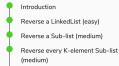


Grokking the Coding Interview: Patterns for **Coding Questions**



LinkedList



- Problem Challenge 1 Solution Review: Problem Challenge 1
- Problem Challenge 2 Solution Review: Problem Challenge 2

Pattern: Tree Breadth First Search



Solution Review: Problem Challenge 1 Problem Challenge 2

Solution Review: Problem Challenge 2

Pattern: Tree Depth First Search



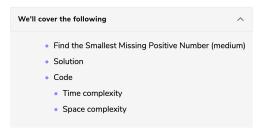
Problem Challenge 1

앟

(

Solution Review: Problem Challenge 1 Problem Challenge 2 Solution Review: Problem

Solution Review: Problem Challenge 2



Find the Smallest Missing Positive Number (medium)

Given an unsorted array containing numbers, find the smallest missing positive number in it.

Example 1:

```
Input: [-3, 1, 5, 4, 2]
Explanation: The smallest missing positive number is '3'
```

Example 2:

```
Input: [3, -2, 0, 1, 2]
```

Example 3:

This problem follows the Cyclic Sort pattern and shares similarities with Find the Missing Number with one big difference. In this problem, the numbers are not bound by any range so we can have any number in the

However, we will follow a similar approach though as discussed in Find the Missing Number to place the numbers on their correct indices and ignore all numbers that are out of the range of the array (i.e., all negative numbers and all numbers greater than or equal to the length of the array). Once we are done with the cyclic sort we will iterate the array and the first index that does not have the correct number will be the smallest missing positive number!

Code

Here is what our algorithm will look like:

```
Python3 G C++
         ction find_first_missing_positive(nums) {
        n = nums.length;
       while (i < n) {
   j = nums[i] - 1;</pre>
          } else {
        for (i = 0; i < n; i++) {
       return nums.length + 1;
22 console.log(find_first_missing_positive([-3, 1, 5, 4, 2]));
23 console.log(find_first_missing_positive([3, -2, 0, 1, 2]));
    console.log(find_first_missing_positive([3, 2, 5, 1]));
                                                                                                                                    03
                                                                                                                               Close
Output
```



Time complexity $^{\circ}$ The time complexity of the above algorithm is O(n).

Space complexity $^{\circ}$ The algorithm runs in constant space O(1).

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you \times applying to them. See how \odot Problem Challenge 2

Problem Challenge 2

Problem Challenge 3

Ask a Question