637 Average of Levels in Binary Tree

2018年4月20日 19:43

Given a non-empty binary tree, return the average value of the nodes on each level in the form of an array.

Example 1:

Input:

```
    3
   / \
  9  20
    /  \
   15   7
```

Output: [3, 14.5, 11]

Explanation:

The average value of nodes on level 0 is 3, on level 1 is 14.5, and on level 2 is 11. Hence return [3, 14.5, 11].

Note:

1. The range of node's value is in the range of 32-bit signed integer.

来自 <https://leetcode.com/problems/average-of-levels-in-binary-tree/description/>

给定一个非空二叉树, 返回一个由每层节点平均值组成的数组.

注意:

1. 节点值的范围在32位有符号整数范围内。

Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution1:
    def averageOfLevels(self, root):
        """
        :type root: TreeNode
        :rtype: List[float]
        """
        from statistics import mean
        return list(map(mean, self.levelOrder(root)))

    def levelOrder(self, root):
        levels = []
        level = [root]
        while any(level):
            levels.append([node.val for node in level])
            level = [kid for node in level for kid in (node.left, node.right) if kid]
        return levels

class Solution2:
    def averageOfLevels(self, root):
        """
        :type root: TreeNode
        :rtype: List[float]
        """
        averages = []
        level = [root]
        while level:
            averages.append(sum(node.val for node in level) / len(level))
            level = [kid for node in level for kid in (node.left, node.right) if kid]
```

```
        return averages
```

Solution for C++:

```
class Solution1 {
public:
    vector<double> averageOfLevels(TreeNode* root) {
        vector<double> res;
        queue<TreeNode*> que;
        que.push(root);
        while (!que.empty()) {
            long tmp = 0;
            int s = que.size();
            for (int i = 0; i < s; i++) {
                TreeNode* t = que.front();
                que.pop();
                if (t->left)
                    que.push(t->left);
                if (t->right)
                    que.push(t->right);
                tmp += t->val;
            }
            res.push_back((double)tmp/s);
        }
        return res;
    }
};
```