如何寻找消失的元素

相关推荐：

- 学习算法和数据结构的思路指南
- 回溯算法最佳实践：括号生成

读完本文，你不仅学会了算法套路，还可以顺便去 LeetCode 上拿下如下题目：

448.找到所有数组中消失的数字

之前也有文章写过几个有趣的智力题，今天再聊一道巧妙的题目。

题目非常简单：

> 给定一个包含 `0, 1, 2, ..., n` 中 *n* 个数的序列，找出 0 .. *n* 中没有出现在序列中的那个数。
>
> **示例 1:**
>
> **输入:** `[3,0,1]`
> **输出:** `2`
>
> **示例 2:**
>
> **输入:** `[9,6,4,2,3,5,7,0,1]`
> **输出:** `8`

给一个长度为 n 的数组，其索引应该在 `[0,n)`，但是现在你要装进去 n + 1 个元素 `[0,n]`，那么肯定有一个元素装不下嘛，请你找出这个缺失的元素。

这道题不难的，我们应该很容易想到，把这个数组排个序，然后遍历一遍，不就很容易找到缺失的那个元素了吗？

或者说，借助数据结构的特性，用一个 HashSet 把数组里出现的数字都储存下来，再遍历 `[0,n]` 之间的数字，去 HashSet 中查询，也可以很容易查出那个缺失的元素。

排序解法的时间复杂度是 O(NlogN)，HashSet 的解法时间复杂度是 O(N)，但是还需要 O(N) 的空间复杂度存储 HashSet。

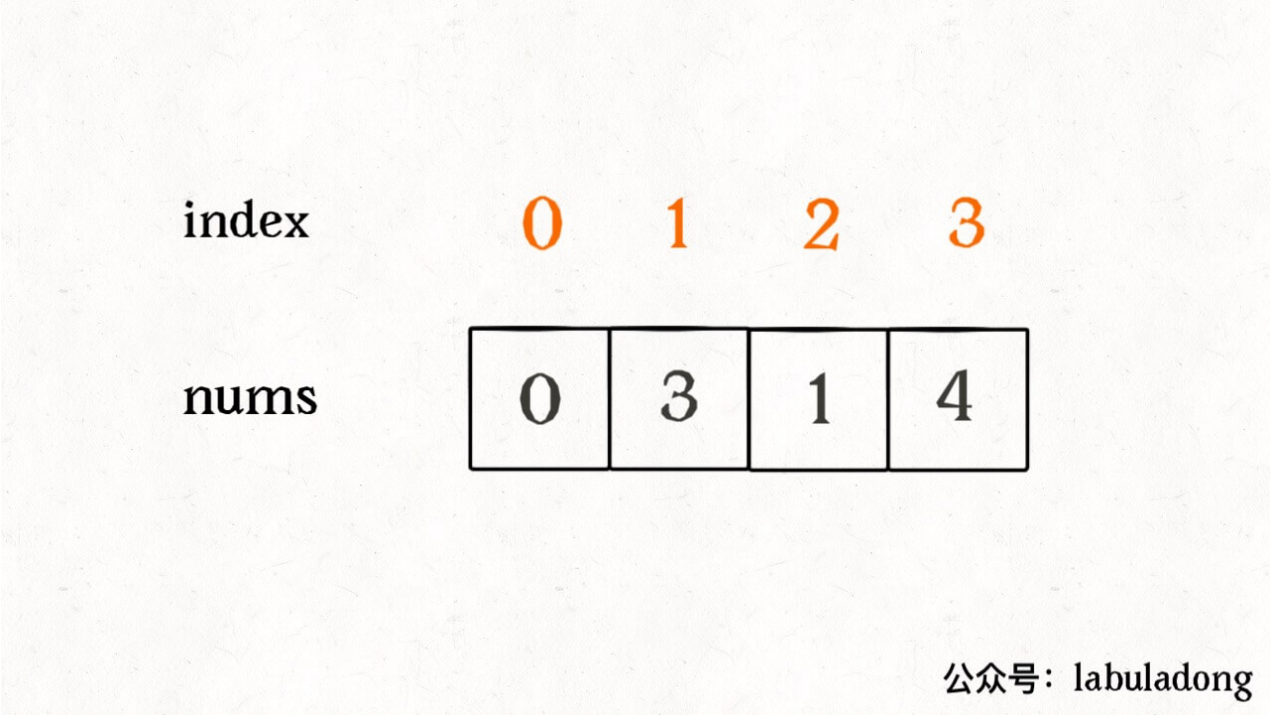
第三种方法是位运算。

对于异或运算（`^`），我们知道它有一个特殊性质：一个数和它本身做异或运算结果为 0，一个数和 0 做异或运算还是它本身。

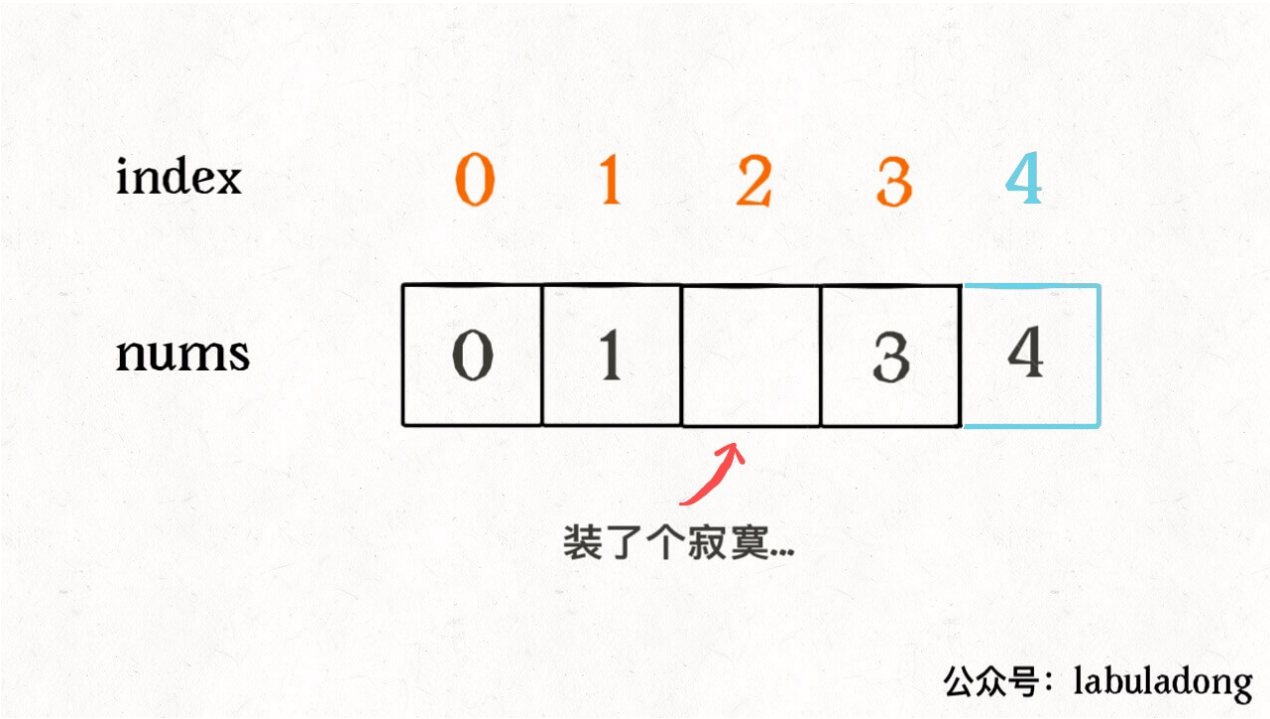
而且异或运算满足交换律和结合律，也就是说：

2 ^ 3 ^ 2 = 3 ^ (2 ^ 2) = 3 ^ 0 = 3

而这道题索就可以通过这些性质巧妙算出缺失的那个元素。比如说 `nums = [0,3,1,4]`：



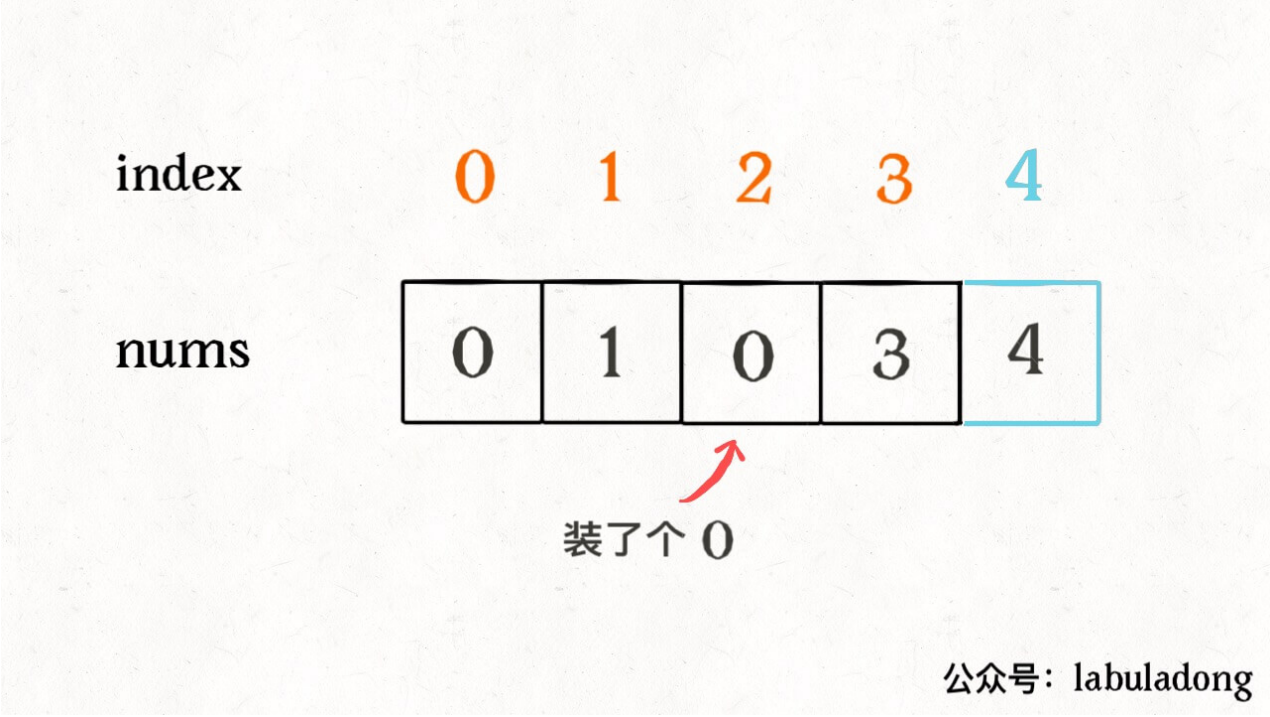
为了容易理解，我们假设先把索引补一位，然后让每个元素和自己相等的索引相对应：



这样做了之后，就可以发现除了缺失元素之外，所有的索引和元素都组成一对儿了，现在如果把这个落单的索引 2 找出来，也就找到了缺失的那个元素。

如何找这个落单的数字呢，**只要把所有的元素和索引做异或运算，成对儿的数字都会消为 0，只有这个落单的元素会剩下**，也就达到了我们的目的。

```
int missingNumber(int[] nums) {
    int n = nums.length;
    int res = 0;
    // 先和新补的索引异或一下
    res ^= n;
    // 和其他的元素、索引做异或
    for (int i = 0; i < n; i++)
        res ^= i ^ nums[i];
    return res;
}
```



由于异或运算满足交换律和结合律，所以总是能把成对儿的数字消去，留下缺失的那个元素的。

至此，时间复杂度 O(N)，空间复杂度 O(1)，已经达到了最优，我们是否就应该打道回府了呢？

如果这样想，说明我们受算法的毒害太深，随着我们学习的知识越来越多，反而容易陷入思维定式，这个问题其实还有一个特别简单的解法：**等差数列求和公式**。

题目的意思可以这样理解：现在有个等差数列 0, 1, 2,..., n，其中少了某一个数字，请你把它找出来。那这个数字不就是 `sum(0,1,..n) - sum(nums)` 嘛？

```
int missingNumber(int[] nums) {
    int n = nums.length;
    // 公式：(首项 + 末项) * 项数 / 2
    int expect = (0 + n) * (n + 1) / 2;

    int sum = 0;
    for (int x : nums)
        sum += x;
    return expect - sum;
```

你看，这种解法应该是最简单的，但说实话，我自己也没想到这个解法，而且我去问了几个大佬，他们也没想到这个最简单的思路。相反，如果去问一个初中生，他也许很快就能想到。

做到这一步了，我们是否就应该打道回府了呢？

如果这样想，说明我们对细节的把控还差点火候。在用求和公式计算 `expect` 时，你考虑过**整型溢出**吗？如果相乘的结果太大导致溢出，那么结果肯定是错误的。

刚才我们的思路是把两个和都加出来然后相减，为了避免溢出，干脆一边求和一边减算了。很类似刚才位运算解法的思路，仍然假设 `nums = [0,3,1,4]`，先补一位索引再让元素跟索引配对：

Index	0	1	2	3
Value	0	3	1	4

$$
\begin{aligned}
missing &= 4 \wedge (0 \wedge 0) \wedge (1 \wedge 3) \wedge (2 \wedge 1) \wedge (3 \wedge 4) \\
&= (4 \wedge 4) \wedge (0 \wedge 0) \wedge (1 \wedge 1) \wedge (3 \wedge 3) \wedge 2 \\
&= 0 \wedge 0 \wedge 0 \wedge 0 \wedge 2 \\
&= 2
\end{aligned}
$$

我们让每个索引减去其对应的元素，再把相减的结果加起来，不就是那个缺失的元素吗？

```
public int missingNumber(int[] nums) {
    int n = nums.length;
    int res = 0;
    // 新补的索引
    res += n - 0;
    // 剩下索引和元素的差加起来
    for (int i = 0; i < n; i++)
        res += i - nums[i];
    return res;
}
```

由于加减法满足交换律和结合律，所以总是能把成对儿的数字消去，留下缺失的那个元素的。

至此这道算法题目经历九曲十八弯，终于再也没有什么坑了。

刷算法，学套路，认准 labuladong，公众号和 在线电子书 持续更新最新文章。

本小抄即将出版，微信扫码关注公众号，后台回复「小抄」限时免费获取，回复「进群」可进刷题群一起刷题，带你搞定 LeetCode。

==其他语言代码==