Experiment 3 A

Student Name: Harsh Raj

Branch: BE-CSE Semester: 6TH

Subject Name: AP Lab-2

UID: 22BCS16961

Section/Group: NTPP-602-A
Date of Performance: 03/02/25

Subject Code: 22CSH-352

1. TITLE:

Maximum Depth of Binary Tree

2. AIM:

Given the root of a binary tree, return its maximum depth.

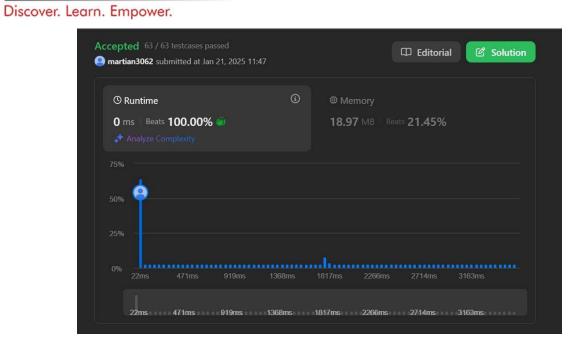
A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

3. Algorithm

- o Start DFS with the root node at depth 0.
- o If the node is null, return the current depth.
- o Recursively explore left and right children, increasing depth by 1.
- o Return the maximum depth from left or right subtree.

Implemetation/Code

```
class Solution:
    def maxDepth(self, root: Optional[TreeNode]) -> int:
        def dfs(root, depth):
        if not root: return depth
        return max(dfs(root.left, depth + 1), dfs(root.right, depth + 1))
        return dfs(root, 0)
```



Time Complexity : O(n)

Space Complexity: O(h)

Learning Outcomes:-

- o Understand how to use depth-first search for tree traversal.
- o Gain skills in calculating the depth or height of binary trees.



Experiment 3 B

Student Name: Reeva UID: 22BCS16744

Branch: BE-CSE Section/Group: Ntpp 602-A
Semester: 6TH Date of Performance: 03/02/25

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

- Perform an in-order traversal of the binary tree starting from the root.
- Use a generator to yield nodes' values one by one in their in-order sequence.
- Iterate up to the kth element of the generator.
- Return the kth smallest element from the traversal.

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
```

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(k)

Space Complexity : O(h)

Learning Outcomes:-

- Learn how to perform and apply in-order traversal in binary trees to solve problems.
- Python generators to manage state and produce results on demand during tree traversal

