Description | Solution | Discuss (383) | Submissions

1120. Maximum Average Subtree

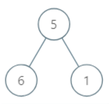
Medium 640 23 Add to List Share

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 <= Node.val <= 10^5`

Accepted 48,794 | Submissions 75,322

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Amazon 8

Related Topics

Tree Depth-First Search Binary Tree

Similar Questions

Count Nodes Equal to Sum of Descendants Medium

Hide Hint 1

Can you find the sum of values and the number of nodes for every sub-tree ?

Hide Hint 2

Can you find the sum of values and the number of nodes for a sub-tree given the sum of values and the number of nodes of it's left and right sub-trees ?

Hide Hint 3

Use depth first search to recursively find the solution for the children of a node then use their solutions to compute the current node's solution.

Java | Autocomplete

```java
class Solution {
    // for each node in the tree, we will maintain three values
    class State {
        // count of nodes in the subtree
        int nodeCount;

        // sum of values in the subtree
        int valueSum;

        // max average found in the subtree
        double maxAverage;

        State(int nodes, int sum, double maxAverage) {
            this.nodeCount = nodes;
            this.valueSum = sum;
            this.maxAverage = maxAverage;
        }
    }

    public double maximumAverageSubtree(TreeNode root) {
        return maxAverage(root).maxAverage;
    }

    State maxAverage(TreeNode root) {
        if (root == null) {
            return new State(0, 0, 0);
        }

        // postorder traversal, solve for both child nodes first.
        State left = maxAverage(root.left);
        State right = maxAverage(root.right);

        // now find nodeCount, valueSum and maxAverage for current node `root`
        int nodeCount = left.nodeCount + right.nodeCount + 1;
        int sum = left.valueSum + right.valueSum + root.val;
        double maxAverage = Math.max(
                (1.0 * (sum)) / nodeCount, // average for current node
                Math.max(right.maxAverage, left.maxAverage) // max average from child nodes
        );

        return new State(nodeCount, sum, maxAverage);
    }
}
```