高斯消元------------终极攻略　　　　　　　　　　林大 chenyu

高消一直是 ACM 中高层次经常用到的算法，虽然线性代数已经学过，但高消求解的问题模型及高消模板的应用变化是高消的最复杂之处。

先介绍一下高消的基本原理：引入互联网 czyuan 的帖子：

高斯消元法，是线性代数中的一个算法，可用来求解线性方程组，并可以求出矩阵的秩，以及求出可逆方阵的逆矩阵。

高斯消元法的原理是：

若用初等行变换将增广矩阵 化为 ，则 AX = B 与 CX = D 是同解方程组。

所以我们可以用初等行变换把增广矩阵转换为行阶梯阵，然后回代求出方程的解。

以上是线性代数课的回顾，下面来说说高斯消元法在编程中的应用。

首先，先介绍程序中高斯消元法的步骤：

(我们设方程组中方程的个数为 equ，变元的个数为 var，注意：一般情况下是 n 个方程，n 个变元，但是有些题目就故意让方程数与变元数不同)

1. 把方程组转换成增广矩阵。

2. 利用初等行变换来把增广矩阵转换成行阶梯阵。

枚举 k 从 0 到 equ – 1，当前处理的列为 col(初始为 0) ，每次找第 k 行以下(包括第 k 行)，col 列中元素绝对值最大的列与第 k 行交换。如果 col 列中的元素全为 0，那么则处理 col + 1 列，k 不变。

3. 转换为行阶梯阵，判断解的情况。

① 无解

当方程中出现(0, 0, …, 0, a)的形式，且 a != 0 时，说明是无解的。

② 唯一解

条件是 k = equ，即行阶梯阵形成了严格的上三角阵。利用回代逐一求出解集。

③ 无穷解。

条件是 k < equ，即不能形成严格的上三角形，自由变元的个数即为 equ – k，但有些题目要求判断哪些变元是不缺定的。

这里单独介绍下这种解法：

首先，自由变元有 var - k 个，即不确定的变元至少有 var - k 个。我们先把所有的变元视为不确定的。在每个方程中判断不确定变元的个数，如果大于 1 个，则该方程无法求解。如果只有 1 个变元，那么该变元即可求出，即为确定变元。

以上介绍的是求解整数线性方程组的求法，复杂度是 O(n3)。浮点数线性方程组的求法类似，但是要在判断是否为 0 时，加入 EPS，以消除精度问题。
以上 czyuan 帖子的基本原理就介绍完了。

-------------------------我对上面的步骤做一下说明--

高斯消元求解的详细步骤：

1） 不用说了，对增广矩阵先消元，化为行阶梯矩阵。

很多人认为这就行了，但有时存在如下情况：

1	2	3	4	3
0	1	2	3	3
0	0	0	1	2
0	0	0	0	0

消元后的矩阵化为行阶梯，上面主对角线元素的主元并没有连续，还应将第 4 列和第 3 列交换才行。

这是网络上很多的模板所没有的，也是造成关键时候出错的根源所在。

所以模板还要增加对列进行检查的代码。

2） 消元完成后，判断是否有解，分三种，无解，有 1 个解和有无穷多解。

3） 对于有唯一解和有无穷多解的时候，都要回带。

啥是回带？ 详细说明之。

```
1   2    3   4   3              1   2   4   3    3
0   1    2   3   3              0   1   3   2    3
0   0    0   1   2--------》    0   0   1   0    2 ------》消元后的矩阵------发现有无穷解
0   0    0   0   0              0   0   0   0    0
（行阶梯）                      （交换列）
```

因为 4 个变量，才 3 行，4-3=1> 0 ,有无穷多解啊。

如果第 4 行不都是 0，回带可以求出这个唯一解。如下：

X1	x2	x3	x4	b1
0	x2	x3	x4	b2
0	0	x3	x4	b3
0	0	0	x4	b4

最后 1 行，有 x4=b4;
第 3 行： x4+x3=b3,已经求出 x4=b4 了，带入第 3 行，可求出 x3;同理，把 x4 x3 带入第 2 行，还可求出 x2;把 x4 x3 x2 带入到第 1 行，可求出 x1;
这就是回带。
回带总结：从最后 1 行，逐一往回带，从最后 1 行代回到第 1 行。

最关键的时候到了：当无穷多解时，最后几行都是 0；没法回带；而某些题目在无穷多解时还要你求最小或最优解，没办法，就得枚举最后行为 0 的那几个解；
如：

1	2	4	3	3
0	1	3	2	3
0	0	1	0	2
0	0	0	0	0

可见最后的 x4 没法解，如果题中给定 x4 的范围，那就枚举 x4，然后回带；枚举 1 次 x4 就回带 1 次得到一组（x1 x2 x3 x4 ）;然后根据题意找最优的，具体题目具体分析。
说得够详细的了，下面拿出有效的高消模板：
----------------------------------再次复习过程--
确定系数矩阵和增广矩阵
消元
If 如果有系数矩阵=0，增广矩阵不为 0，则无解；
{
例： 0 0 0 0 2
有这样的行就没解，0*x1+0*x2+0*x3+0*x4=2;没这样的 x

}
If (var-k>0) //有多个解//var 变量数，k 是主对角线上连续 1 的个数
{
枚举最后几行（全为 0） 的那几个解；枚举或 dfs 都行
回带可以求出很多组解；
}
If (var-k==0) //唯一解
{
直接回带
}
----------------------------复习完成--

----------------------------------下为模版--

```
int a[maxn][maxn+1],x[maxn];//a 是系数矩阵和增广矩阵，x 是最后存放的解
// a[][maxn]中存放的是方程右面的值（bi）
int equ,var;//equ 是系数阵的行数，var 是系数矩阵的列数（变量的个数）
int free_num,ans=100000000;
int abs1(int num)    //取绝对值
{
        if (num>=0) return num;
        else
        return -1*num;



}
void Debug(void)    //调试输出，看消元后的矩阵值，提交时，就不用了
{
        int i, j;
        for (i = 0; i < equ; i++)
        {
                for (j = 0; j < var + 1; j++)
                {
                        cout << a[i][j] << " ";
                }
                cout << endl;
        }
        cout << endl;
}

inline int gcd(int a, int b) //最大公约数
{
        int t;
        while (b != 0)
        {
                t = b;
                b = a % b;
                a = t;
        }
        return a;
```

```
}

inline int lcm(int a, int b)    //最小公倍数
{
        return a * b / gcd(a, b);
}
void swap(int &a,int &b){int temp=a;a=b;b=temp;}    //交换 2 个数
int Gauss()
{
        int k,col = 0;    //当前处理的列
        for(k = 0;k < equ && col < var;++k,++col)
        {
                int max_r = k;
                for(int i = k+1;i < equ; ++i)
                        if(a[i][col] > a[max_r][col])
                                max_r = i;
                if(max_r != k)
                {
                        for(int i = k;i < var + 1; ++i)
                                swap(a[k][i],a[max_r][i]);
                }
                if(a[k][col] == 0)
                {
                        k--;
                        continue;
                }
                for(int i = k+1;i < equ; ++i)
                {
                        if(a[i][col] != 0)
                        {
                                int LCM = lcm(a[i][col],a[k][col]);
                                int ta = LCM/a[i][col], tb = LCM/a[k][col];
                                if(a[i][col]*a[k][col] < 0)
                                        tb = -tb;
                                for(int j = col;j < var + 1; ++j)
                                        a[i][j] = ( (a[i][j]*ta)%2 - (a[k][j]*tb)%2    +    2 ) % 2;        //a[i][j]只有
0 和 1 两种状态
                        }
```

```
            }
        }
        //上述代码是消元的过程，行消元完成
//解下来 2 行，判断是否无解
//注意 K 的值，k 代表系数矩阵值都为 0 的那些行的第 1 行
for(int i = k;i < equ; ++i)
            if(a[i][col] != 0)      return -1;    //  无解返回  -1
//Debug();
        //唯一解或者无穷解,k<=var
        //var-k==0    唯一解；var-k>0  无穷多解，自由解的个数=var-k
      //能执行到这，说明肯定有解了，无非是 1 个和无穷解的问题。
        //下面这几行很重要，保证秩内每行主元非 0，且按对角线顺序排列，就是检查列
            for(int i = 0; i <equ; ++i)//每一行主元素化为非零
          if(!a[i][i])
          {
              int j;
              for(j = i+1;j<var;++j)
                  if(a[i][j])
                      break;
               if(j == var)
                   break;
               for(int k = 0;k < equ; ++k)
                   swap(a[k][i],a[k][j]);
            }
// ----处理保证对角线主元非 0 且顺序，检查列完成
      // free_num=k;
       if (var-k>0) {
       //无穷多解，先枚举解，然后用下面的回带代码进行回带；
       //这里省略了下面的回带的代码；不管唯一解和无穷解都可以回带，只不过无穷解
//回带时，默认为最后几个自由变元=0 而已。
       }

       if (var-k==0)//唯一解时
         {

         //下面是回带求解代码，当无穷多解时，最后几行为 0 的解默认为 0；
               for(int i = k-1;i >= 0; --i) //从消完元矩阵的主对角线非 0 的最后 1 行，开始往
//回带
```

```
            {
            int tmp = a[i][var] % 2;
            for(int j = i+1;j < var; ++j) //x[i]取决于 x[i+1]--x[var]啊，所以后面的解对前面的解有影响。
                if(a[i][j] != 0)
                    tmp = ( tmp - (a[i][j]*x[j])%2    +    2 ) % 2;
            //if (a[i][i]==0) x[i]=tmp;     //最后的空行时，即无穷解得
            //else
            x[i] = (tmp/a[i][i]) % 2; //上面的正常解
            }
           //回带结束了

  }

}
```

--------------------------对上面模板的说明--

因为我们要解决的问题，基本上都是解都是出于 0-1 这 2 个数，所以要对 2 取余；

如果想对普通的正数解方程，那自己把对 2 取余删掉就行了；

下面要仔细看：

以前讲述的列方程的方法需要考虑连动的情况，主要是列方程的思路，而很多牛人也都是按照，按的这个格，和这个格都翻转的格子，一起构成 1 个方程；如下面的 poj1222 的我 AC 的代码；后来发现有 2 道题要考虑“连动”，而 zoj 的 3353 更是必须要数据 a[i][j] 调换输入才行，而 3353 题中没有“连动”意思；最后和 struggle_mind 讨论，发现下面这个建立方程的办法更有说服力，而且经过验证，所有题目都能 A ，而且不用再考虑“连动”啥的了，是有理论作为支撑的。

对方程的建立方法的说明：　　　还是常规方法，按 1 个格，则它的自己本身和上、下、左和右格子都会发生翻转。黑变白，白能变黑。

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

分析办法： bi 为初始状态；每次按一个格，发生翻转的记为 1，按列排列；而每行表示：该行号的格子在整个一系列过程中被按下的值。 记住：每次按格子，把发生翻转的格子按列来标记为 1；

	按第 1 个格	按第 2 个格	按第 3 个格	按第 4 个格	bi
第 1 个格	1	1	0	0	b1
第 2 个格	1	1	1	0	b2
第 3 个格	0	1	1	1	b3
第 4 个格	0	0	1	1	b4
第 5 个格	1	0	0	0	b5
第 6 个格	0	1	0	0	b6
第 7 个格	0	0	1	0	b7
第 8 个格	0	0	0	1	b8

以上只给出了按前 4 个格的矩阵的记录。

一句话：就是从前的方法中，把 a[i][j],录入时把 i 和 j 的位置调换一下就行；所有题目还可以 AC 的。

下面这几道题要各个搞定才行：

POJ 1222 EXTENDED LIGHTS OUT

http://acm.pku.edu.cn/JudgeOnline/problem?id=1222

POJ 1681 Painter's Problem

http://acm.pku.edu.cn/JudgeOnline/problem?id=1681

POJ 1753 Flip Game

http://acm.pku.edu.cn/JudgeOnline/problem?id=1753

POJ 1830 开关问题

http://acm.pku.edu.cn/JudgeOnline/problem?id=1830

POJ 3185 The Water Bowls

http://acm.pku.edu.cn/JudgeOnline/problem?id=3185

开关窗户，开关灯问题，很典型的求解线性方程组的问题。方程数和变量数均为行数*列数，直接套模板求解即可。但是，当出现无穷解时，需要枚举解的情况，因为无法判断哪种解是题目要求最优的。

POJ 2947 Widget Factory   poj 2065

http://acm.pku.edu.cn/JudgeOnline/problem?id=2947

求解同余方程组问题。与一般求解线性方程组的问题类似，只要在求解过程中加入取余即可。

注意：当方程组唯一解时，求解过程中要保证解在[3, 9]之间。

分下类： 1222 有唯一解   太简单

1753 3185 2965 高斯+枚举，我要详细讲（黑板上）

2947　2065　同余方程

先说 1222 这题：这题的代码是老方法的，新方法把 a[i][j]录入时调换位置；

题目大意：给你一个 5*6 的矩阵，矩阵里每一个单元都有一个灯和一个开关，如果按下此开关，那么开关所在位置的那个灯和开关前后左右的灯的状态都会改变(即由亮到不亮或由不亮到亮)。给你一个初始的灯的状态，问怎样控制每一个开关使得所有的灯最后全部熄灭（此题保证有唯一解）。

解题思路：高斯消元。很显然每个灯最多只需要按 1 下（因为按两下和没有按是一个效果）。我们可以定义 30 和未知数 x0、x1.......x29 代表每一个位置的开关是否被按。那么对于每一个灯的状态可以列一个方程，假设位置(i,j)处的开关为 x(i*6+j),那么我们就可以列出方程:
x(i*6+j)+x((i-1)*6+j)+x((i+1)*6+j)+x(i*6+j-1)+x(i*6+j+1) = bo(mod 2)
(括号里的数字为 x 的下标，这里假设这些下标都是符合要求的，即都在矩形内，如果不在则可以去掉，当这个灯初始时是开着的，那么 bo 为 1，否则为 0)
这样可以列出 30 个方程，然后用高斯消元解这个方程组即可。

我的补充：题目给定后，我立刻知道了该题，呵呵，有唯一解；

怎么判断出的呢？　5 行 6 列，已经固定了；每个灯的方程也固定了（系数矩阵）；只有开始时灯的状态会变，开始时灯的状态只能决定增广矩阵的最后 1 列；因为系数矩阵式固定的，而系数矩阵能决定该方程是否有唯一解和多个解；编好代码后，自己 deBUG 一下，也就是把 a 这个矩阵输出一下，自己 look 一下，发现没有都是 0 的行，秩=5；

所以模板里只要算一下　(var-k==0)是，回带就行了；

```
#include <iostream>
#include <cstring>
#include <cmath>
#include <stdio.h>
using namespace std;
```

```
const int maxn = 30;

int equ, var; // 有 equ 个方程，var 个变元。增广阵行数为 equ, 分别为 0 到 equ - 1，列数为
var + 1，分别为 0 到 var.
int a[maxn][maxn+1];
int x[maxn]; // 解集.
bool free_x[maxn+1]; // 判断是否是不确定的变元.
int free_num;
int abs1(int num)
{
	if (num>=0) return num;
	else
	return -1*num;



}
void Debug(void)
{
	int i, j;
	for (i = 0; i < equ; i++)
	{
		for (j = 0; j < var + 1; j++)
		{
			cout << a[i][j] << " ";
		}
		cout << endl;
	}
	cout << endl;
}

inline int gcd(int a, int b)
{
	int t;
	while (b != 0)
	{
		t = b;
		b = a % b;
		a = t;
```

```
    }
    return a;
}

inline int lcm(int a, int b)
{
    return a * b / gcd(a, b);
}
  void swap(int &a,int &b){int temp=a;a=b;b=temp;}
int Gauss_new()
{
    int k,col = 0;   //当前处理的列
    for(k = 0;k < equ && col < var;++k,++col)
    {
        int max_r = k;
        for(int i = k+1;i < equ; ++i)
            if(a[i][col] > a[max_r][col])
                max_r = i;
        if(max_r != k)
        {
            for(int i = k;i < var + 1; ++i)
                swap(a[k][i],a[max_r][i]);
        }
        if(a[k][col] == 0)
        {
            k--;
            continue;
        }
        for(int i = k+1;i < equ; ++i)
        {
            if(a[i][col] != 0)
            {
                int LCM = lcm(a[i][col],a[k][col]);
                int ta = LCM/a[i][col], tb = LCM/a[k][col];
                if(a[i][col]*a[k][col] < 0)
                    tb = -tb;
                for(int j = col;j < var + 1; ++j)
                    a[i][j] = ( (a[i][j]*ta)%2 - (a[k][j]*tb)%2   +   2 ) % 2;      //a[i][j]只有0和
```

1 两种状态

```
                }
            }
        }
        for(int i = k;i < equ; ++i)
            if(a[i][col] != 0)       return -1;      //  无解返回  -1

        //唯一解或者无穷解,k<=var，这里直接回带了，知道有唯一解啊
        for(int i = k-1;i >= 0; --i)
        {
            int tmp = a[i][var] % 2;
            for(int j = i+1;j < var; ++j)
                if(a[i][j] != 0)
                    tmp = ( tmp - (a[i][j]*x[j])%2    +    2 ) % 2;
            x[i] = (tmp/a[i][i]) % 2;
        }
        return 0;
}


int main(void)
{
      // freopen("Input.txt", "r", stdin);
        int i, j,t,t1;
        cin>>t;
        t1=t;
        equ=30;
        var=30;
        while (t--)
        {
            memset(a, 0, sizeof(a));
      memset(x, 0, sizeof(x));
      //memset(free_x, 1, sizeof(free_x)); //  一开始全是不确定的变元.
            //下面要根据位置计算 a[i][j];



            for (i = 0; i < 5; i++)
```

```
        {
            for (j = 0; j < 6; j++)
            {
              /* for(int k=0;k<4;k++)
                {
                    int ni=i+di[k];
                    int nj=j+dj[k];
                    if(inlim(ni,nj))
                    {
                        a[i*6+j][ni*6+nj]=1;
                    }
                }
                */
              //下面要把 a[i][j]的位置调换一下更科学，我代码没调换啊 ， 也能 A
              if (i-1>=0) a[i*6+j][(i-1)*6+j]=1; //计算上面的位置，应为 a[(i-1)*6+j][i*6+j]=1;
              if (i+1<=4) a[i*6+j][(i+1)*6+j]=1;//计算下面的位置
              if (j-1>=0)    a[i*6+j][i*6+j-1]=1;//计算左面的位置
              if    (j+1<=5)    a[i*6+j][i*6+j+1]=1; //计算右面的位置
               a[i*6+j][i*6+j]=1;//别忘了计算自己
               cin>>a[i*6+j][30];

                //scanf("%d", &a[i][j]);
            }
        }
//        Debug
//free_num = Gauss();
free_num=Gauss_new();
        if (free_num == -1) printf("无解!\n");
          else if (free_num >= 0)
        {

            int na_num=0;
            printf("PUZZLE #%d\n",t1-t);
            for (i = 0; i < var; i++)
            {
                na_num++;
                if (na_num%6==0) {printf("%d\n",x[i]);}
                else
```

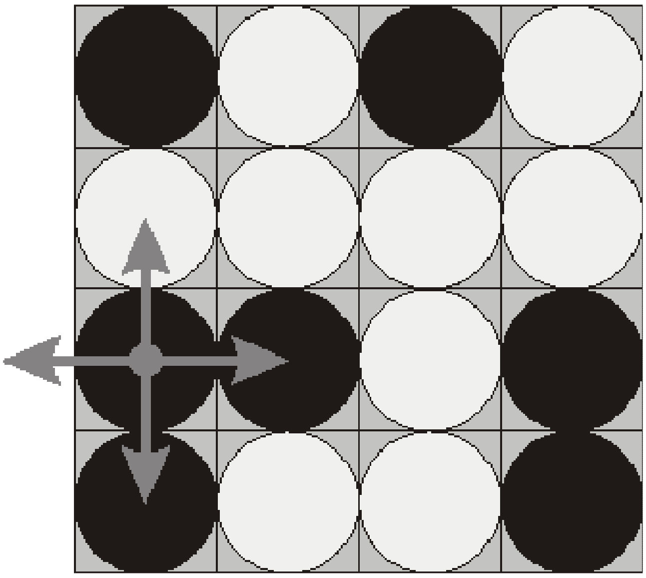
```
                    printf("%d ",x[i]);
                }
            }
          // printf("\n");
      }
      return 0;
}
```

-----------------------------------1222-----------over---

重点说下 1753 这题：本题用 guass 还是比较牛的，速度很快啊

题意：

有 4*4 的正方形，每个格子要么是黑色，要么是白色，当把一个格子的颜色改变(黑->白或者白->黑)时，其周围上下左右(如果存在的话)的格子的颜色也被反转，问至少反转几个格子可以使 4*4 的正方形变为纯白或者纯黑？



本题求都变白或变黑的最小格子数。　也就是初值取反一下（黑变白）

分析：求都变白的最小格子数；再求变黑的最小格子数，输出 2 者最小值即可；

这里说下本类问题的 bi 的赋值，bi 就是曾广矩阵的最后 1 列；

其实就是开始的矩阵（0-1），和最终矩阵的（0-1）做异或；相同的格子为 0，不同的格子值为 1；这类问题都一样的；

本体 4*4 的矩阵固定，方法固定，只是初始状态不固定，还是先把代码先敲进去，用样例输入，用 debug()把消元后的 a 的结果输出到屏幕上看看，看看有几个解是未知的，共 16 个

方程：

1100100000000000

0111001000000000

0010110000000000

0001110100000000

0000110101000000

0000011110000000

0000001101010000

0000000111100000

0000000010111000

0000000001111010

0000000000110100

0000000000010011

0000000000000000

0000000000000000

0000000000000000

0000000000000000

最后 4 行都是 0，则最后 4 个解变量是无穷解；

X15　x14　x13　x12 只能取 0 或 1，枚举 2^4 共 16 中情况，枚举 1 次，回带 1 次，算出 1 组解，记录解变量和最小的。

0 000　0001　0010　0011 ----------1111 共 16 个，说得太详细了，枚举可以循环

For (i=1----16)

{

0 0 0 0　// 每次根据 i 值，产生 0 0 0 0　还是　00 10　，就是 x15 x14　x13　x12 的值

回带

得到解

和是否最小

}

这是 16 种情况，要是更多，还是用 dfs()吧，也简单的要命！

--------------------------------下位直接枚举的代码---超短--------------------------------

- for(i=0,min=17;i<16;++i){　　　　　　　　　　//枚举有解的 16 种情况
- 　　for(j=15;j>11;--j)r[j]=(1<<(j-12))&i?1:0; //最后 4 变量是自由变量

仔细体会上面这 2 句，如何产生 0 0 0　0　----1111　的

----------------------------------高消+枚举-------poj1753--

```
#include <iostream>
#include <cstring>
#include <stdio.h>
using namespace std;
const int maxn=16;
int a[maxn][maxn+1],x[maxn],b[maxn][maxn+1];
int equ,var;
bool free_x[maxn+1]; // 判断是否是不确定的变元.
int free_num,ans=100000000;
int abs1(int num)
{
    if (num>=0) return num;
    else
    return -1*num;



}
void Debug(void)
{
    int i, j;
    for (i = 0; i < equ; i++)
    {
        for (j = 0; j < var + 1; j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}

inline int gcd(int a, int b)
{
    int t;
    while (b != 0)
    {
        t = b;
        b = a % b;
```

```
            a = t;
        }
        return a;
}

inline int lcm(int a, int b)
{
        return a * b / gcd(a, b);
}
int    dfs(int p)    //枚举自由解，只能取 0-1，枚举完就回带，找到最小的
{
if (p<=free_num-1) //深入到了主对角线元素非 0 的行了
{    //下面就是回带的代码啊
    for(int i = free_num-1;i >= 0; --i)
            {
            int tmp = a[i][var] % 2;
            for(int j = i+1;j < var; ++j) //x[i]取决于 x[i+1]--x[var]啊，所以后面的解对前面的解有影
响。
                    if(a[i][j] != 0)
                            tmp = ( tmp - (a[i][j]*x[j])%2    +    2 ) % 2;

            x[i] = (tmp/a[i][i]) % 2; //上面的正常解
            } //回带完成了
          //计算解元素为 1 的个数；
            int sum=0;
            for(int i=0;i<var;i++) sum+=x[i];
            if (ans>sum) ans=sum;
    return 0;
}
x[p]=0;dfs(p-1);
x[p]=1;dfs(p-1);

}
void swap(int &a,int &b){int temp=a;a=b;b=temp;}
int Gauss_1()
{
        int k,col = 0;    //当前处理的列
        for(k = 0;k < equ && col < var;++k,++col)
```

```
    {
        int max_r = k;
        for(int i = k+1;i < equ; ++i)
            if(a[i][col] > a[max_r][col])
                max_r = i;
        if(max_r != k)
        {
            for(int i = k;i < var + 1; ++i)
                swap(a[k][i],a[max_r][i]);
        }
        if(a[k][col] == 0)
        {
            k--;
            continue;
        }
        for(int i = k+1;i < equ; ++i)
        {
            if(a[i][col] != 0)
            {
                int LCM = lcm(a[i][col],a[k][col]);
                int ta = LCM/a[i][col], tb = LCM/a[k][col];
                if(a[i][col]*a[k][col] < 0)
                    tb = -tb;
                for(int j = col;j < var + 1; ++j)
                    a[i][j] = ( (a[i][j]*ta)%2 - (a[k][j]*tb)%2   +   2 ) % 2;      //a[i][j]只有0和
1两种状态
            }
        }
    }
    for(int i = k;i < equ; ++i)
        if(a[i][col] != 0)      return -1;     //  无解返回  -1
//上述代码是消元的过程，消元完成
//Debug();
    //唯一解或者无穷解,k<=var
    //var-k==0   唯一解；var-k>0  无穷多解，自由解的个数=var-k

    //下面这几行很重要，保证秩内每行主元非0，且按对角线顺序排列
        for(int i = 0; i <equ; ++i)//每一行主元素化为非零
```

```
            if(!a[i][i])
            {
                int j;
                for(j = i+1;j<var;++j)
                    if(a[i][j])
                        break;
                 if(j == var)
                     break;
                 for(int k = 0;k < equ; ++k)
                     swap(a[k][i],a[k][j]);
             }
// ----处理保证对角线主元非 0 且顺序，完成



        free_num=k;
        if (var-k>0) { dfs(var-1); return ans;}

        if  (var-k==0)//唯一解时，poj1753 本题没唯一解，当题目具体操作给出后，系数矩阵已
经固定了！
  {

        //下面是回带求解，当无穷多解时，最后几行为 0
                 for(int i = k-1;i >= 0; --i)
             {
             int tmp = a[i][var] % 2;
             for(int j = i+1;j < var; ++j) //x[i]取决于 x[i+1]--x[var]啊，所以后面的解对前面的解有影
响。
                 if(a[i][j] != 0)
                     tmp = ( tmp - (a[i][j]*x[j])%2    +    2 ) % 2;
             //if (a[i][i]==0) x[i]=tmp;      //最后的空行时，即无穷解得
             //else
             x[i] = (tmp/a[i][i]) % 2; //上面的正常解
             }
             int sum=0;
             for(int i=0;i<var;i++) sum+=x[i];
             return sum;
```

```
    }

}

int main()
{
int k,free_num;
char c1[20];
        memset(a,0,sizeof(a));
        memset(x,0,sizeof(x));
        ans=1000000000;
        for (int i=0;i<4;i++)
        {
                cin>>c1;
                        //构造系数矩阵 A
                for (int j=0;j<4;j++)
                {
                        k = 4*i+j;
                        a[k][k]=1;
                        if (i>0) a[k][k-4]=1; //上， 也是 a[i][j]  没调换位置的，调换更科学,a[k-4][k]=1
                        if (i<3)            a[k][k+4]=1;            //下，   a[k+4][k]=1
                        if (j>0)            a[k][k-1]=1;            //左       a[k-1][k]=1
                        if (j<3)            a[k][k+1]=1;            //右       a[k+1][k]=1
                        if (c1[j]=='b')
                        a[k][maxn] = 1;
                }
        }
        for(int i=0;i<16;i++)
        for(int j=0;j<=16;j++)
        b[i][j]=a[i][j];
        for(int k=0;k<16;k++)
        b[k][16]^=1;
equ=var=16;
int j1=Gauss_1();
int min1=ans;

for(int i=0;i<16;i++)
        for(int j=0;j<=16;j++)
```

```
        a[i][j]=b[i][j];
ans=100000000;
int j2=Gauss_1();
int min2=ans;
    if (j1==-1&&j2==-1) cout<<"Impossible"<<endl;
    else
cout<<min(ans,min1)<<endl;
      // cout << "Hello world!" << endl;
        return 0;
}
```

对于 2947 和 2065 我是用下面的模板过的

下是高斯消元+同余方程的模板

```
#define maxn 80
int equ,var,prime;
char st[80];
int aa[maxn][maxn],x[maxn];

 inline int abs1(int x)
 {

        if (x>=0) return x;
        else
        return -1*x;

 }
 inline int gcd(int a, int b)
  {
          int t;
          while (b != 0)
          {
                 t = b;
                 b = a % b;
               a = t;
```

```
    }
    return a;
}

inline int lcm(int a, int b)
{
    return a * b / gcd(a, b);
}

 int extgcd(int a, int b, int & x, int & y){
    if (b == 0) { x=1; y=0; return a; }
    int d = extgcd(b, a % b, x, y);
    int t = x; x = y; y = t - a / b * y;
    return d;
}

 int Gauss(){
    int i,j,k;
    int max_r , col , temp;
    int LCM , GCD;
    int ta,tb;
    col = 0;
    for(k=0 ; k<equ && col < var ; k++,col++)
    {
        max_r = k;
        for(i=k+1 ; i<equ ; i++)
        {    if(abs1(aa[i][col]) > abs1(aa[max_r][col]))max_r = i;}
        if(max_r != k)
        {    for(j=k ; j<var+1 ; j++)swap(aa[k][j],aa[max_r][j]);}
        if(aa[k][col] == 0)
        {    k--;continue;
        }

        for(i=k+1 ; i<equ ; i++)
        {
```

```
                if(aa[i][col] != 0)
                {
                    LCM = lcm(abs1(aa[i][col]) , abs1(aa[k][col]));
                    ta = LCM/abs1(aa[i][col]) ;
                    tb = LCM/abs1(aa[k][col]);
                    if(aa[i][col] * aa[k][col] < 0)tb = -tb;
                    for(j=col ; j<var+1 ; j++)
                    {       aa[i][j] = (aa[i][j]*ta-aa[k][j]*tb) % prime;
                    }
                }
            }//for
        }


    for(i=var-1 ; i>=0 ; i--)
    {
        temp = aa[i][var];
        for(j=i+1 ; j<var ; j++)
        {       if(aa[i][j] != 0)temp =(temp - aa[i][j]*x[j]%prime) ;
        }
        temp = (temp%prime + prime) % prime;
        GCD = extgcd(aa[i][i] , prime , x[i] , k);
        x[i] = ( (x[i]*(temp/GCD) % prime) + prime) % prime;
    }
    return 0;
}
```

用以上的代码　很快把 poj 2065 A 掉了

POJ 2947 高斯消元　　解题报告

题目大意：有 n 种装饰物，m 个已知条件,每个已知条件的描述如下：

p start end

a1,a2......ap (1<=ai<=n)

第一行表示从星期 start 到星期 end 一共生产了 p 件装饰物(工作的天数为 end-start+1+7*x,

加 7*x 是因为它可能生产很多周)，第二行表示这 p 件装饰物的种类(可能出现相同的种类，即 ai=aj)。规定每件装饰物至少生产 3 天，最多生产 9 天。问每种装饰物需要生产的天数。如果没有解，则输出“Inconsistent data.”，如果有多解，则输出“Multiple solutions.”，如果只有唯一解，则输出每种装饰物需要生产的天数。

解题思路：高斯消元。设每种装饰物需要生产的天数为 xi(1<=i<=n)。每一个条件就相当于给定了一个方程式，假设生产 1 类装饰物 a1 件、2 类装饰物 a2 件、i 类装饰物 ai 件所花费的天数为 b，则可以列出下列方程：

a1*x1+a2*x2+...an*xn = b (mod 7)

这样一共可以列出 m 个方程式，然后使用高斯消元来解此方程组即可。

```
#include <iostream>
#include <cstring>
#include <stdio.h>
using namespace std;
#define maxn 305
int equ,var,prime;
//char st[80];
char data[7][5]={"MON","TUE","WED","THU","FRI","SAT","SUN"};
int aa[maxn][maxn],x[maxn];

 inline int abs1(int x)
 {

     if (x>=0) return x;
     else
     return -1*x;

 }
 inline int gcd(int a, int b)
  {
      int t;
      while (b != 0)
      {
          t = b;
```

```
         b = a % b;
        a = t;
    }
    return a;
}

inline int lcm(int a, int b)
{
    return a * b / gcd(a, b);
}

 int extgcd(int a, int b, int & x, int & y){
    if (b == 0) { x=1; y=0; return a; }
    int d = extgcd(b, a % b, x, y);
    int t = x; x = y; y = t - a / b * y;
    return d;
}

 int Gauss(){
    int i,j,k;
    int max_r , col , temp;
    int LCM , GCD;
    int ta,tb;
    col = 0;
    for(k=0 ; k<equ && col < var ; k++,col++)
    {
        max_r = k;
        for(i=k+1 ; i<equ ; i++)
        {    if(abs1(aa[i][col]) > abs1(aa[max_r][col]))max_r = i;}
        if(max_r != k)
        {    for(j=k ; j<var+1 ; j++)swap(aa[k][j],aa[max_r][j]);}
        if(aa[k][col] == 0)
        {    k--;continue;
        }
```

```
            for(i=k+1 ; i<equ ; i++)
            {
                if(aa[i][col] != 0)
                {
                    LCM = lcm(abs1(aa[i][col]) , abs1(aa[k][col]));
                    ta = LCM/abs1(aa[i][col]) ;
                    tb = LCM/abs1(aa[k][col]);
                    if(aa[i][col] * aa[k][col] < 0)tb = -tb;
                    for(j=col ; j<var+1 ; j++)
                    {       aa[i][j] = (aa[i][j]*ta-aa[k][j]*tb) % prime;
                            aa[i][j]=(aa[i][j]%7+7)%7;
                    }
                }
            }//for
        }
        for(i=k;i<equ;i++)
        {
            if (aa[i][col]!=0) return -1;//无解
        }

        if (k<var)    return var-k;// 无穷多解

       for(i=var-1 ; i>=0 ; i--)
       {
            temp = aa[i][var];
            for(j=i+1 ; j<var ; j++)
            {       if(aa[i][j] != 0) temp =(temp - aa[i][j]*x[j]%prime) ;
                //temp=temp-aa[i][j]*x[j];
            }
            temp = (temp%prime + prime) % prime;
            GCD = extgcd(aa[i][i] , prime , x[i] , k);
            x[i] = ( (x[i]*(temp/GCD) % prime) + prime) % prime;

          // while(temp%aa[i][i])
// temp+=7;
```

```
  //x[i]=temp/aa[/i][i];


            //x[i]=(x[i]%7+7)%7;
            while(x[i]<3) x[i]=x[i]+7;
            //while(x[i]>9) x[i]=x[i]-7;
        }
        return 0;
 }

//void init()
//{
 // int n=var;


//}
int comp_1(char a[])
{
    int j=0;
    for(int i=0;i<7;i++)
    if (strcmp(a,data[i])==0) j=i;
    cout<<"s="<<j<<endl;
    return j;

}

int main()
{
    //`MON', `TUE', `WED', `THU', `FRI', `SAT' and `SUN'.
    int m,n,k,typ;
    int num1,num2;
    char s1[5],s2[5];
    prime=7;
    while(scanf("%d%d",&m,&n))
    {
```

```
        if (m==0&&n==0) break;
        var=m; //这个要注意,var!=equ 啊
        equ=n;
        //var=equ=m;
        memset(aa,0,sizeof(aa));
        memset(x,0,sizeof(x));
        for(int i=0;i<n;i++)
        {
            scanf("%d",&k);
            scanf("%s %s",s1,s2);
            for(int i=0;i<7;i++)
            {
                if (strcmp(s1,data[i])==0)
                num1=i;
                if (strcmp(s2,data[i])==0)
                num2=i;
            }

            int day=(num2-num1+1+7)%7;
           // cout<<day<<endl;

            for(int j=0;j<k;j++)
            {
                scanf("%d",&typ);
                aa[i][typ-1]++;
                aa[i][typ-1]%=7;
            }

            aa[i][m]=day;
        }
    int free_num=Gauss();
    if (free_num==-1)    cout<<"Inconsistent data."<<endl;
    if (free_num>0) cout<<"Multiple solutions."<<endl;
    if (free_num==0)
     {
```

```
            for(int i=0;i<var-1;i++)
            cout<<x[i]<<" ";
            cout<<x[var-1]<<endl;


        }
      }


      //cout << "Hello world!" << endl;
      return 0;
}
//本取余模板很好用，注意参数别犯低级错误
```

还有要补充的：

对于像 1830 这样“连动“的开关的问题，采用逆向思维，以最终的灯列方程，不能以开始的灯列方程；就是 a[i]j[]的系数矩阵录入时，实际录入的是 a[j][i],就行了。

啥叫连动，就是 1 个灯亮和不亮由很多灯控制，具体体会 1830 题的文字描述，最后能把 HDU 的 3364 和 4200AC 掉，就算掌握“连动“了。

注：现在基本不用考虑“连动”了，按列来列方程就行了；也可以说是每个题都按“连动”来处理（a[i][j]中，初始化时 i 和 j 互换).

做一下 zoj 的 3353 吧!

作业：把这 9 个题都搞定，并尝试 同余的模板能用开始的那个模板来自己改写吗？

终极目标：举 1 反 3，掌握求解这类问题的本质，对于打亚洲赛的同学，要会使用 double 型的高斯消元（矩阵值是整数，但解是双精度的），同时要能使用高斯+高精度 AC 掉 2008 亚洲哈尔滨的那道纯高消题，这是终极目标。

高消的考试 1 周后进行！4 道简单题。

陈宇 2012/8/31

可以参考宋忠平同学的博客

http://blog.csdn.net/struggle_mind/article/details/7932371