Experiment- 5A

Student Name: Roshan Kumar **UID:** 22BCS16490
Branch:BE-CSE **Section/Group:** NTPP 602-A
Semester:6th **Date of Performance:**10/02/25
Subject Name: AP Lab-2 **Subject Code: 22CSH-**352

1. TITLE:

Maximum Depth of Binary Tree

2. AIM:

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

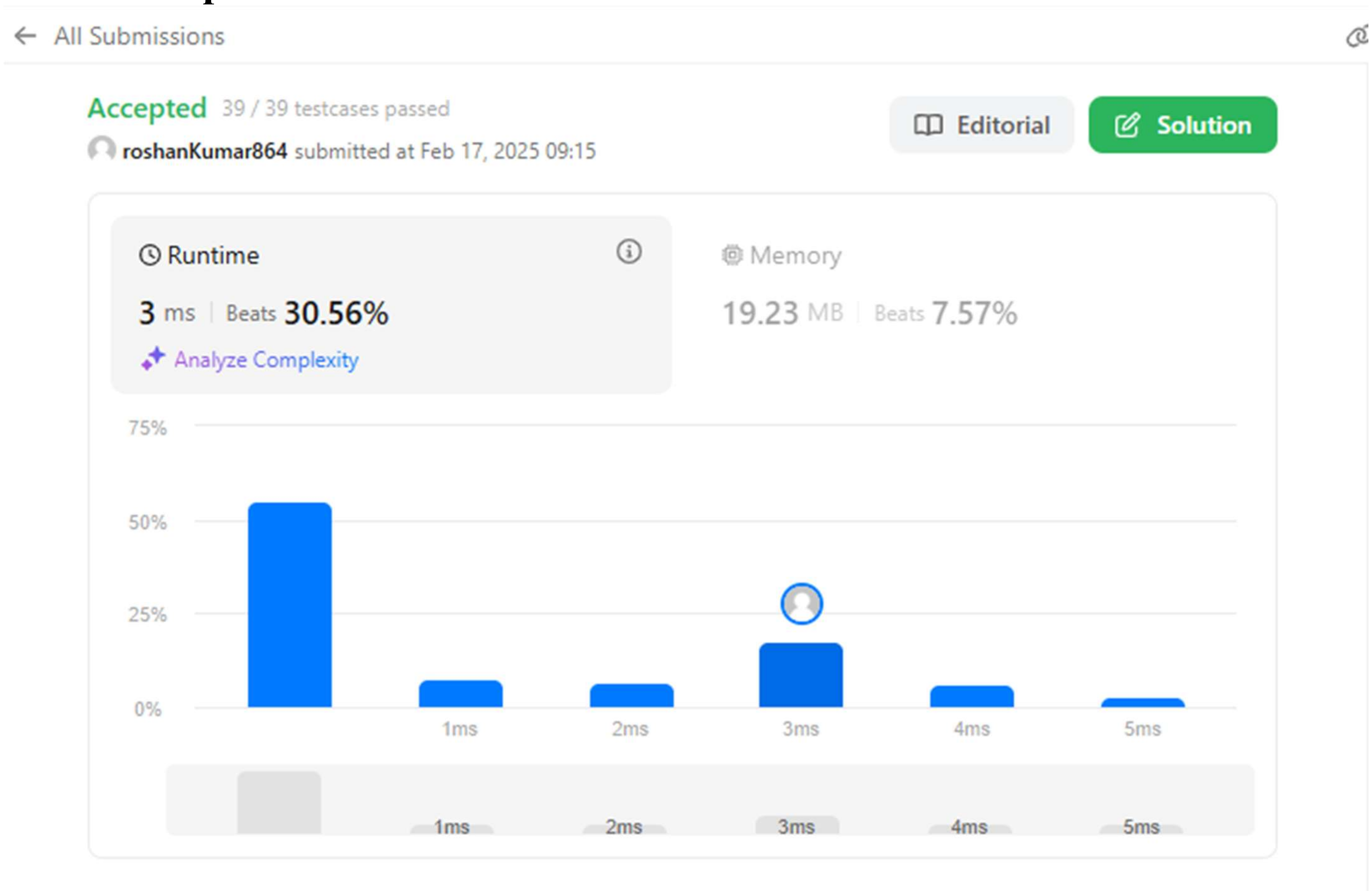
3. Algorithm

- If the node is None, return 0 (an empty tree has depth 0).
- Compute the depth of the left subtree.
- Compute the depth of the right subtree.
- The maximum depth is 1 + max(left_depth, right_depth).

Implemetation/Code

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

Output:



Time Complexity : O(n)

Space Complexity : O(h)

Learning Outcomes:-

1. Gain a deeper understanding of **binary tree structures**, specifically how to calculate the depth or height of a tree.
2. Learn the concept of **maximum depth**, which is the number of nodes along the longest path from the root node down to the farthest leaf node.

Experiment - 2B

Student Name: Roshan Kumar **UID:** 22BCS16490
Branch: BE-CSE **Section/Group:** NTPP- 602_A
Semester: 6th **Date of Performance:** 10/02/25
Subject Name: AP Lab-2 **Subject Code:** 22CSH-352

1. TITLE:

Symmetric Tree.

2. AIM:

Given the root of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

3. Algorithm

- If the tree is empty (`root is None`), return `True`.
- Check if the **left subtree** is a mirror of the **right subtree**.
- Two subtrees are mirrors if:
- Their root values are equal.
- The **left subtree of one** matches the **right subtree of the other**.

Implemetation/Code:

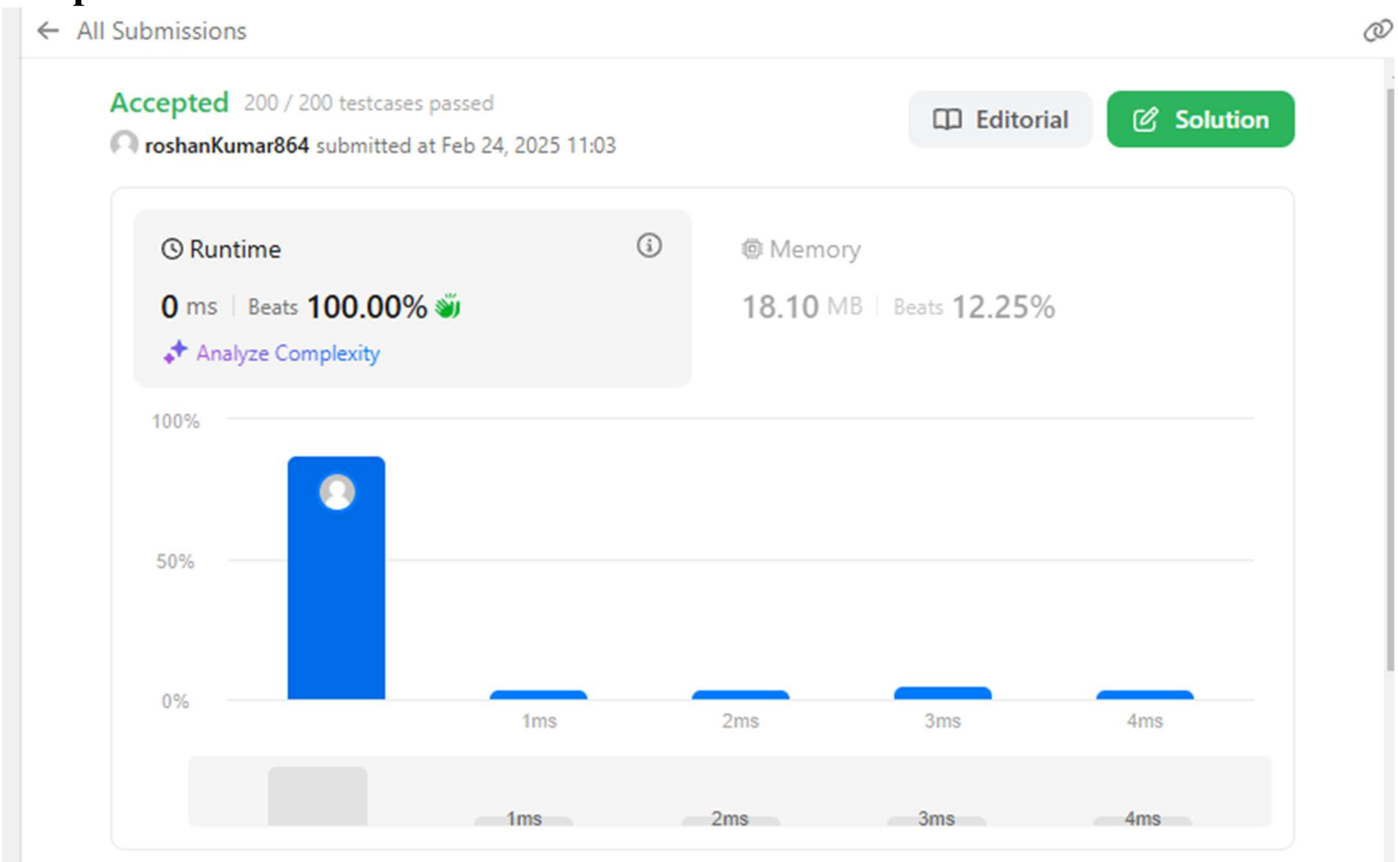
```
class Solution {
public:
bool isSymmetric(TreeNode* root) {
return isSymmetric(root, root);
}

private:
bool isSymmetric(TreeNode* p, TreeNode* q) {
if (!p || !q)
return p == q;

return p->val == q->val &&          //
isSymmetric(p->left, q->right) &&  //
isSymmetric(p->right, q->left);
}
```

};

Output:



Time Complexity : O(N)

Space Complexity : O(h)

Learning Outcomes:-

1. Gain a deeper understanding of **binary tree structures**, specifically how to navigate left and right subtrees.
2. Learn the concept of **symmetric trees**, where the left and right subtrees are mirror images of each other.