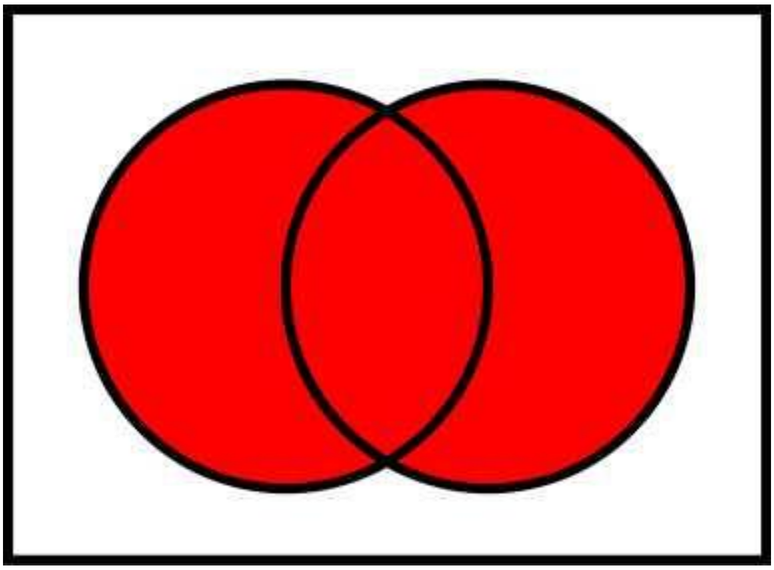
XOR(Exclusive or)

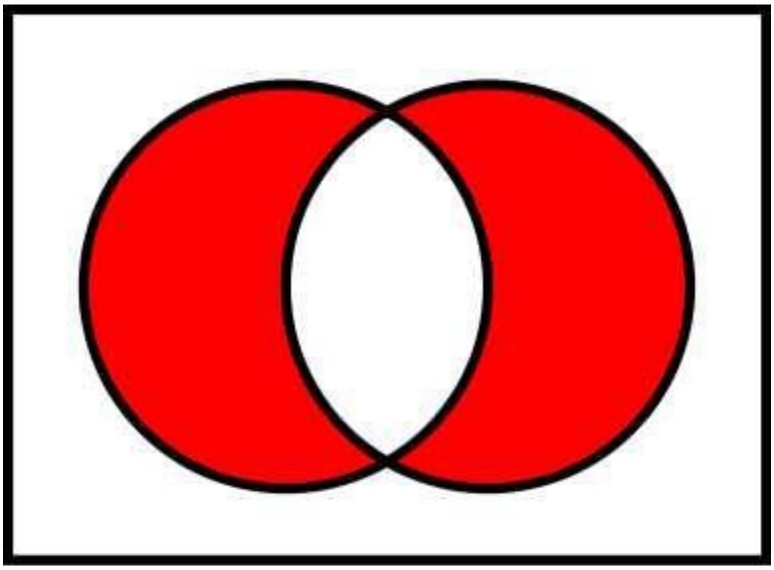
异或是 Z_2 群的加法运算，满足加法结合律和交换律.

exclusive 有一个意思是『排外的』。如果我们说 A 和 B 是『mutually exclusive』，那就是说 A 和 B 互不相容，互相排斥。回想一下『或』这个词在数学里的意思—『一个元素在集合 A 或集合 B 里』往往意味着『它只在 A 里』、『它只在 B 里』、『它在 A 和 B 的交集里』这三种情况中的一种。

『或』的维恩图如下

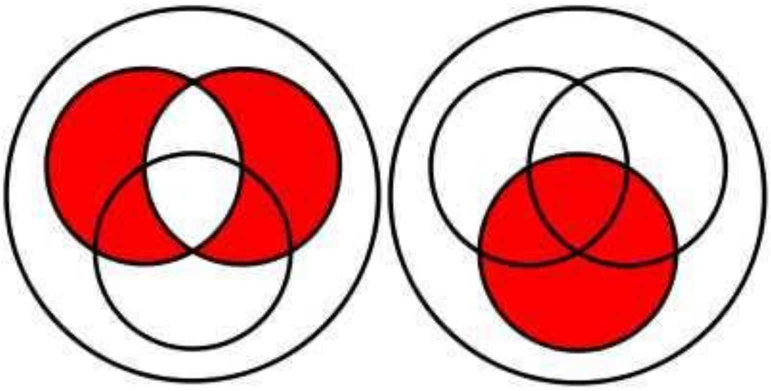


『异或』则不同，不允许『共存』的情况，因为是 exclusive 的，所以维恩图如下：

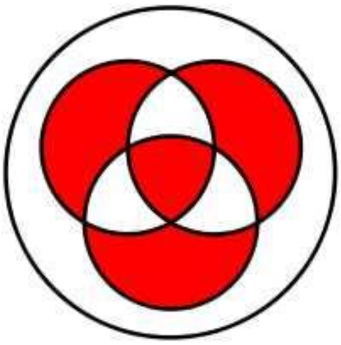


那三个圈的维恩图是什么样的呢？

在两个圈的基础上加上一个圈：



异或一下，得到：



在维恩图中，当我们用一个新的圈来异或已有的圈，我们相当于是在做『反色』操作。所以，异或操作是交换且结合的.

关于 a^b^a=b,即 异或 是 异或 的逆运算。

假设有两个数 A=a 和 B=b，我们要不依靠第三个变量交换它们。首先把 B 加到 A 上。

A = A+B = a+b

然后 A 现在就是 a 和 b 的和。B 仍然是 b。所以我们现在要让 B 变为 a 就是 B = A – B = a+b-b=a. 然后我们让 A 变成 b,

A = A-B=a+b-a=b

那么我们用二进制的性质。

下面举例子：（数字后的 b 表示这是个二进制数）

A = 1101b B = 1011b

然后我们按照下面的规则分别对每一位做加法：

1+0=1 0+1=1 0+0=0 如果有进位，我们就不要了：1+1=0

我们把这个运算用“^”符号表示。

和上面步骤一样：先求“和”

A=A^B=1101b^1011b=0110b

然后我们要算上面的“和” “减” B 的值来得到原来 a 的值。减是上面的“^”的逆运算，那么就是：

1-0=1 1-1=0 0-0=0 同时我们规定，减法的借位也不需要：0-1=1

然后我们继续来算：

B=A^B=0110b^1011b=1101b

A=A^B=0110b^1101b=1011b

和上面的利用加法的交换原理是一样的。

我们现在再来看我们刚才定义的“^”运算：

0^0=0

0^1=1

1^0=1

1^1=0

这就是异或。

不用算术运算符实现两个数的加法(按位异或)

对于二进制的加法运算，若不考虑进位，则 1+1=0,1+0=1,0+1=1,0+0=0,通过对比异或，不难发现，此方法与异或运算类似。因而排除进位，加法可用异或来实现。

然后考虑进位，0+0 进位为 0,1+0 进位为 0,0+1 进位为 0，1+1 进位为 1，该操作与位运算的&操作相似。

那么加法运算可以这样实现：

1）先不考虑进位，按位计算各位累加（用异或实现），得到值 a；

2）然后再考虑进位，并将进位的值左移，得值 b，若 b 为 0，则 a 就是加法运算的结果，若 b 不为 0，则 a+b 即得结果（递归调用该函数）。

```
int bitAdd(int a,int b)
{
    if(b==0)
        return a;
    int sum = a^b;
    int carry =(a&b)<<1;
    return bitAdd(sum,carry);
}
```

题目

给定正整数 X， Y,询问满足 a+b=X 且 a^b=Y 的数对(a,b)(a<=b)中， a 最小的情况；

输出该情况下的 a 值并输出总情况数。

如果不存在这样的数对 ， 输出 No solution.

多组数据。

对于 100%的数据, 满足数据组 t<=100000;X,Y<=1e18.

样例：

Input:

3

7 7

4 2

9 7

Output:

0 4

1 1

No Solution

Sol:

考虑到 a+b=a^b(不考虑进位)+(a&b)<<1（考虑进位）。

所以设 Z = （X-Y）>> 1。

则(a&b) = Z;

则最小的 a 值为 Z.

考虑(a^b) = Y,则对于 Y 的二进制，用加法实现异或，只要 Y 的首位 1 分给 b,则之后可任意分配 1，又考虑到 Y 为 0 时只有一种情况，所以总情况数为 $2^{\max(0,\text{bit（1）count}(Y)-1)}$

考虑 No solution 的情况，即（x-y）<0 或 (x-y)为奇数 或 Z&Y 不为 0（考虑加法 与 异或 的关系）。