常用公式：

一些数论公式

- 当 $x \ge \phi(p)$ 时有 $a^x \equiv a^{x \ mod \ \phi(p)+\phi(p)} \pmod p$
- $\mu^2(n)=\sum_{d^2|n}\mu(d)$
- $\sum_{d|n}\varphi(d)=n$
- $\sum_{d|n}2^{\omega(d)}=\sigma_0(n^2)$，其中 ω 是不同素因子个数
- $\sum_{d|n}\mu^2(d)=2^{\omega(d)}$

一些数论函数求和的例子

- $\sum_{i=1}^n i[gcd(i,n)=1]=\frac{n\varphi(n)+[n=1]}{2}$
- $\sum_{i=1}^n\sum_{j=1}^m[gcd(i,j)=x]=\sum_d\mu(d)\lfloor\frac{n}{dx}\rfloor\lfloor\frac{m}{dx}\rfloor$
- $\sum_{i=1}^n\sum_{j=1}^m gcd(i,j)=\sum_{i=1}^n\sum_{j=1}^m\sum_{d|gcd(i,j)}\varphi(d)=\sum_d\varphi(d)\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$
- $S(n)=\sum_{i=1}^n\mu(i)=1-\sum_{i=1}^n\sum_{d|i,d<i}\mu(d)\overset{t=\frac{i}{d}}{=}1-\sum_{t=2}^n S(\lfloor\frac{n}{t}\rfloor)$
  - 利用 $[n=1]=\sum_{d|n}\mu(d)$
- $S(n)=\sum_{i=1}^n\varphi(i)=\sum_{i=1}^n i-\sum_{i=1}^n\sum_{d|i,d<i}\varphi(i)\overset{t=\frac{i}{d}}{=}\frac{i(i+1)}{2}-\sum_{t=2}^n S(\frac{n}{t})$
  - 利用 $n=\sum_{d|n}\varphi(d)$
- $\sum_{i=1}^n\mu^2(i)=\sum_{i=1}^n\sum_{d^2|n}\mu(d)=\sum_{d=1}^{\lfloor\sqrt{n}\rfloor}\mu(d)\lfloor\frac{n}{d^2}\rfloor$

- $\sum_{i=1}^n \sum_{j=1}^n gcd^2(i,j) = \sum_d d^2 \sum_t \mu(t) \lfloor \frac{n}{dt} \rfloor^2$
  $\overset{x=dt}{=} \sum_x \lfloor \frac{n}{x} \rfloor^2 \sum_{d|x} d^2 \mu(\frac{x}{d})$
- $\sum_{i=1}^n \varphi(i) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [i \perp j] - 1 = \frac{1}{2} \sum_{i=1}^n \mu(i) \cdot \lfloor \frac{n}{i} \rfloor^2 - 1$

斐波那契数列性质

- $F_{a+b} = F_{a-1} \cdot F_b + F_a \cdot F_{b+1}$
- $F_1 + F_3 + \cdots + F_{2n-1} = F_{2n}, F_2 + F_4 + \cdots + F_{2n} = F_{2n+1} - 1$
- $\sum_{i=1}^n F_i = F_{n+2} - 1$
- $\sum_{i=1}^n F_i^2 = F_n \cdot F_{n+1}$
- $F_n^2 = (-1)^{n-1} + F_{n-1} \cdot F_{n+1}$
- $gcd(F_a, F_b) = F_{gcd(a,b)}$
- 模 n 周期（皮萨诺周期）
  - $\pi(p^k) = p^{k-1}\pi(p)$
  - $\pi(nm) = lcm(\pi(n), \pi(m)), \forall n \perp m$
  - $\pi(2) = 3, \pi(5) = 20$
  - $\forall p \equiv \pm 1 \pmod{10}, \pi(p) | p - 1$
  - $\forall p \equiv \pm 2 \pmod{5}, \pi(p) | 2p + 2$

常见生成函数

- $(1+ax)^n = \sum_{k=0}^n \binom{n}{k} a^k x^k$
- $\dfrac{1-x^{r+1}}{1-x} = \sum_{k=0}^n x^k$
- $\dfrac{1}{1-ax} = \sum_{k=0}^\infty a^k x^k$
- $\dfrac{1}{(1-x)^2} = \sum_{k=0}^\infty (k+1) x^k$
- $\dfrac{1}{(1-x)^n} = \sum_{k=0}^\infty \binom{n+k-1}{k} x^k$
- $e^x = \sum_{k=0}^\infty \dfrac{x^k}{k!}$
- $\ln(1+x) = \sum_{k=0}^\infty \dfrac{(-1)^{k+1}}{k} x^k$

佩尔方程

若一个丢番图方程具有以下的形式：$x^2 - ny^2 = 1$。且 n 为正整数，则称此二元二次不定方程为**佩尔方程**。

若 n 是完全平方数，则这个方程式只有平凡解 $(\pm 1, 0)$（实际上对任意的 n，$(\pm 1, 0)$ 都是解）。对于其余情况，拉格朗日证明了佩尔方程总有非平凡解。而这些解可由 $\sqrt{n}$ 的连分数求出。

$$x = [a_0; a_1, a_2, a_3] = x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}}$$

设 $\frac{p_i}{q_i}$ 是 $\sqrt{n}$ 的连分数表示: $[a_0; a_1, a_2, a_3, \ldots]$ 的渐近分数列, 由连分数理论知存在 i 使得 (p_i, q_i) 为佩尔方程的解。取其中最小的 i, 将对应的 (p_i, q_i) 称为佩尔方程的基本解, 或最小解, 记作 (x_1, y_1), 则所有的解 (x_i, y_i) 可表示成如下形式: $x_i + y_i\sqrt{n} = (x_1 + y_1\sqrt{n})^i$。或者由以下的递回关系式得到:

$x_{i+1} = x_1 x_i + n y_1 y_i, y_{i+1} = x_1 y_i + y_1 x_i$。

但是： 佩尔方程千万不要去推（虽然推起来很有趣，但结果不一定好看，会是两个式子）。记住佩尔方程结果的形式通常是 $a_n = ka_{n-1} - a_{n-2}$（a_{n-2} 前的系数通常是 -1）。暴力 / 凑出两个基础解之后加上一个 0，容易解出 k 并验证。

Burnside & Polya

- $|X/G| = \frac{1}{|G|}\sum_{g\in G}|X^g|$

注：X^g 是 g 下的不动点数量，也就是说有多少种东西用 g 作用之后可以保持不变。

- $|Y^X/G| = \frac{1}{|G|}\sum_{g\in G}m^{c(g)}$

注：用 m 种颜色染色，然后对于某一种置换 g，有 $c(g)$ 个置换环，为了保证置换后颜色仍然相同，每个置换环必须染成同色。

皮克定理

$2S = 2a + b - 2$

- S 多边形面积
- a 多边形内部点数
- b 多边形边上点数

莫比乌斯反演

- $g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n}\mu(d)g(\frac{n}{d})$
- $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d}\mu(\frac{d}{n})f(d)$

低阶等幂求和

- $\sum_{i=1}^{n} i^1 = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$
- $\sum_{i=1}^{n} i^3 = \left[\frac{n(n+1)}{2}\right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$
- $\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$
- $\sum_{i=1}^{n} i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2$

一些组合公式

- 错排公式：$D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-1} + D_{n-2}) = n!(\frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n\frac{1}{n!}) = \lfloor \frac{n!}{e} + 0.5 \rfloor$
- 卡塔兰数（n 对括号合法方案数，n 个结点二叉树个数，$n \times n$ 方格中对角线下方的单调路径数，凸 $n+2$ 边形的三角形划分数，n 个元素的合法出栈序列数）：$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$

数论

1. 光速幂：

```
int p = 998244353;
const int BL = 1<<16;
int qp[BL+10][2]; //0:a^t%p 1:a^is%p
int base;
void init(int a)
{
    base = sqrt(p);
    qp[0][0]=qp[0][1]=1;
    for(int i=1;i<=base;i++)qp[i][0]=qp[i-1][0]*a%p;
    for(int i=1;i<=base;i++)qp[i][1]=qp[i-1][1]*qp[base][0]%p;
}
```

```
int phi(int x)
{
    int res=x;
    for(int i=2;i*i<=x;i++)
    {
        if(x%i==0) res=res/i*(i-1);
        while(x%i==0) x/=i;
    }
    if(x>1) res=res/x*(x-1);
    return res;
}

int ask(int b)//return a^b%p
{
    b%=phi(p);//这里phi(p)可以预处理
    return qp[b%base][0]*qp[b/base][1]%p;
}
```

2. 二次剩余：
   求解x^2 = a(mod p)的解，x至多有两个，至少一个

```
namespace QR//二次剩余相关，只考虑模数为奇素数的情况
{
    int ksm(int a,int b,int p)
    {
        int ans=1;
        while(b>0)
        {
            if(b&1)ans=ans*a%p;
            a=a*a%p;
            b>>=1;
        }
        return ans;
    }
    //欧拉判别法，判断a是否是p的二次剩余
    int euler(int a,int p)
    {
        if(ksm(a,(p-1)/2,p)==1)return 1;
        else return 0;
    }


    int w,mod;//w^2=(r*r-a) (mod p) , mod=p;
    struct Complex//类似复数的数域
    {
        int x,y;
        Complex(int xx=0,int yy=0)
        {
            x=xx%mod;y=yy%mod;
        }
    };
```

```
    Complex operator+(Complex a,Complex b)
    {
        return Complex((a.x+b.x)%mod,(a.y+b.y)%mod);
    }
    Complex operator-(Complex a,Complex b)
    {
        return Complex((a.x-b.x+mod)%mod,(a.y-b.y+mod)%mod);
    }
    Complex operator*(Complex a,Complex b)
    {
        return Complex((a.x*b.x+a.y*b.y%mod*w)%mod,(a.x*b.y+a.y*b.x)%mod);
    }
    Complex power(Complex a,int b)
    {
        Complex ans(1,0);
        while(b>0)
        {
            if(b&1)ans=ans*a;
            a=a*a;
            b>>=1;
        }
        return ans;
    }

    //Cipolla算法求解x^2=a(mod p)
    //返回一个解（-1无解），相反数为另一个解
    int cipolla(int a,int p)
    {
        mod=p;
        srand((unsigned)time(NULL));
        if(a%p==0)return 0;
        if(!euler(a,p))return -1;
        int r;
        do{
            r=rand()%p;
            w=(r*r%p-a+p)%p;
            if(!euler(w,p))break;
        }while(1);
        Complex ans(r,1);
        ans=power(ans,(p+1)/2);
        return ans.x;
    }
}
```

3. 扩展欧几里得算法： 给出ax+by=gcd(a,b)的一组解

```
int exgcd(int a,int b,int &x,int &y)
{
    if(b==0)
    {
        x=1,y=0;
        return a;
```

```
    }
    int d=exgcd(b,a%b,x,y);
    int k=x;
    x=y;
    y=k-a/b*y;
    return d;
}
```

4. 前缀异或和：计算1~n的异或和

```
int calc_xor(int n){
    if(n < 0) return 0;
    int rem = n % 4;
    if(rem == 0) return n;
    if(rem == 1) return 1;
    if(rem == 2) return n + 1;
    return 0;
}
```

5. 莫比乌斯函数：

```
//线性筛求莫比乌斯函数

const int MAXN = 1e6+10;
int mu[MAXN],p[MAXN],flg[MAXN];//莫比乌斯函数，素数序列，是否为素数
int tot=0;//素数个数

void getMu(int n)//得出n内的莫比乌斯函数
{
    mu[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(!flg[i])p[++tot]=i,mu[i]=-1;
        for(int j=1;j<=tot&&i*p[j]<=n;j++)
        {
            flg[i*p[j]]=1;
            if(i%p[j]==0)
            {
                mu[i*p[j]]=0;
                break;
            }
            mu[i*p[j]]=-mu[i];
        }
    }
}
```

6. 杜教筛：用于求解积性函数前缀和

```
//杜教筛用于处理一类数论函数的前缀和

//求莫比乌斯函数前缀

const int MAXN = 2e6+10;

int mu[MAXN],p[MAXN],flg[MAXN];//莫比乌斯函数，素数序列，是否为素数
int tot=0;//素数个数
int con=2e6;

void getMu(int n)//得出n内的莫比乌斯函数
{
    mu[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(!flg[i])p[++tot]=i,mu[i]=-1;
        for(int j=1;j<=tot&&i*p[j]<=n;j++)
        {
            flg[i*p[j]]=1;
            if(i%p[j]==0)
            {
                mu[i*p[j]]=0;
                break;
            }
            mu[i*p[j]]=-mu[i];
        }
    }
}
//考虑杜教筛求莫比乌斯函数前缀，欧拉函数前缀用迪利克雷卷积求得
map<int,int> pre_mu,pre_phi;//记录

int getsummu(int n)
{
    if(n<=con)return mu[n];
    if(pre_mu[n]!=0)return pre_mu[n];
    int sum=0,i=2;
    while(i<=n)
    {
        int j=n/(n/i);
        sum+=(j-i+1)*getsummu(n/i);
        i=j+1;
    }
    pre_mu[n]=1-sum;
    return 1-sum;
}

int getsumphi(int n)
{
    if(pre_phi[n]!=0)return pre_phi[n];
    int ans=0,i=1;
    while(i<=n)
    {
        int j=n/(n/i);
```

```
        ans+=(getsummu(j)-getsummu(i-1))*(n/i)*(n/i);
        i=j+1;
    }
    pre_phi[n]=(ans+1)/2;
    return (ans+1)/2;
}
```

7. 中国剩余定理(与扩展中国剩余定理)：

```
//给出ax+by=gcd(a,b)的一组解（这里用来求逆元,x是a的逆元）
//注意x可能为负数，x=(x+b)%b即可;
int exgcd(int a,int b,int &x,int &y)
{
    if(b==0)
    {
        x=1,y=0;
        return a;
    }
    int d=exgcd(b,a%b,x,y);
    int k=x;
    x=y;
    y=k-a/b*y;
    return d;
}

//求解一元线性同余方程组

const int MAXN = 2e5+10;
int a[MAXN],r[MAXN];//r为模数数组

int CRT(int k)//k为方程数目
{
    int n=1,ans=0;
    for(int i=1;i<=k;i++)n=n*r[i];
    for(int i=1;i<=k;i++)
    {
        int m=n/r[i],m1,x,y;
        exgcd(m,r[i],x,y);
        m1=(x+r[i])%r[i];//注意x才是结果
        ans+=m*m1*a[i];
        ans%=n;
    }
    return ans;
}


//扩展中国剩余定理（模数不互质的情况）
int excrt(int k)//k为方程数目
{
    int ma=a[1],mp=r[1];
    for(int i=2;i<=k;i++)
    {
```

```
        int x,y;
        if((ma-a[i])%__gcd(mp,r[i])!=0)return -1;
        exgcd(mp,r[i],x,y);
        if(x<0)x=(x+r[i])%r[i];
        x=x*(a[i]-ma)/__gcd(mp,r[i]);
    }
}
```

8. BSGS算法： 求解a^x = b (mod p)的一个解

```
typedef long long ll;

map<ll, ll> mp;
ll a, b, p, ans;

ll exgcd(ll a, ll b, ll& x, ll& y) {  //求逆元和 gcd 都可以使用扩欧，非常方便 QwQ
    if(!b) {
        x = 1, y = 0;
        return a;
    }
    ll ans = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return ans;
}

ll inv(ll a, ll p) {  //求 a 模 p 的逆元
    ll x, y;
    exgcd(a, p, x, y);
    return (x % p + p) % p;
}

ll qpow(ll a, ll b, ll p) {  //快速幂
    ll ans = 1;
    while(b) {
        if(b & 1) ans = (ans * a) % p;
        a = (a * a) % p; b >>= 1;
    }
    return ans;
}

ll BSGS(ll a, ll b, ll p) {  //BSGS 主体，不解释了
    ll unit = (ll)ceil(sqrt(p)), tmp = qpow(a, unit, p);

    for(int i = 0; i <= unit; ++i)
        mp[b] = i, b = (b * a) % p;

    b = 1;

    for(int i = 1; i <= unit; ++i) {
        b = (b * tmp) % p;
        if(mp[b]) return i * unit - mp[b];
    }
```

```
    return -1;
}

ll exBSGS(ll a, ll b, ll p) {
    //特判几个特殊情况
    for(int i=0;i<=5;i++)
    {
        if(qpow(a,i,p)%p==b%p)
        {
            return i;
        }
    }
    ll x, y, g = exgcd(a, p, x, y), k = 0, tmp = 1;
    while(g != 1) {  //当 gcd(a, p) 不为 1 时，就不断除以 gcd(a, p) 直到 a 与 p 互质
        if(b % g) return -1;  //b 不能被 gcd(a, p) 整除，当然不互质
        ++k, b /= g, p /= g, tmp = tmp * (a / g) % p;
        if(tmp == b) return k;
        g = exgcd(a, p, x, y);
    }
    //用传统 BSGS 来解决问题
    ll ans = BSGS(a, b * inv(tmp, p) % p, p);
    if(ans == -1) return -1;
    return ans + k;
}
```

组合数学

1. 十二重计数法：

```
int ksm(int a,int b,int p)
{
    int ans=1;
    while(b>0)
    {
        if(b&1)ans=ans*a%p;
        a=a*a%p;
        b>>=1;
    }
    return ans;
}
int a[MAXN],inv[MAXN];//阶乘与逆元
void init(int n)
{
    a[0]=1;
    for(int i=1;i<=4e5;i++)a[i]=a[i-1]*i%mod;
    inv[0]=1,inv[(int)4e5]=ksm(a[(int)4e5],mod-2,mod);
    for(int i=4e5-1;i>=1;i--)inv[i]=inv[i+1]*(i+1)%mod;
}
int c(int n,int m)//组合数
{
```

```
    if(m<0 || m>n)return 0;
    return a[n]*inv[m]%mod*inv[n-m]%mod;
}
//有n个球和m个盒子，要全部装进盒子里。还有一些限制条件，那么有多少种方法放球?

//1.球之间互不相同，盒子之间也互不相同。
int count1(int n,int m)
{
    return ksm(m,n,mod);
}

//2.球之间互不相同，盒子之间互不相同，每个盒子至多装一个球。
int count2(int n,int m)
{
    if(n>m)return 0;
    return c(m,n)*a[n]%mod;
}

//3.球之间互不相同，盒子之间互不相同，每个盒子至少装一个球。(对有几个空盒进行容斥)
int count3(int n,int m)
{
    if(m>n)return 0;
    int ans=0,f=1;
    for(int i=0;i<=m;i++)
    {

        if((m-i)%2==1)ans=(ans+(mod-c(m,i)*ksm(i,n,mod)%mod)%mod)%mod;
        else ans=(ans+(c(m,i)*ksm(i,n,mod)%mod)%mod+mod)%mod;
    }
    return ans;
}


//4.球之间互不相同，盒子全部相同。 (第二类斯特林数行的和)
int count4(int n,int m)
{
    int limit=getl((n+1)<<1);
    vector<int> f1(limit),g1(limit);
    int sign=1;
    for(int i=0;i<=n;i++)
    {
        f1[i]=(sign*inv[i]+mod)%mod;
        g1[i]=ksm(i,n,mod)*inv[i]%mod;
        sign*=-1;
    }
    NTT(f1,limit,1);NTT(g1,limit,1);
    for(int i=0;i<limit;i++)f1[i]=f1[i]*g1[i]%mod;
    NTT(f1,limit,0);
    int ans=0;
    for(int i=0;i<=min(m,n);i++)ans=(ans+f1[i])%mod;
    return ans;
}

//5.球之间互不相同，盒子全部相同，每个盒子最多装一个。
```

```
int count5(int n,int m)
{
    if(n<=m)return 1;
    else return 0;
}

//6.球之间互不相同，盒子全部相同，每个盒子最少装一个。
int count6(int n,int m)
{
    if(m>n)return 0;
    int ans=0,f=1;
    for(int i=0;i<=m;i++)
    {
        if((m-i)%2==1)ans=(ans+(mod-inv[m-
i]*inv[i]%mod)*ksm(i,n,mod)%mod+mod)%mod;
        else ans=(ans+inv[m-i]*inv[i]%mod*ksm(i,n,mod)%mod+mod)%mod;
    }
    return ans;
}

//7.球相同，盒子不同
int count7(int n,int m)
{
    return c(n+m-1,m-1);
}

//8.球全部相同，盒子之间互不相同，每个盒子至多装一个球。
int count8(int n,int m)
{
    if(n>m)return 0;
    return c(m,n);
}

//9.球全部相同，盒子之间互不相同，每个盒子至少装一个球。
int count9(int n,int m)
{
    if(m>n)return 0;
    return c(n-1,m-1);
}
vector<int> f1(MAXN),g1(MAXN);
//10.球全部相同，盒子全部相同
int count10(int n,int m)
{
    int limit=getl(n+1);
    for(int i=1;i<=m;i++)
    {
        for(int j=i;j<=n;j+=i)
        {
            f1[j]+=ksm(j/i,mod-2);
        }
    }
    polyexp(f1,g1,limit);
    int t=g1[n];
    for(int i=0;i<limit;i++)f1[i]=g1[i]=0;
```

```
    return t;
}

//11.球全部相同，盒子全部相同，每个盒子至多装一个球。
int count11(int n,int m)
{
    if(m<n)return 0;
    return 1;
}

//12.球全部相同，盒子全部相同，每个盒子至少装一个球。
int count12(int n,int m)
{
    if(m>n)return 0;
    return count10(n-m,m);
}
```

线性代数

1. 矩阵与向量：

```
const int MAXN = 101;
#define T double
//向量直接vector表示

struct matrix//下标从1开始
{
    int n=101,m=101;//默认为100x100矩阵
    matrix(int x,int y)
    {
        n=x;m=y;
    }
    vector<vector<T>> a;
    void init(void)
    {
        for(int i=0;i<=n;i++)a.push_back(vector<T>(m+1));
    }
};
matrix operator+(matrix a,matrix b)
{
    int n=a.n,m=a.m;
    matrix res(n,m);
    res.init();
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)res.a[i][j]=a.a[i][j]+b.a[i][j];
    }
    return res;
}
matrix operator-(matrix a,matrix b)
{
```

```
    int n=a.n,m=a.m;
    matrix res(n,m);
    res.init();
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)res.a[i][j]=a.a[i][j]-b.a[i][j];
    }
    return res;
}
matrix operator*(matrix a,matrix b)
{
    int n1=a.n,m1=a.m,n2=b.n,m2=b.m;
    matrix res(n1,m2);
    res.init();
    for(int i=1;i<=n1;i++)
    {
        for(int j=1;j<=m2;j++)
        {
            for(int k=1;k<=m1;k++)res.a[i][j]+=a.a[i][k]*b.a[k][j];
        }
    }
    return res;
}
vector<T> operator*(vector<T> a,T k)//向量数乘
{
    vector<T> res(a.size());
    for(int i=0;i<a.size();i++)res[i]=a[i]*k;
    return res;
}
vector<T> operator-(vector<T> a,vector<T> b)//向量减法
{
    vector<T> res(a.size());
    for(int i=0;i<a.size();i++)res[i]=a[i]-b[i];
    return res;
}
vector<T> operator+(vector<T> a,vector<T> b)//向量减法
{
    vector<T> res(a.size());
    for(int i=0;i<a.size();i++)res[i]=a[i]+b[i];
    return res;
}
void show(matrix a)
{
    for(int i=1;i<=a.n;i++)
    {
        for(int j=1;j<=a.m;j++)
        {
            cout<<a.a[i][j]<<" ";
        }
        cout<<"\n";
    }
}
```

2. 高斯消元：

```
//利用增广矩阵进行高斯消元（可以直接输入）
//高斯消元可以用于求行列式的值，转化为对角线即可

const double esp = 1e-9;//处理精度问题

matrix operator+(matrix a,vector<int> b)//得到增广矩阵
{
    matrix res(a.n,a.m+1);
    for(int i=1;i<=a.n;i++)
    {
        for(int j=1;j<=res.m;j++)
        {
            if(j==res.m)
            {
                res.a[i][j]=b[i];
            }
            else res.a[i][j]=a.a[i][j];
        }
    }
    return res;
}

matrix Gauss(matrix a)
{
    int n=a.n,m=a.m-1;
    for(int i=1;i<=m;i++)
    {
        int tmp=-1;
        for(int j=1;j<=n;j++)
        {
            if(abs(a.a[j][i])>=esp)//不为0
            {
                int f=1;
                for(int k=1;k<=j-1;k++)
                {
                    if(abs(a.a[k][i])>=esp)
                    {
                        f=0;
                        break;
                    }
                }
                if(f)tmp=j;
                break;
            }
        }
        if(tmp==-1)continue;
        for(int j=1;j<tmp;j++)
        {
            if(abs(a.a[j][i])<esp)continue;
            a.a[j]=a.a[j]-a.a[tmp]*(a.a[j][i]/a.a[tmp][i]);
        }
```

```
        for(int j=tmp+1;j<=n;j++)
        {
            if(abs(a.a[j][i])<esp)continue;
            a.a[j]=a.a[j]-a.a[tmp]*(a.a[j][i]/a.a[tmp][i]);
        }
    }
    return a;
}
```

多项式与生成函数:

1. 多项式全家桶:

```
double pi = acos(-1.0);
const int mod=998244353,g=3,ig=332748118;//模数，原根，ig是g的逆元
//一般多项式长度不超过8e6可以用上面的模数
const int MAXN = 4e6+10;
int inv2=(mod+1)/2;

vector<int> t1(MAXN),t2(MAXN),t3(MAXN),t4(MAXN);

int ksm(int a,int b)
{
    int ans=1;
    while(b>0)
    {
        if(b&1)ans=ans*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return ans;
}

void fft(vector<complex<double>> &a,int n,int mode)
{
    if(n==1)return;
    vector<complex<double>> a1(n/2),a2(n/2);
    for(int i=0;i<n;i+=2)
    {
        a1[i>>1]=a[i];a2[i>>1]=a[i+1];
    }
    fft(a1,n>>1,mode);
    fft(a2,n>>1,mode);
    complex<double> Wn{cos(2.0*pi/n),mode*sin(2.0*pi/n)},k{1,0};//单位根,k次根

    for(int i=0;i<(n>>1);i++,k=k*Wn)
    {
        a[i]=a1[i]+k*a2[i];
        a[i+(n>>1)]=a1[i]-k*a2[i];
    }
}
```

```
//mode=1为DFT，mode=0为逆变换
//注意所有运算要在模mod意义下进行
/*
****** for(int i=0;i<limit;++i)rev[i]=(rev[i/2]/2+(i%2)*limit/2);
       rev数组处理方法，limit为数组元素上限，是2的幂次
*/
int rev[2100000];

int getl(int len)
{
    int n=1;
    while(n<=len)n<<=1;
    for(int i=0;i<n;i++)rev[i]=(rev[i>>1]>>1)|((i&1)?(n>>1):0);
    return n;
}
void NTT(vector<int> &x,int n,int mode)//每次NTT前调用getl进行初始化rev,n为2的幂次
{
    for(int i=0;i<n;i++)if(i<rev[i])swap(x[i],x[rev[i]]);
    for(int len=1;len<n;len<<=1)
    {
        int Wn=ksm(mode?g:ig,(mod-1)/(2*len));
        for(int i=0;i<n;i+=2*len)
        {
            int W=1,X,Y;
            for(int j=i;j<i+len;j++)
            {
                X=x[j];Y=W*x[j+len]%mod;
                x[j]=(X+Y)%mod;x[j+len]=(X-Y+mod)%mod;
                W=(W*Wn)%mod;
            }
        }
    }
    if(!mode)
    {
        int invs=ksm(n,mod-2);
        for(int i=0;i<n;i++)x[i]=(x[i]*invs)%mod;
    }
}

void polymul(vector<int> &a,vector<int> &b,vector<int> &c,int n,int m)//n为
deg(a), m为deg(b)  // c = a * b
{
    int limit=1,len=0;
    limit=getl(n+m);
    for(int i=0;i<=n;i++)t1[i]=a[i];
    for(int i=0;i<=m;i++)t2[i]=b[i];
    NTT(t1,limit,1);NTT(t2,limit,1);
    for(int i=0;i<limit;i++)t1[i]=t1[i]*t2[i]%mod;
    NTT(t1,limit,0);
    for(int i=0;i<=m+n;i++)c[i]=t1[i];
}
```

```
//以下所有的n均为2的幂次，即先求一次getl(deg(f)+1);
void polyinv(vector<int> &f,vector<int> &g,int n)//求f的乘法逆元，f->g
{
    if(n==1){g[0]=ksm(f[0],mod-2);return;}
    polyinv(f,g,n>>1);
    int l=getl(n);
    for(int i=0;i<n;i++)t1[i]=f[i],t2[i]=g[i];
    for(int i=n;i<l;i++)t1[i]=t2[i]=0;
    NTT(t1,l,1);NTT(t2,l,1);
    for(int i=0;i<l;i++)t1[i]=t1[i]*t2[i]%mod*t2[i]%mod;
    NTT(t1,l,0);
    for(int i=0;i<n;i++)g[i]=(2*g[i]-t1[i]+mod)%mod;
}

//n=getl(deg(f)+1)
void polysqrt(vector<int> &f,vector<int> &g,int n)
{
    if(n==1){g[0]=1;return;}//这里根据题目确定
    polysqrt(f,g,n>>1);
    int l=getl(n<<1);
    for(int i=0;i<n;i++)t3[i]=f[i],t4[i]=0;
    for(int i=n;i<l;i++)t3[i]=t4[i]=0;
    polyinv(g,t4,n);l=getl(n<<1);
    NTT(t3,l,1);NTT(t4,l,1);
    for(int i=0;i<l;i++)t3[i]=t3[i]*t4[i]%mod;
    NTT(t3,l,0);
    for(int i=0;i<n;i++)g[i]=(g[i]+t3[i])*inv2%mod;
}

//多项式对数函数
//n=getl(def(f)+1)
void polyln(vector<int> &f,vector<int> &g,int n)//g=ln(f) 一定要有[0]f=1否则没有符
合条件的
{
    g[0]=0;
    polyinv(f,t3,n);//求逆
    for(int i=0;i<n-1;i++)t4[i]=f[i+1]*(i+1)%mod;//求导
    int limit=n<<1;
    NTT(t3,limit,1);NTT(t4,limit,1);
    for(int i=0;i<limit;i++)t3[i]=t3[i]*t4[i]%mod;
    NTT(t3,limit,0);
    for(int i=1;i<n;i++)g[i]=t3[i-1]*ksm(i,mod-2)%mod;//积分
    for(int i=0;i<limit;i++)t3[i]=t4[i]=0;//求完后更新t3,t4为0，不然exp会错
}

vector<int> t5(MAXN),t6(MAXN);
//多项式指数函数
//n=getl(def(f)+1)
void polyexp(vector<int> &f,vector<int> &g,int n)//g=exp(f) 一定要有[0]f=0,否则没有
符合条件的
{
    if(n==1){g[0]=1;return;}
    polyexp(f,g,n>>1);
    polyln(g,t5,n);
```

```
    int l=getl(n<<1);
    for(int i=0;i<n;i++)t6[i]=f[i];
    for(int i=n;i<l;i++)t5[i]=t6[i]=0;
    NTT(t5,l,1);NTT(t6,l,1);NTT(g,l,1);
    for(int i=0;i<l;i++)g[i]=g[i]*(1-t5[i]+t6[i]+mod)%mod;
    NTT(g,l,0);
    for(int i=n;i<l;i++)g[i]=0;
}
```