

Grokking the Coding Interview: Patterns for Coding Questions

52% completed



- Find the Duplicate Number (easy)
- Find all Duplicate Numbers (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: In-place Reversal of a LinkedList

- Introduction
- Reverse a LinkedList (easy)
- Reverse a Sub-list (medium)
- Reverse every K-element Sub-list (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

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

Solution Review: Problem Challenge 3

We'll cover the following

- Find the First K Missing Positive Numbers (hard)
- Solution
- Code
 - Time complexity
 - Space complexity

Find the First K Missing Positive Numbers (hard)

Given an unsorted array containing numbers and a number 'k', find the first 'k' missing positive numbers in the array.

Example 1:

```
Input: [3, -1, 4, 5, 5], k=3
Output: [1, 2, 6]
Explanation: The smallest missing positive numbers are 1, 2 and 6.
```

Example 2:

```
Input: [2, 3, 4], k=3
Output: [1, 5, 6]
Explanation: The smallest missing positive numbers are 1, 5 and 6.
```

Example 3:

```
Input: [-2, -3, 4], k=2
Output: [1, 2]
Explanation: The smallest missing positive numbers are 1 and 2.
```

Solution

This problem follows the **Cyclic Sort** pattern and shares similarities with [Find the Smallest Missing Positive Number](#). The only difference is that, in this problem, we need to find the first 'k' missing numbers compared to only the first missing number.

We will follow a similar approach as discussed in [Find the Smallest Missing Positive Number](#) to place the numbers on their correct indices and ignore all numbers that are out of the range of the array. Once we are done with the cyclic sort we will iterate through the array to find indices that do not have the correct numbers.

If we are not able to find 'k' missing numbers from the array, we need to add additional numbers to the output array. To find these additional numbers we will use the length of the array. For example, if the length of the array is 4, the next missing numbers will be 4, 5, 6 and so on. One tricky aspect is that any of these additional numbers could be part of the array. Remember, while sorting, we ignored all numbers that are greater than or equal to the length of the array. So all indices that have the missing numbers could possibly have these additional numbers. To handle this, we must keep track of all numbers from those indices that have missing numbers. Let's understand this with an example:

```
nums: [2, 1, 3, 6, 5], k =2
```

After the cyclic sort our array will look like:

```
nums: [1, 2, 3, 6, 5]
```

From the sorted array we can see that the first missing number is '4' (as we have '6' on the fourth index) but to find the second missing number we need to remember that the array does contain '6'. Hence, the next missing number is '7'.

Code

Here is what our algorithm will look like:

```
Java Python3 C++ JS
2 const n = nums.length;
3 let i = 0;
4 while (i < n) {
5     const j = nums[i] - 1;
6     if (nums[i] > 0 && nums[i] <= n && nums[i] !== nums[j]) {
7         [nums[i], nums[j]] = [nums[j], nums[i]]; // swap
8     } else {
9         i += 1;
10    }
11 }
```

Challenge 4

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 Letter (medium)

Number Range (medium)

Search in a Sorted Infinite Array (medium)

Minimum Difference Element (medium)

Bitonic Array Maximum (easy)

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

```
12 missingNumbers = [];  
13 const extraNumbers = new Set();  
14 for (i = 0; i < n; i++) {  
15   if (missingNumbers.length < k) {  
16     if (nums[i] !== i + 1) {  
17       missingNumbers.push(i + 1);  
18       extraNumbers.add(nums[i]);  
19     }  
20   }  
21 }  
22  
23 // add the remaining missing numbers  
24 i = 1;  
25 while (missingNumbers.length < k) {  
26   const candidateNumber = i + n;  
27   // ignore if the array contains the candidate number  
28   if (!extraNumbers.has(candidateNumber)) {  
29     missingNumbers.push(candidateNumber);
```

RUN

SAVE

RESET

Close

Output1.746s

[1, 2, 6]
[1, 5, 6]
[1, 2]

Time complexity

The time complexity of the above algorithm is $O(n + k)$, as the last two `for` loops will run for $O(n)$ and $O(k)$ times respectively.

Space complexity

The algorithm needs $O(k)$ space to store the `extraNumbers`.

Interviewing soon? We've partnered with [Hired](#) so that companies apply to you instead of you applying to them. [See how](#)

← Back

Next →

Problem Challenge 3

Introduction

Mark as Completed

Report an Issue

Ask a Question