WORKSHEET 5

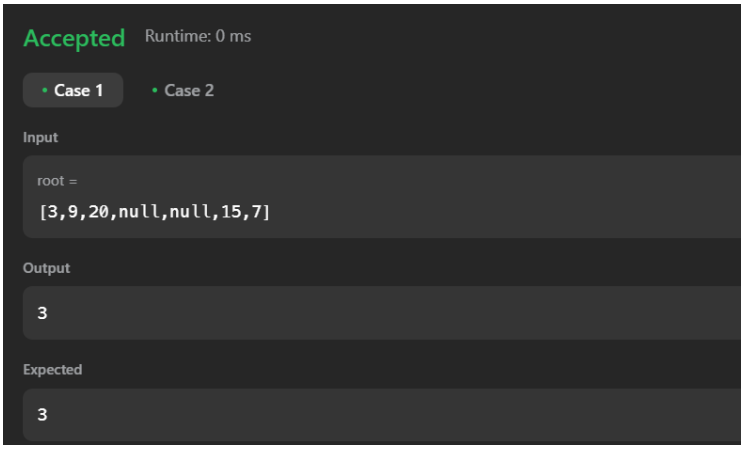
Student Name: Reeva
UID: 22BCS16744
Branch: BE-CSE
Section/Group: 22BCS_NTPP-602-A
Semester: 6th
Date of Performance: 17/02/2025
Subject Name: AP LAB - II
Subject Code: 22CSP-351

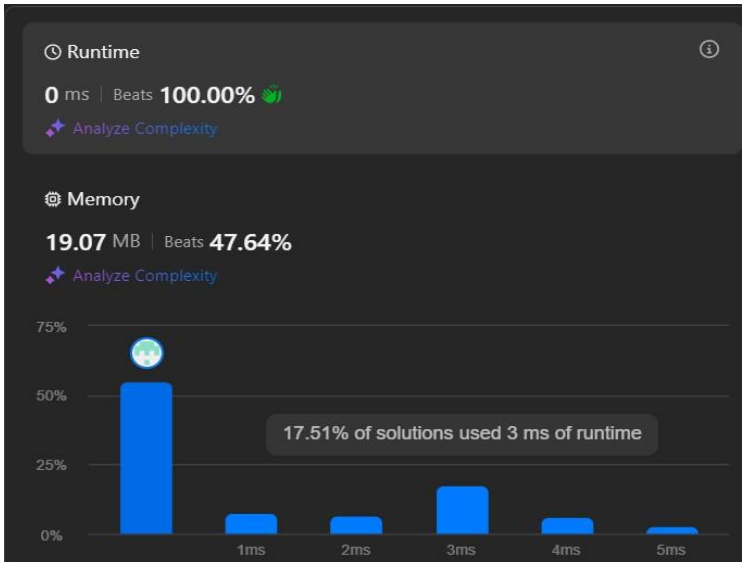
1. **Aim:** Given the root of a binary tree, return *its maximum depth*.
   A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

2. **Source 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__":
3.      Solution()
```

4. Screenshots of outputs:





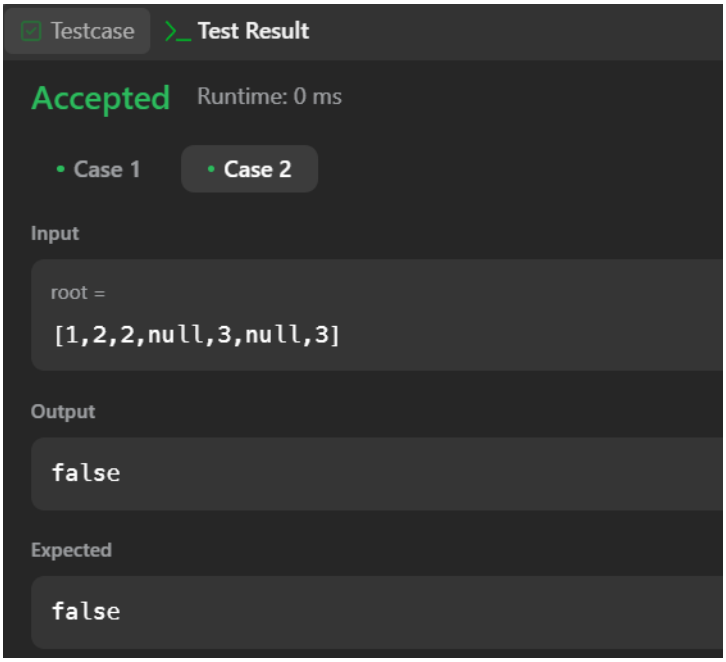
2. Aim: Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

Source Code:

```
from typing import Optional
from collections import deque

class Solution:
```

```
def isSymmetric(self, root: Optional['TreeNode']) -> bool:
    q = deque([(root, root)])
    while q:
        t1, t2 = q.popleft()
        if not t1 and not t2:
            continue
        if not t1 or not t2 or t1.val != t2.val:
            return False
        q.append((t1.left, t2.right))
        q.append((t1.right, t2.left))
    return True
```

Screenshots of outputs:



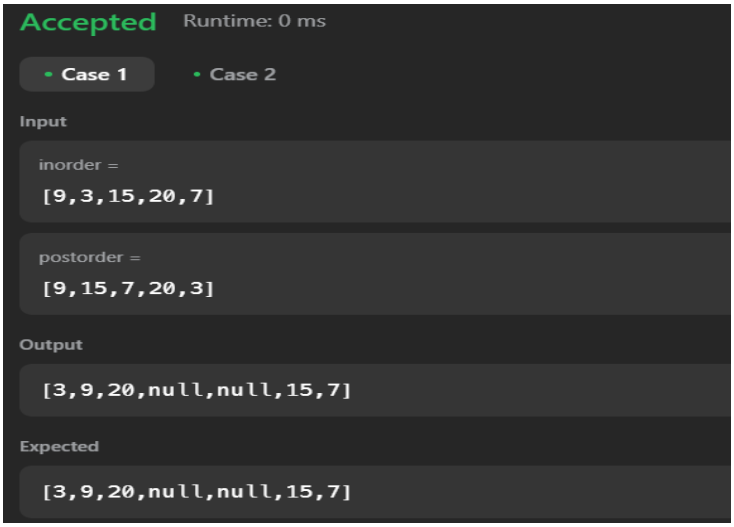
3. Aim: Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.

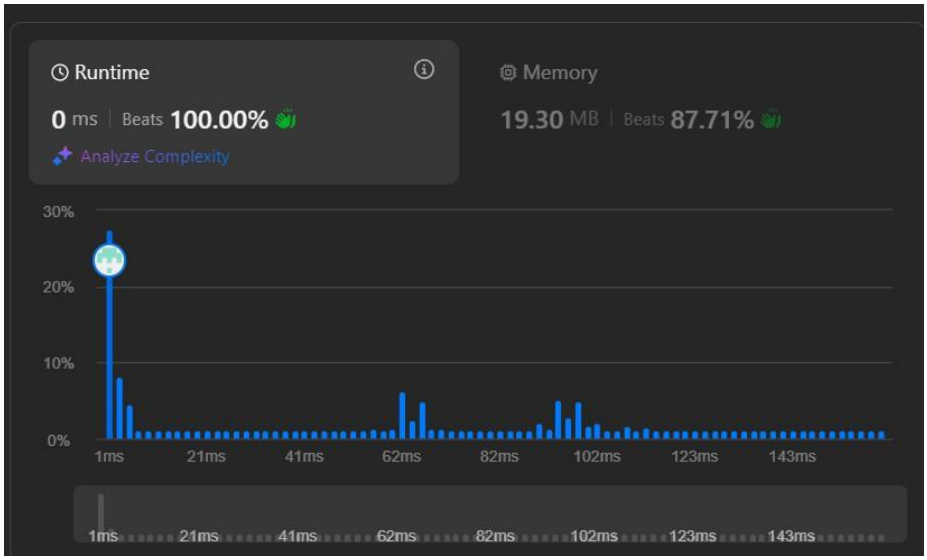
Source Code:

```
class Solution:
```

```
    def buildTree(self, inorder: List[int], postorder: List[int]) ->
Optional[TreeNode]:
        if not inorder or not postorder:
            return None

        inorder_map = {val: idx for idx, val in enumerate(inorder)}

        def construct(left: int, right: int) -> Optional[TreeNode]:
            if left > right:
                return None
            root_val = postorder.pop()
            root = TreeNode(root_val)
            inorder_idx = inorder_map[root_val]
            root.right = construct(inorder_idx + 1, right)
            root.left = construct(left, inorder_idx - 1)
            return root

        return construct(0, len(inorder) - 1)
```

4. Screenshots of outputs:




Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
19.30 MB | Beats 87.71%
30%
20%
10%
0%
1ms
21ms
41ms
62ms
82ms
102ms
123ms
143ms