0260. 只出现一次的数字 III

ITCharge 大约 2 分钟

- 标签：位运算、数组
- 难度：中等

题目链接

- 0260. 只出现一次的数字 III - 力扣

题目大意

描述：给定一个整数数组 $nums$。$nums$ 中恰好有两个元素只出现一次，其余所有元素均出现两次。

要求：找出只出现一次的那两个元素。可以按任意顺序返回答案。要求时间复杂度是 $O(n)$，空间复杂度是 $O(1)$。

说明：

- $2 \leq nums.length \leq 3 \times 10^4$。
- $-2^{31} \leq nums[i] \leq 2^{31} - 1$。
- 除两个只出现一次的整数外，$nums$ 中的其他数字都出现两次。

示例：

- 示例 1：

```py
输入：nums = [1,2,1,3,2,5]
输出：[3,5]
解释：[5, 3] 也是有效的答案。
```

- 示例 2：

```py
输入：nums = [-1,0]
输出：[-1,0]
```

解题思路

思路 1：位运算

求解这道题之前，我们先来看看如何求解「一个数组中除了某个元素只出现一次以外，其余每个元素均出现两次。」即「136. 只出现一次的数字 」问题。

我们可以对所有数不断进行异或操作，最终可得到单次出现的元素。

下面我们再来看这道题。

如果数组中有两个数字只出现一次，其余每个元素均出现两次。那么经过全部异或运算。我们可以得到只出现一次的两个数字的异或结果。

根据异或结果的性质，异或运算中如果某一位上为 1，则说明异或的两个数在该位上是不同的。根据这个性质，我们将数字分为两组：

1. 一组是和该位为 0 的数字，
2. 一组是该位为 1 的数字。

然后将这两组分别进行异或运算，就可以得到最终要求的两个数字。

思路 1：代码

py

```
class Solution:
    def singleNumbers(self, nums: List[int]) -> List[int]:
        all_xor = 0
        for num in nums:
            all_xor ^= num
        # 获取所有异或中最低位的 1
        mask = 1
        while all_xor & mask == 0:
            mask <<= 1

        a_xor, b_xor = 0, 0
        for num in nums:
            if num & mask == 0:
```

```
                a_xor ^= num
            else:
                b_xor ^= num

        return a_xor, b_xor
```

思路 1：复杂度分析

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