EXPERIMENT – 5

Student Name: Yuktam Singla **UID: 22BCS10908**

Branch: BE-CSE **Section/Group: NTPP-601-A**

Semester: 6 **Date of Performance: 13-02-2025**

Subject Name: Advanced Programming 2 **Subject Code: 22CSP-351**

1. Aim:

(a) **Same Tree**: Given the roots of two binary trees p and q, write a function to check if they are the same or not.

(b) **Balanced Binary Tree:** Given a binary tree, determine if it is height-balanced.

2. Objectives:

- Given the roots of two binary trees check if they are the same or not.
- Check if height-balanced or not.

3. Algorithm:

➢ **Same Tree:**

- If either tree is None, return whether both are None.
- If node values differ, return False.
- Compare left and right subtrees recursively.

➢ **Balanced Binary Tree:**

- If the tree is empty (root is None), return True.
- Compute the height (maxDepth) of the left and right subtrees.
- If their height difference is more than 1, return False.
- Ensure both left and right subtrees are also balanced.

4. Implementation/Code:

(a) Same Tree:

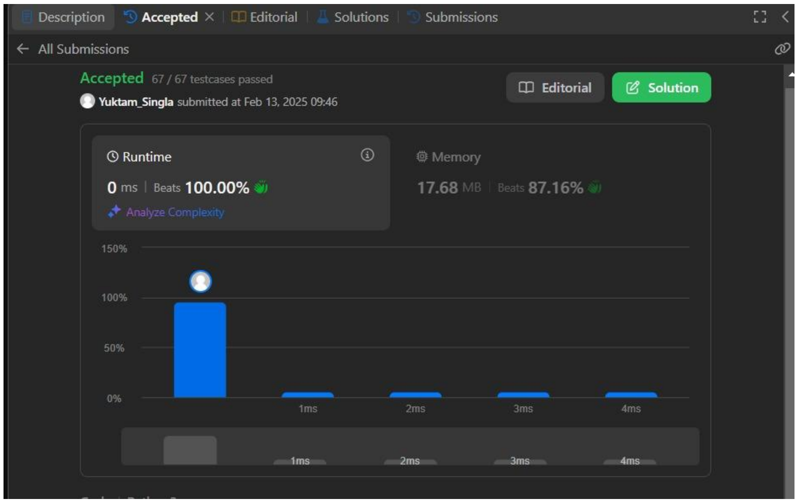
```
class Solution:
 def isSameTree(self, p: TreeNode | None, q: TreeNode | None) -> bool:
  if not p or not q:
   return p == q
  return (p.val == q.val and
      self.isSameTree(p.left, q.left) and
      self.isSameTree(p.right, q.right))
```

(b) Balanced Binary Tree:

```
class Solution:

 def isBalanced(self, root: TreeNode | None) -> bool:

  if not root:

   return True

  def maxDepth(root: TreeNode | None) -> int:

   if not root:

    return 0

   return 1 + max(maxDepth(root.left), maxDepth(root.right))

  return (abs(maxDepth(root.left) - maxDepth(root.right)) <= 1 and

      self.isBalanced(root.left) and

      self.isBalanced(root.right))
```

5. Output:

(a) Same Tree:



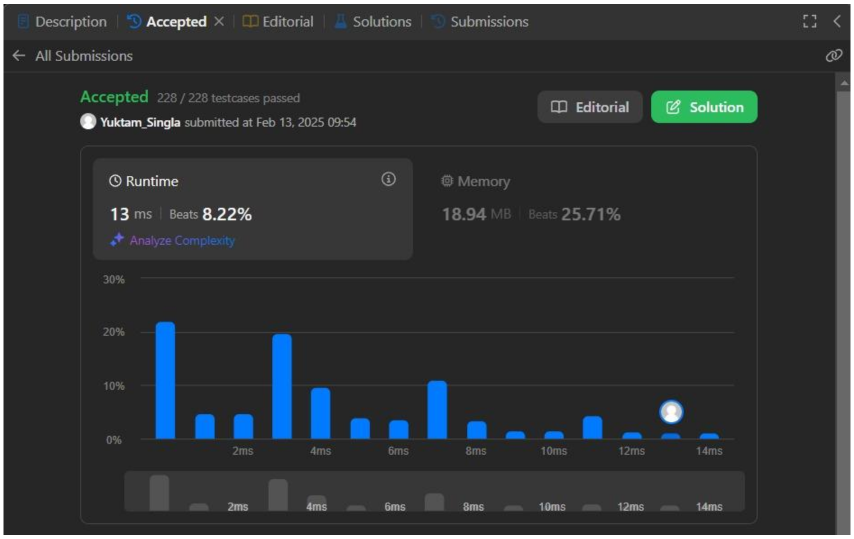
Accepted 67 / 67 testcases passed

Yuktam_Singla submitted at Feb 13, 2025 09:46

Runtime
0 ms | Beats 100.00%

Memory
17.68 MB | Beats 87.16%

(b) Balanced Binary Tree:



Accepted 228 / 228 testcases passed

Yuktam_Singla submitted at Feb 13, 2025 09:54

Runtime
13 ms | Beats 8.22%

Memory
18.94 MB | Beats 25.71%

6. Learning Outcomes:

- Understand tree traversal (recursive comparison of nodes).
- Compare node values and recursively verify left and right subtrees.
- Understand tree height calculation using recursion.
- Learn the definition of a height-balanced tree .
- Apply recursive depth-first traversal to check balance.