Problem of the Week – ***Count Unival Subtrees***

the question asked by **Google**

Problem Description:

Scenario:
In many systems, especially in distributed trees or replicated data structures, it's important to find substructures that are uniform. A **unival tree** (short for *universal value tree*) is a **subtree** where **all the nodes have the same value**.

You are given the **root of a binary tree**, and your task is to **count the number of unival subtrees** present in the tree.

A single node is trivially considered a unival subtree.

Input Format:

You will be given the root of a binary tree. Each node contains:

- An integer value
- Left and right children

For coding practice, you may build the tree manually or from helper functions.

Output Format:

- Print a single integer: the number of unival subtrees.

Constraints:

- Number of nodes ≤ 1000
- Node values can be any integer (positive or negative)

Example Tree:

```
     0
    / \
   1   0
      / \
     1   0
    / \
   1   1
```

Example Output:

```
5
```

Explanation:

The unival subtrees are:

1. The left leaf with value `1`
2. The rightmost leaf with value `0`
3. The two `1` leaves under the left of right subtree
4. The subtree rooted at that `1` node (both children are also `1`)

Hence, **5 unival subtrees** total.

Approach Hint:

Use **post-order traversal** to:

- Recursively check left and right subtrees
- Determine if current node forms a unival subtree:
    - Left and right subtrees are unival
    - Node's value matches children's (if they exist)

Maintain a counter to keep track of valid unival subtrees.

Expected Time Complexity:

- O(N), where N is the number of nodes in the tree

Practice Links:

- 🔗 GeeksforGeeks – Count Unival Subtrees
- 🔗 Leetcode – Count Univalue Subtrees (Problem 250)

Video Explanation:

- 🎥 YouTube – Count Univalue Subtrees (DFS)

Sample Starter Code (Python):

```
class Node:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def count_unival_subtrees(root):
    count = [0]

    def is_unival(node):
        if not node:
            return True
        left = is_unival(node.left)
        right = is_unival(node.right)

        if not left or not right:
            return False

        if node.left and node.left.val != node.val:
            return False
        if node.right and node.right.val != node.val:
            return False

        count[0] += 1
        return True

    is_unival(root)
    return count[0]

# Example Tree
root = Node(0,
            Node(1),
            Node(0,
                 Node(1, Node(1), Node(1)),
                 Node(0)))

print(count_unival_subtrees(root))
```