AP Assignment-3

Name: Shivangi Gupta UID: 22BCS15008 Section: 601-A

94. https://leetcode.com/problems/binary-tree-inorder-traversal/

Code:

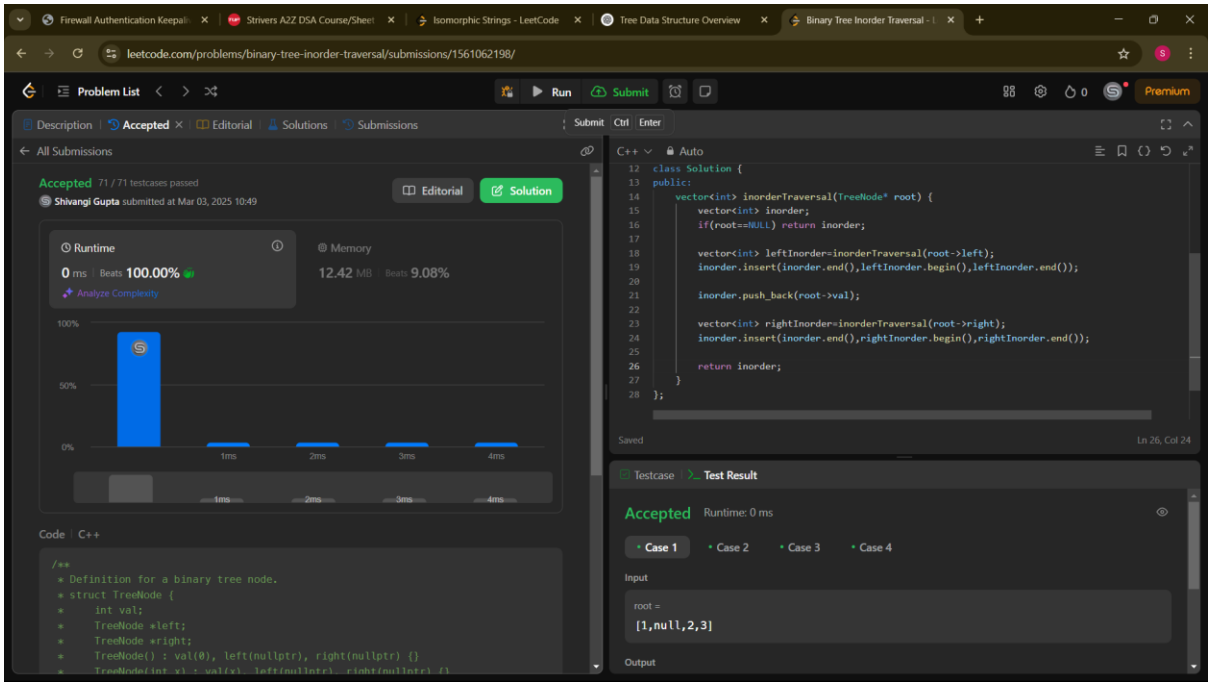
```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> inorder;
        if(root==NULL) return inorder;

        vector<int> leftInorder=inorderTraversal(root->left);
        inorder.insert(inorder.end(),leftInorder.begin(),leftInorder.end());

        inorder.push_back(root->val);

        vector<int> rightInorder=inorderTraversal(root->right);
        inorder.insert(inorder.end(),rightInorder.begin(),rightInorder.end());

        return inorder;
    }
};
```

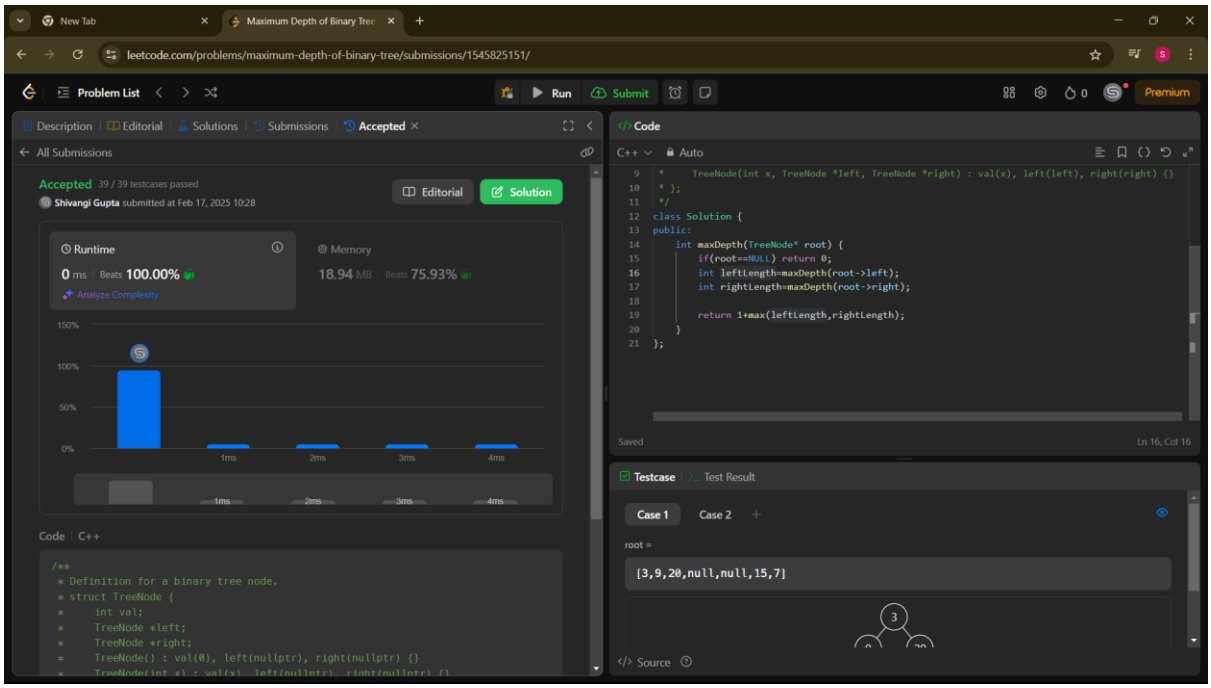


104. https://leetcode.com/problems/maximum-depth-of-binary-tree/submissions/1545825151/

Code:

```
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if(root==NULL) return 0;
        int leftLength=maxDepth(root->left);
        int rightLength=maxDepth(root->right);

        return 1+max(leftLength,rightLength);
    }
};
```



102. https://leetcode.com/problems/binary-tree-level-order-traversal/description/

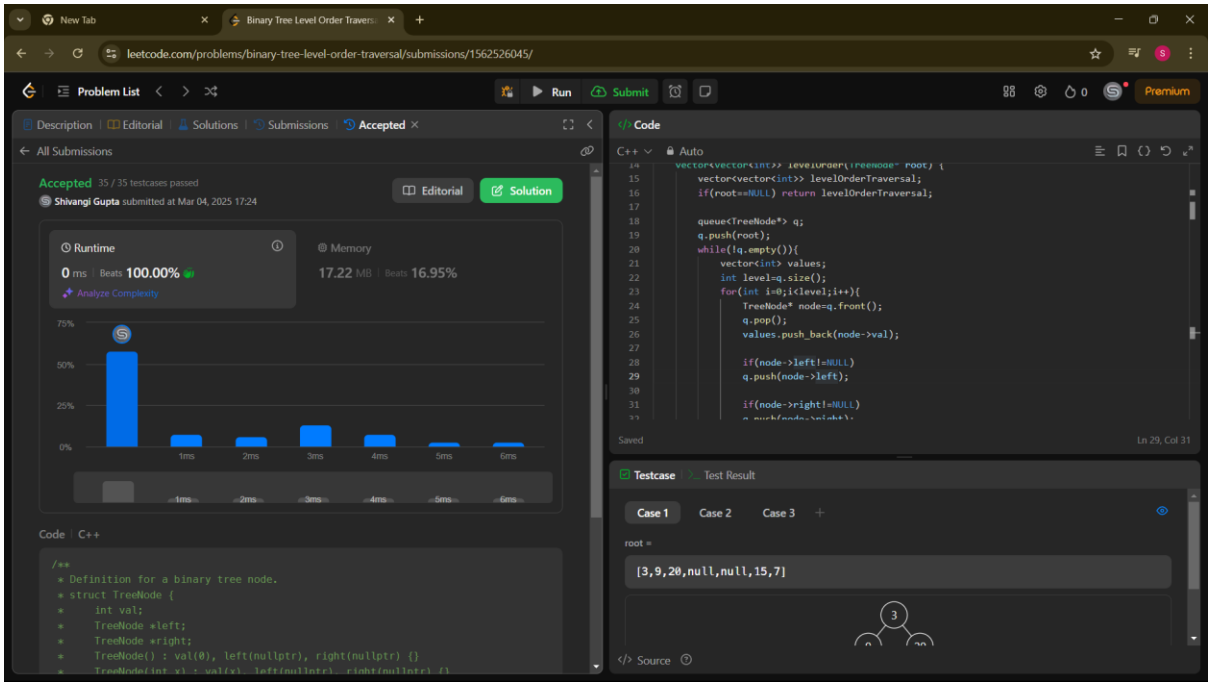
Code:

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> levelOrderTraversal;
        if(root==NULL) return levelOrderTraversal;
```

```
        queue<TreeNode*> q;
        q.push(root);
        while(!q.empty()){
            vector<int> values;
            int level=q.size();
            for(int i=0;i<level;i++){
                TreeNode* node=q.front();
                q.pop();
                values.push_back(node->val);

                if(node->left!=NULL)
                q.push(node->left);

                if(node->right!=NULL)
                q.push(node->right);
            }
            levelOrderTraversal.push_back(values);
        }
        return levelOrderTraversal;
    }
};
```



107. https://leetcode.com/problems/binary-tree-level-order-traversal-ii/description/

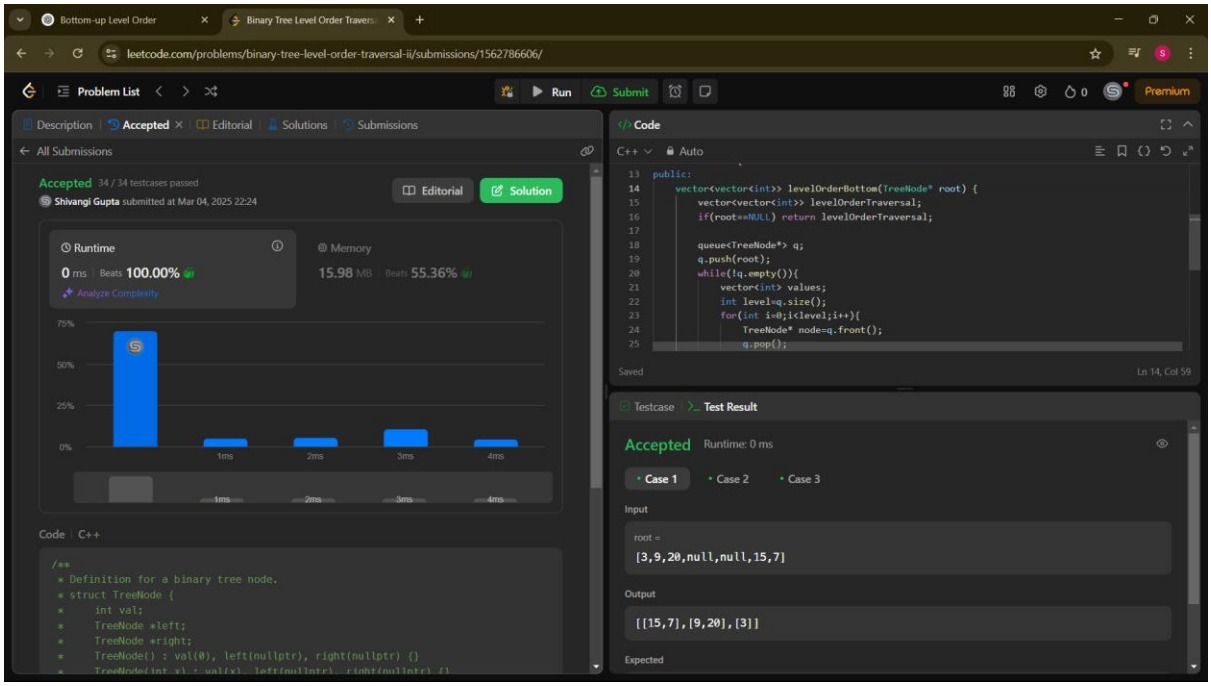
Code:

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

        queue<TreeNode*> q;
        q.push(root);
        while(!q.empty()){
            vector<int> values;
            int level=q.size();
            for(int i=0;i<level;i++){
                TreeNode* node=q.front();
                q.pop();
                values.push_back(node->val);

                if(node->left!=NULL)
                q.push(node->left);
```

```
            if(node->right!=NULL)
            q.push(node->right);
          }
          levelOrderTraversal.push_back(values);
        }
        reverse(levelOrderTraversal.begin(),levelOrderTraversal.end());
        return levelOrderTraversal;
    }
};
```



101. https://leetcode.com/problems/symmetric-tree/description/

Code:

```
bool isMirror(TreeNode* t1, TreeNode* t2) {
    if (t1 == NULL && t2 == NULL) {
        return true;
    }
    if (t1 == NULL || t2 == NULL) {
        return false;
```

```
        }
        return (t1->val == t2->val) &&
                isMirror(t1->left, t2->right) &&
                isMirror(t1->right, t2->left);
    }
    bool isSymmetric(TreeNode* root) {
        if (root == NULL) {
            return true;
        }
        return isMirror(root->left, root->right);
    }
};
```