

Problem 1120: Maximum Average Subtree

Problem Information

Difficulty: Medium

Acceptance Rate: 66.87%

Paid Only: Yes

Tags: Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, return the maximum **average** value of a **subtree** of that tree. Answers within 10^{-5} of the actual answer will be accepted.

A **subtree** of a tree is any node of that tree plus all its descendants.

The **average** value of a tree is the sum of its values, divided by the number of nodes.

Example 1:



Input: `root = [5,6,1]` **Output:** `6.00000` **Explanation:** For the node with value = 5 we have an average of $(5 + 6 + 1) / 3 = 4$. For the node with value = 6 we have an average of $6 / 1 = 6$. For the node with value = 1 we have an average of $1 / 1 = 1$. So the answer is 6 which is the maximum.

Example 2:

Input: `root = [0,null,1]` **Output:** `1.00000`

Constraints:

* The number of nodes in the tree is in the range $[1, 10^4]$. * $0 \leq \text{Node.val} \leq 10^5$

Code Snippets

C++:

```
/**
 * 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:
    double maximumAverageSubtree(TreeNode* root) {

    }
};
```

Java:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *   int val;
 *   TreeNode left;
 *   TreeNode right;
 *   TreeNode() {}
 *   TreeNode(int val) { this.val = val; }
 *   TreeNode(int val, TreeNode left, TreeNode right) {
 *     this.val = val;
 *     this.left = left;
 *     this.right = right;
 *   }
 * }
 */
class Solution {
    public double maximumAverageSubtree(TreeNode root) {
```

```
}  
}
```

Python3:

```
# Definition for a binary tree node.  
# class TreeNode:  
#     def __init__(self, val=0, left=None, right=None):  
#         self.val = val  
#         self.left = left  
#         self.right = right  
class Solution:  
    def maximumAverageSubtree(self, root: Optional[TreeNode]) -> float:
```