Experiment-5

Student Name: Nikhil Kumar
UID: 22BCS15501
Branch: BE-CSE
Section/Group: KPIT-901/B
Semester: 6th
Date of Performance: 18/01/25
Subject Name: Advanced Programming Lab - 2
Subject Code: 22CSP-351

1. **Aim:** Tree.
   1. Problem: 104. Maximum Depth of Binary Tree.
   2. Problem: 108. Convert Sorted Array to Binary Search Tree.

2. **Objective:**
   1. Maximum Depth of Binary Tree: Determine the longest path from the root to the farthest leaf node in a binary tree.
   2. Convert Sorted Array to BST: Convert a sorted array into a height-balanced binary search tree (BST).

3. **Implementation/Code:**

1.)

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

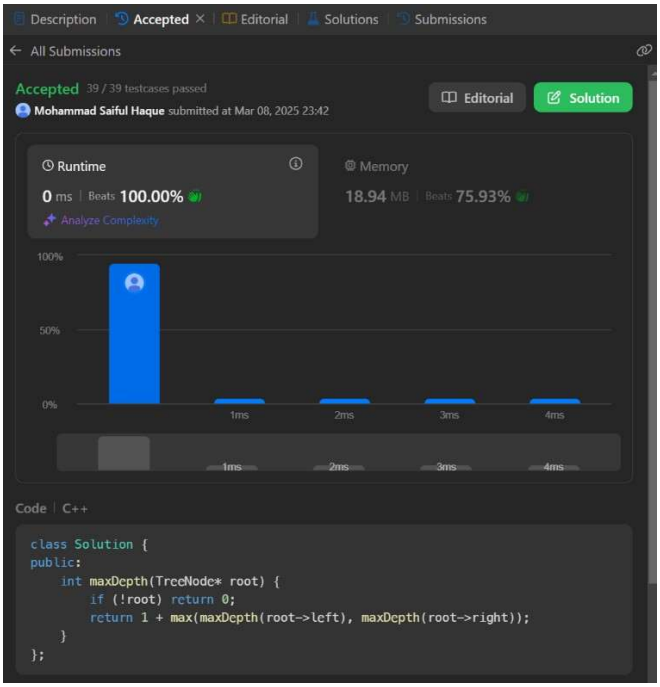
2.)

```
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
```

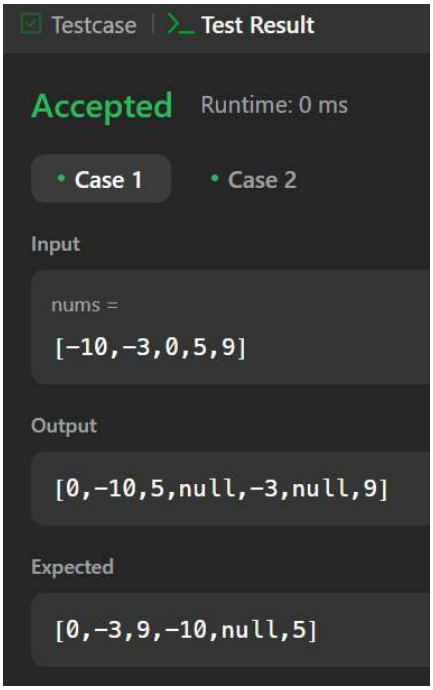
```
        return helper(nums, 0, nums.size() - 1);
    }

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;
    }
};
```

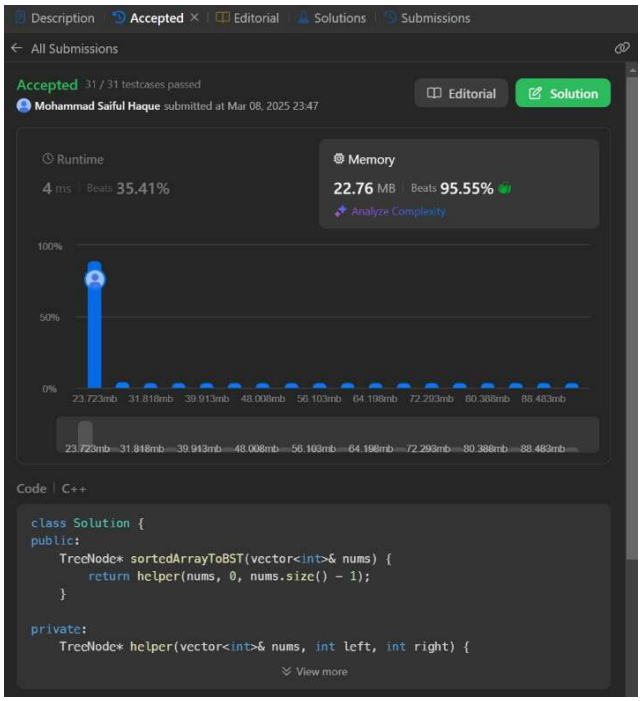
4. Output:

1.



2.





5. Time Complexity:

1. O(n)
2. O(n)

6. Space Complexity:

1. O(h)
2. O(log(n))

7. Learning Outcome:

1. Understand recursion and depth-first search (DFS) in trees.
2. Learn how to calculate tree depth efficiently.
3. Apply the divide and conquer approach to construct a balanced BST.
4. Gain insight into binary search and its application in tree construction.