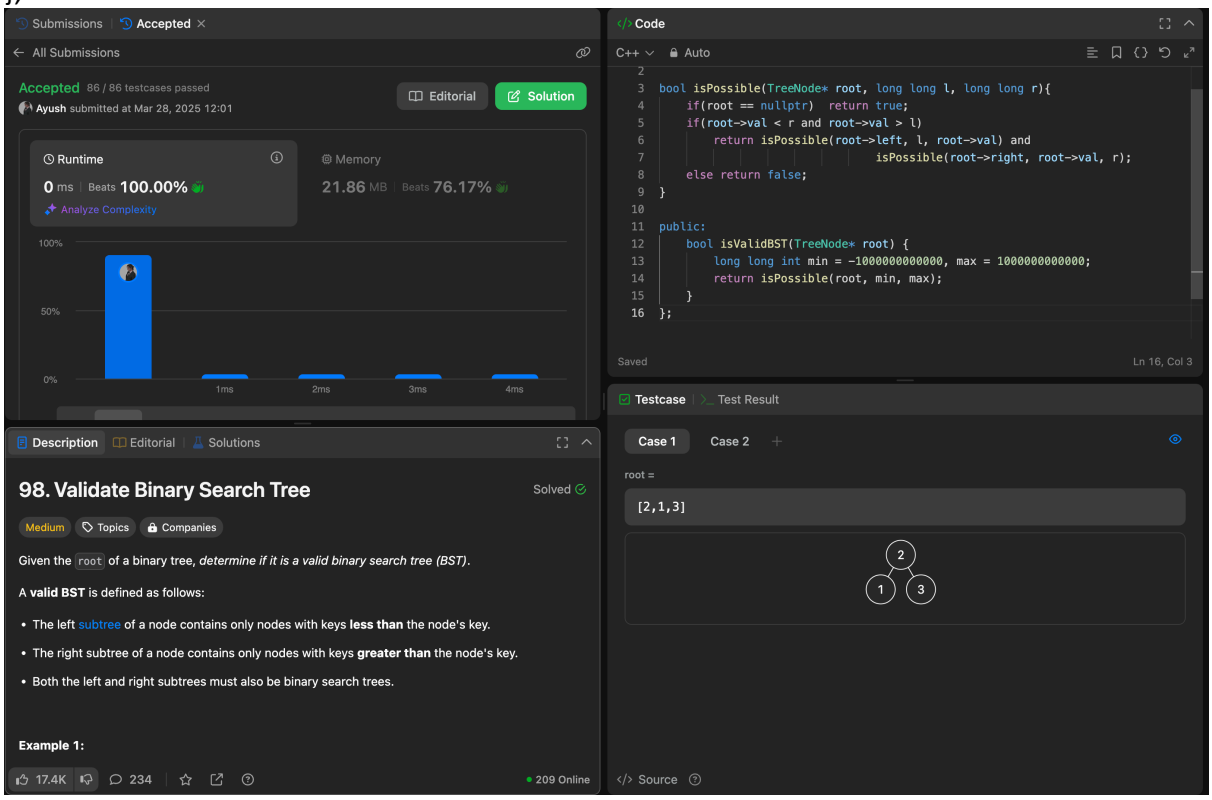
Problem 1 - [Maximum Depth of Binary Tree]

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



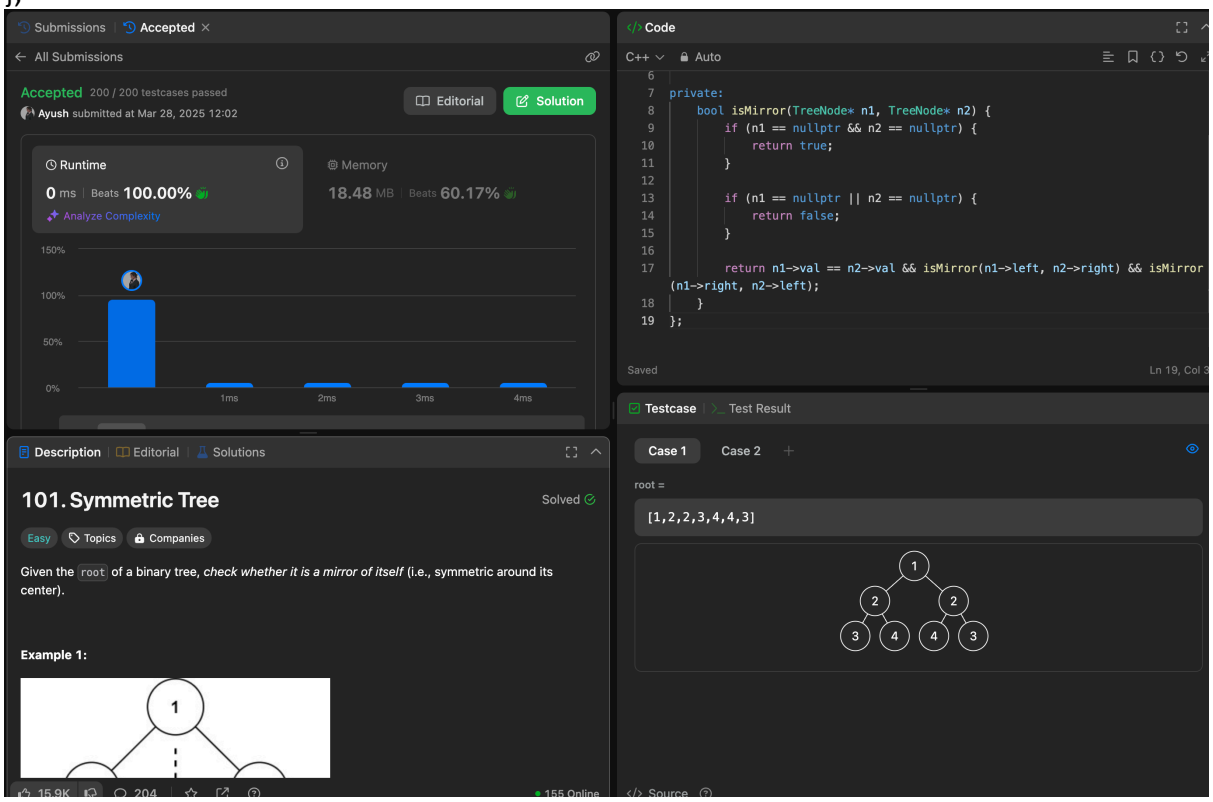
Problem 2 - [Validate Binary Search Tree]

```
class Solution {

bool isPossible(TreeNode* root, long long l, long long r){
   if(root == nullptr)  return true;
   if(root->val < r and root->val > l)
      return isPossible(root->left, l, root->val) and
                        isPossible(root->right, root->val, r);
   else return false;
}

public:
   bool isValidBST(TreeNode* root) {
      long long int min = -1000000000000, max = 1000000000000;
      return isPossible(root, min, max);
   }
};
```



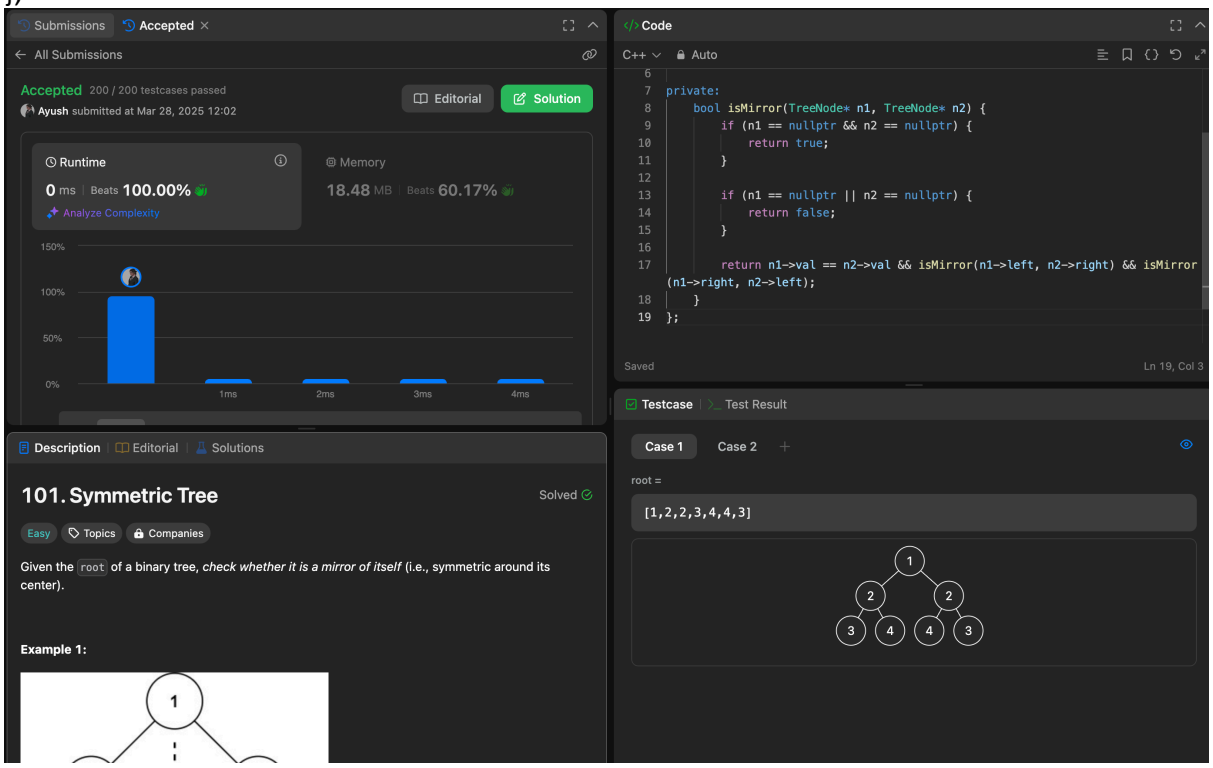
Problem 3 – [Symmetric Tree]

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

private:
    bool isMirror(TreeNode* n1, TreeNode* n2) {
        if (n1 == nullptr && n2 == nullptr) {
            return true;
        }

        if (n1 == nullptr || n2 == nullptr) {
            return false;
        }

        return n1->val == n2->val && isMirror(n1->left, n2->right) && isMirror(n1->right, n2->left);
    }
};
```



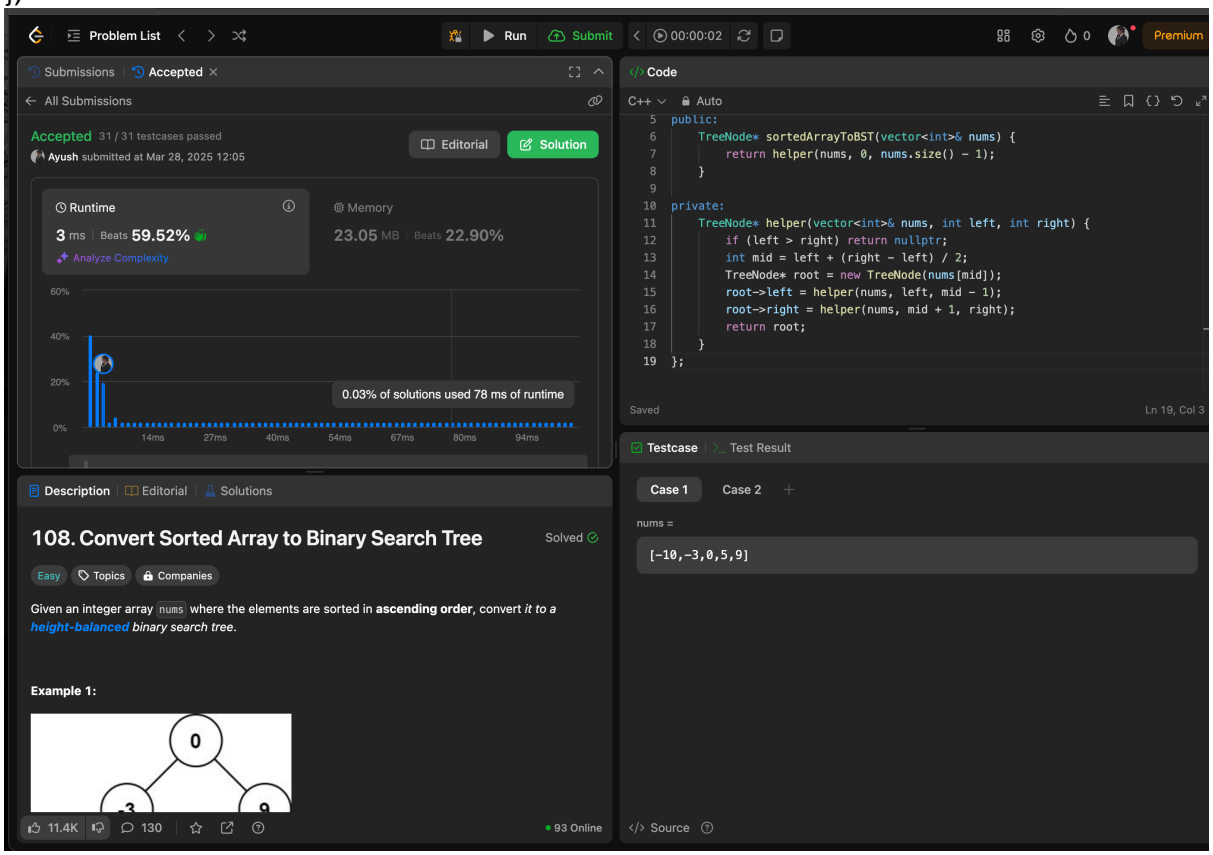
Problem 4 - [Binary Tree Level Order Traversal]

```
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;
   }
};
```



Problem 5- [Convert Sorted Array to Binary Search Tree]

```
#include <vector>
using namespace std;

class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        return helper(nums, 0, nums.size() - 1);
    }

private:
    TreeNode* helper(vector<int>& nums, int left, int right) {
        if (left > right) return nullptr;
        int mid = left + (right - left) / 2;
        TreeNode* root = new TreeNode(nums[mid]);
        root->left = helper(nums, left, mid - 1);
        root->right = helper(nums, mid + 1, right);
        return root;
    }
};
```



Problem 6 - [Binary Tree Inorder Traversal]

```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
    vector<int> ans;
    if (root == NULL) return ans;
    vector<int> left = inorderTraversal(root->left);
    ans.insert(ans.end(), left.begin(), left.end());
    ans.push_back(root->val);
    vector<int> right = inorderTraversal(root->right);
    ans.insert(ans.end(), right.begin(), right.end());
    return ans;
}

};
```