等比数列

等比数列求和公式

$$S_n = n \times a_1 \quad (q = 1)$$

$$S_n = a_1 \cdot \frac{1 - q^n}{1 - q} = \frac{a_1 - a_n \cdot q}{1 - q} \quad (q \neq 1)$$

- 等比数列求和

$∑(1+q^1+q^2+...+q^n)modp$ 已知q,p,n都很大可以到10^9，且q,p不一定互质而且p不一定为质数

- 分治

  (1) 1+q1+q2+...+qn fun(n,q)

```
long long fun(long long n, int f)
{
    if (n == 0) return 1;
    if (n & 1) return ((1 + qPow(f, n / 2 + 1)) % mod*fun(n/2, f) % mod) % mod;
    else
        return ((1 + qPow(f, n / 2 + 1)) % mod*fun((n - 1) / 2,f) % mod + qPow(f, n / 2) % mod) % mod;
}
```

  (2) q1+q2+...+qn (fun(n,q)%p+1%p)%p

```
long long fun(long long n,int f)
{
    if (n == 1)return f;
    if (n & 1)
    {
        return ((fun(n / 2,f) % mod)*(qPow(f, n / 2) + 1) + qPow(f, n) % mod) % mod;
    }
    return ((fun(n / 2,f) % mod)*(qPow(f, n / 2) + 1)) % mod;
}
```

- 求逆元

  $(q^{n+1}-1)/(q-1) modp=((q^{n+1}-1) modp*inv(q-1))modp$

  求逆元的方法：

(1)线性打表法（数组有限）

```cpp
#include<stdio.h>
const int MAXN=100000;
const long long mod=1e9+7;
long long inv[MAXN+10];
void getInv(){
    inv[1]=1;
    for(long long i=2;i<=MAXN;i++)
        inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    return ;
}
```

(2) 费马小定理：p是质数而且q，p互质 $inv(q)=(q^{(p-2)})modp$

(3) 欧拉定理：q，p互质 $inv(q)=q^{φ(p)-1} modp$ 线性打表求欧拉，数组有限

(4) 拓展欧几里得：q，p互质

2.3 扩展欧几里得算法（求 ax+by=gcd 的解以及逆元）

```cpp
//*******************************
//返回 d=gcd(a,b); 和对应于等式 ax+by=d 中的 x,y
long long extend_gcd(long long a,long long b,long long &x,long long
    &y){
    if(a==0&&b==0) return -1;//无最大公约数
    if(b==0){x=1;y=0;return a;}
    long long d=extend_gcd(b,a%b,y,x);
    y-=a/b*x;
```



```cpp
    return d;
}
//********* 求逆元 ******************
//ax = 1(mod n)
long long mod_reverse(long long a,long long n){
    long long x,y;
    long long d=extend_gcd(a,n,x,y);
    if(d==1) return (x%n+n)%n;
    else return -1;
}
```

逆元不存在，即意味着gcd(b,p)!=1：(a/b)modp=(amod(bp))/b

等比数列的逆元不存在：$((q^{n+1}-1)/(q-1)) modp$ 即q-1%p=0 即q%p=1 则等比数列1+q1+q2+...+qn的每一项%p=1，一共有n+1项，所以求余的结果是n+1（限制条件：p一定是质数的时候，当q-1能

整除p可用）

素数

N	π（N）：1到N之间素数的个数	两个相邻素数间隔的平均值
10	4	2.5
100	25	4.0
1,000	168	6.0
10,000	1,229	8.1
100,000	9,592	10.4
1,000,000	78,498	12.7
10,000,000	664,579	15.0
100,000,000	5,761,455	17.4
1,000,000,000	50,847,534	19.7
10,000,000,000	455,052,511	22.0

- 素数分布
- 所有大于2的素数都可以唯一地表示成两个平方数之差。
- 当n为大于2的整数时，2^n+1和2^n-1两个数中，如果其中一个数是素数，那么另一个数一定是合数。
- 孪生素数：3和5等。

素性检测

- 结合哥德巴赫猜想，两个大于2的偶数=两个质数相加(偶数的范围可以达到10^{18}) 当质数很大，难用常规方法判断。

```
typedef unsigned long long ll;
//ll*ll可能会溢出，所以乘法化加法
ll ModMul(ll a,ll b,ll n)//快速积取模 a*b%n
{
    ll ans=0;
    while(b)
    {
        if(b&1)
          ans=(ans+a)%n;
        a=(a+a)%n;
        b>>=1;
    }
    return ans;
}
ll ModExp(ll a,ll b,ll n)//快速幂取模 a^b%n
{
    ll ans=1;
    while(b)
    {
        if(b&1)
          ans=ModMul(ans,a,n);
        a=ModMul(a,a,n);
        b>>=1;
    }
    return ans;
}
bool miller_rabin(ll n)//Miller-Rabin素数检测算法
{
    ll i,j,a,x,y,t,u,s=10;
    if(n==2)
      return true;
    if(n<2||!(n&1))
      return false;
    for(t=0,u=n-1;!(u&1);t++,u>>=1);//n-1=u*2^t
    for(i=0;i<s;i++)
    {
        a=rand()%(n-1)+1;
        x=ModExp(a,u,n);
        for(j=0;j<t;j++)
        {
            y=ModMul(x,x,n);
            if(y==1&&x!=1&&x!=n-1)
              return false;
            x=y;
        }
        if(x!=1)
          return false;
    }
    return true;
}
```

对数的敏感性

- log(2)=0.693147
- $sqrt(2)=1.414$ $sqrt(3)=1.732$
- 被9整除 $k*(10^x-1)$

调和级数 欧拉公式

- 在n较小的时候没有公式，当n较大(大于一万)

  $f(n)=ln(n)+C+1/(2*n)$

  $C≈0.57721566490153286060651209$

```
//对数函数y = lnx的表示方法为log
f=C+log(n)+1.0/(2.0*n);
```

加解密

RSA

- 选取大素数p、q，计算乘积n=p*q
- 根据欧拉函数$r=Phi(n)=Phi(p)*Phi(q)=(p-1)*(q-1)$
- 选择一个小于r并与r互质的整数e，求得e关于r的模反元素，命名为d（ed = 1(mod r)模反元素存在，当且仅当e与r互质），e我们通常取65537。
- e、n公开，d私钥。

  A明文，B密文

  $A=B^d mod n$

  $B=A^e mod n$

质因子分解

- n = $p1^{a1}$ x $p2^{a2}$ x ... x pk^{ak} 其中，pi是n的质因子
- 复杂度：质数等于O(sqrt(n))(优化：先打好素数表，是素数就跳出)，合数小于O(sqrt(n))

```
typedef long long ll;
int fac[maxn],cnt[maxn];
```

```
int getFac(ll n){
        int num=0,sum=0,m=sqrt(n+0.5);
        for(int k=2;k<=m;k++){
                sum=0;
                while(n%k==0){
                        n/=k;sum++;
                }
                if(sum!=0){
                        fac[num]=k;
                        cnt[num++]=sum;
                }
                m=sqrt(n+0.5);
        }
        if(n!=1){
                fac[num]=n;
                cnt[num++]=1;
        }
        return num;
}
```

- 因子总个数 $D = (a1+1) * (a2+1) * ... * (ak+1)$
- 因子之和 $S = Π[1 + pi + pi^2 + ... + pi^{ai}]$
- 阶乘的因子分解：给定正整数n，求n!的因子分解式中质因子p的数量，可以用以下公式求解：$S(p)=Σ[n/p+n/(p^2)+n/(p^3)+...+n/(p^k)]$，其中p^k<=n，时间复杂度为O(log(n))。

GCD & LCM

GCD的性质

- (1.结合律) GCD(a,b,c)=GCD(GCD(a,b),c)
- (2.区间) GCD(al,...,ar)=GCD(GCD(al,...,am-1),GCD(am,...,ar))
- (3.分配律) GCD(k*a*,*k*b)=k*GCD(a,b)
- (4.互质) 若GCD(a,b)=p，则a/p与b/p互质
- (5.线性变换) GCD(a+k*b,b)=GCD(a,b)
- (6.因子分解) GCD(a,b)=$Π[pi^{min(ai,bi)}]$

```
int gcd(int a,int b){
        return b==0?a:gcd(b,a%b);
}
```

LCM的性质

- (1.结合律) LCM(a,b,c)=LCM(LCM(a,b),c)
- (2.分配律) LCM(k*a,k*b)=k*LCM(a,b)
- (3.因子分解) LCM(a,b)=$Π[pi^{max(ai,bi)}]$

欧拉函数

- φ(n)表示所有小于正整数n并且与n互质的正整数的个数
- φ(n) = $Π[pi^{(ai-1)}] * Π[pi-1]$
- (性质 1) 对任意正奇数n，φ(n)=φ(2*n)，特别规定φ(1)=1
- (性质 2) 对任意质数n，φ(n)=n-1，$φ(n^k)=(n-1)*n^{(k-1)}$
- (性质 3) 对于正整数n的所有因子di，有Σ[φ(di)]=n
- 积性性质：若a,b互质，则有φ(a*b)=φ(a)*φ(b)
- 有若p为质数，则$φ(p^k)=p^k-p^{(k-1)}=p^k*(1-1/p)$
- 复杂度：O(sqrt(n))

```
int Euler(int n){
        int ret=n;
        for(int i=2;i<=sqrt(n);i++){
                if(n%i==0){
                        ret=ret/i*(i-1);
                        while(n%i==0)n/=i;
                }
        }
        if(n>1)ret=ret/n*(n-1);
        return ret;
}
```

欧拉函数线性筛

- 复杂度O(n)

```
const int MAXN=1e7,MAXP=1e6;
int prime[MAXP+10],check[MAXN+10],phi[MAXN+10],num=0;
void GetPhi(){
        phi[1]=1;
        memset(check,0,sizeof(check));
        for(int i=2;i<MAXN;i++){
                if(!check[i]){
                        prime[num++]=i;phi[i]=i-1;//素数
                }
                for(int j=0;j<num;j++){
                        if(i*prime[j]>MAXN) break;
                        check[i*prime[j]]=1;
                        if(i%prime[j]==0){ //i mod p = 0, 那么phi(i * p)=p *
phi(i)
                                phi[i*prime[j]]=phi[i]*prime[j];break;
                        }
```

```
                        else{
                                phi[i*prime[j]]=phi[i]*(prime[j]-1);
//phi(prime[j])=prime[j]-1
                        }
                }
        }
        return;
}
```

模运算

快速幂求模

```
long long qPow(long long b,long long p,long long k){
        long long a=b,ans=1;
        while(p){
                if(1&p)ans=(ans*a)%k;
                a=(a*a)%k;
                p>>=1;
        }
        return ans%k;
}
```

- 性质

    1. (a+b)%p=(a%p+b%p)%p
    2. (a-b)%p=(a%p-b%p+p)%p
    3. (a*b)%p=((a%p)*(b%p))%p
    4. (a/b)%p=((a%p)*inv(b))%p，其中inv(b)表示b%p的逆元

- 求正整数a在模mod下逆元的方法

    1. 线性打表法 (只要求mod是质数，时间复杂度O(n))
    2. 费马小定理 (要求mod是质数且与a互质，快速幂优化)
    3. 欧拉定理 (只要求a与mod互质，需要欧拉函数与快速幂)
    4. 拓展欧几里德 (只要求a与mod互质，时间复杂度O(log(n)))

同余方程

线性同余方程

- 线性同余方程：a*x≡b(mod n)
- 拓展欧几里德定理：对于不完全为0的非负整数a和b，必定存在整数对(x,y)，使等式a*x+b*y=gcd(a,b)成立
- 求二元一次不定方程a*x+b*y=c的整数解。

当且仅当c%gcd(a,b)=0时，不定方程存在整数解。

特别注意：若不定方程存在整数解，则整数解有无穷多组，拓展欧几里德只能求出其中的一组解，并且求出的解可能是负的。

通解的求法：若(x0,y0)是线性方程a*x+b*y=c的一组特解，那么对于任意的整数t，(x0+(b/gcd(a,b))*t,y0-(a/gcd(a,b))*t)都是线性方程的解。

```
bool LinearEqu(int a,int b,int c,int &x,int &y){
    int d=exgcd(a,b,x,y);
    if(c%d==0){
        int k=c/d; x*=k; y*=k;
        return true;
    }
    return false;
}
```

- 求线性同余方程a*x≡b(mod n)的最小正整数解。

  解法：首先将方程改写为a*x*-*n*y=b的形式，然后使用拓展欧几里德求出一组特解(x0,y0)。

  题目要求找到最小的正整数解，可以令k=n/gcd(a,n)，这样x的最小正整数解可以通过表达式x=(x0%k+k)%k求出。（x0可能是负数）

```
bool ModularEqu(int a,int b,int m,int &x0){
    int x,y,k;
    int d=exgcd(a,m,x,y);
    if(b%d==0){
        x0=x*(b/d)%m; k=m/d; x0=(x0%k+k)%k;
        return true;
    }
    return false;
}
```

二次同余方程

- Let p = 1000000007. Given two integers b and c, please find two integers x and y(0≤x≤y<p)(0≤x≤y<p), such that (x+y)modp=b 和 (x×y)modp=c 可将题目转化为 x^2-bx+c=0(mod p)求出在模p情况下的二次方程的根

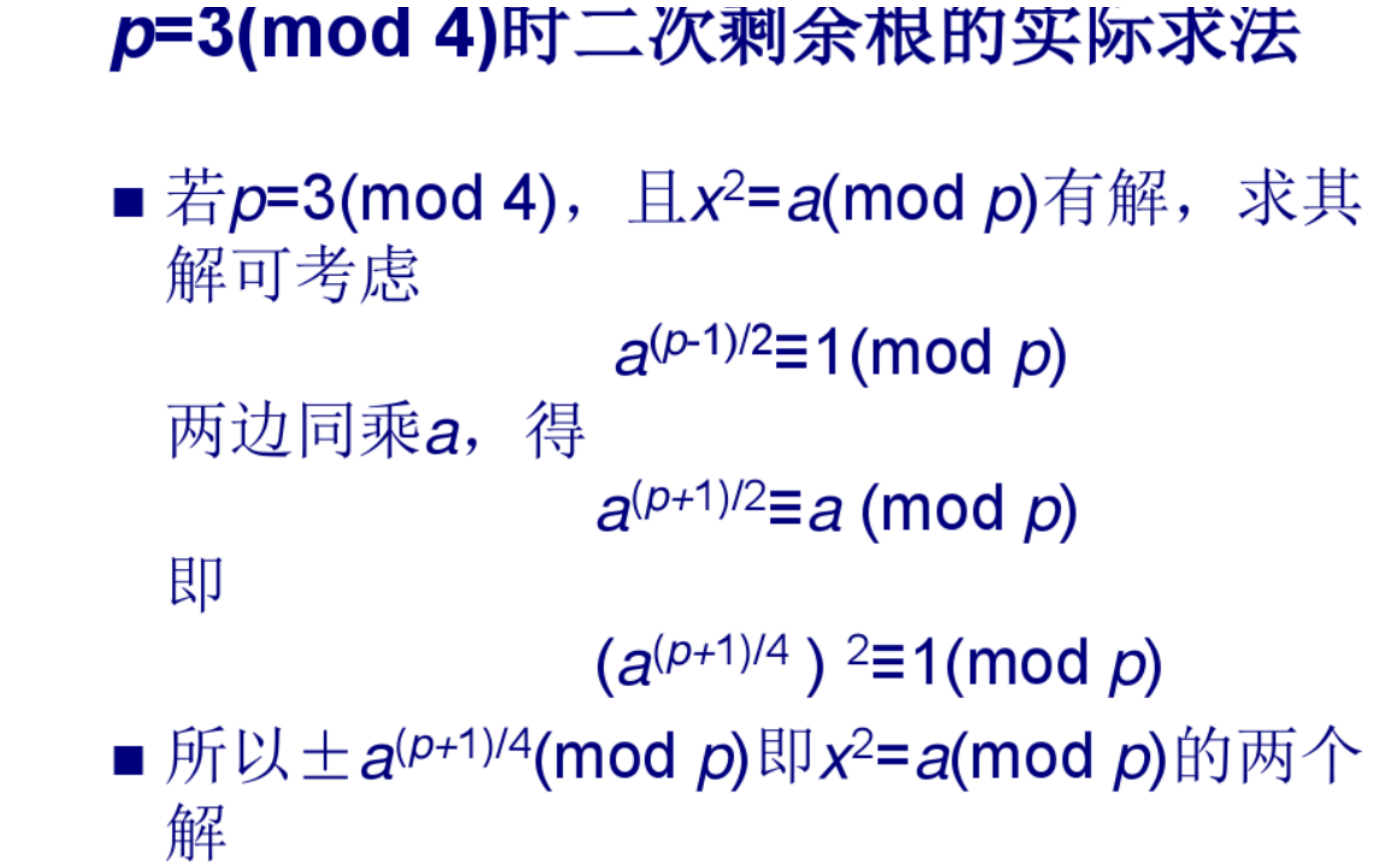
- 欧拉准则

  等于1一定有解，等于0可能有，等于-1（或者在求余意义下的其他值）一定没有解

  In number theory, **Euler's criterion** is a formula for determining whether an integer is a quadratic residue modulo a prime.

  Let p be an odd prime and a an integer coprime to p. Then[1][2]

$$a^{\frac{p-1}{2}} \equiv \begin{cases} 1 \pmod{p} & \text{if there is an integer } x \text{ such that } a \equiv x^2 \pmod{p} \\ -1 \pmod{p} & \text{if there is no such integer.} \end{cases}$$



中国剩余定理

- 设正整数N满足线性同余方程组N≡ai(mod pi)，其中 1<=i<=n，pi两两互质，则 $N=Σ[aiWiinv(Wi,pi)]%M$。 其中，$M=Πpi=p1p2.....*pn$，$Wi=M/pi$，$inv(Wi,pi)$表示Wi在模pi下的逆元。

```
#include<stdio.h>
int N=3,PP=1;
int exgcd(int a,int b,int &x,int &y);
int inv(int n,int m){
    int x,y,d=exgcd(n,m,x,y);
    return (x%m+m)%m;
}
int China(int p[],int a[]){
    int ans=0;
    for(int i=0;i<N;i++) PP*=p[i];
    for(int i=0;i<N;i++){
        int W=PP/p[i];
        ans=(ans+a[i]*W*inv(W,p[i]))%PP;
    }
    return ans;
}
```

拓展中国剩余定理

```
//luogu P4777
//first pass luogu & niuke
#include<stdio.h>
#include<iostream>
using namespace std;
const int maxn=1e5+5;
typedef __int128 ll;
int n;
ll a[maxn],b[maxn];
int scan(ll &x){
    x=0;int sgn=1;
    char ch;
    while(ch=getchar()){
        if(ch==EOF)return EOF;
        else if(ch=='-')sgn=-sgn;
        else if(ch>='0'&&ch<='9'){
            x=x*10+(ch-'0');break;
        }
    }
    while((ch=getchar())>='0'&&ch<='9')x=x*10+(ch-'0');
    x*=sgn;return 1;
}
void _print(ll x){
    if(x>9)_print(x/10);
    putchar(x%10+'0');
}
void print(ll x){
    if(x<0){x=-x;putchar('-');}
    _print(x);
}
//a*x+b*y=gcd(a,b)
ll exgcd(ll a,ll b,ll &x,ll &y){
    if(a==0&&b==0)return -1;
    if(b==0){x=1;y=0;return a;}
    ll d=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}
//china
ll excrt(){
    ll x,y,aa,bb,gcd;
    ll m=a[0],ans=b[0];//a[] divide b[] mod
    for(int i=1;i<n;i++){
        aa=a[i],bb=b[i];
        bb=((bb-ans)%aa+aa)%aa;
        gcd=exgcd(m,aa,x,y);
        if(bb%gcd)return -1;
        x=x*(bb/gcd)%(aa/gcd);
        x=(x+aa)%(aa/gcd);//change the sgn of x, because x maybe - change to +
        ans+=x*m;
```

```
        m=m/gcd*aa;
    }
    return ans;
}
int main(){
    scanf("%d",&n);
    for(int i=0;i<n;i++)scan(a[i]),scan(b[i]);
    print(excrt());
    return 0;
}

//extra method
//china x=ai(mod bi) a[] mod b[] divider
ll excrt(){
    ll x,y,a0=a[0],b0=b[0],a1,b1,gcd;
    for(int i=1;i<n;i++){
        a1=((a[i]-a0)%b[i]+b[i])%b[i],b1=b[i];
        gcd=exgcd(b0,b1,x,y);
        if(a1%gcd)return -1;
        x=x*(a1/gcd)%b1;
        a0+=x*b0;
        b0*=b1/gcd;// b0 finally will be the lcm of all numbers
        a0=(a0+b0)%b0;//change the sgn of a0
    }
    return a0;
}

//niuke https://ac.nowcoder.com/acm/contest/890/D
//second pass luogu & niuke
#include<stdio.h>
#include<iostream>
using namespace std;
const int maxn=1e2+5;
typedef __int128 ll;
int n;
ll a[maxn],b[maxn];
int scan(ll &x){
    x=0;
    int sgn=1;
    char ch;
    while(ch=getchar()){
        if(ch==EOF)return EOF;
        else if(ch=='-')sgn=-sgn;
        else if(ch>='0'&&ch<='9'){
            x=x*10+(ch-'0');break;
        }
    }
    while((ch=getchar())>='0'&&ch<='9')x=x*10+(ch-'0');
    x*=sgn;return 1;
}
void _print(ll x){
    if(x>9)_print(x/10);
    putchar(x%10+'0');
}
```

```
void print(ll x){
    if(x<0){x=-x;putchar('-');}
    _print(x);
}
ll exgcd(ll a,ll b,ll &x,ll &y){
    if(a==0&&b==0)return -1;
    if(b==0){x=1;y=0;return a;}
    ll d=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}
ll excrt(){
    ll a1=b[0],n1=a[0],a2,n2,d,x,y,gcd;
    //if(a1==0)a1=a[0]; require positive number, so 0 is not correct
    for(int i=1;i<n;i++){
        a2=b[i];n2=a[i];
        d=a2-a1;
        gcd=exgcd(n1,n2,x,y);
        if(d%gcd)return -1;
        x=((x*d/gcd)%(n2/gcd)+(n2/gcd))%(n2/gcd);//k1=x*(d/gcd),k1=(k1%(n2/gcd)+
(n2/gcd))%(n2/gcd)
        a1=x*n1+a1;//a1+k1*n1
        n1=n1*n2/gcd;//lcm(n1,n2)
    }
    return a1;
}
int main(){
    scanf("%d",&n);
    ll num;scan(num);
    for(int i=0;i<n;i++)scan(a[i]),scan(b[i]);
    ll ans=excrt();
    if(ans==-1)puts("he was definitely lying");
    else if(ans>num)puts("he was probably lying");
    else print(ans);
    return 0;
}



ll exgcd(ll a,ll b,ll &x,ll &y){
    //cout<<"hh"<<endl;
//  if(a==0&&b==0) return -1;
    ll t,res;
    if(b==0){
        x = 1;
        y = 0;
        return a;
    }
    res = exgcd(b,a%b,x,y);
    t = x;
    x = y;
    y=t - a/b*y;
    return res;
```

```
}
ll excrt(){
    //cout<<"hh"<<endl;
    int flag ;
    flag = 0;
    ll b1=b[0],a1=a[0],b2,a2,k1,k2,x0,gcd,c;
    ll t;
    for(int i=1;i<n;i++){
        b2=b[i],a2=a[i];
        c=b2-b1;
        gcd=exgcd(a1,a2,k1,k2);//解得：n1*k1+n2*k2=gcd(n1,n2)
        if(c%gcd){
            flag=1;
            //return -1;//无解
            break;
        }
        x0 = k1;
        k1=k1*c/gcd;//n1*x0+n2*(c/gcd*k2)=c  PS:k1/gcd*c错误!
        t=a2/gcd;
        k1=(k1%t+t)%t;//求n1*x0+n2*y=c的x0的最小解
        b1=a1*k1+b1;
        a1=a1*a2/gcd;
        //b1 = (b1%a1+a1)%a1;
   }
    //int res = (int )((N-a1)/n1+(a1==0?0:1));

    //int ans  = (N-b1)/a1 + (b1==0? 0:1);
    //int ans = 125;
    if(flag==1) b1 = -1;
    //cout<<"hh"<<endl;
    if(b1==0&&n>1) {
        b1 = a1;
    }
    //if(b1==0&&n==1) b1 =a[0];
    return b1;

}
```

四色猜想

任何一张地图只用四种颜色就能使具有共同边界的国家着上不同的颜色

费马大定理

当整数n >2时，关于x, y, z的方程 $x^n + y^n = z^n$ 没有正整数解

康威常数

外观数列 （康威常数）

今天在图书馆修炼，无意中看到一本数学书上写了这么一个数列
1
11
21
1211
111221
312211
13112221
1113213211 ……（你知道下一个是什么吗？）

哈哈，一看到这个我就亮了。记得在学校时，有个同学在黑板上出了这道题，并说明这是某重点小学的入学考试题。当时我为了解这道题，放着厚厚的作业不做，一直想了2h。oh my lady gaga,我还是失败了。
现在终于了解，这个数列有专有名词，叫做外观数列。规律就是对数的描述。
11表示有1个1；21表示有2个1；1211表示1个2，1个1。

实数域不可拆分多项式只有两种：一次多项式和二次的(b^2<4ac)

*求多项式anx^n+an-1x^(n-1)+…+a1x+a0在实数域上是否可约？n=1，不可约；n=2，根的判别式；n=3，一定实数解（不算虚根i），可以有根号

- 艾森斯坦因判别法：有理数域不可约，即一定要整数解

  艾森斯坦判别法是代数的定理，给出了判定整系数多项式不能分解为整系数多项式乘积的充分条件。由高斯定理，这判别法也是多项式在有理数域不可约的充分条件。

  艾森斯坦判别法是说：给出下面的整系数多项式

  $$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

  如果存在素数p，使得

  - p不整除a_n，但整除其他a_i；
  - p^2不整除a_0，那么$f(x)$是不可约的。

阶乘

阶乘的位数

```
scanf("%d", &n);
double sum = 0;
for (int i = n; i >= 1; i--)
        sum += log10(i);
printf("%d\n", (int)sum + 1);
```

- 自己模拟(int)log10(number)+1
- 斯特林公式(n=0需要特判，无法求出)
  - 写法（pi和e一定要取多一点，不然会有精度问题，对精度要求高）
  - double PI=acos(double(-1)); 反余弦函数 -1.0为pi，1为0
  - double e=exp(double(1));

公式为:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

这就是说，对于足够大的整数n，这两个数互为近似值。更加精确地:

$$\lim_{n\to\infty} \frac{n!}{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n} = 1$$

或

$$\lim_{n\to\infty} \frac{e^n\, n!}{n^n\sqrt{n}} = \sqrt{2\pi}.$$

```
#define pi 3.1415926535
#define e 2.718281828459
int main()
{
    scanf("%d", &t);
    while (t--) {
        scanf("%d", &n);
        int ans = (int)(log10(sqrt(2.0 * pi * n))+n*(log10(1.0*n/e)));
        printf("%d\n", ans+1);
    }
    return 0;
```

阶乘求模

(((n!)!)!)%p=? 阶乘求模

1. 1!!=1
2. 2!!=2
3. 3!!=720
4. 4!!=62044840173323943936000
5. 5!!很大

所以前面特判，后面的一定能够整除p

- 阶乘末尾0的个数
    - 10！=3,628,800
    - 20! =2432902008176640000
    - 25!末尾有6个0
- 阶乘算到几位会爆long long n=20的时候会变成负数

斐波那契数列

f(n)=f(n-1)+f(n-2)

1. 递推的方法能够求到80多项，太多无法用long long储存
2. 递归的方法求能够很快到40多项，时间复杂度高

- 斐波那契因子

Fibonomial triangle [edit]

The Fibonomial coefficients (sequence A010048 in the OEIS) are similar to binomial coefficients and can be displayed in a triangle similar to Pascal's triangle. The first eight rows are shown below.

$$
\begin{array}{ll}
n=0 & 1 \\
n=1 & 1 \quad 1 \\
n=2 & 1 \quad 1 \quad 1 \\
n=3 & 1 \quad 2 \quad 2 \quad 1 \\
n=4 & 1 \quad 3 \quad 6 \quad 3 \quad 1 \\
n=5 & 1 \quad 5 \quad 15 \quad 15 \quad 5 \quad 1 \\
n=6 & 1 \quad 8 \quad 40 \quad 60 \quad 40 \quad 8 \quad 1 \\
n=7 & 1 \quad 13 \quad 104 \quad 260 \quad 260 \quad 104 \quad 13 \quad 1
\end{array}
$$

The recurrence relation

$$\binom{n}{k}_F = F_{n-k+1}\binom{n-1}{k-1}_F + F_{k-1}\binom{n-1}{k}_F$$

日期转换星期

- 基姆拉尔森计算公式 Week = (d+2*m*+3(m+1)/5+y+y/4-y/100+y/400+1) mod 7；

0-星期日，1-星期一，2-星期二，3-星期三，4-星期四，5-星期五，6-星期六

其中： d为几号，m为月份，y为年份，注：把一月和二月看为是上一年的十三月和十四月

```
int check(int y, int m, int d)
 {
   if (m == 1 || m == 2) m += 12, y = y - 1;
   return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400 ) % 7; }
```

- 蔡勒公式

1582年10月4日后：w = (d + 1+ 2*m*+3(m+1)/5+y+y/4-y/100+y/400)%7;

1582年10月4日前：w = (d+1+2*m*+3(m+1)/5+y+y/4+5) % 7;

或者1752年9月3日为例 1752年9月3日后：w = (d + 2*m*+3(m+1)/5+y+y/4-y/100+y/400)%7;

比上面的少1 说明 6-星期日，0-星期一，1-星期二，2-星期三，3-星期四，4-星期五，5-星期六

1752年9月3日前：w = (d+2*m*+3(m+1)/5+y+y/4+5) % 7;

注：罗马教皇决定在1582年10月4日后使用格利戈里历法；而英国则是在1752年9月3日后才接受使用格利戈里历法。

注意： 当年的1,2月要当成上一年的13,14月进行计算

```
int celar(int y, int m, int d)
{
        if (m == 1 || m == 2) {
                m += 12, y--;
        }
        int c = y / 100;
        y = y % 100;
        int w = (c/4-2*c+y+y/4+13*(m+1)/5+d-1) % 7;
        return (w % 7 + 7) % 7;
}
```

矩阵快速幂

- 复杂度：O(m^3*log(n)) m是矩阵的大小，第n项

- 一些推导

斐波拉契数列： $F_i = F_{i-1} + F_{i-2}$：

$$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}^{i-1} \times \begin{bmatrix}F_1\\F_0\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 0\end{bmatrix} \times \begin{bmatrix}F_{i-1}\\F_{i-2}\end{bmatrix} = \begin{bmatrix}F_i\\F_{i-1}\end{bmatrix}$$

递推式 - 1： $G_i = a \times G_{i-1} + b \times G_{i-2}$：

$$\begin{bmatrix}a & b\\1 & 0\end{bmatrix}^{i-1} \times \begin{bmatrix}G_1\\G_0\end{bmatrix} = \begin{bmatrix}a & b\\1 & 0\end{bmatrix} \times \begin{bmatrix}G_{i-1}\\G_{i-2}\end{bmatrix} = \begin{bmatrix}G_i\\G_{i-1}\end{bmatrix}$$

递推式 - 2： $G_i = a \times G_{i-1} + i^2$：

$$\begin{bmatrix}a & 1 & 0 & 0\\0 & 1 & 2 & 1\\0 & 0 & 1 & 1\\0 & 0 & 0 & 1\end{bmatrix}^{i} \times \begin{bmatrix}G_0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}a & 1 & 0 & 0\\0 & 1 & 2 & 1\\0 & 0 & 1 & 1\\0 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}G_{i-1}\\i^2\\i\\1\end{bmatrix} = \begin{bmatrix}G_i\\(i+1)^2\\i+1\\1\end{bmatrix}$$

递推式 - 3： $G_i = a \times G_{i-1} + i^3$：

$$\begin{bmatrix}a & 1 & 0 & 0 & 0\\0 & 1 & 3 & 3 & 1\\0 & 0 & 1 & 2 & 1\\0 & 0 & 0 & 1 & 1\\0 & 0 & 0 & 0 & 1\end{bmatrix}^{i} * \begin{bmatrix}G_0\\1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}a & 1 & 0 & 0 & 0\\0 & 1 & 3 & 3 & 1\\0 & 0 & 1 & 2 & 1\\0 & 0 & 0 & 1 & 1\\0 & 0 & 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}G_{i-1}\\i^3\\i^2\\i\\1\end{bmatrix} = \begin{bmatrix}G_i\\(i+1)^3\\(i+1)^2\\i+1\\1\end{bmatrix}$$

递推式 - 4： $G_i = a \times G_{i-1} + b^i$：

$$\begin{bmatrix}a & 1\\0 & b\end{bmatrix}^{i} \times \begin{bmatrix}G_0\\b\end{bmatrix} = \begin{bmatrix}a & 1\\0 & b\end{bmatrix} \times \begin{bmatrix}G_{i-1}\\b^i\end{bmatrix} = \begin{bmatrix}G_i\\b^{i+1}\end{bmatrix}$$

1.f(n)=a*f(n-1)+b*f(n-2)+c;　(a,b,c是常数)

$$\begin{pmatrix}a & b & 1\\1 & 0 & 0\\0 & 0 & 1\end{pmatrix} * \begin{pmatrix}f_{n-1}\\f_{n-2}\\c\end{pmatrix} = \begin{pmatrix}f_n\\f_{n-1}\\c\end{pmatrix}$$

2.f(n)=c^n-f(n-1)；　(c是常数)

$$\begin{pmatrix}-1 & c\\0 & c\end{pmatrix} * \begin{pmatrix}f_{n-1}\\c^{n-1}\end{pmatrix} = \begin{pmatrix}f_n\\c^n\end{pmatrix}$$

- 建立矩阵

```
const int N = 100;//N*N的矩阵
typedef struct Matrix
{
    long long val[N+1][N+1];
    Matrix()
    {
        memset(val, 0, sizeof(val));
    }
}Matrix;
```

- 初始化矩阵

```
Matrix I, O;
void init()
{
    I = Matrix();
    O = Matrix();
    for (int i = 0; i < N; i++)
    {
        I.val[i][i] = 1;
    }
    return;
}
```

- 矩阵相乘

```
Matrix multi(Matrix A, Matrix B)
{
    Matrix ans = O;
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            for (int k = 0; k < N; k++)
                ans.val[i][j] = (ans.val[i][j] + A.val[i][k] * B.val[k][j]) % mod;
    return ans;
}
```

- 矩阵快速幂

```
Matrix qPow(Matrix A, long long p)
{
    Matrix ans = I;
    while (p)
    {
        if (p & 1)ans = multi(ans, A);
        A = multi(A, A);
        p >>= 1;
    }
    return ans;
}
```

- 矩阵乘法

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        for (int k = 0; k < n; k++)
        {
            ans[i][j] += M1[i][k] * M2[k][j];
        }
    }
}
```

组合数学

- 组合数的性质

  组合数的定义：从n个不同的元素中选出m个元素，共有C(n,m)中不同的选法。

  计算方法：C(n,m)=n!/(m!*(n-m)!)。

  (性质1) C(n,k)=C(n,n-k)。(对称性)

  (性质2) C(n,k-1)+C(n,k)=C(n+1,k)。(递推公式)

  (性质3) ΣC(n,i)=C(n,0)+C(n,1)+...+C(n,n)=2^n。(横向求和)

  (性质4) 若n&m==m，则C(n,m)为奇数。(Lucas定理)

  (性质5) 若p为质数，则C(p,k)%p=0，其中1<=k<=n-1。

- 组合数打表

前几个组合数打表结果如下：

1							
1	1						
1	2	1					
1	3	3	1				
1	4	6	4	1			
1	5	10	10	5	1		
1	6	15	20	15	6	1	
1	7	21	35	35	21	7	1

特别注意：C(30,15)=155117520（int范围中最大的组合数），C(64,32)=1.83x10^18（long long范围中最大的组合数）。

计算

在不取模的情况下，要求n的范围非常小才行

1. 定义计算

- 复杂度：O(m) 组合数横向递推
- 注意乘法溢出，如果要取模则需要预处理逆元
- C(n,0)=1，C(n,k+1)=C(n,k)*(n-k)/(k+1)

```
typedef long long ll;
ll C(int n,int m){
    ll ans=1;
    for(int i=0;i<m;i++){
        ans=ans*(n-i)/(i+1);
    }
    return ans;
}
```

2. 通过递推关系打表，$O(n^2)$打表，O(1)查询，杨辉三角纵向递推

   用O(n^2)的时间求出C(0,0)~C(n,n)

   递推关系如下：C(i,0)=1，C(i,j)=C(i-1,j-1)+C(i-1,j)

   乘法运算，加法的递推不容易溢出。注意在不取模的情况下，n不能超过64。在取模的情况下，n主要受到空间的限制，大概是3000以内。

```
void getC(){
    for(int i=0;i<=60;i++){
        c[i][0]=1;c[i][i]=1;
    }
    for(int i=1;i<=60;i++){
        for(int j=1;j<i;j++){
            c[i][j]=c[i-1][j-1]+c[i-1][j];
        }
    }
    return;
}
```

组合数取模

1. 卢卡斯定理

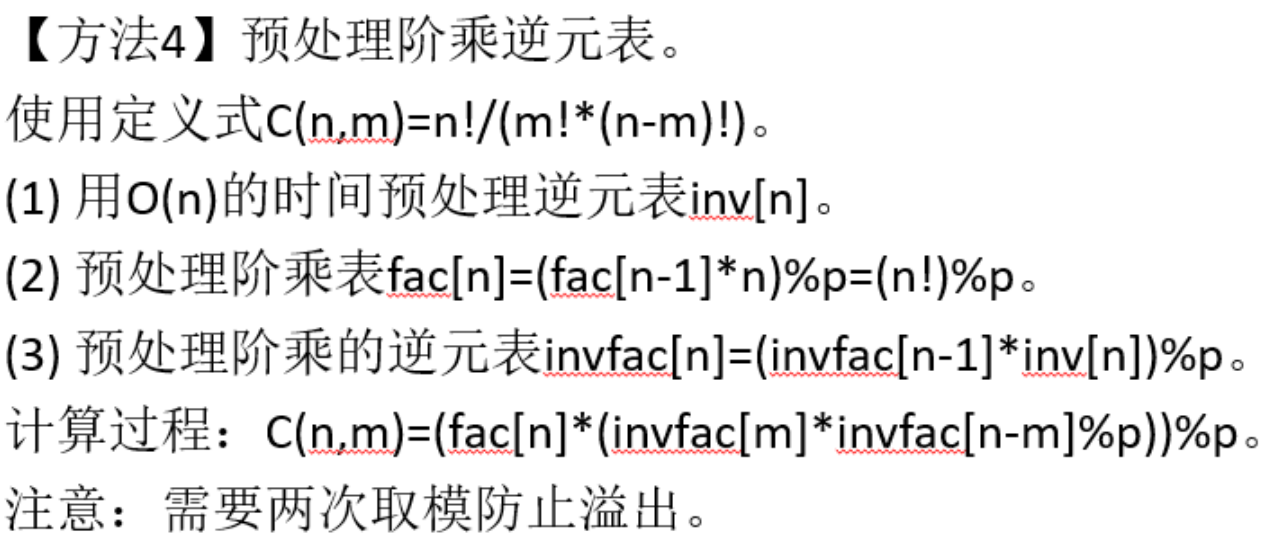
- **模p是质数且相对比较小**
- 时间复杂度O(plog(n))
- 设N和M是较大的非负整数，p是较小的质数，将n和m写成p进制数，设N[i]与M[i]分别表示N和M在p进制下的第i位数，那么组合数C(N,M)%p可以写成Π[C(a[i],b[i])]%p连乘的形式。 例如：把11写成二进制是(1011)，把5写成二进制是(0101)。 因此C(11,5)%2=(C(1,0)*C(0,1)*C(1,0)*C(1,1))%2=0。

```
typedef long long ll;
ll C(ll n,ll m,ll p);
ll Lucas(ll n,ll m,ll p){
    return m?Lucas(n/p,m/p,p)*C(n%p,m%p,p)%p:1;
}
//while循环版本
ll C(ll n,ll m,ll p);
ll Lucas2(ll n,ll m,ll p){
    ll ans=1;
    while(n&&m&&ans){
        ans=(ans*C(n%p,m%p,p))%p;
        n/=p;m/=p;
    }
    return ans;
}
```

2. 预处理阶乘逆元表

- **模p为质数**

- O(n)打表，O(1)查询



- 线性预处理逆元

```
void setInv(int n){
    inv[0]=inv[1]=1;
    for(int i=2;i<=n;i++)
        inv[i]=1ll*(mod-mod/i)*inv[mod%i]%mod;
}
```

- 预处理阶乘以及阶乘逆元求组合数

```
void setFac(int n){
    fac[0]=facInv[0]=1;
    for(int i=1;i<=n;i++){
        fac[i]=1ll*fac[i-1]*i%mod;
        facInv[i]=1ll*facInv[i-1]*inv[i]%mod;
    }
}
int C(int n,int m){
    if(n<m)return 0;
    if(n<0||m<0)return 0;
    int ans=fac[n];
    ans=1ll*ans*facInv[m]%mod;
    ans=1ll*ans*facInv[n-m]%mod;
    return ans;
}
```

正整数拆分

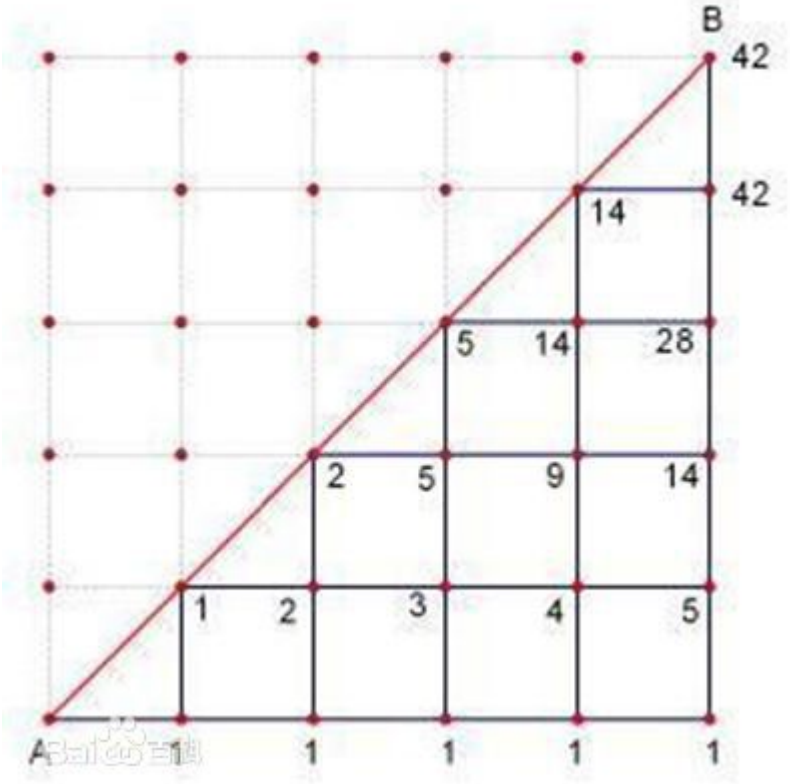
设f(n,k)表示正整数n被拆成k个数的不同拆法。 使用挡板法，f(n,k)=C(n-1,k-1)。 所以S(n)=Σf(n,k)=ΣC(n-1,k-1)=2^(n-1)

小球盒子

杨辉三角

卡特兰数

- 其前几项为（从第零项开始）：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452



- 卡特兰数的一些性质

卡特兰数：$K(n)=\frac{1}{n+1}C_{2n}^{n}$。

递推形式：

K(0)=0，K(1)=1，

当n>=2时，$K(n)=\sum K(i)*K(n-i-1)$，0<=i<=n-1。

或者，$K(n)=K(n-1)*(4n-2)/(n+1)$。

此外，还有另一种通项公式，$K(n)=C_{2n}^{n}-C_{2n}^{n-1}$。

错排问题

- 递推

错排公式：$D(n)=n!\left(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\frac{1}{4!}+\cdots+(-1)^n\frac{1}{n!}\right)$。

递推形式：$D(n)=(n-1)(D(n-1)+D(n-2))$。

证明：第n个人共有(n-1)种选法，不妨假设第n个人选中了第k个人的帽子。接下来考虑第k个人，如果他恰好选到了第n个人的帽子，则剩下的(n-2)个人的选法就是D(n-2)。如果他选到的是另外(n-2)个人的帽子，那么他可以把这顶帽子交给第n个人，换回自己的帽子，于是就转化为(n-1)个人的错排问题。

容斥原理

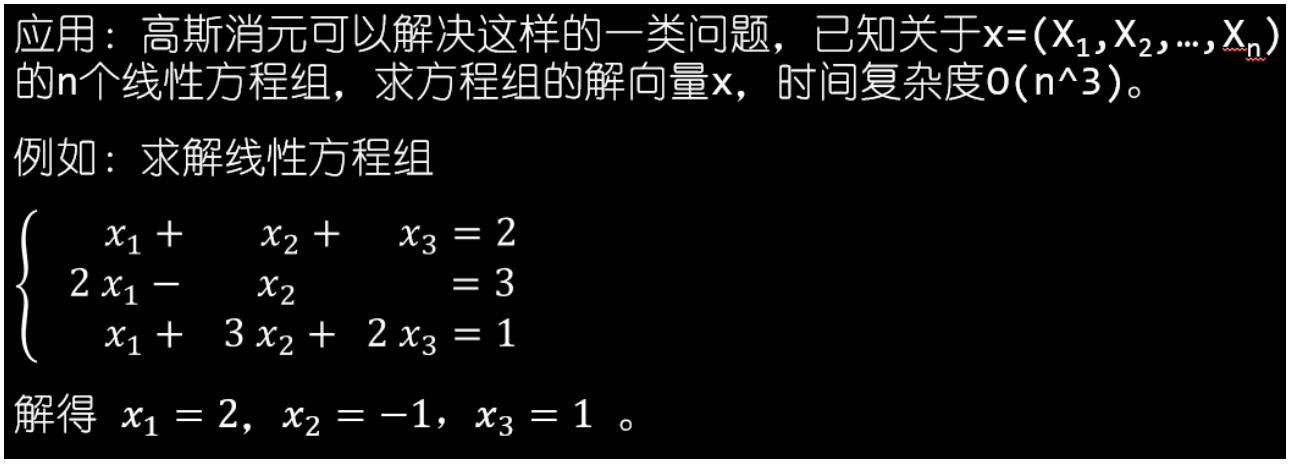
- $(A\cup B)=A+B-(A\cap B)$
- $(A\cup B\cup C)=A+B+C-(A\cap B)-(B\cap C)-(C\cap A)+(A\cap B\cap C)$
- 在1到N的范围内，有多少个正整数可以被2、3、5中的任意一个整除?能被2整除的数有N/2个，能被3整除的数有N/3个，能被5整除的数有N/5个，然后考虑重叠部分。能同时被2和3整除的有N/6个，能同时被2和5整除的有N/10个，能同时被3和5整除的有N/15个，能同时被2和3和5整除的有N/30个。

鸽巢原理&抽屉原理

- 把(n+1)个元素放到n个集合中去，必定会有两个元素属于同一个集合，也必定会有一个集合中至少有两个元素。 推论：把(m*n+1)个元素放到n个集合中去，则至少有一个集合中存在不少于(m+1)个元素。
- 拉姆塞定理：在6个人当中，至少有3个人互相认识或者互相不认识。

高斯消元

- 求解线性方程组Ax=B，已知关于x=(X1,X2,...,Xn)的n个线性方程组，求方程组的解向量x
- 时间复杂度O(n^3)
- 高斯消元



- 模板

```cpp
#include<stdio.h>
#include<math.h>
const double eps=1e-6;
const int N=102,M=102;
double a[N][M];
int Gauss(int n,int m){
    // 总共有 n 个方程，m 个未知数，增广矩阵是n*(m+1)的
    int r,c,k,pst,var=0;
    double tmp,maxr;
    for(r=0,c=0;r<n&&c<m;r++,c++){
    maxr=eps; pst=n;
    // 将 k 设置为绝对值最大的行
    for(k=r;k<n;k++)
        if(fabs(a[k][c])>maxr){
            maxr=fabs(a[k][c]); pst=k;
        }
        k=pst;
        if(k==n){ r--; continue; }
        // 选中第 k 行与第 r 行交换
        if(k!=r){
        for(int j=c;j<=m;j++){
            tmp=a[k][j]; a[k][j]=a[r][j]; a[r][j]=tmp;
        }
    }
    // 将第 r 行的第 c 列元素消成 1.0，将其余的行消成 0.0
    for(int j=c+1;j<=m;j++) a[r][j]/=a[r][c];
    a[r][c]=1.0;
    for(int i=0;i<n;i++){
        if(i==r||fabs(a[i][c])<eps) continue;
        for(int j=c+1;j<=m;j++)
            a[i][j]-=a[i][c]*a[r][j];
            a[i][c]=0.0;
        }
    }
    for(int i=r;i<n;i++){
        if(fabs(a[i][m])>eps){
            return -1; // 无解时返回-1
        }
    }
    var=m-r; // 方程有唯一解则返回 0
    return var; // 返回自由变量的个数
}

double x[102][102],x0[102];
bool isfree[102];
```

```
/*  这里的 m 是自由变量的个数而不是增广矩阵列数  */
int Solve(int n,int m){
    int r,c,num=0;
    // 寻找自由变量
    for(int i=0;i<m;i++) isfree[i]=true;
    for(r=0,c=0;r<n&&c<m;r++,c++){
        if(fabs(a[r][c])<eps){
            isfree[c]=true; r-=1;
        }else{
            isfree[c]=false;
        }
    }
    // 寻找 Ax=0 的通解
    for(int i=0;i<m;i++){
        if(isfree[i]){
            r=0;
            for(int j=0;j<m;j++){
                if(!isfree[j]){
                    x[num][j]=-a[r][i]; r=r+1;
                }else{
                    if(i==j) x[num][j]=1.0;
                    else x[num][j]=0.0;
                }
            }
            num=num+1;
        }
    }
    // 寻找 Ax=0 的特解
    for(int i=0;i<m;i++){
        r=0;
        if(!isfree[i]){
            x0[i]=a[i][m]; r=r+1;
        }else{
            x0[i]=0.0;
        }
    }
    return num; // 返回自由变量的个数
}

int main()
{
    int n,m;
    while(scanf("%d %d",&n,&m)!=EOF){
        for(int i=0;i<n;i++){
            for(int j=0;j<=m;j++) scanf("%lf",&a[i][j]);
        }
        Gauss(n,m);
        Solve(n,m);
        for(int i=0;i<m;i++) printf("x[%d]=%.4f ",i+1,x0[i]);
        printf("\n");
    }
    return 0;
}
```

取模问题

- 要求6/4输出2，ans=(n-1)/k+1;

条件概率

- 条件概率
  - 在已知事件B发生的条件下，A发生的概率，记为P(A|B)。计算公式：P(A|B)=P(AB)/P(B)
  - 全概率公式：P(A)=P(B1)xP(A|B1)+...+P(Bn)xP(A|Bn)。B1,...,Bn是互不相容的事件，且U(B1,...,Bn)=全集Ω

期望

- 性质
  1. E(c)=0，其中c为任意常数
  2. E(a*X)=a*E(X)，其中a为任意常数。
  3. E(a*X+b*Y)=a*E(X)+b*E(Y)，期望的线性性

期望的线性性

- 期望的线性性
  - 随机游走(链和图)

  1. 一个长度为n的链，从一端走到另一端的期望时间？设dp[i]表示从i第一次到i+1所需要的期望时间，则dp[i]=1/2+1/2*(1+dp[i-1]+dp[i]) 得dp[i]=dp[i-1]+2。dp[1]=1（1到2只能前进）。ans=dp[1]+dp[2]+.....+dp[n-1]=(n-1)^2

  2. N个点的完全图，问从S走到T的期望时间？ans=1/(n-1)+(n-2)/(n-1)*(ans+1)得1/(n-1)*ans=1得ans=n-1。

例题3：有n(n<=20)个格子，每次会涂一个格子，其中涂第i个格子的概率是p_i（保证∑p_i=1）。求每个格子都被涂色的期望次数。（类似的例题：HDU 4336）

状压dp+概率：二进制1认为当前格子已经被涂色。dp[1<<n-1]=0,dp[0]为所求的答案。

转移：从0开始转移。

```
int sum=(1<<n)-1;
dp[sum]=0;//涂满的期望次数为0
for(int i=sum-1;i>=0;i--){
        dp[i]=1;
        double tmp=0;
        for(int j=0;j<n;j++){
                if((i&(1<<j))==0){
                        dp[i]+=dp[i+(1<<j)]*p[j];
                        tmp+=p[j];
                }
        }
```

```
            dp[i]/=tmp;
    }
    cout<<dp[0]<<endl;
```