是非成败转头空 *异或运算的应用*

万精油

上期趣味数学专栏的题目是棋盘定位。为方便解答，我们把上期题目再列一遍。

上期题目：

棋盘定位：教授把你叫到他的办公室，给你一个国际象棋棋盘（8×8 方格），棋盘上有些格子里有棋子，有些格子里没有棋子。棋子都是翻过来的，也就是说每个棋子都一样。教授在棋盘上随便指了一格。你现在需要做下面两件事中的一个：

1. 选一个有棋子的格子，把棋子从格子中拿走。

或者

2. 选一个没有棋子的格子，放一个棋子到格子中。

教授再把你的朋友叫进来，把你改变过的棋盘给他看。他的任务是从棋盘中看出教授所指的那个格子。

注意，棋盘的初始状态是随机的，教授所指的格子也是随机的。两个动作你只能选一样来做。你可以事先与你的朋友商量好一套利用棋盘上的棋子位置的信息传递系统，使得你的朋友总可以确定教授所指的格子。

解答：将棋盘上的位置按 0, 1,⋯, 63 标号。对任何一个格局 A，设 $F(A) = x_1 \hat{} x_2 \hat{} \cdots \hat{} x_k$，其中 $x_1, \cdots x_k$ 为其位置上有棋子的点的标号。符号 ^ 为对应于每一个比特的异或运算。对于教授指定的任何一点 p，只要变动 $F(A) \hat{} p$ 位置上的棋盘即可。如果该点没有子则加子，有子则把它拿掉。

你的朋友看到格局 B 时，只需算出 $F(B)$, $F(B)$ 所对应的点即为教授所指定的点 p。

对于熟悉异或运算的人来说，这个解法一目了然，非常漂亮。对于不熟悉异或运算的人来说，可能看不懂。我们现在就来介绍一下这个异或运算。

异或运算是一个逻辑运算。当两个输入量不同时，输出 1，反之输出 0。这个运算英文叫 ExclusiveOR，计算机语言里常用 XOR 表示对每一个比特的异或运算。比如，12 与 9，表示成 2 进制就是 1100, 1001，12^9 = 1100^1001 = 0101 = 5。显然，在有多个数做异或运算时，运算结果由每一个比特上的 1 的奇偶性决定。奇数个 1 结果就是 1，偶数个 1 结果就是 0。

异或运算表

输入		输出
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

考虑一下模 2 的加法运算。0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 ≡ 0(mod2)。可以看出，异或运算与模 2 的加法等价。它满足交换律、结合律。

从这个模 2 等价可以得出异或运算的一个重要性能，我把它称为还原功能。具体说起来就是，对任意 x, y : x^(x^y) = y。因为任何比特自己与自己相加总是偶数，模 2 就是 0。所以，别的数不受影响。

当 y 与 x 进行异或运算后，表面上看起来，y 的信息消失了。但是，如果我们把这个结果再与 x 进行异或运算，消失的 y 又出现了。

有了这个还原性，我们可以来解释一下本文的标题。第一次运算，是是非非都成了 0（是与是，非与非都变成了 0），正好对应了“是非成败转头空”。第二次运算，“转头空”的东西又出现了，正好对应那句诗的后一句，“青山依旧在”。

有了这些准备工作，我们可以来解释前面的解答了。

假设原来的棋盘格局是 A，教授指定点 p，我们变动的是 $F(A)$^p 点，其中 $F(A)$ 是对 A 中所有有棋子的点的标号做的连续异或运算。$F(A) = x_1$^x_2^⋯^x_k，其中 x_1,⋯x_k 为其位置上有棋子的点的标号。

如果 $F(A)$^p 点没有棋子，我们在那里加一个子。设变动后的格局为 B。格局 B 就是格局 A 加上 $F(A)$^p，所以，

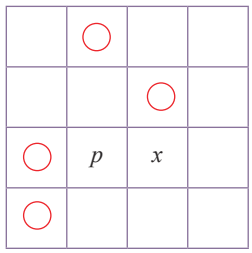
$F(B) = x_1$^x_2^⋯^x_k^($F(A)$^p) = $F(A)$^($F(A)$^p)。利用异或运算的还原性，两个 $F(A)$ 消掉了，结果正好是教授指定的点 p。

如果 $F(A)$^p 点有棋子，设该点为 x_i, 我们把 x_i 点的棋子去掉，相当于对 x_i 异或运算两次（两次就把该点化为 0）。所以，

$F(B) = x_1$^x_2^⋯x_{i-1}^x_{i+1}...^$x_k = x_1$^x_2^⋯x_{i-1}^x_{i+1}⋯^x_k^(x_i^x_i) = x_1^x_2^⋯x_{i-1}^ x_i^x_{i+1}⋯^x_k^x_i = $F(A)$^x_i = $F(A)$^($F(A)$^p)，结果还是教授指定的点 p。

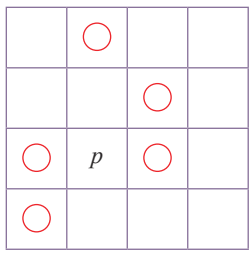
前面的解释或许太抽象，不好懂。我们来看一个具体例子：

假设棋盘是 4×4 的大小。初始格局如下图（有圆圈的格子表示有棋子）。



初始的格局 A

我们把棋盘从左到右，从上到下标为 0, 1, 2, …, 15。有棋子的点的标号是 1, 6, 8, 12。在 2 进制下用比特表示这些数就是 0001, 0110, 1000, 1100。对这些标号做异或运算，我们有：$F(A)$ = 0001^0110^1000^1100 = 0011（只需数每个比特上的 1 是奇数还是偶数）；假设教授指定的点是 p（图中有 p 的点），这点的标号是 9，那么我们有：$F(A)$^9 = 0011^1001 = 1010。这是标号为 10 的点（图中有 x 的点）。根据我们的解答方法，由于这个点没有棋子，我们要在这个点加一个棋子。加完后我们得到格局 B 如下图



变换后的格局 B

你的朋友进来后看到格局 B，对格局 B 做异或运算 F(B) = 0001^0110^1000^1010^1100 = 1001，这是标号为 9 的点，正好是教授指定的点。解答完毕。

简单的异或运算漂亮地解决了这个初看起来似乎不可能的题目，异或运算大显神通。

这个题目实际上可以推广到大小为 $2^N \times 2^N$ 的任意棋盘。其证明不难。稍难一点的是证明只有这种大小的棋盘才可行。不是 $2^N \times 2^N$ 大小的棋盘就没有解。这个证明留给读者做练习。

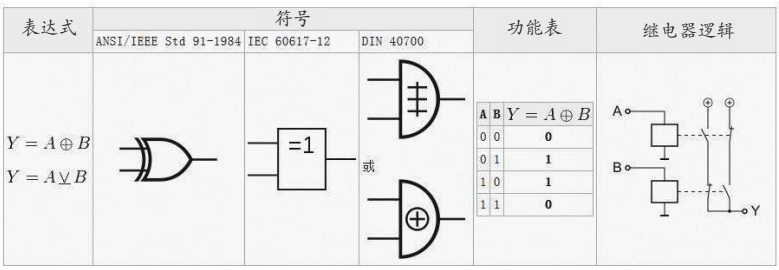
我们再回头来说异或运算。

学计算机编程语言的人基本上都会遇到一个经典题目，要求不用第三个变量，把两个变量的值互换。显然，如果用第三个变量，很容易互换两个变量的值。比如我们要把 X 与 Y 互换，只需引入一个临时变量 Z。$Z = X$; $X = Y$; $Y = Z$；X 与 Y 的值就互换了。如果没有第三个变量，一个变量赋予了一个值，那么它原来的值就丢失了，不能再传给另一个变量。初看起来这是一个几乎不可能完成的任务。这个时候异或运算的还原性就派上了用场。$X = X$^Y; $Y = Y$^X; $X = Y$^X。X 与 Y 巧妙的得到了互换。

说到不用第三个变量互换两个变量的值，想起一道小学时候的趣味题目。说是两个人在路上相遇，他们想交换各自大布袋里的货物（比如一个装的是小米，一个装的是大米），但又想把自己的袋子带回家。没有第三个袋子来倒腾，路面很脏，也不可能倒在路上。问他们怎样完成这个任务。（注：假设布袋很大，货物不到一半）。

这个题比较有趣，但与异或运算没有太大关系。我们还是回头来说异或运算。

异或运算的引入当然不是为了来解决这些趣味题目。它在逻辑、数字电路、计算机里有很广泛的应用。数字电路里的异或门就是异或运算的直接对应。若两个输入的电平相异，则输出为高电平（1）；若两个输入的电平相同，则输出为低电平（0）。



异或运算还可以应用在密码、储存、校码等等诸多方面。其基本原理都是依赖于前面提到的还原功能。

利用异或运算的简单密码：一个信息 X，与钥匙码 K 做异或运算后得 Y，传给另一方。另一方再用钥匙码 K 对 Y 做异或运算，恢复 X。

比如 X 为字符串 math，钥匙码为 12，Y = math^12 = amxd, 另一方得到字符串 amxd 后，与钥匙码 12 异或，amxd^12 = (math^12)^12 = math。原来的字符串得到恢复。不同的钥匙码会对原始码进行不同的加密。当然，这种加密法实在是太过于简单，很容易破。我们这里只是展现它的一个功能。

异或运算在储存与校码上的应用就比较实际了。假设有码 A，B 需要传递。我们传递 A, B 的同时也传递 A^B。对方收到 A, B 后可以用 A^B 来验证。用这种方法来做储存，如果其中任意一个有损坏，都可通过其它两个来恢复。A^(A^B) = B, B^(A^B) = A。

异或运算还有很多其它应用，我们不可能全部介绍完，有兴趣的读者可以自己去找相关资料来看。

本期趣味题目：

本期题目是被公认为最有趣的含数学原理的一个扑克牌游戏。

扑克传信：一副牌 52 张，没有大小王。一个观众从中随机抽出五张。你从其中选出一张藏起来，把剩下的四张放在桌上，让你的朋友根据桌上这四张牌的面值及顺序来推出藏起来的那张牌是什么。也就是说请你和你的朋友设计一套信号系统，使得不管抽出的是哪五张牌，你都可以用其中的四张牌来表示另一张牌。注意，你的朋友可以利用的信息只能是四张牌的面值与顺序。诸如把某张牌翻过来或是放斜一点，高一点，角上折一下之类的旁门左道都不能用。

上面的题做出来以后，请再考虑一下更进一步的情况——不是一副牌，而是一副麻将。筒、条、万各 36 张，中、发、白各四张，再加上春、夏、秋、冬，总共 124 张牌。能不能设计一套信号系统，使得不管抽出的是哪五张牌，你都可以用其中的四张牌来表示另一张牌？注：我们假设每张牌都是不同的。比如四个八万，应该可以像春夏秋冬一样区分。八万春，八万夏，八万秋，八万冬，诸如此类。也就是说我们有 124 张不同的牌。