给定一个**非空**整数数组，除了某个元素只出现一次以外，其余每个元素均出现两次。找出那个只出现了一次的元素。

说明：

你的算法应该具有线性时间复杂度。 你可以不使用额外空间来实现吗？

示例 1:

```
输入: [2,2,1]
输出: 1
```

示例 2:

```
输入: [4,1,2,1,2]
输出: 4
```

这个题与有一个数组data[n-1]存储了1~n中的n-1个数，问data中缺少的数字是多少【每日一题】实质上是一样的。此处仅把其中最优的位操作法重复举一下，更详细的请参考上述博文。

首先，这个方案使用的位运算中的**异或**（^），a^b当a和b不相等时为1，相等时为0。

算法描述

将data中的所有元素进行异或运算，然后再将结果与1~n每个元素依次异或，最后得到的结果就是缺少的元素（只出现了一次的元素）。

我们来论证一下这个算法的正确性：

1. 0 ^ 1 = 1, 1 ^ 0 = 1, 0 ^ 0 = 0, 1 ^ 1 = 0
2. 对于任意整数n，n ^ 0 = n, n ^ n = 0

（1）当n与0异或时，由于0的所有二进制位均为0，因此，n的二进制位中为1的与0相应位的二进制位0异或结果为1，n的二进制位中为0的与0相应位的二进制位0异或结果为0，因此异或后的结果与n本身完全相同；（2）当n与n异或时，由于其二进制位完全相同，而根据1中0 ^ 0 = 0，1 ^ 1 = 0，n ^ n结果的所有位均为0，所以结果为0。

3. 异或运算满足交换结合律 a ^ b ^ c = a ^ c ^ b.

其实我们可以将所有的abc均看做二进制形式，其结果可以看做是如下运算：

```
00000000 00000000 00000000 00000010 a = 2
^
00000000 00000000 00000000 00000001 b = 1
^
00000000 00000000 00000000 00000100 c = 4
-----------------------------------------
00000000 00000000 00000000 00000111 result = 7
```

即所有运算数的每一位分别异或，因此不论运算顺序如何，结果都相同。

结论

综合1、2、3，然后再根据我们的数据的特点，有2n-1个数，其中有n-1个数出现了两次，只有一个数出现了1次，那么我们将所有的2n-1个数进行异或时，可以看成如下过程，对于出现了两次的元素，x ^ x = 0，然后是n-1个0和剩余的那个只出现了一次的y进行异或，n-1个0异或的结果还是0，最后再与y异或结果是y，y就是我们要找的缺失的元素，因此上述算法是正确的。

这个算法，需要将所有元素做异或运算，**时间复杂度O(n)，空间复杂度O(c)，而且不会有溢出的问题**，这是面试官最喜欢的答案了。

代码

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int tem=0;
        for(int i=0;i<nums.size();++i){
            tem^=nums[i];
        }
        return tem;
    }
};
```