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

Given the root of a binary tree, return the average value of the nodes on each level in the form of an array. Answers within 10^-5 of the actual answer will be accepted.

**Example 1:**

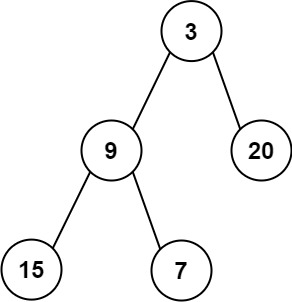


**Input:** root = [3,9,20,null,null,15,7]

**Output:** [3.00000,14.50000,11.00000]

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].

**Example 2:**



**Input:** root = [3,9,20,15,7]

**Output:** [3.00000,14.50000,11.00000]

**Constraints:**

The number of nodes in the tree is in the range [1, 104].

-2^31 <= Node.val <= 2^31 - 1

**Attempt 1: 2023-03-10**

**Solution 1: BFS + Level Order Traversal (10 min)**

/\*\*

 \* 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 List<Double> averageOfLevels(TreeNode root) {

        List<Double> result = new ArrayList<>();

        Queue<TreeNode> q = new LinkedList<>();

        q.offer(root);

        while(!q.isEmpty()) {

            int size = q.size();

            double sum = 0.0;

            for(int i = 0; i < size; i++) {

                TreeNode node = q.poll();

                sum += node.val;

                if(node.left != null) {

                    q.offer(node.left);

                }

                if(node.right != null) {

                    q.offer(node.right);

                }

            }

            result.add(sum / size);

        }

        return result;

    }

}

Time Complexity: O(N)

The time complexity of the given code is O(N), where N is the number of nodes in the binary tree.

This is because the algorithm uses a queue to traverse each node exactly once. During the traversal,

each node's value is accessed and added to a sum, and its children are potentially added to the queue.

Space Complexity: O(W)

The space complexity of the code can be considered as O(W), where W is the maximum width of the tree

or the maximum number of nodes at any level of the tree. This occurs because the queue stores a level

of the tree at most, which, in the worst case, can be all the leaf nodes of a full binary tree at the

last level.