2265. Count Nodes Equal to Average of Subtree

My Submissions (/contest/weekly-contest-292/problems/count-nodes-equal-to-average-of-subtree/submissions/)

Back to Contest (/contest/weekly-contest-292/)

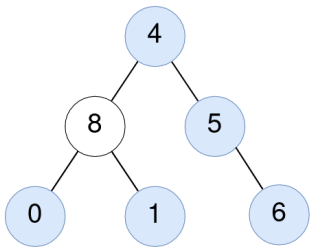
Given the `root` of a binary tree, return *the number of nodes where the value of the node is equal to the **average** of the values in its **subtree***.

User Accepted:	4687
User Tried:	4909
Total Accepted:	5008
Total Submissions:	6046
Difficulty:	Medium

Note:

- The **average** of `n` elements is the **sum** of the `n` elements divided by `n` and **rounded down** to the nearest integer.
- A **subtree** of `root` is a tree consisting of `root` and all of its descendants.

Example 1:




```
Input: root = [4,8,5,0,1,null,6]
Output: 5
Explanation:
For the node with value 4: The average of its subtree is (4 + 8 + 5 + 0 + 1 + 6) / 6 = 24 / 6 = 4.
For the node with value 5: The average of its subtree is (5 + 6) / 2 = 11 / 2 = 5.
For the node with value 0: The average of its subtree is 0 / 1 = 0.
For the node with value 1: The average of its subtree is 1 / 1 = 1.
For the node with value 6: The average of its subtree is 6 / 1 = 6.
```

Example 2:


```
Input: root = [1]
Output: 1
Explanation: For the node with value 1: The average of its subtree is 1 / 1 = 1.
```

Constraints:

- The number of nodes in the tree is in the range `[1, 1000]`.
- `0 <= Node.val <= 1000`

Discuss (https://leetcode.com/problems/count-nodes-equal-to-average-of-subtree/discuss)

Go

```
/**
 * Definition for a binary tree node.
 * type TreeNode struct {
 *     Val int
 *     Left *TreeNode
 *     Right *TreeNode
 * }
 */
func averageOfSubtree(root *TreeNode) int {
    var averageOfSubtreeSum func(node *TreeNode) (sum, nnode, found int)
    averageOfSubtreeSum = func(node *TreeNode) (sum, nnode, found int) {
        if node == nil {
            return 0, 0, 0
        } else {
```

```
            s, nn, f := averageOfSubtreeSum(node.Left)
            sum, nnode, found = sum+s, nnode+nn, found+f
            s, nn, f = averageOfSubtreeSum(node.Right)
            sum, nnode, found = sum+s, nnode+nn, found+f
        }
        sum, nnode = sum+node.Val, nnode+1
        if sum/nnode == node.Val {
            found++
        }
        return
    }
    _, _, result := averageOfSubtreeSum(root)
    return result
}
```

☐ **Custom Testcase** Use Example Testcases

Run Submit