

Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

32% completed

Search Course

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Problem Challenge 3

Solution Review: Problem Challenge 3

# Solution Review: Problem Challenge 1

We'll cover the following

- Find the Corrupt Pair (easy)
- Solution
- Code
- Time complexity
- Space complexity

## Find the Corrupt Pair (easy)#

We are given an unsorted array containing ‘n’ numbers taken from the range 1 to ‘n’. The array originally contained all the numbers from 1 to ‘n’, but due to a data error, one of the numbers got duplicated which also resulted in one number going missing. Find both these numbers.

Example 1:

```
Input: [3, 1, 2, 5, 2]
Output: [2, 4]
Explanation: '2' is duplicated and '4' is missing.
```

Example 2:

```
Input: [3, 1, 2, 3, 6, 4]
Output: [3, 5]
Explanation: '3' is duplicated and '5' is missing.
```

## Solution#

This problem follows the **Cyclic Sort** pattern and shares similarities with [Find all Duplicate Numbers](#). Following a similar approach, we will place each number at its correct index. Once we are done with the cyclic sort, we will iterate through the array to find the number that is not at the correct index. Since only one number got corrupted, the number at the wrong index is the duplicated number and the index itself represents the missing number.

## Code#

Here is what our algorithm will look like:

JavaPython3C++JS

```
5 while (i < nums.length) {
6     if (nums[i] != nums[nums[i] - 1])
7         swap(nums, i, nums[i] - 1);
8     else
9         i++;
10 }
11
12 for (i = 0; i < nums.length; i++)
13     if (nums[i] != i + 1)
14         return new int[] { nums[i], i + 1 };
15
16 return new int[] { -1, -1 };
17 }
18
19 private static void swap(int[] arr, int i, int j) {
20     int temp = arr[i];
21     arr[i] = arr[j];
22     arr[j] = temp;
23 }
24
25 public static void main(String[] args) {
26     int[] nums = FindCorruptNums.findNumbers(new int[] { 3, 1, 2, 5, 2 });
27     System.out.println(nums[0] + ", " + nums[1]);
28     nums = FindCorruptNums.findNumbers(new int[] { 3, 1, 2, 3, 6, 4 });
29     System.out.println(nums[0] + ", " + nums[1]);
30 }
31 }
32
```

RunSaveReset

## Time complexity#

The time complexity of the above algorithm is  $O(n)$ .

## Space complexity#

The algorithm runs in constant space  $O(1)$ .

Want to work at Google, Facebook, or Amazon? Get hired faster with anonymous mock interviews conducted by senior engineers from those companies. Detailed feedback helps you prep. [See how](#)