educative

Solution Review: Problem Challenge 1

We'll cover the following

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:

Java Python3 C++ JS

```java
class FindCorruptNums {

  public static int[] findNumbers(int[] nums) {
    int i = 0;
    while (i < nums.length) {
      if (nums[i] != nums[nums[i] - 1])
        swap(nums, i, nums[i] - 1);
      else
        i++;
    }

    for (i = 0; i < nums.length; i++)
      if (nums[i] != i + 1)
        return new int[] { nums[i], i + 1 };

    return new int[] { -1, -1 };
  }

  private static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
  }

  public static void main(String[] args) {
    int[] nums = FindCorruptNums.findNumbers(new int[] { 3, 1, 2, 5, 2 });
    System.out.println(nums[0] + ", " + nums[1]);
    nums = FindCorruptNums.findNumbers(new int[] { 3, 1, 2, 3, 6, 4 });
```

Run Save Reset

Time complexity

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

Space complexity

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

← Back
Problem Challenge 1

Next →
Problem Challenge 2

✓ Mark as Completed