**2265. Count Nodes Equal to Average of Subtree: -**

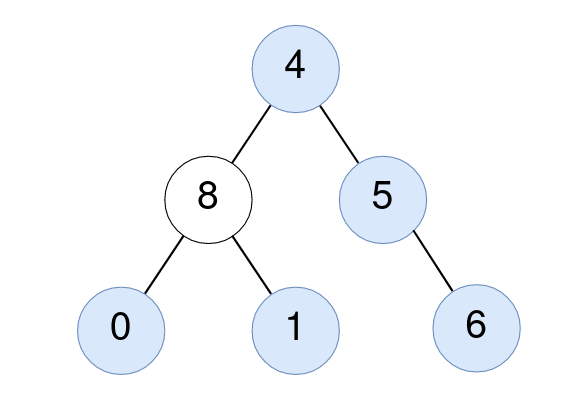
Medium Accepted: 79.7K Submissions: 92.2K Acceptance Rate: 86.4%

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***.

**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

**Code: -**

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

    int ans;

    pair<long long,long long> helper(TreeNode \*root){

        // base case

        if(!root)

            return {0,0};

        // recursive case

        pair<long long, long long> leftsubtree = helper(root->left);

        long long leftsum = leftsubtree.first;

        long long leftcount = leftsubtree.second;

        pair<long long, long long> rightsubtree = helper(root->right);

        long long rightsum = rightsubtree.first;

        long long rightcount = rightsubtree.second;

        // answer calculation

        long long totalsum = leftsum + root->val + rightsum;

        long long totalcount = leftcount + 1 + rightcount;

        if(totalsum / totalcount == root->val)

            ++ans;

        // return from current state

        return {totalsum, totalcount};

    }

    int averageOfSubtree(TreeNode\* root) {

        ans = 0;

        helper(root);

        return ans;

    }

};

**T.C: - O(N)**

**S.C: - O(1)**

**Excluding recursive call stack**