Photo credit: **Unsplash** (https://unsplash.com/photos/m9LlUwkPvT8)

Coding Patterns: Cyclic Sort

4 minute read

In **Coding Patterns (https://emre.me/categories/#coding-patterns)** series, we will try to *recognize* common patterns *underlying* behind each algorithm question, using real examples from Leetcode (https://leetcode.com/).

Previous posts were about Sliding Window (https://emre.me/coding-patterns/sliding-window/), Two Pointers (https://emre.me/coding-patterns/two-pointers/), Fast & Slow Pointers (https://emre.me/coding-patterns/fast-slow-pointers/) and Merge Intervals (https://emre.me/coding-patterns/merge-intervals/) patterns and today, we will introduce Cyclic Sort (https://emre.me/coding-patterns/cyclic-sort/) pattern which is very useful to solve the problems involving arrays containing numbers in a given *range*, finding the *missing* or *duplicate* numbers.

Problem: Missing Number

LeetCode 268 - Missing Number [*easy*] (https://leetcode.com/problems/missing-number/)

Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.

Example 1:

```
Input: [3, 0, 1]
Output: 2
```

Example 2:

```
Input: [9, 6, 4, 2, 3, 5, 7, 0, 1]
Output: 8
```

Note:

Your algorithm should run in *linear* runtime complexity. Could you implement it using only constant extra space complexity?

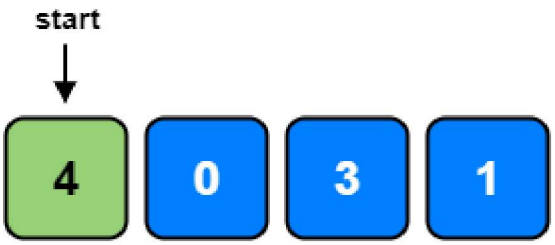
Cyclic Sort Solution

As we know, the input array contains numbers in the range of `0` to `n`. We can use this fact to devise an efficient way to sort the numbers.

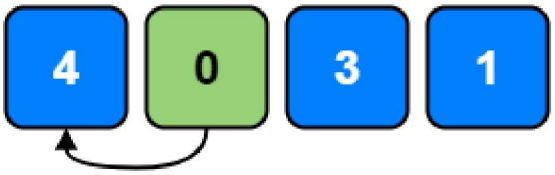
Since all numbers are unique, we can try placing each number at its correct place, for example, placing **0** at index `0`, placing **1** at index `1`, and so on.

Once we have every number in its correct place, we can iterate the array to find the index which does not have the correct number, and that index will be our missing number.

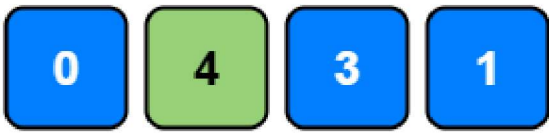
Since the array will have **n** numbers, which means array indices will range from `0` to `n-1`. Therefore, we will ignore the number `n` as we can't place it in the array, so `=> nums[i] < len(nums)`



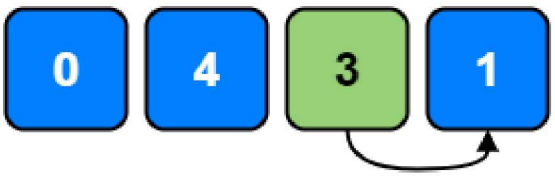
4 is number "n" and we are going to ignore it



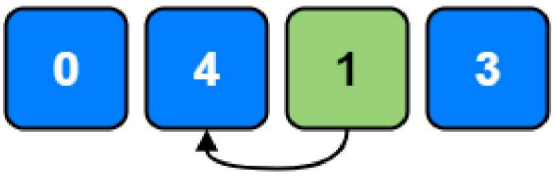
0 is not its correct place, let's swap



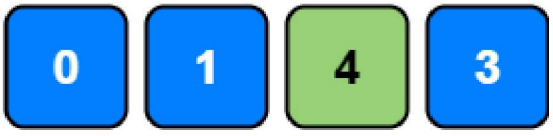
4 is number "n" and we are going to ignore it



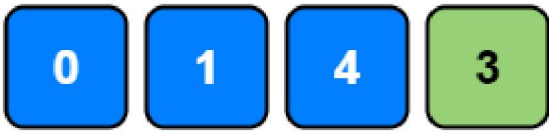
3 is not its correct place, let's swap



1 is not its correct place, let's swap



4 is number "n" and we are going to ignore it



3 is its correct place, we can now iterate the array to find the index which does not have the correct

number

```python
class Solution:
    def missingNumber(self, nums: List[int]) -> int:
        start = 0

        while start < len(nums):
            num = nums[start]
            if num < len(nums) and num != start:
                nums[start], nums[num] = nums[num], nums[start]
            else:
                start += 1

        for i in range(len(nums)):
            if nums[i] != i:
                return i

        return len(nums)
```

Time Complexity: **O(N) + O(N - 1)** which is asymptotically equivalent to **O(N)**

Space Complexity: **O(1)**, algorithm runs in constant space.

How to identify?

This approach is quite useful when dealing with numbers in a given range and asking to find the duplicates/missing ones etc.

When the problem involving arrays containing numbers in a given range, you should think about Cyclic Sort (https://emre.me/coding-patterns/cyclic-sort/) pattern.

Similar LeetCode Problems

- LeetCode 442 - Find All Duplicates in an Array [*medium*] (https://leetcode.com/problems/find-all-duplicates-in-an-array/)
- LeetCode 448 - Find All Numbers Disappeared in an Array [*easy*] (https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/)
- LeetCode 645 - Set Mismatch [*easy*] (https://leetcode.com/problems/set-mismatch/)

Tags: algorithms python

Categories: coding-patterns

Updated: October 28, 2019