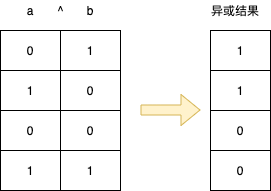
异或 XOR （exclusive OR ）

程序中的所有数在计算机内存中都是以二进制的形式储存的。位操作是程序设计中对位模式或二进制数的一元和二元操作。在许多古老的微处理器上，位运算比加减运算略快，通常位运算比乘除法运算要快很多。在现代架构中，情况并非如此：位运算的运算速度通常与加法运算相同（仍然快于乘法运算）。（摘自维基百科）

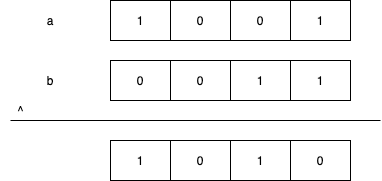
常见的位运算符有：与&，或|，异或^，取反~（在go中为^），左移，右移。

那我们今天主要看下位运算中的异或。

如下图所示，异或，也就是我们常说的无进位相加。对于a，b两个变量，当他们的取值为[]int{0,1}时，异或的结果为：相同为0，不同为1。



我们再来看一个例子：



根据相同为0，不同为1的规则，可以看到a，b异或后为1010。

由于整型可转换成二进制的0和1来表示，所以，我们可以得出

0 ^ 任意数 = 任意数

任意数 ^ 它自己 = 0

并且，异或运算满足交换律和结合律（不受次序影响）。

A ^ B = B ^ A

A ^ B ^ C = A ^ ( B ^ C )

【结合例题理解】

https://leetcode-cn.com/problems/single-number/

# 题目：[136. 只出现一次的数字](https://leetcode-cn.com/problems/single-number/)

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。

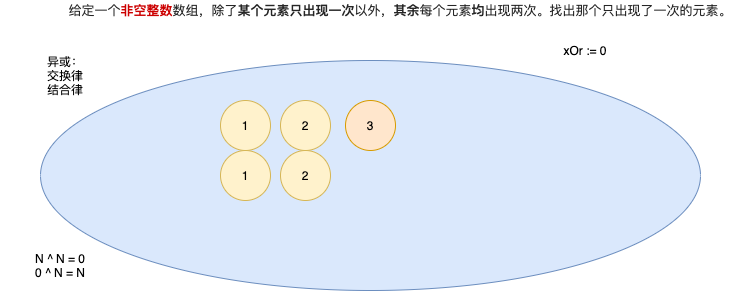
注意：时间复杂度线性且不使用额外的空间。

思路：

两个相同的数，异或，结果为0

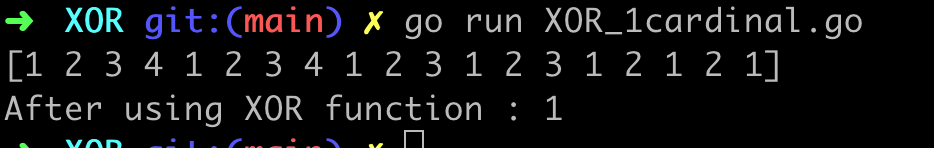
两个不同的数，异或，结果为那个数

（具体过程可以参见目录下视频：*【异或】有一个数-奇数次出现.mov*）



**代码：XOR\_1cardinal.go**

运行结果：



<https://leetcode-cn.com/problems/swap-numbers-lcci/>

# 题目：[面试题 16.01. 交换数字](https://leetcode-cn.com/problems/swap-numbers-lcci/)

编写一个函数，不用临时变量，直接交换numbers = [a, b]中a与b的值。

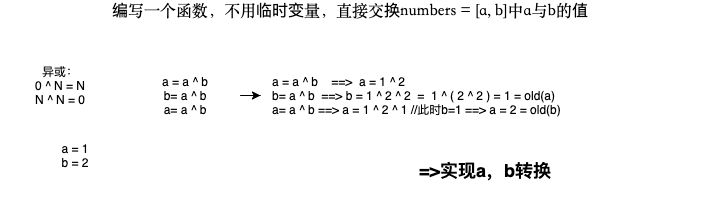
注意：不使用临时变量。

思路：

两个相同的数，异或，结果为0

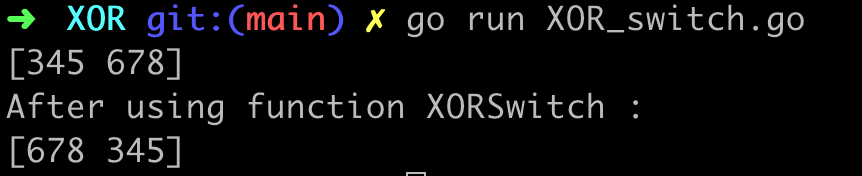
两个不同的数，异或，结果为那个数

（具体过程可参见视频：*【异或】函数不引入新变量实现交换.mov）*



**代码：XOR\_switch.go**

运行结果：



<https://leetcode-cn.com/problems/shu-zu-zhong-shu-zi-chu-xian-de-ci-shu-lcof/>

# 题目：[剑指 Offer 56 - I. 数组中数字出现的次数](https://leetcode-cn.com/problems/shu-zu-zhong-shu-zi-chu-xian-de-ci-shu-lcof/)

一个整型数组 nums 里除两个数字之外，其他数字都出现了两次。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)。

注意：时间复杂度是O(n)，空间复杂度是O(1)。

思路：

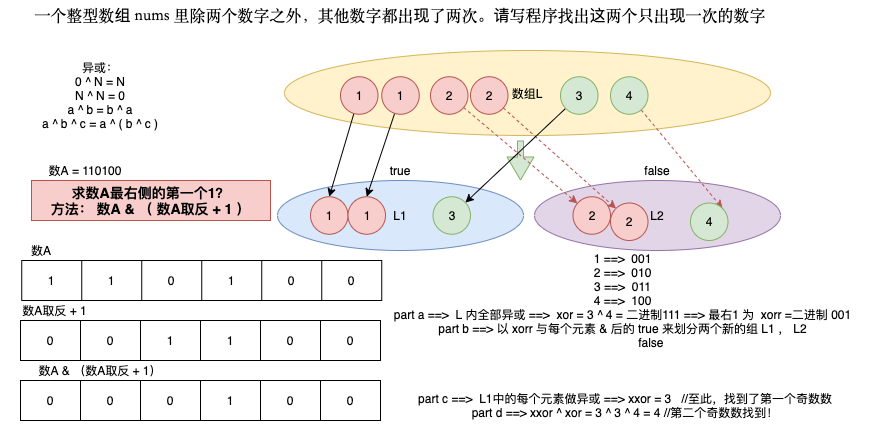
两个相同的数，异或，结果为0

两个不同的数，异或，结果为那个数

**两个出现奇数次的数，分堆进行**

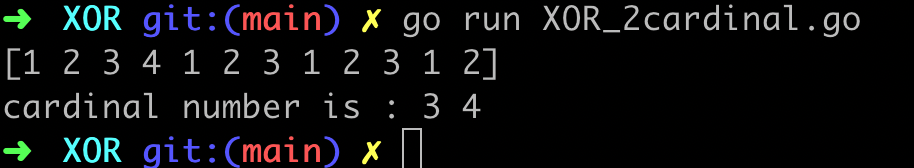
数A & （ 数A取反 + 1 ）🡺 可得最右边的第一个1

（具体过程可以参见目录下视频：*【异或】有两个数-奇数次出现.mov*）



**代码：XOR\_2cardinal.go**

运行结果：



<https://leetcode-cn.com/problems/chalkboard-xor-game/>

# 题目：[810. 黑板异或游戏](https://leetcode-cn.com/problems/chalkboard-xor-game/)

黑板上写着一个非负整数数组 nums[i] 。Alice 和 Bob 轮流从黑板上擦掉一个数字，Alice 先手。如果擦除一个数字后，剩余的所有数字按位异或运算得出的结果等于 0 的话，当前玩家游戏失败。 (另外，如果只剩一个数字，按位异或运算得到它本身；如果无数字剩余，按位异或运算结果为 0。）

并且，轮到某个玩家时，如果当前黑板上所有数字按位异或运算结果等于 0，这个玩家获胜。

假设两个玩家**每步都使用最优解**，当且仅当 Alice 获胜时返回 true。

思路：涉及下棋及异或。

两个相同的数，异或，结果为0

**两个不同的数，异或，结果一定不为0**

异或满足交换律和结合律

1. 初始为0.Alice胜
2. 初始不为0

* 偶数个元素数，必存在 a ^ b != 0 则 Alice胜
* 奇数个元素数，
* a ^ a ^ b 情况 🡺 0 ^ b 则 **Alice输**
* a ^ b ^ c 情况 🡪 则 **Alice输**

**代码：XOR\_BlackboardGame.go**

运行结果：

