

# 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	C++
------	---------	----	-----

```
1 def find_missing_numbers(nums):
2     missingNumbers = []
3     # TODO: Write your code here
4     return missingNumbers
5
```

## Solution #

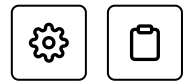
This problem follows the **Cyclic Sort** pattern and shares similarities with Find the Missing Number

(<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
<pre>1 def find_missing_numbers(nums): 2     i = 0 3     while i &lt; len(nums): 4         j = nums[i] - 1 5         if nums[i] != nums[j]: 6             nums[i], nums[j] = nums[j], nums[i] # swap 7         else: 8             i += 1 9 10    missingNumbers = [] 11 12    for i in range(len(nums)): 13        if nums[i] != i + 1: 14            missingNumbers.append(i + 1) 15 16    return missingNumbers 17 18 19 def main(): 20     print(find_missing_numbers([2, 3, 1, 8, 2, 3, 21     print(find_missing_numbers([2, 4, 1, 2])) 22     print(find_missing_numbers([2, 3, 2, 1])) 23 24 25 main() 26</pre>			
<div> </div>			

## 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  $O(1)$ .

[← 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](https://discuss.educative.io/tag/find-all-missing-numbers-easy__pattern-cyclic-sort__grokking-the-coding-interview-patterns-for-coding-questions))