0041. 缺失的第一个正数

ITCharge 大约 2 分钟

- 标签：数组、哈希表
- 难度：困难

题目链接

- 0041. 缺失的第一个正数 - 力扣

题目大意

描述：给定一个未排序的整数数组 `nums` 。

要求：找出其中没有出现的最小的正整数。

说明：

- $1 \leq nums.length \leq 5 * 10^5$。
- $-2^{31} \leq nums[i] \leq 2^{31} - 1$。
- 要求实现时间复杂度为 `O(n)` 并且只使用常数级别额外空间的解决方案。

示例：

- 示例 1：

```py
输入：nums = [1,2,0]
输出：3
```

- 示例 2：

```py
输入：nums = [3,4,-1,1]
输出：2
```

解题思路

思路 1：哈希表、原地哈希

如果使用普通的哈希表，我们只需要遍历一遍数组，将对应整数存入到哈希表中，再从 `1` 开始，依次判断对应正数是否在哈希表中即可。但是这种做法的空间复杂度为 $O(n)$，不满足常数级别的额外空间要求。

我们可以将当前数组视为哈希表。一个长度为 `n` 的数组，对应存储的元素值应该为 `[1, n + 1]` 之间，其中还包含一个缺失的元素。

1. 我们可以遍历一遍数组，将当前元素放到其对应位置上（比如元素值为 `1` 的元素放到数组第 `0` 个位置上、元素值为 `2` 的元素放到数组第 `1` 个位置上，等等）。
2. 然后再次遍历一遍数组。遇到第一个元素值不等于下标 + 1 的元素，就是答案要求的缺失的第一个正数。
3. 如果遍历完没有在数组中找到缺失的第一个正数，则缺失的第一个正数是 `n + 1` 。
4. 最后返回我们找到的缺失的第一个正数。

思路 1：代码

py

```
class Solution:
    def firstMissingPositive(self, nums: List[int]) -> int:
        size = len(nums)

        for i in range(size):
            while 1 <= nums[i] <= size and nums[i] != nums[nums[i] - 1]:
                index1 = i
                index2 = nums[i] - 1
                nums[index1], nums[index2] = nums[index2], nums[index1]

        for i in range(size):
            if nums[i] != i + 1:
                return i + 1
        return size + 1
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 为数组 `nums` 的元素个数。
- **空间复杂度**：$O(1)$。