Experiment 5 A

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:17/02/25**

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

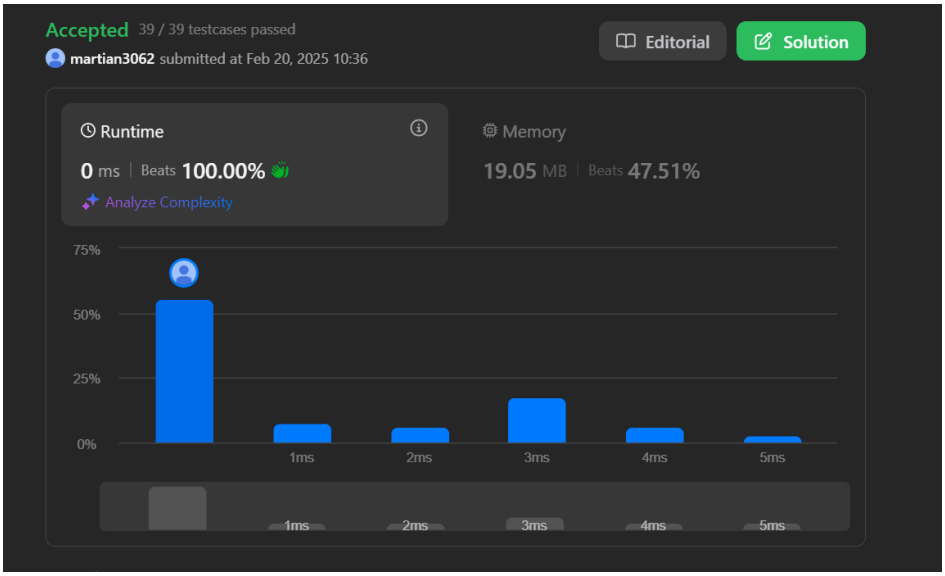
- Define TreeNode class to represent each node with val, left, and right.
- Define maxDepth function to return 0 for None and 1 + max of left and right subtree depths.
- Call maxDepth on the root node.
- Return depth of the tree.

Implemetation/Code

```
class TreeNode:
    def _init_(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def maxDepth(self, root):
        if not root: return 0
        return 1 + max(self.maxDepth(root.left), self.maxDepth(root.right))
if __name__ == "__main__":
    Solution()
```

Output



Time Complexity : O(n)

Space Complexity : O(n)

Learning Outcomes:-

- Learn how to traverse a tree recursively to calculate its maximum depth.
- Understand how to handle base cases in recursion, like when the node is None.

Experiment 5 B

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:17/02/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Kth Smallest Element in a BST

2. AIM:

Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of the nodes in the tree.

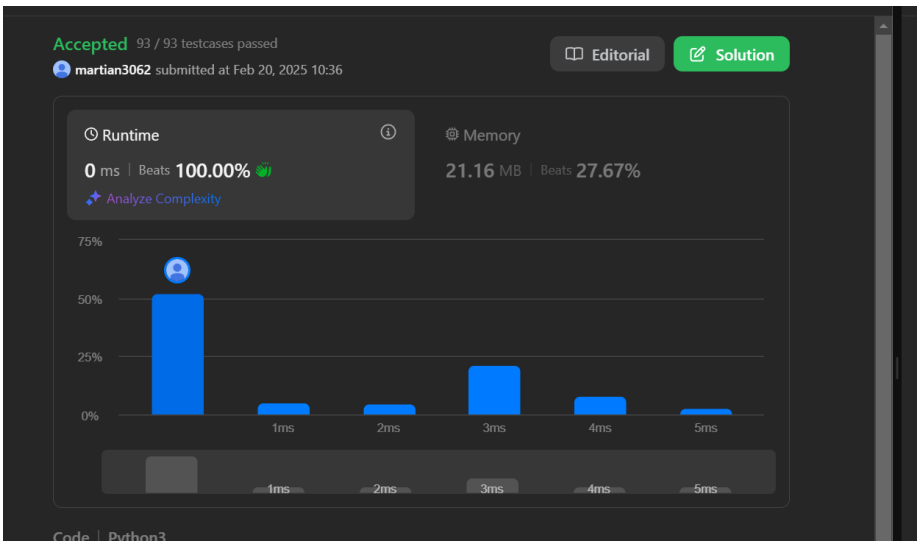
3. Algorithm

- Define an inorder function to recursively yield values of the tree in sorted order.
- Create a generator from the inorder function starting at the root node.
- Iterate k times to get the kth smallest value by calling next() on the generator.
- Return the kth smallest value once it's retrieved from the generator.

Implemetation/Code:

```
class Solution:
    def kthSmallest(self, root: TreeNode, k: int) -> int:
        def inorder(node):
            if not node:
                return
            yield from inorder(node.left)
            yield node.val
            yield from inorder(node.right)
        gen = inorder(root)
        for _ in range(k):
            result = next(gen)
        return result
```

Output



Time Complexity : O(n)

Space Complexity : O(h)

Learning Outcomes:-

- Implement an inorder traversal using a generator to yield node values in sorted order.
- kth smallest element in a binary search tree by utilizing an efficient traversal approach.