0169. 多数元素

ITCharge 大约 2 分钟

- 标签：数组、哈希表、分治、计数、排序
- 难度：简单

题目链接

- 0169. 多数元素 - 力扣

题目大意

描述：给定一个大小为 n 的数组 `nums` 。

要求：返回其中相同元素个数最多的元素。

说明：

- $n == nums.length$。
- $1 \leq n \leq 5 * 10^4$。
- $-10^9 \leq nums[i] \leq 10^9$。

示例：

- 示例 1：

```py
输入：nums = [3,2,3]
输出：3
```

- 示例 2：

```py
输入：nums = [2,2,1,1,1,2,2]
输出：2
```

解题思路

思路 1：哈希表

1. 遍历数组 `nums` 。
2. 对于当前元素 `num` ，用哈希表统计每个元素 `num` 出现的次数。
3. 再遍历一遍哈希表，找出元素个数最多的元素即可。

思路 1：代码

```py
class Solution:
    def majorityElement(self, nums: List[int]) -> int:
        numDict = dict()
        for num in nums:
            if num in numDict:
                numDict[num] += 1
            else:
                numDict[num] = 1
        max = float('-inf')
        max_index = -1
        for num in numDict:
            if numDict[num] > max:
                max = numDict[num]
                max_index = num
        return max_index
```

思路 1：复杂度分析

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

思路 2：分治算法

如果 `num` 是数组 `nums` 的众数，那么我们将 `nums` 分为两部分，则 `num` 至少是其中一部分的众数。

则我们可以用分治法来解决这个问题。具体步骤如下：

1. 将数组 `nums` 递归地将当前序列平均分成左右两个数组，直到所有子数组长度为 `1` 。
2. 长度为 1 的子数组众数肯定是数组中唯一的数，将其返回即可。
3. 将两个子数组依次向上两两合并。
    1. 如果两个子数组的众数相同，则说明合并后的数组众数为：两个子数组的众数。
    2. 如果两个子数组的众数不同，则需要比较两个众数在整个区间的众数。
4. 最后返回整个数组的众数。

思路 2：代码

```py
class Solution:
    def majorityElement(self, nums: List[int]) -> int:
        def get_mode(low, high):
            if low == high:
                return nums[low]

            mid = low + (high - low) // 2
            left_mod = get_mode(low, mid)
            right_mod = get_mode(mid + 1, high)

            if left_mod == right_mod:
                return left_mod

            left_mod_cnt, right_mod_cnt = 0, 0
            for i in range(low, high + 1):
                if nums[i] == left_mod:
                    left_mod_cnt += 1
                if nums[i] == right_mod:
                    right_mod_cnt += 1
```

```
            if left_mod_cnt > right_mod_cnt:
                return left_mod
            return right_mod

        return get_mode(0, len(nums) - 1)
```

思路 2：复杂度分析

- **时间复杂度**：$O(n \times \log n)$。
- **空间复杂度**：$O(\log n)$。