Grokking the Coding Interview: Patterns for Coding Questions

31% completed

Search Course

Pattern: Tree Breadth First Search

- Introduction
- Binary Tree Level Order Traversal (easy)
- Reverse Level Order Traversal (easy)
- Zigzag Traversal (medium)
- Level Averages in a Binary Tree (easy)
- Minimum Depth of a Binary Tree (easy)
- Level Order Successor (easy)
- Connect Level Order Siblings (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Tree Depth First Search

- Introduction
- Binary Tree Path Sum (easy)
- All Paths for a Sum (medium)
- Sum of Path Numbers (medium)
- Path With Given Sequence (medium)
- Count Paths for a Sum (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Two Heaps

- Introduction
- Find the Median of a Number Stream (medium)
- Sliding Window Median (hard)
- Maximize Capital (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

Pattern: Subsets

- Introduction
- Subsets (easy)
- Subsets With Duplicates (easy)
- Permutations (medium)
- String Permutations by changing case (medium)
- Balanced Parentheses (hard)
- Unique Generalized Abbreviations (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

Level Averages in a Binary Tree (easy)

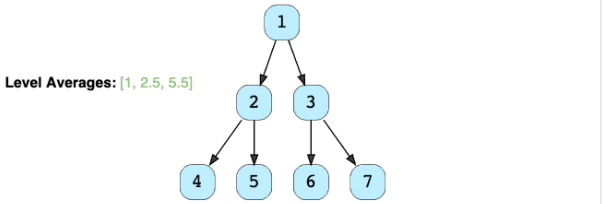
We'll cover the following

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

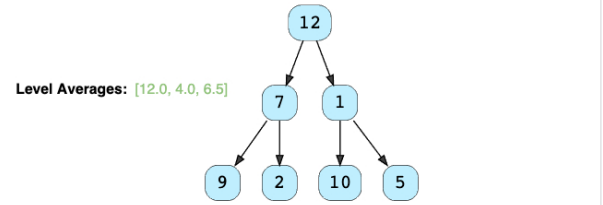
Problem Statement

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

Example 1:



Example 2:



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
class TreeNode {

  constructor(value) {
    this.value = value;
    this.left = null;
    this.right = null;
  }
};

const find_level_averages = function(root) {
  result = [];
  // TODO: Write your code here
  return result;
};


var 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)

console.log(`Level averages are: ${find_level_averages(root)}`)

```

RUN SAVE RESET

Problem Challenge 2
Solution Review: Problem Challenge 2
Problem Challenge 3
Solution Review: Problem Challenge 3

Pattern: Modified Binary Search
Introduction
Order-agnostic Binary Search (easy)
Ceiling of a Number (medium)
Next Letter (medium)
Number Range (medium)
Search in a Sorted Infinite Array (medium)
Minimum Difference Element (medium)
Bitonic Array Maximum (easy)
Problem Challenge 1
Solution Review: Problem Challenge 1
Problem Challenge 2
Solution Review: Problem Challenge 2
Problem Challenge 3
Solution Review: Problem Challenge 3

Pattern: Bitwise XOR
Introduction
Single Number (easy)
Two Single Numbers (medium)
Complement of Base 10 Number (medium)
Problem Challenge 1
Solution Review: Problem Challenge 1

Pattern: Top 'K' Elements
Introduction
Top 'K' Numbers (easy)
Kth Smallest Number (easy)
'K' Closest Points to the Origin (easy)
Connect Ropes (easy)
Top 'K' Frequent Numbers (medium)
Frequency Sort (medium)
Kth Largest Number in a Stream (medium)
'K' Closest Numbers (medium)
Maximum Distinct Elements (medium)
Sum of Elements (medium)
Rearrange String (hard)
Problem Challenge 1
Solution Review: Problem Challenge 1
Problem Challenge 2
Solution Review: Problem Challenge 2
Problem Challenge 3
Solution Review: Problem Challenge 3

Pattern: K-way merge
Introduction
Merge K Sorted Lists (medium)
Kth Smallest Number in M Sorted Lists (Medium)

Solution

This problem follows the Binary Tree Level Order Traversal 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

```
const Deque = require('./collections/deque'); //http://www.collectionsjs.com


class TreeNode {
  constructor(val) {
    this.val = val;
    this.left = null;
    this.right = null;
  }
}

function find_level_averages(root) {
  result = [];
  if (root === null) {
    return result;
  }

  const queue = new Deque();
  queue.push(root);
  while (queue.length > 0) {
    let levelSize = queue.length,
      levelSum = 0.0;
    for (i = 0; i < levelSize; i++) {
      currentNode = queue.shift();
      // 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) {
```

RUN | SAVE | RESET

Close

Output 5.169s

```
Level averages are: 12,4,6.5
```

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)
```


← Back
Zigzag Traversal (medium)

Next →
Minimum Depth of a Binary Tree (easy)

Create

Kth Smallest Number in a Sorted