

## 剑指Offer（四十）：数组中只出现一次的数字

🕒 2018年1月17日 10:39:12 🗂 8 🌡 8,302 °C 📝 编辑



## 一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：[牛客网](#)

书籍下载：[共享资源](#)

## 二、题目

一个整型数组里除了两个数字之外，其他的数字都出现了两次。请写程序找出这两个只出现一次的数字。要求时间复杂度是 $O(n)$ ，空间复杂度是 $O(1)$ 。

### 1、思路

大家首先想到的是顺序扫描法，但是这种方法的时间复杂度是 $O(n^2)$ 。接着大家又会考虑用哈希表的方法，但是空间复杂度不是 $O(1)$ 。

应该怎么做才能即满足时间复杂度是 $O(n)$ 又满足空间复杂度是 $O(1)$ 的要求呢？

我们可以想一想“异或”运算的一个性质，我们直接举例说明。

举例： $\{2,4,3,6,3,2,5,5\}$

这个数组中只出现一次的两个数分别是4和6。怎么找到这两个数字呢？

我们先不看找到两个的情况，先看这样一个问题，如何在一个数组中找到一个只出现一次的数字呢？比如数组： $\{4,5,5\}$ ，唯一一个只出现一次的数字是4。

我们知道异或的一个性质是：**任何一个数字异或它自己都等于0**。也就是说，如果我们从头到尾依次异或数组中的每一个数字，那么最终的结果刚好是个只出现一次的数字。比如数组 $\{4,5,5\}$ ，我们先用数组中的第一个元素4（二进制形式：0100）和数组中的第二个元素5（二进制形式：0101）进行异或操作，0100和0101异或得到0001，用这个得到的元素与数组中的第三个元素5（二进制形式：0101）进行异或操作，0001和0101异或得到0100，正好是结果数字4。这是因为数组中相同的元素异或是为0的，因此就只剩下那个不成对的孤苦伶仃元素。

现在好了，我们已经知道了如何找到一个数组中找到一个只出现一次的数字，那么我们如何在一个数组中找到两个只出现一次的数字呢？如果，我们可以将原始数组分成两个子数组，使得每个子数组包含一个只出现一次的数字，而其他数字都成对出现。这样，我们就可以用上述方法找到那个孤苦伶仃的元素。

我们还是从头到尾一次异或数组中的每一个数字，那么最终得到的结果就是两个只出现一次的数组的异或结果。因为其他数字都出现了两次，在异或中全部抵消了。由于两个数字肯定不一样，那么异或的结果肯定不为0，也就是说这个结果数组的二进制表示至少有一个位为1。我们在结果数组中找到第一个为1的位的位置，记为第n位。现在我们以第n位是不是1为标准把原数组中的数字分成两个子数组，第一个子数组中每个数字的第n位都是1，而第二个子数组中每个数字的第n位都是0。

举例： $\{2,4,3,6,3,2,5,5\}$

我们依次对数组中的每个数字做异或运行之后，得到的结果用二进制表示是0010。异或得到结果中的倒数第二位是1，于是我们根据数字的倒数第二位是不是1分为两个子数组。第一个子数组 $\{2,3,6,3,2\}$ 中所有数字的倒数第二位都是1，而第二个子数组 $\{4,5,5\}$ 中所有数字的倒数第二位都是0。接下来只要分别两个子数组求异或，就能找到第一个子数组中只出现一次的数字是6，而第二个子数组中只出现一次的数字是4。

2、代码

C++:


C++

```
1 class Solution {
2 public:
3     void FindNumsAppearOnce(vector<int> data,int* num1,int *num2) {
4         int length = data.size();
5         if(length < 2){
6             return;
7         }
8
9         // 对原始数组每个元素求异或
10        int resultExclusiveOR = 0;
11        for(int i = 0; i < length; ++i){
12            resultExclusiveOR ^= data[i];
13        }
14
15        unsigned int indexOf1 = FindFirstBitIs1(resultExclusiveOR);
16
17        *num1 = *num2 = 0;
18        for(int j = 0; j < length; j++){
19            if(IsBit1(data[j], indexOf1)){
20                *num1 ^= data[j];
21            }
22            else{
23                *num2 ^= data[j];
24            }
25        }
26    }
27 private:
28    // 找到二进制数num第一个为1的位数，比如0010，第一个为1的位数是2。
29    unsigned int FindFirstBitIs1(int num){
30        unsigned int indexBit = 0;
31        // 只判断一个字节的
32        while((num & 1) == 0 && (indexBit < 8 * sizeof(unsigned int))){
33            num = num >> 1;
34            indexBit++;
35        }
36        return indexBit;
37    }
38    // 判断第indexBit位是否为1
39    bool IsBit1(int num, unsigned int indexBit){
40        num = num >> indexBit;
41        return (num & 1);
42    }
43 };
```

Python:

Python

```
1 # -*- coding:utf-8 -*-
2 class Solution:
3     # 返回[a,b] 其中ab是出现一次的两个数字
4     def FindNumsAppearOnce(self, array):
5         # write code here
6         if len(array) <= 0:
7             return []
8         resultExclusiveOR = 0
9         length = len(array)
10        for i in array:
11            resultExclusiveOR ^= i
12        firstBitIs1 = self.FindFisrtBitIs1(resultExclusiveOR)
13        num1, num2 = 0, 0
14        for i in array:
15            if self.BitIs1(i, firstBitIs1):
16                num1 ^= i
17            else:
18                num2 ^= i
19        return num1, num2
20
21    def FindFisrtBitIs1(self, num):
22        indexBit = 0
23        while num & 1 == 0 and indexBit <= 32:
24            indexBit += 1
25            num = num >> 1
26        return indexBit
27
28    def BitIs1(self, num, indexBit):
29        num = num >> indexBit
30        return num & 1
```



微信公众号

分享技术，乐享生活：微信公众号搜索「JackCui-AI」关注一个在互联网摸爬滚打的潜行者。

虽然迷茫与痛苦过，但也曾天真的笑过。--- 岸本齐史《火影忍者》