

Grokking the Coding Interview: Patterns for Coding Questions

34% completed

(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

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 Greater Element

Sum of Path Numbers (medium)

We'll cover the following

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

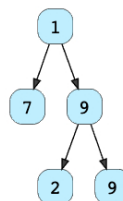
Problem Statement

Given a binary tree where each node can only have a digit (0-9) value, each root-to-leaf path will represent a number. Find the total sum of all the numbers represented by all paths.

Example 1:

Output: 408

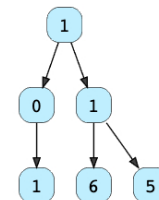
Explanation: The sum of all path numbers: 17 + 192 + 199



Example 2:

Output: 332

Explanation: The sum of all path numbers: 101 + 116 + 115



Try it yourself

Try solving this question here:

Java

Python3

JS

C++

```
1 class TreeNode {
2
3   constructor(value) {
4     this.value = value;
5     this.left = null;
6     this.right = null;
7   }
8 };
9
10
11 const find_sum_of_path_numbers = function(root) {
12   // TODO: Write your code here
13   return -1;
14 };
15
16
17 var root = new TreeNode(1)
18 root.left = new TreeNode(0)
19 root.right = new TreeNode(1)
20 root.left.left = new TreeNode(1)
21 root.right.left = new TreeNode(6)
22 root.right.right = new TreeNode(5)
23 console.log('Total Sum of Path Numbers: ' + find_sum_of_path_numbers(root))
24
25
```

RUN

SAVE

RESET



Solution

This problem follows the [Binary Tree Path Sum](#) pattern. We can follow the same DFS approach. The additional thing we need to do is to keep track of the number representing the current path.

How do we calculate the path number for a node? Taking the first example mentioned above, say we are at

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

MW

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

node '7'. As we know, the path number for this node is '17', which was calculated by: $1 * 10 + 7 \Rightarrow 17$. We will follow the same approach to calculate the path number of each node.

Code

Here is what our algorithm will look like:

JavaPython3C++JS

```
13
14
15 function find_root_to_leaf_path_numbers(currentNode, pathSum) {
16     if (currentNode === null) {
17         return 0;
18     }
19
20     // calculate the path number of the current node
21     pathSum = 10 * pathSum + currentNode.val;
22
23     // if the current node is a leaf, return the current path sum
24     if (currentNode.left === null && currentNode.right === null) {
25         return pathSum;
26     }
27
28     // traverse the left and the right sub-tree
29     return find_root_to_leaf_path_numbers(currentNode.left, pathSum) +
30           find_root_to_leaf_path_numbers(currentNode.right, pathSum);
31 }
32
33
34 const root = new TreeNode(1);
35 root.left = new TreeNode(0);
36 root.right = new TreeNode(1);
37 root.left.left = new TreeNode(1);
38 root.right.left = new TreeNode(6);
39 root.right.right = new TreeNode(5);
40 console.log(`Total Sum of Path Numbers: ${find_sum_of_path_numbers(root)}`);
```

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 $O(N)$ in the worst case. This space will be used to store the recursion stack. The worst case will happen when the given tree is a linked list (i.e., every node has only one child).

Interviewing soon? We've partnered with [Hired](#) so that companies apply to you, instead of the other way around. [See how](#)

← Back

Next →

MARK AS COMPLETED

Path With Given Sequence (medium)

All Paths for a Sum (medium)