0283. 移动零

ITCharge 大约 2 分钟

- 标签：数组、双指针
- 难度：简单

题目链接

- 0283. 移动零 - 力扣

题目大意

描述：给定一个数组 $nums$。

要求：将所有 0 移动到末尾，并保持原有的非 0 数字的相对顺序。

说明：

- 只能在原数组上进行操作。
- $1 \leq nums.length \leq 10^4$。
- $-2^{31} \leq nums[i] \leq 2^{31} - 1$。

示例：

- 示例 1：

```py
输入：nums = [0,1,0,3,12]
输出：[1,3,12,0,0]
```

- 示例 2：

```py
输入：nums = [0]
输出：[0]
```

解题思路

思路 1：冒泡排序（超时）

冒泡排序的思想，就是通过相邻元素的比较与交换，使得较大元素从前面移到后面。

我们可以借用冒泡排序的思想，将值为 0 的元素移动到数组末尾。

因为数据规模为 10^4，而冒泡排序的时间复杂度为 $O(n^2)$。所以这种做法会导致超时。

思路 1：代码

```py
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        for i in range(len(nums)):
            for j in range(len(nums) - i - 1):
                if nums[j] == 0 and nums[j + 1] != 0:
                    nums[j], num    + 1] = nums[j + 1], nums[j]
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$。
- **空间复杂度**：$O(1)$。

思路 2：快慢指针

1. 使用两个指针 $slow$，$fast$。$slow$ 指向处理好的非 0 数字数组的尾部，$fast$ 指针指向当前待处理元素。
2. 不断向右移动 $fast$ 指针，每次移动到非零数，则将左右指针对应的数交换，交换同时将 $slow$ 右移。
3. 此时，$slow$ 指针左侧均为处理好的非零数，而从 $slow$ 指针指向的位置开始， $fast$ 指针左边为止都为 0。

遍历结束之后，则所有 0 都移动到了右侧，且保持了非零数的相对位置。

思路 2：代码

```py
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        slow = 0
        fast = 0
        while fast < len(nums):
            if nums[fast] != 0:
                nums[slow], nums[fast] = nums[fast], nums[slow]
                slow += 1
            fast += 1
```

思路 2：复杂度分析

- **时间复杂度**：$O(n)$。
- **空间复杂度**：$O(1)$。