educative(/learn)

Find all Missing Numbers (easy)

We'll cover the following ⌃

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

Problem Statement#

We are given an unsorted array containing numbers taken from the range 1 to 'n'. The array can have duplicates, which means some numbers will be missing. Find all those missing numbers.

Example 1:

```
Input: [2, 3, 1, 8, 2, 3, 5, 1]
Output: 4, 6, 7
Explanation: The array should have all numbers from 1 to 8, due
to duplicates 4, 6, and 7 are missing.
```

Example 2:

```
Input: [2, 4, 1, 2]
Output: 3
```

Example 3:

```
Input: [2, 3, 2, 1]
Output: 4
```

Try it yourself#

Try solving this question here:

| Java	Python3	JS JS	C++

```java
 1  import java.util.*;
 2
 3  class AllMissingNumbers {
 4
 5    public static List<Integer> findNumbers(int[] nums) {
 6      List<Integer> missingNumbers = new ArrayList<>();
 7      // TODO: Write your code here
 8      return missingNumbers;
 9    }
10  }
11
```

Test   Save   Reset

Solution#

This problem follows the **Cyclic Sort** pattern and shares similarities with Find the Missing Number (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5319932059320320/) with one difference. In this problem, there can be many duplicates whereas in 'Find the Missing Number' there were no duplicates and the range was greater than the length of the array.

However, we will follow a similar approach though as discussed in Find the Missing Number (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5319932059320320/) to place the numbers on their correct indices. Once we are done with the cyclic sort we will iterate the array to find all indices that are missing the correct numbers.

Code#

Here is what our algorithm will look like:

☕ Java	🐍 Python3	⊕ C++	JS JS

```java
1   import java.util.*;
2
3   class AllMissingNumbers {
4
5     public static List<Integer> findNumbers(int[] nums) {
6       int i = 0;
7       while (i < nums.length) {
8         if (nums[i] != nums[nums[i] - 1])
9           swap(nums, i, nums[i] - 1);
```

```
10        else
11             i++;
12      }
13
14      List<Integer> missingNumbers = new ArrayList<>();
15      for (i = 0; i < nums.length; i++)
16        if (nums[i] != i + 1)
17          missingNumbers.add(i + 1);
18
19      return missingNumbers;
20    }
21
22    private static void swap(int[] arr, int i, int j) {
23      int temp = arr[i];
24      arr[i] = arr[j];
25      arr[j] = temp;
26    }
27
28    public static void main(String[] args) {
```

Run   Save   Reset   ⌂

Time complexity#

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

Space complexity#

Ignoring the space required for the output array, the algorithm runs in constant space $O(1)$.

Interviewing soon? We've partnered with Hired so that companies apply to

utm_source=educative&utm_medium=lesson&utm_location=US&utm_can

ⓘ

← **Back**

Next →

Find the Missing Number (easy)   Find the Duplicate Number (easy)

☑ Mark as Completed

⚠ Report an Issue

⍰ Ask a Question
(https://discuss.educative.io/tag/find-all-missing-numbers-easy__pattern-cyclic-sort__grokking-the-coding-interview-patterns-for-coding-questions?
open=true&ctag=grokking-the-coding-interview-patterns-for-coding-questions__design-gurus&aid=5668639101419520&cid=5671464854355968&pid=4893412715134976)