



Level Averages in a Binary Tree (easy)

We'll cover the following



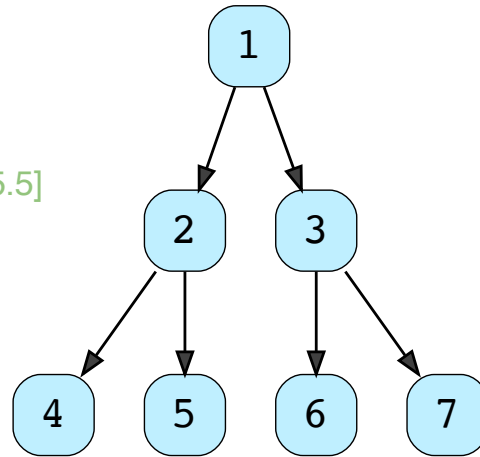
- Problem Statement
- Try it yourself
- Solution
- Code
 - Time complexity
 - Space complexity
- Similar Problems

Problem Statement#

Given a binary tree, populate an array to represent the **averages of all of its levels**.

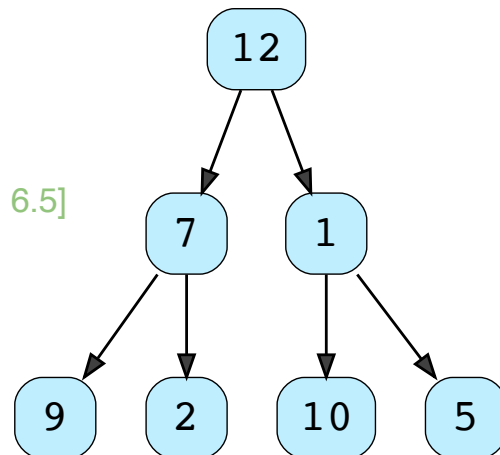
Example 1:

Level Averages: [1, 2.5, 5.5]







Example 2:

Level Averages: [12.0, 4.0, 6.5]



Try it yourself#

Try solving this question here:

 Java	 Python3	 JS	 C++
--	---	--	---

📄 ⬇

```
import java.util.*;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
};

class LevelAverage {
    public static List<Double> findLevelAverages(TreeNode root) {
        List<Double> result = new ArrayList<>();
        // TODO: Write your code here
        return result;
    }

    public static void main(String[] args) {
        TreeNode root = new TreeNode(12);
        root.left = new TreeNode(7);
        root.right = new TreeNode(1);
        root.left.left = new TreeNode(9);
        root.left.right = new TreeNode(2);
        root.right.left = new TreeNode(10);
        root.right.right = new TreeNode(5);
        List<Double> result = LevelAverage.findLevelAverages(root);
        System.out.print("Level averages are: " + result);
    }
}
```

Run

Save

Reset





⌕

Solution#

This problem follows the Binary Tree Level Order Traversal (<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5726607939469312/>) pattern. We can follow the same **BFS** approach. The only difference will be that instead of keeping track of all nodes of a level, we will only track the running sum of the values of all nodes in each level. In the end, we will append the average of the current level to the result array.

Code#

Here is what our algorithm will look like; only the highlighted lines have changed:

 Java	 Python3	 C++	 JS
--	---	---	--

```
import java.util.*;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
};

class LevelAverage {
    public static List<Double> findLevelAverages(TreeNode root) {
        List<Double> result = new ArrayList<>();
        if (root == null)
            return result;

        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);
        while (!queue.isEmpty()) {
            int levelSize = queue.size();
```

```
double levelSum = 0;
for (int i = 0; i < levelSize; i++) {
    TreeNode currentNode = queue.poll();
    // add the node's value to the running sum
    levelSum += currentNode.val;
    // insert the children of current node to the queue
    if (currentNode.left != null)
        queue.offer(currentNode.left);
    if (currentNode.right != null)
        queue.offer(currentNode.right);
}
// append the current level's average to the result array
result.add(levelSum / levelSize);
}

return result;
}

public static void main(String[] args) {
    TreeNode root = new TreeNode(12);
    root.left = new TreeNode(7);
    root.right = new TreeNode(1);
    root.left.left = new TreeNode(9);
    root.left.right = new TreeNode(2);
    root.right.left = new TreeNode(10);
    root.right.right = new TreeNode(5);
    List<Double> result = LevelAverage.findLevelAverages(root);
    System.out.print("Level averages are: " + result);
}
}
```

RunSaveReset

Time complexity#

The time complexity of the above algorithm is $O(N)$, where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

Space complexity#

The space complexity of the above algorithm will be $O(N)$ which is required for the queue. Since we can have a maximum of $N/2$ nodes at any level (this could happen only at the lowest level), therefore we will need $O(N)$ space to store them in the queue.

Similar Problems#

Problem 1: Find the largest value on each level of a binary tree.

Solution: We will follow a similar approach, but instead of having a running sum we will track the maximum value of each level.

```
maxValue = max(maxValue, currentNode.val)
```

Interviewing soon? We've partnered with Hired so that companies apply to [utm_source=educative&utm_medium=lesson&utm_location=US&utm_can](https://www.educative.io/courses/grokking-the-coding-interview/YQWkA2l67GW)



← Back

Next →

Zigzag Traversal (medium)

Minimum Depth of a Binary Tree (easy)



Mark as Completed



Report an Issue

