高斯消元

两个概念

（1）系数矩阵：由方程式未知数系数构成的矩阵叫做系数矩阵。

比如：$\begin{cases} x_1+x_2+x_3=3 \\ x_1-x_2+2x_3=-2 \\ 3x_1+2x_2-5x_3=6 \end{cases}$ 的系数矩阵为 $\begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \\ 3 & 2 & -5 \end{pmatrix}$

（2）增广矩阵：增广矩阵就是在系数矩阵的右边添上一列，这一列是线性方程组的等号右边的值。

比如：$\begin{cases} x_1+x_2+x_3=3 \\ x_1-x_2+2x_3=-2 \\ 3x_1+2x_2-5x_3=6 \end{cases}$ 的增广矩阵为 $\left(\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 1 & -1 & 2 & -2 \\ 3 & 2 & -5 & 6 \end{array}\right)$

概述:

高斯消元法主要用于求解线性方程组，也可以求矩阵的秩、矩阵的逆等，是一个重要的数学方法。

其时间复杂度主要与方程组个数、方程组未知数个数有关，一般来说，时间复杂度为$O(n^3)$

线性方程组:

线性方程组：有多个未知数，且每个未知数的次数均为一次，这样多个未知数所组成的方程组。

其形式为：$\begin{cases} a_{11}x_1+a_{12}x_2+...+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+...+a_{2n}x_n=b_2 \\ \qquad\qquad ... \\ a_{n1}x_1+a_{n2}x_2+...+a_{nn}x_n=b_n \end{cases}$

记为矩阵形式，有：

$$Ax=B, A=\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix}, B=\begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{bmatrix}$$

高斯消元法:

高斯消元法的基本思想是：通过一系列的加减消元运算，直到得到类似 kx=b 的式子，然后逐一回代求解 x 向量

1.解方程组过程

以以下线性方程组为例

$$\begin{cases} 3x+2y+z=6 \quad (1) \\ 2x+2y+2z=4 \quad (2) \\ 4x-2y-2z=2 \quad (3) \end{cases}$$

首先进行消元操作：

将 (1) 除以 3，把 x 的系数化为 1，得：$x+\frac{2}{3}y+\frac{1}{3}z=2 \quad (1)'$

再令 (2)-2*(1)'、(3)-4*(1)'，将 (2)、(3) 的 x 消去，得：

$$\begin{cases} x+\frac{2}{3}y+\frac{1}{3}z=2 & (1)' \\ 0x+\frac{2}{3}y+\frac{4}{3}z=0 & (2)' \\ 0x-\frac{14}{3}y-\frac{10}{3}z=-6 & (3)' \end{cases}$$

将令 (2)' 除以 2/3，将 y 系数化为 1，得：

$y+2z=0 \quad (2)''$

再令$(3)'-(-14/3)*(2)''$，将$(3)'$的y消去，得：

$$\begin{cases} x+\frac{2}{3}y+\frac{1}{3}z=2 & (1)' \\ 0x+y+2z=0 & (2)'' \\ 0x+0y+\frac{18}{3}z=-6 & (3)'' \end{cases}$$

由 $(3)''$可得：$z=-1$,最后进行回代操作：

将$z=-1$带回 (2)'' 可得：$y=2$

将$y=2$、$z=-1$带回$(1)'$可得：$x=1$

即结果为：$\begin{cases} x=1 \\ y=2 \\ z=-1 \end{cases}$

2.矩阵运算消元过程

同样以以下线性方程组为例:

$$\begin{cases} 3x+2y+z=6 & (1) \\ 2x+2y+2z=4 & (2) \\ 4x-2y-2z=2 & (3) \end{cases}$$

将其记为矩阵形式，有：$\begin{bmatrix} x & y & z & val \\ 3 & 2 & 1 & 6 \\ 2 & 2 & 2 & 4 \\ 4 & -2 & -2 & 2 \end{bmatrix}$

消去 x，有：$\begin{bmatrix} x & y & z & val \\ 1 & \frac{2}{3} & \frac{1}{3} & 2 \\ 0 & \frac{2}{3} & \frac{4}{3} & 0 \\ 0 & -\frac{14}{3} & -\frac{10}{3} & -6 \end{bmatrix}$

消去 y，有：$\begin{bmatrix} x & y & z & val \\ 1 & \frac{2}{3} & \frac{1}{3} & 2 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & \frac{18}{3} & -6 \end{bmatrix}$

回代得解：$\begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$

3.解的判断

无解：当消元完毕后，发现有一行系数都为 0，但是常数项不为 0，此时无解

例如：$\begin{bmatrix} x & y & z & val \\ 1 & \frac{2}{3} & \frac{1}{3} & 2 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & -6 \end{bmatrix}$

多解：当消元完毕后，发现有多行系数、常数项均为 0，此时多解，有几行为全为 0，就有几个自由元，即变量的值可以任取，有无数种情况可以满足给出的方程组

例如：$\begin{bmatrix} x & y & z & val \\ 1 & \frac{2}{3} & \frac{1}{3} & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$，此时自由元个数为 2

普通高斯消元法

Luogu P3389 【模板】高斯消元法

题目描述

给定一个线性方程组，对其求解

输入格式

第一行，一个正整数 n

第二至 $n+1$ 行，每行 $n+1$ 个整数，为$a_1, a_2 \cdots a_n$ 和 b，代表一组方程。

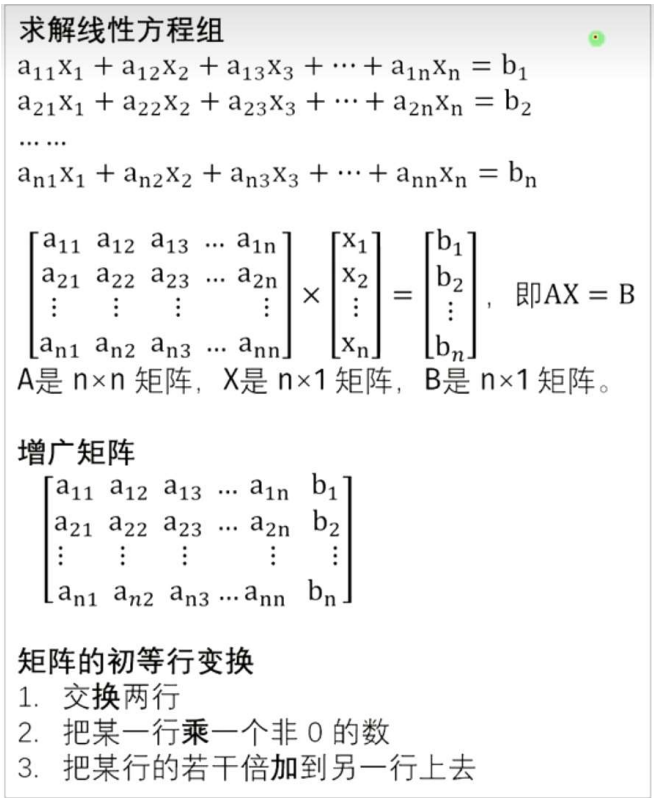
输出格式

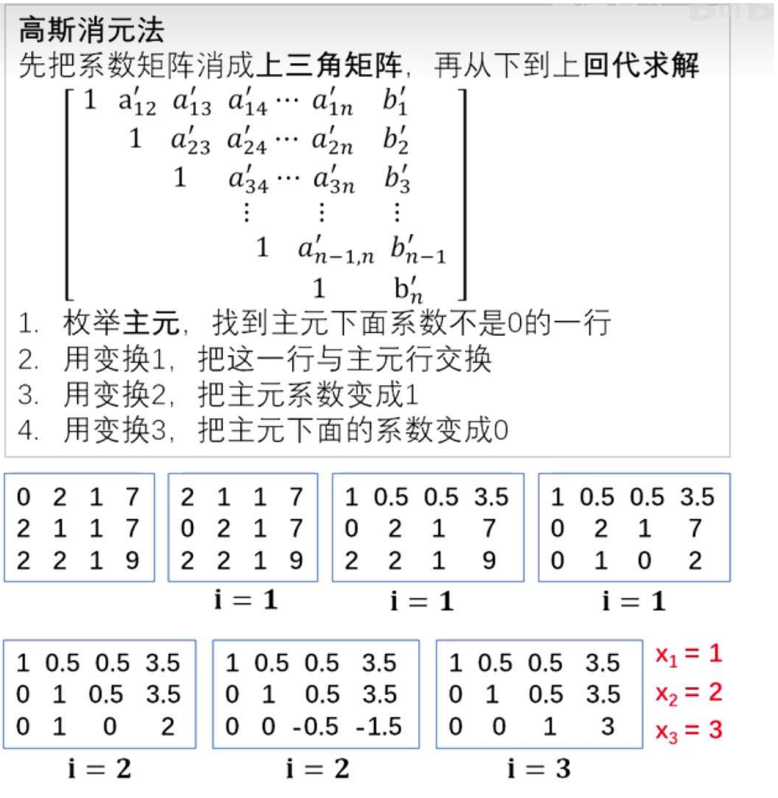
共 n 行，每行一个数，第 i 行为 x_i （保留 2 位小数）

如果不存在唯一解，在第一行输出 `No Solution .`

提示

$1 \leq n \leq 100, |a_i| \leq 10^4, |b| \leq 10^4$





```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 110;
const double eps = 1e-6;
int n;
double a[N][N]; //增广矩阵

int gauss(){
  for(int i=1; i<=n; i++){ //枚举主元行
    int r = i;
    for(int k=i; k<=n; k++) //找非0行
      if(fabs(a[k][i])>eps)
        {r = k; break;}
    if(r!=i) swap(a[r],a[i]); //换行
    if(fabs(a[i][i])<eps) return 0;

    for(int j=n+1; j>=i; j--) //主元变1 ，逆序的原因是主元这一行所有系
    //数都要除以主元的系数，如果要是正着枚举，a[i][i]这个主元系数就变了
      a[i][j] /= a[i][i];
    for(int k=i+1; k<=n; k++) //主元所在i行以下系数变0
      for(int j=n+1; j>=i; j--)
        a[k][j]-=a[k][i]*a[i][j];
        //a[k][i]*a[i][j]：a[k][i]指下面每一行的主元的那一列*主元那一行的每个位置（a[i][j]），
  }
  for(int i=n-1; i>=1; i--) //回代
    for(int j=i+1; j<=n; j++)
      a[i][n+1]-=a[i][j]*a[j][n+1]; //把最终每个未知数的解存在a[i][n+1]
      //内，当要求第i行的主元值a[i][n+1]时，a[i][j]*a[j][n+1]表示：
      //a[i][j]表示主元后面的位置，a[j][n+1]表示第j行的主元的值，j是从
      //i+1到n循环的，就是说要用a[i][n+1]减去a[i][i]这个主元后面左右已经
      //求得的主元值乘以相应系数，最后a[i][n+1]就是主元a[i][i]的值，比
      //如i=2时候，x3已经求得=2，x2=a[2][n+1]-0.5*a[3][n+1](也就是x3的值)=2，
     // 当i=1时，x1=a[1][n+1]=a[1][n+1]-0.5*x2-0.5*x3
  return 1; //存在唯一解
}
```

时间复杂度o(n^3)

```cpp
int main(){
  cin >> n;
  for(int i=1; i<=n; i++)
    for(int j=1; j<=n+1; j++)
      cin >> a[i][j];
  if(gauss())
    for(int i=1; i<n+1; i++)
      printf("%.2lf\n", a[i][n+1]);
  else puts("No Solution");
  return 0;
}
```

```
bool gauss(){
  for(int i=1; i<=n; i++){ //枚举行列
    int r = i;
    for(int k=i; k<=n; k++) //找非0行
      if(fabs(a[k][i])>eps)
        {r = k; break;}
    if(r!=i) swap(a[r],a[i]); //换行
    if(fabs(a[i][i])<eps) return 0;

    for(int j=n+1; j>=i; j--) //变1
      a[i][j] /= a[i][i];
    for(int k=i+1; k<=n; k++) //变0
      for(int j=n+1; j>=i; j--)
        a[k][j]-=a[k][i]*a[i][j];
  }
  for(int i=n-1; i>=1; i--) //回代
    for(int j=i+1; j<=n; j++)
      a[i][n+1]-=a[i][j]*a[j][n+1];
  return 1; //存在唯一解
}
```

高斯消元法

先把系数矩阵消成**上三角矩阵**，再从下到上**回代求解**

$$\begin{bmatrix} 1 & a'_{12} & a'_{13} & a'_{14} & \cdots & a'_{1n} & b'_1 \\ & 1 & a'_{23} & a'_{24} & \cdots & a'_{2n} & b'_2 \\ & & 1 & a'_{34} & \cdots & a'_{3n} & b'_3 \\ & & & \vdots & & \vdots & \vdots \\ & & & & 1 & a'_{n-1,n} & b'_{n-1} \\ & & & & & 1 & b'_n \end{bmatrix}$$

1. 枚举**主元**，找到主元下面系数不是0的一行
2. 用变换1，把这一行与主元行交换
3. 用变换2，把主元系数变成1
4. 用变换3，把主元下面的系数变成0

解分三种情况：

1. 唯一解：i 可以枚举完 n 行
2. 无解：$a[i][i]=0, b\neq 0$
3. 无穷多解：$a[i][i]=0, b=0$

$$\begin{bmatrix} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 8 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

$x_2+2x_3=3$
$x_2+2x_3=6$

$$\begin{bmatrix} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$x_2+2x_3=3$
$2x_2+4x_3=6$

高斯-约旦消元法

```
bool Gauss_Jordan(){
  for(int i=1; i<=n; ++i){ //枚举行列
    int r = i;
    for(int k=i; k<=n; ++k) //找非0行
      if(fabs(a[k][i])>eps)
        {r = k; break;}
    if(r!=i) swap(a[r],a[i]); //换行
    if(fabs(a[i][i])<eps) return 0;

    for(int k=1; k<=n; ++k){ //对角化
      if(k == i) continue;
      double t=a[k][i]/a[i][i];
      for(int j=i; j<=n+1; ++j)
        a[k][j] -= t*a[i][j];
    }
  }
  for(int i=1; i<=n; ++i)
    a[i][n+1]/=a[i][i]; //除以主元
  return 1; //存在唯一解
}
```

高斯-约旦消元法

先把系数矩阵消成**主对角矩阵**，再除以**主元**

$$\begin{bmatrix} a'_{11} & & & & b'_1 \\ & a'_{22} & & & b'_2 \\ & & a'_{33} & & b'_3 \\ & & & \ddots & \vdots \\ & & & a'_{nn} & b'_n \end{bmatrix}$$

每轮循环，主元所在行不变，主元所在列消成 0

```
1 2 1 8
2 2 1 9
2 1 1 7
```

```
1 2  1  8
0 -2 -1 -7
0 -3 -1 -9
```

i = 1

```
1 0  0   1
0 -2 -1  -7
0  0 0.5 1.5
```

i = 2

```
1 0  0  1
0 -2 0  -4
0  0 0.5 1.5
```

i = 3

```
1 0  0  1
0 -2 0  2
0  0 0.5 3
```

除以主元

```cpp
#include <iostream>
#include <algorithm>
#include <cmath>
using namespace std;

const int N = 110;
const double eps = 1e-6;
int n;
double a[N][N]; //增广矩阵

bool Gauss_Jordan(){
  for(int i=1; i<=n; ++i){ //枚举行列
    int r = i;
    for(int k=i; k<=n; ++k) //找非0行
      if(fabs(a[k][i])>eps)
        {r = k; break;}
    if(r!=i) swap(a[r],a[i]); //换行
    if(fabs(a[i][i])<eps) return 0;

    for(int k=1; k<=n; ++k){ //对角化
      if(k == i) continue;//跳过主元那一行，保持不变，因为要消其他行为0
      double t=a[k][i]/a[i][i];
      for(int j=i; j<=n+1; ++j)
        a[k][j] -= t*a[i][j];
    }
  }
  for(int i=1; i<=n; ++i)
    a[i][n+1]/=a[i][i]; //除以主元
  return 1; //存在唯一解
}
int main(){
  cin >> n;
  for(int i=1; i<=n; i++)
    for(int j=1; j<=n+1; j++)
      cin >> a[i][j];
  if(Gauss_Jordan())
    for(int i=1; i<=n; i++)
      printf("%.2lf\n",a[i][n+1]);
  else puts("No Solution");
  return 0;
}
```

poj 1830 开关问题

题目描述:

有N个相同的开关，每个开关都与某些开关有着联系，每当你打开或者关闭某个开关的时候，其他的与此开关相关联的开关也会相应地发生变化，即这些相联系的开关的状态如果原来为开就变为关，如果为关就变为开。

你的目标是经过若干次开关操作后使得最后N个开关达到一个特定的状态。

对于任意一个开关，最多只能进行一次开关操作。

你的任务是，计算有多少种可以达到指定状态的方法。（不计开关操作的顺序）

输入格式

输入第一行有一个数K，表示以下有K组测试数据。

每组测试数据的格式如下：

第一行：一个数N。

第二行：N个0或者1的数，表示开始时N个开关状态。

第三行：N个0或者1的数，表示操作结束后N个开关的状态。

接下来每行两个数I, J，表示如果操作第I个开关，第J个开关的状态也会变化。

每组数据以0 0结束。

输出格式

每组数据输出占一行。

如果有可行方法，输出总数，否则输出 Oh,it's impossible~!! 。

数据范围

$1 \leq K \leq 10,0 < N < 29$

输入样例：

2
3
0 0 0
1 1 1
1 2
1 3
2 1
2 3
3 1
3 2
0 0
3
0 0 0
1 0 1
1 2
2 1
0 0

输出样例：

4
Oh,it's impossible~!!

题解1：

1、描述

有一些开始状态的开关，题目让我们操控开关，使得开关从开始状态变成指定状态。

注意，当你操作一个开关，其关联的开关也会被操控。例如输入样例一，开始状态为000的三个开关，你要操作使其变成111。那么有以下四种方法：

- 只打开开关1，$2\ and\ 3$和1关联，所以$2\ and\ 3$也变成1
- 只打开开关2,3
- 只打开开关3,4
- 打开开关$1, 2, 3$

> 初始状态 0 0 0 0
> 终止状态 1 1 1 1
> 按下i号开关，会影响j号开关:
> 1 3
> 3 1
> 3 4
> 4 1
> 4 3
> 我能影响谁a_i
> a_1->(1,0,1,0)->{1,3}
> a_2->(0,1,0,0)->{2}
> a_3->{1, 3, 4}
> a_4->{1, 3, 4}
> 谁能影响我k_i

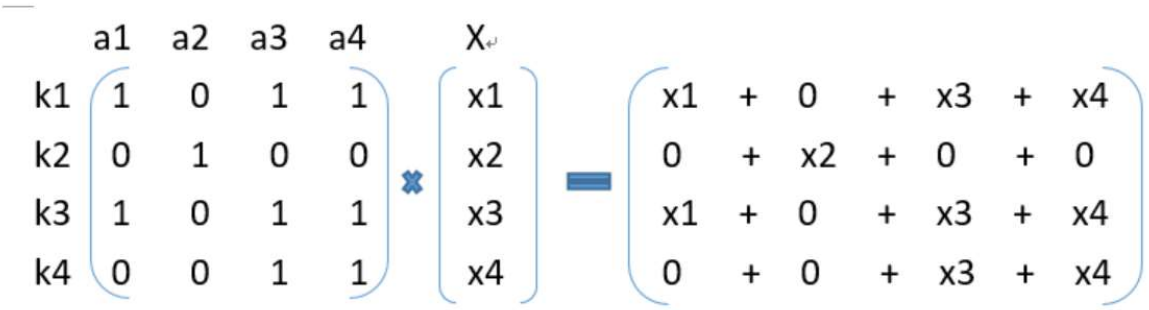
操作每个开关后，影响的情况

2、问题解决

我们用**线性方程组**来求解。

$$[a_1, a_2, a_3] * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = b, 即 A_x = b$$

由上图表达式，我们可以思考，倘若**乘号左边是开关的关联关系**，**乘号右边是开关操作**，那么两个矩阵相乘，**结果就是开关变化**。由此，我们可以让b为开关变化。
意思为当开关从0变为1，那么**开关有变化**，b_i为1；**当开关无变化**，则b_i为0。



例如上面例子，$a_1 = (1, 0, 1, 0)$表示，操作开关1，开关3也会改变。即$a[i][j]$表示操作开关j，开关i也会变化。

所以，我们可以列出矩阵乘法表达式，然后进行高斯消元求解。

Q:为什么这里有反着记录的呢?

答：我们希望以方程组的形式对原问题进行求解。那么，由于有$n盏灯，每个方程都要描述一盏灯的变化情况，这样的话，就共需要n$个方程。

我们以第1盏灯为例，我们知道它的初始状态和终止状态，我们就可以研究它是在初始状态下，通过什么样的操作变化到终止状态的。因为灯开关的特殊性，关联一次就变化一次，所以，我们还可以取一巧：看看开始状态和终止状态是一样的呢，还是有了变化。

- 一样的
  中间的变化是偶数次。用异或运算来描述就是异或和等于0。
- 不一样
  中间的变化是奇数次。用异或运算来描述就是异或和等1。

总结：我们关心的是状态的变化情况

1号灯的变化，受和它相关联灯的制约，以上面的图为例说明：
第1盏灯的变化，受$1,3,4$三个灯的影响，我们可以把1盏灯的变化情况看作

$$1*x_1+0*x_2+1*x_3+1*x_4$$

描述的说是:

- 1号灯操作，影响1号灯的状态
- 2号灯操作，不影响1号灯的状态
- 3号灯操作，影响1号灯的状态
- 4号灯操作，影响1号灯的状态

当然，只是说影响，但每个灯还有是权力决定自己是不是要操作的。
这就引出了一个系数的问题：
"**我是一号灯，谁能影响我的状态?**"

题解2:
对于开关i,j:代表改变开关i的状态，j的状态也会改变;
由于j的状态是因为i的变化而变化，所以，对于我构造的矩阵$a[i][j]$
i为行代表方程的编号，即此方程结果对应的开关编号，j代表未知数的系数；开关j随着开关i变化，那么这里的i和j要相互交换位置！！！！因为行代表的是对应编号开关的状态;

我们可以很显然的知道，某个等的亮灭情况将体现出各个开关的复合结果，因此我们可以得到这样的方程组:

$E(a)=x_a*A_{11}\wedge x_b*A_{12}\wedge x_c*A_{13}\wedge S(a)$;

$E(b)=x_a*A_{21}\wedge x_b*A_{22}\wedge x_c*A_{23}\wedge S(b)$;

$E(c)=x_a*A_{31}\wedge x_b*A_{32}\wedge x_c*A_{33}\wedge S(c)$;

其中$S(a)$表示初始状态，$E(a)$表示a号灯最终的情况，x_a表示a号开关是否按下，A_{13}表示3号开关是否影响1号灯的亮灭，异或操作体现了操作的具体影响。

将S项移到方程的左边，我们抽象出系数矩阵，然后构造出增广矩阵，然后就是进行高斯消元，高斯消元的意思可以看作是符合一系列两个逻辑关系的情况下，对变元要求的变换。只不过这里用的全部都是异或操作，最后我们进行秩的判定，如果矩阵的秩和邻接矩阵的秩不相等的话那么就无解，如果相同的话，那么说明达到理想灯的亮灭情况下至少需要确定的状态的开关数为矩阵的秩。那么解就是剩下开关的任意选择了，即2^X（每个开关对应关或者是开）。

```cpp
#include<bits/stdc++.h>
#define E 1e-9
#define PI acos(-1.0)
#define INF 0x3f3f3f3f
#define LL long long
const int MOD=6;
const int N=1000+5;
const int dx[]= {-1,1,0,0};
const int dy[]= {0,0,-1,1};
using namespace std;

int a[N][N];//增广矩阵
int x[N];//解集
int freeX[N];//自由变元
int Gauss(int equ,int var){//返回自由变元个数
    /*初始化*/
    for(int i=0;i<=var;i++){
        x[i]=0;
        freeX[i]=0;
    }

    /*转换为阶梯阵*/
    int col=0;//当前处理的列
    int num=0;//自由变元的序号
    int k;//当前处理的行
    for(k=0;k<equ&&col<var;k++,col++){//枚举当前处理的行
        int maxRow=k;//当前列绝对值最大的行
        for(int i=k+1;i<equ;i++){//寻找当前列绝对值最大的行
            if(abs(a[i][col])>abs(a[maxRow][col]))
                maxRow=i;
        }
        if(maxRow!=k){//与第k行交换
            for(int j=k;j<var+1;j++)
                swap(a[k][j],a[maxRow][j]);
        }
        if(a[k][col]==0){//col列第k行以下全是0，处理当前行的下一列
            freeX[num++]=col;//记录自由变元
            k--;//k--之后，再运行循环，k又变成了当前k得值，col+1.变成下一列
            continue;
        }

        for(int i=k+1;i<equ;i++){
            if(a[i][col]!=0){//倒三角，要消掉下面得1
                for(int j=col;j<var+1;j++){//对于下面出现该列中有1的行，需要把1消掉
                    a[i][j]^=a[k][j];
                }
            }
        }
    }

    /*求解*/
    //无解：化简的增广阵中存在(0,0,...,a)这样的行，且a!=0
    for(int i=k;i<equ;i++)
        if(a[i][col]!=0)
            return -1;
    return var-k;//自由变元有var-k个
}
int start[N];//开始状态
int endd[N];//结束状态
int main(){
    int t;
    scanf("%d",&t);

    while(t--){
        int n;
        scanf("%d",&n);

        for(int i=0;i<n;i++)//开始状态
            scanf("%d",&start[i]);
        for(int i=0;i<n;i++)//最终状态
            scanf("%d",&endd[i]);

        memset(a,0,sizeof(a));
        for(int i=0;i<n;i++)//每个方程的解
            a[i][n]=start[i]^endd[i];
        for(int i=0;i<n;i++)//第i个开关受第i个开关影响
            a[i][i]=1;
```

```
        int t1,t2;
        while(scanf("%d%d",&t1,&t2)!=EOF&&(t1+t2)){
            //a[t1-1][t2-1]=1;
            a[t2-1][t1-1]=1;
        }

        int freeNum=Gauss(n,n);
        if(freeNum==-1)
            printf("Oh,it's impossible~!!\n");
        else
            printf("%d\n",1<<freeNum);
    }
    return 0;
}
```

```cpp
#include <bits/stdc++.h>

using namespace std;
const int N = 30;
const double eps = 10e-8;
int n;
int start[N]; //开始状态
int stop[N];  //结束状态

//高斯消元模板
double a[N][N]; //增广矩阵
int gauss() {
    int c, r; //当前列,行
    // 1、枚举系数每一列
    for (c = 0, r = 0; c < n; c++) {
        // 2、找出系数最大的行
        int t = r;
        for (int i = r; i < n; i++) //行
            if (abs(a[i][c]) > abs(a[t][c])) t = i;
        // 3、最大系数为0，直接下一列
        if (abs(a[t][c]) < eps) continue;
        // 4、交换
        if (r != t) // POJ中，如果二维数组，直接swap(a[t],a[r])会报编译错误，没办法，只好用了循环
            for (int i = 0; i < N; i++) swap(a[t][i], a[r][i]);
        // 5、倒序，每项除a[r][c],化系数为1,处理的是方程左右两端，需要带着a[r][n]
        for (int i = n; i >= c; i--) a[r][i] /= a[r][c];
        // 6、用当前行将下面所有的列消成0
        for (int i = r + 1; i < n; i++)
            for (int j = n; j >= c; j--)
                a[i][j] -= a[r][j] * a[i][c];
        // 7、下一行
        r++;
    }

    if (r < n) {
        for (int i = r; i < n; i++)
            if (abs(a[i][n]) > eps)
                return -1; //无解
        return n - r;      //自由元个数
    }

    //倒三角，将已知解代入
    for (int i = n - 2; i >= 0; i--)
        for (int j = i + 1; j < n; j++)
            a[i][n] -= a[i][j] * a[j][n];

    //唯一解：0
    return 0;
}

int main() {
    int T;
    cin >> T;
    while (T--) {
        cin >> n;
        memset(a, 0, sizeof(a));
        memset(start, 0, sizeof(start)); //初始化开始状态
        memset(stop, 0, sizeof(stop));   //初始化终止状态
        //输入起始状态(下标从0开始)
        for (int i = 0; i < n; i++) cin >> start[i];

        //输入终止状态(下标从0开始)
        for (int i = 0; i < n; i++) cin >> stop[i];

        //输入增广矩阵
        int x, y;
        while (cin >> x >> y && x != 0 && y != 0)
            a[y - 1][x - 1] = 1; //反着存入y-1受x-1影响

        for (int i = 0; i < n; i++) {
            a[i][n] = start[i] ^ stop[i]; //状态变化 start^stop
            a[i][i] = 1;                  //自己影响自己
        }

        //高斯消元模板
        int t = gauss();

        if (t == -1)
```

```
            cout << "Oh,it's impossible~!!" << endl;
        else
            cout << (1 << t) << endl;
    }
    return 0;
}
```

poj1830 poj 1222 hdu 5755