多项式

FFT（快速傅里叶变换）

```
const int N=5e6+5;
const double PI=acos(-1);
struct Complex{
    double x,y;
    Complex(){}
    Complex(double x,double y):x(x),y(y){};
    Complex operator+(const Complex&a)const{return {x+a.x,y+a.y};}
    Complex operator-(const Complex&a)const{return {x-a.x,y-a.y};}
    Complex operator*(const Complex&a)const{return {x*a.x-y*a.y,x*a.y+y*a.x};}
};

int R[N],limit=1,L;
void FFT(Complex*a,int flag){
    for(int i=0;i<limit;i++){
        if(i<R[i])swap(a[i],a[R[i]]);
    }
    for(int mid=1;mid<limit;mid<<=1){
        Complex wn(cos(PI/mid),flag*sin(PI/mid));
        int pos=0,len=2*mid;
        while(pos<limit){
            Complex w(1,0);
            for(int i=0;i<mid;i++){
                Complex x=a[pos+i],y=w*a[pos+i+mid];
                a[pos+i]=x+y;
                a[pos+i+mid]=x-y;
                w=w*wn;
            }
            pos+=len;
        }
    }
    if(flag==1)return;
    for(int i=0;i<limit;i++){
        a[i].x/=limit;
    }
}

int n,m;
void solve(){
    cin>>n>>m;
    for(int i=0;i<=n;i++)cin>>a[i].x;
    for(int i=0;i<=m;i++)cin>>b[i].x;
    while(limit<=n+m){
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=((R[i>>1]>>1)|((i&1)<<(L-1)));
    }
```

```
    FFT(a,1);
    FFT(b,1);
    for(int i=0;i<limit;i++)a[i]=a[i]*b[i];
    FFT(a,-1);
    for(int i=0;i<=n+m;i++){
        cout<<(int)(a[i].x+0.5)<<' ';
    }
}
```

分治 NTT (vector版)

$f_i = \sum_{j=1}^{n} f_{i-j} * g_j, \qquad f_0 = 1$

```
// f_i = \sum_{j=1}^{n} {f_{i-j}*g_j}   |   f_0 = 1
VI g;
void cdq(VI&f,int l,int r){
    if(r==0)f[0]=1;
    if(l==r)return;
    int mid=(l+r)>>1;
    VI a(mid-l+1,0),b(r-mid,0);
    for(int i=0;i<=mid-l;i++)a[i]=f[i];
    for(int i=0;i<=r-mid-1;i++)b[i]=f[i+(mid-l+1)];
    cdq(a,l,mid);
    VI tmp(r-l+1);
    for(int i=0;i<=r-l;i++)tmp[i]=g[i];
    tmp=tmp*a;
    for(int i=0;i<=r-mid-1;i++)b[i]=(b[i]+tmp[i+(mid-l+1)])%mod;
    cdq(b,mid+1,r);
    for(int i=0;i<=mid-l;i++)f[i]=a[i];
    for(int i=0;i<=r-mid-1;i++)f[i+(mid-l+1)]=b[i];
}
```

FWT（快速沃尔什变换）

$c_k = \sum_{i|j=k} a_i b_j$

```
#define int long long
const int N=2e6+5;
const int mod=998244353;
void OR(int*f,int x,int n){     // [0,n-1]
    for(int len=2;len<=n;len*=2){
        for(int i=0;i<n;i+=len){
            int mid=len>>1;
            for(int j=0;j<mid;j++){
                f[i+j+mid]=(f[i+j+mid]+f[i+j]*x)%mod;
                f[i+j+mid]=(f[i+j+mid]+mod)%mod;
            }
        }
    }
}
void get_or(int*a,int*b,int n){   // or 卷积
    OR(a,1,n);
    OR(b,1,n);
    for(int i=0;i<n;i++)a[i]=a[i]*b[i]%mod;
```

```
    OR(a,-1,n);
    OR(b,-1,n);
}
void AND(int*f,int x,int n){     // [0,n-1]
    for(int len=2;len<=n;len*=2){
        for(int i=0;i<n;i+=len){
            int mid=len>>1;
            for(int j=0;j<mid;j++){
                f[i+j]=(f[i+j]+f[i+j+mid]*x)%mod;
                f[i+j]=(f[i+j]+mod)%mod;
            }
        }
    }
}
void get_and(int*a,int*b,int n){   // and 卷积
    AND(a,1,n);
    AND(b,1,n);
    for(int i=0;i<n;i++)a[i]=a[i]*b[i]%mod;
    AND(a,-1,n);
    AND(b,-1,n);
}
int inv_2=499122177;       // qpow(2,mod-2)
void XOR(int*f,int x,int n){     // [0,n-1]
    for(int len=2;len<=n;len*=2){
        for(int i=0;i<n;i+=len){
            int mid=len>>1;
            for(int j=0;j<mid;j++){
                f[i+j]=(f[i+j]+f[i+j+mid])%mod;
                f[i+j+mid]=(f[i+j]-2*f[i+j+mid]%mod)%mod;
                f[i+j+mid]=(f[i+j+mid]+mod)%mod;
                f[i+j]=f[i+j]*x%mod;
                f[i+j+mid]=f[i+j+mid]*x%mod;
            }
        }
    }
}
void get_xor(int*a,int*b,int n){   // xor 卷积
    XOR(a,1,n);
    XOR(b,1,n);
    for(int i=0;i<n;i++)a[i]=a[i]*b[i]%mod;
    XOR(a,inv_2,n);
    XOR(b,inv_2,n);
}
```

子集卷积

$c_k = \sum_{i|j=k,i\&j=0} a_i b_j$

```
// c_k=\sum_{i|j=k,i&j=0}{a_i*b_j}
const int mod=1e9+9;
void OR(int*f,int x,int n){     // [0,n-1]
    for(int len=2;len<=n;len*=2){
        for(int i=0;i<n;i+=len){
            int mid=len>>1;
            for(int j=0;j<mid;j++){
```

```
                f[i+j+mid]=(f[i+j+mid]+f[i+j]*x)%mod;
                f[i+j+mid]=(f[i+j+mid]+mod)%mod;
            }
        }
    }
}
int a[25][1<<20],b[25][1<<20],c[25][1<<20];
void solve(){
    int n;
    cin>>n;
    for(int i=0;i<(1<<n);i++){
        int x;
        cin>>x;
        a[__builtin_popcount(i)][i]=x;
    }
    for(int i=0;i<(1<<n);i++){
        int x;
        cin>>x;
        b[__builtin_popcount(i)][i]=x;
    }
    for(int i=0;i<=n;i++)OR(a[i],1,1<<n),OR(b[i],1,1<<n);
    for(int i=0;i<=n;i++){
        for(int j=0;j<=i;j++){
            for(int k=0;k<(1<<n);k++){
                c[i][k]=(c[i][k]+a[j][k]*b[i-j][k]%mod)%mod;
            }
        }
        OR(c[i],-1,1<<n);
    }
    for(int i=0;i<(1<<n);i++)cout<<c[__builtin_popcount(i)][i]<<' ';
}
```

固定模数NTT

```
/*
原根：3
模数：104857601, 167772161, 469762049, 469762049, 998244353, 1004535809,
2281701377
*/
#define int long long
const int N=5e6+5;
const int G=3,Gi=332748118,mod=998244353;
// Gi=qpow(G,mod-2)
int qpow(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;
}
int R[N],limit=1,L,a[N],b[N];
```

```
void NTT(int*a,int flag){
    for(int i=0;i<limit;i++){
        if(i<R[i])swap(a[i],a[R[i]]);
    }
    for(int mid=1;mid<limit;mid<<=1){
        int wn=qpow(G,(mod-1)/(2*mid));
        if(flag==-1)wn=qpow(wn,mod-2);
        int pos=0,len=2*mid;
        while(pos<limit){
            int w=1;
            for(int i=0;i<mid;i++){
                int x=a[pos+i],y=w*a[pos+i+mid]%mod;
                a[pos+i]=(x+y)%mod;
                a[pos+i+mid]=(x-y+mod)%mod;
                w=w*wn%mod;
            }
            pos+=len;
        }
    }
    if(flag==1)return;
    int inv=qpow(limit,mod-2);
    for(int i=0;i<limit;i++){
        a[i]=(a[i]*inv)%mod;
    }
}
int n,m;
void solve(){
    cin>>n>>m;
    for(int i=0;i<=n;i++){
        cin>>a[i];
        a[i]=(a[i]+mod)%mod;
    }
    for(int i=0;i<=m;i++){
        cin>>b[i];
        b[i]=(b[i]+mod)%mod;
    }
    while(limit<=n+m){
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=((R[i>>1]>>1)|((i&1)<<(L-1)));
    }
    NTT(a,1);
    NTT(b,1);
    for(int i=0;i<limit;i++)a[i]=a[i]*b[i]%mod;
    NTT(a,-1);
    for(int i=0;i<=n+m;i++){
        cout<<a[i]<<' ';
    }
}
```

任意模数NTT（三模数MTT）

```
#define int long long
const int N=5e5+5,G=3;
const int mod1=998244353,mod2=1004535809,mod3=469762049;
int qpow(int a,int b,int mod){
    int ans=1;
    while(b>0){
        if(b&1)
            ans=(ans*a)%mod;
        a=(a*a)%mod;
        b>>=1;
    }
    return ans;
}
int R[N],a[N],b[N],x[N],y[N],c1[N],c2[N],c3[N];
int limit=1,L;
void NTT(int*a,int flag,int mod){
    for(int i=0;i<limit;i++){
        if(i<R[i])swap(a[i],a[R[i]]);
    }
    for(int mid=1;mid<limit;mid<<=1){
        int wn=qpow(G,(mod-1)/(2*mid),mod);
        if(flag==-1)wn=qpow(wn,mod-2,mod);
        int pos=0,len=2*mid;
        while(pos<limit){
            int w=1;
            for(int i=0;i<mid;i++){
                int x=a[pos+i],y=w*a[pos+i+mid]%mod;
                a[pos+i]=(x+y)%mod;
                a[pos+i+mid]=(x-y+mod)%mod;
                w=w*wn%mod;
            }
            pos+=len;
        }
    }
    if(flag==1)return;
    int inv=qpow(limit,mod-2,mod);
    for(int i=0;i<limit;i++){
        a[i]=(a[i]*inv)%mod;
    }
}
int n,m,mod;
void MTT(int*cc,int mod){
    for(int i=0;i<limit;i++)x[i]=a[i];
    for(int i=0;i<limit;i++)y[i]=b[i];
    NTT(x,1,mod);
    NTT(y,1,mod);
    for(int i=0;i<limit;i++)cc[i]=x[i]*y[i]%mod;
    NTT(cc,-1,mod);
}
void solve(){
    cin>>n>>m>>mod;
    for(int i=0;i<=n;i++)cin>>a[i];
    for(int i=0;i<=m;i++)cin>>b[i];
```

```
    while(limit<=n+m){
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=((R[i>>1]>>1)|((i&1)<<(L-1)));
    }
    MTT(c1,mod1);
    MTT(c2,mod2);
    MTT(c3,mod3);
    int mod4=mod1*mod2;
    int inv1=qpow(mod1,mod2-2,mod2),inv2=qpow(mod4%mod3,mod3-2,mod3);
    for(int i=0;i<limit;i++){
        c1[i]=(((c2[i]-c1[i])%mod2+mod2)%mod2*inv1%mod2*mod1+c1[i])%mod4;
        c3[i]=(((c3[i]-c1[i])%mod3+mod3)%mod3*inv2%mod3*
(mod4%mod)%mod+c1[i]%mod)%mod;
    }
    for(int i=0;i<=n+m;i++){
        cout<<c3[i]<<' ';
    }
}
```

任意模数NTT（拆系数FFT）

```
#define int long long
#define double long double
#define VI vector<int>
const int LG=18;
const int N=(1<<LG);
const double PI=acos(-1);
struct Complex{
    double x,y;
    Complex(){}
    Complex(double x,double y):x(x),y(y){};
    Complex operator+(const Complex&a)const{return {x+a.x,y+a.y};}
    Complex operator-(const Complex&a)const{return {x-a.x,y-a.y};}
    Complex operator*(const Complex&a)const{return {x*a.x-y*a.y,x*a.y+y*a.x};}
};
int R[N],mod;    // 非质数模数
Complex p1[N],p2[N],p3[N];
Complex w[N]={Complex(1.0,0.0)};
void FFT(Complex*a,int op,int n){
    for(int i=0;i<n;i++)
        if(i<R[i])swap(a[i],a[R[i]]);
    for(int mid=1;mid<n;mid<<=1){
        Complex wn=Complex(cos(PI/mid),sin(PI/mid)*op);
        for(int i=mid-2;i>=0;i-=2)w[i]=w[i>>1],w[i+1]=w[i>>1]*wn;
        for(int k=0;k<n;k+=2*mid){
            for(int p=0;p<mid;p++){
                Complex sav=w[p]*a[k|mid|p];
                a[k|mid|p]=a[k|p]-sav;
                a[k|p]=a[k|p]+sav;
            }
        }
    }
```

```
}
VI operator*(VI a,VI b){
    int n=a.size()+b.size()-1;
    int s=(1<<__lg(2*n-1));
    a.resize(s),b.resize(s);
    for(int i=0;i<s;i++){
        p1[i]=Complex(a[i]>>15,a[i]&32767);
        p2[i]=Complex(a[i]>>15,-(a[i]&32767));
        p3[i]=Complex(b[i]>>15,b[i]&32767);
    }
    for(int i=1;i<s;i++)
        R[i]=(R[i>>1]>>1) | ((i&1)?(s>>1):0);
    FFT(p1,1,s),FFT(p2,1,s),FFT(p3,1,s);
    for(int i=0;i<s;i++){
        p3[i].x/=s,p3[i].y/=s;
        p1[i]=p1[i]*p3[i];
        p2[i]=p2[i]*p3[i];
    }
    FFT(p1,-1,s),FFT(p2,-1,s);
    for(int i=0;i<n;i++){
        int u,v,p,q;
        u=(int)floor((p1[i].x+p2[i].x)/2+0.5)%mod;
        v=(int)floor((p1[i].y+p2[i].y)/2+0.5)%mod;
        p=((int)floor(p1[i].y+0.5)-v)%mod;
        q=((int)floor(p2[i].x+0.5)-u)%mod;
        a[i]=(( ((u<<15)+v+p) << 15)+q)%mod;
        if(a[i]<0)a[i]+=mod;
    }
    a.resize(n);
    return a;
}
void solve(){
    int n,m;
    cin>>n>>m>>mod;
    VI a(n+1),b(m+1);
    for(int i=0;i<=n;i++)cin>>a[i];
    for(int i=0;i<=m;i++)cin>>b[i];
    VI ans=a*b;
    for(int i=0;i<=n+m;i++)cout<<ans[i]<<' ';
}
```

拉格朗日插值

```
#define int long long
const int mod=998244353;
const int N=2e4+5;
int n,inv[N];
int tmp[N],num[N],res[N],x[N],y[N];
int qpow(int a,int b){
    int ans=1;
    while(b){
        if(b&1)ans=(ans*a)%mod;
        a=(a*a)%mod;
        b>>=1;
    }
```

```
    return ans;
}
void pre(){     //预处理
    num[0]=1;
    for(int i=1;i<=n;i++){
        tmp[0]=0;
        inv[i]=qpow(mod-x[i],mod-2);
        for(int j=1;j<=i;j++){
            tmp[j]=num[j-1];
        }
        for(int j=0;j<=i;j++){
            tmp[j]+=num[j]*(mod-x[i])%mod;
            tmp[j]%=mod;
        }
        swap(num,tmp);
    }
}
void lglr(){        //求出多项式系数
    pre();
    for(int i=1;i<=n;i++){
        int tt=1,lst=0;
        for(int j=1;j<=n;j++){
            if(i==j)continue;
            tt=(tt*(x[i]-x[j]+mod))%mod;
        }
        tt=y[i]*qpow(tt,mod-2)%mod;
        for(int j=0;j<n;j++){
            tmp[j]=(num[j]-lst+mod)*inv[i]%mod;
            res[j]+=tt*tmp[j]%mod;
            res[j]%=mod;
            lst=tmp[j];
        }
    }
}
int cal(int k){     //计算多项式单点值
    int ans=0,tmp=1;
    for(int i=0;i<n;i++){
        ans=(ans+res[i]*tmp)%mod;
        tmp=tmp*k%mod;
    }
    return ans;
}
void solve(){
    int k;
    cin>>n>>k;
    for(int i=1;i<=n;i++){
        cin>>x[i]>>y[i];
    }
    lglr();
    cout<<cal(k)<<endl;
    return;
}
```

连续拉格朗日插值

```
int f(int x){}      // 多项式 f(x)
int inv[N],pre[N],suf[N];
void init(){    // 预处理逆元
    inv[0]=inv[1]=1;
    for(int i=2;i<N;i++)inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    for(int i=2;i<N;i++)inv[i]=inv[i]*inv[i-1]%mod;
}
// k+1 个点插出 k 次多项式
int cal(int x,int*val,int k){     // 求 k 次多项式 f(x) 的值
    // 处理前缀积、后缀积
    pre[0]=1,pre[1]=x-1;
    for(int i=2;i<=k+1;i++)pre[i]=pre[i-1]*(x-i)%mod;
    suf[k+2]=1,suf[k+1]=x-(k+1);    // 需要 k+1 个点
    for(int i=k;i>=1;i--)suf[i]=suf[i+1]*(x-i)%mod;
    int ans=0;
    for(int i=1;i<=k+1;i++){
        int op=1;
        if((k+1-i)%2==1)op=mod-1;
        ans+=op*val[i]%mod*pre[i-1]%mod*suf[i+1]%mod*inv[i-1]%mod*inv[k+1-
i]%mod;
        ans%=mod;
    }
    return (ans+mod)%mod;
}
int val[N];
void solve(){
    int x,k;
    cin>>x>>k;
    for(int i=1;i<=k+1;i++)val[i]=f(i); // 求 k+1 个点值
    init();
    cout<<cal(x,val,k)<<endl;
}

//  下标为 [0,k]
int cal(int x,int*val,int k){     // 求 k 次多项式 f(x) 的值
    // 处理前缀积、后缀积
    pre[0]=x;
    for(int i=1;i<=k;i++)pre[i]=pre[i-1]*(x-i)%mod;
    suf[k+1]=1; // 需要 k+1 个点
    for(int i=k;i>=0;i--)suf[i]=suf[i+1]*(x-i)%mod;
    int ans=0;
    for(int i=0;i<=k;i++){
        int op=1;
        if((k-i)%2==1)op=mod-1;
        if(i>0)op=op*pre[i-1]%mod;
        ans+=op*val[i]%mod*suf[i+1]%mod*inv[i]%mod*inv[k-i]%mod;
        ans%=mod;
    }
    return (ans+mod)%mod;
}
```

多项式乘法

```
void get_mul(int*a,int*b,int n,int m){      //n次*m次
    limit=1,L=0;
    while(limit<=2*(n+m)){
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));
    }
    NTT(a,1);
    NTT(b,1);
    for(int i=0;i<limit;i++)a[i]=a[i]*b[i]%mod;
    NTT(a,-1);
    NTT(b,-1);
}
void get_mul(Complex*a,Complex*b,int deg){      //FFT多项式乘法
    limit=1,L=0;
    while(limit<=deg){
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));
    }
    FFT(a,1);
    FFT(b,1);
    for(int i=0;i<limit;i++)a[i]=a[i]*b[i];
    FFT(a,-1);
    FFT(b,-1);
}
```

多项式求导积分

```
void get_dev(int*a,int*b,int n){    //多项式求导
    for(int i=1;i<n;i++){
        b[i-1]=i*a[i]%mod;
    }
    b[n-1]=0;
}
void get_indev(int*a,int*b,int n){      //多项式求积分
    for(int i=1;i<n;i++){
        b[i]=a[i-1]*qpow(i,mod-2)%mod;
    }
    b[0]=0;
}
```

多项式全家桶

```
#define int long long
const int N=7e5+5,G=3;
const int mod=998244353;
int qpow(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;
}
int R[N],a[N],b[N],f[N],g[N],c[N],aa[N],bb[N];
int limit=1,L;
void NTT(int*a,int flag){
    for(int i=0;i<limit;i++){
        if(i<R[i])swap(a[i],a[R[i]]);
    }
    for(int mid=1;mid<limit;mid<<=1){
        int wn=qpow(G,(mod-1)/(2*mid));
        if(flag==-1)wn=qpow(wn,mod-2);
        int pos=0,len=2*mid;
        while(pos<limit){
            int w=1;
            for(int i=0;i<mid;i++){
                int x=a[pos+i],y=w*a[pos+i+mid]%mod;
                a[pos+i]=(x+y)%mod;
                a[pos+i+mid]=(x-y+mod)%mod;
                w=w*wn%mod;
            }
            pos+=len;
        }
    }
    if(flag==1)return;
    int inv=qpow(limit,mod-2);
    for(int i=0;i<limit;i++){
        a[i]=(a[i]*inv)%mod;
    }
}
void get_inv(int*a,int*b,int deg){      //多项式求逆
    if(deg==1){
        b[0]=qpow(a[0],mod-2);
        return;
    }
    get_inv(a,b,(deg+1)/2);
    limit=1;
    while(limit<=(deg<<1)){
        limit<<=1;
    }
    for(int i=0;i<limit;i++){
        R[i]=(R[i>>1]>>1)|((i&1)?(limit>>1):0);
        c[i]=(i<deg?a[i]:0);
    }
    NTT(b,1);
    NTT(c,1);
    for(int i=0;i<limit;i++){
        b[i]=(2ll-c[i]*b[i]%mod+mod)%mod*b[i]%mod;
    }
```

```
    NTT(b,-1);
    fill(b+deg,b+limit,0);
}
void get_dev(int*a,int*b,int n){    //多项式求导
    for(int i=1;i<n;i++){
        b[i-1]=i*a[i]%mod;
    }
    b[n-1]=0;
}
void get_indev(int*a,int*b,int n){      //多项式求积分
    for(int i=1;i<n;i++){
        b[i]=a[i-1]*qpow(i,mod-2)%mod;
    }
    b[0]=0;
}
const int inv2=499122177;       // 模意义下的 1/2
void get_sqrt(int*a,int*b,int deg){       // a[0]=1时，多项式开根
    if(deg==1){
        b[0]=1;
        return;
    }
    get_sqrt(a,b,(deg+1)>>1);
    limit=1;
    while(limit<2*deg)limit<<=1;
    fill(f,f+limit,0);
    get_inv(b,f,deg);
    copy(a,a+deg,c);
    fill(c+deg,c+limit,0);
    NTT(b,1);
    NTT(c,1);
    NTT(f,1);
    for(int i=0;i<limit;i++){
        b[i]=inv2*(b[i]+c[i]*f[i]%mod)%mod;
    }
    NTT(b,-1);
    fill(b+deg,b+limit,0);
}
void get_mul(int*a,int*b,int deg){      //多项式乘法
    limit=1,L=0;
    while(limit<=deg){
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));
    }
    NTT(a,1);
    NTT(b,1);
    for(int i=0;i<limit;i++)a[i]=a[i]*b[i]%mod;
    NTT(a,-1);
    NTT(b,-1);
}
void get_mul(int*a,int*b,int n,int m){      //n次*m次 多项式乘法
    limit=1,L=0;
    while(limit<=2*(n+m)){
```

```
        limit<<=1;
        L++;
    }
    for(int i=0;i<limit;i++){
        R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));
    }
    NTT(a,1);
    NTT(b,1);
    for(int i=0;i<limit;i++)a[i]=a[i]*b[i]%mod;
    NTT(a,-1);
    NTT(b,-1);
}
int ar[N],br[N],ar_inv[N],br_inv[N];              //多项式除法
void get_div(int*a,int*b,int*f,int*g,int n,int m){      // f=a/b g=a%b
    for(int i=0;i<=n;i++)ar[i]=a[n-i];
    for(int i=0;i<=n-m+1;i++)br[i]=b[m-i];
    for(int i=n-m+2;i<=m;i++)br[i]=0;
    get_inv(br,br_inv,n-m+1);
    get_mul(ar,br_inv,n,n-m);
    for(int i=0;i<=n-m;i++)f[i]=ar[n-m-i];
    get_mul(b,f,m,n-m);
    for(int i=0;i<m;i++)g[i]=(a[i]-b[i]+mod)%mod;
}
void get_ln(int*f,int*g,int n){          //多项式对数函数
    get_dev(f,a,n);
    get_inv(f,b,n);
    get_mul(a,b,n);
    get_indev(a,g,n);
    for(int i=0;i<=(2*n);i++){
        a[i]=0,b[i]=0;
    }
}
void get_exp(int*a,int*b,int deg){       //多项式指数函数
    if(deg==1){
        b[0]=1;
        return;
    }
    get_exp(a,b,(deg+1)>>1);
    get_ln(b,aa,deg);
    aa[0]=(a[0]+1-aa[0]+mod)%mod;
    for(int i=1;i<deg;i++){
        aa[i]=(a[i]-aa[i]+mod)%mod;
    }
    get_mul(b,aa,deg);
    for(int i=deg;i<(2*deg);i++){
        b[i]=0,aa[i]=0;
    }
}
void get_qpow(int*a,int*b,int deg,int k){     //多项式快速幂
    get_ln(a,bb,deg);
    for(int i=0;i<deg;i++){
        bb[i]=bb[i]*k%mod;
    }
    for(int i=deg;i<=(deg<<1);i++)bb[i]=0;
    get_exp(bb,b,deg);
```

```
}
```

多项式Vector板子

```
#define int long long
#define VI vector<int>
const int LG=21; //内存参数，约为log(多项式长度)      LG==21 -> 30MB (注意调节)
const int N=(1<<LG);    // 有用的，不能乱设
const int mod=998244353;
int qpow(int a,int b){
    int ans=1;
    while(b){
        if(b&1)ans=ans*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return ans;
}
int w[N],inv[N];
int cp=[]{
    w[N/2]=1;
    int s=qpow(3,(mod-1)/N);
    for(int i=N/2+1;i<N;i++){
        w[i]=s*w[i-1]%mod;
    }
    for(int i=N/2-1;i>0;i--){
        w[i]=w[i*2];
    }
    inv[1]=1;
    for(int i=2;i<N;i++)
        inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    return 0;
}();
void dft(int*a,int n){
    for(int k=n/2;k;k>>=1)
    for(int i=0;i<n;i+=2*k)
    for(int j=0;j<k;j++){
        int t=a[i+j+k];
        a[i+j+k]=(a[i+j]-t+mod)*w[k+j]%mod;
        a[i+j]=(a[i+j]+t)%mod;
    }
}
void idft(int*a,int n){
    for(int k=1;k<n;k*=2)
    for(int i=0;i<n;i+=2*k)
    for(int j=0;j<k;j++){
        int u=a[i+j],v=a[i+j+k]*w[j+k]%mod;
        a[i+j+k]=(u-v+mod)%mod;
        a[i+j]=(u+v)%mod;
    }
    for(int i=0;i<n;i++)
        a[i]=a[i]*(mod-(mod-1)/n)%mod;
    reverse(a+1,a+n);
}
VI operator*(VI a,VI b){
```

```
    int n=a.size()+b.size()-1;
    int s=(1<<__lg(2*n-1));
    a.resize(s),b.resize(s);
    dft(a.data(),a.size());
    dft(b.data(),b.size());
    for(int i=0;i<s;i++){
        a[i]=a[i]*b[i]%mod;
    }
    idft(a.data(),a.size());
    a.resize(n);
    return a;
}
//注意 vector 表示多项式中，第 0 项表示常数项
VI get_inv(const VI&a){
    int n=a.size();
    if(n==1)return {qpow(a[0],mod-2)};
    VI a2(a.begin(),a.begin()+(n+1)/2);
    VI b=get_inv(a2),c=a*b;
    for(int i=0;i<c.size();i++)c[i]=(mod-c[i])%mod;
    c[0]=(c[0]+2)%mod;
    c=c*b;
    c.resize(n);
    return c;
}
VI get_ln(const VI&a){
    int n=a.size();
    VI b(n,0);
    for(int i=1;i<n;i++)
        b[i-1]=a[i]*i%mod;
    b=b*get_inv(a);
    b.resize(n);
    for(int i=n-1;i;i--)
        b[i]=b[i-1]*inv[i]%mod;
    b[0]=0;
    return b;
}
VI get_exp(const VI&a){
    int n=a.size();
    if(n==1)return {1};
    VI a2(a.begin(),a.begin()+(n+1)/2);
    VI b=get_exp(a2);
    b.resize(n);
    VI c=get_ln(b);
    for(int i=0;i<n;i++)
        c[i]=(a[i]-c[i]+mod)%mod;
    c[0]=(c[0]+1)%mod;
    c=c*b;
    c.resize(n);
    return c;
}
void get_div(const VI&a,const VI&b,VI&f,VI&g){
    //  f=a/b   |   g=a%b
    int n=a.size()-1,m=b.size()-1;
    VI ar(n+1),br(n-m+2,0);
    for(int i=0;i<=n;i++)ar[i]=a[n-i];
```

```
    for(int i=0;i<=min(m,n-m+1);i++)br[i]=b[m-i];
    br=get_inv(br);
    ar=ar*br;
    f.resize(n-m+1);
    for(int i=0;i<=n-m;i++)f[i]=ar[n-m-i];
    br=b*f;
    g.resize(m);
    for(int i=0;i<m;i++){
        g[i]=(a[i]-br[i]+mod)%mod;
    }
}
VI get_mod(const VI&a,const VI&b){
    VI f,g;
    int n=a.size()-1,m=b.size()-1;
    if(n<m)return a;
    VI ar(n+1),br(n-m+2,0);
    for(int i=0;i<=n;i++)ar[i]=a[n-i];
    for(int i=0;i<=min(m,n-m+1);i++)br[i]=b[m-i];
    br=get_inv(br);
    ar=ar*br;
    f.resize(n-m+1);
    for(int i=0;i<=n-m;i++)f[i]=ar[n-m-i];
    br=b*f;
    g.resize(m);
    for(int i=0;i<m;i++)g[i]=(a[i]-br[i]+mod)%mod;
    return g;
}
VI get_sqrt(const VI&a){
    int n=a.size();
    //  如果 a_0 不等于 1 ，要求 a_0 的二次剩余
    if(n==1)return {1};
    VI a2(a.begin(),a.begin()+(n+1)/2);
    VI b=get_sqrt(a2);
    b.resize(n);
    VI f=b*b;
    f.resize(n);
    for(int i=0;i<n;i++)f[i]=(f[i]+a[i])%mod;
    for(auto&i:b)i=i*2%mod;
    f=f*get_inv(b);
    f.resize(n);
    return f;
}
VI get_dev(const VI&a){         // 求导
    int n=a.size();
    VI b(n-1);
    for(int i=1;i<n;i++)b[i-1]=a[i]*i%mod;
    return b;
}
VI get_ksm(VI&a,string k){          // 传字符串
    int x=0,n=a.size(),len=k.size();
    while(x<n&&!a[x])x++;
    int k1=0,k2=0;
    for(int i=0;i<len;i++){
        k1=(k1*10%mod+k[i]-'0')%mod;
        k2=(k2*10+k[i]-'0')%(mod-1);
```

```
    }
    if(x && len>=6 || k1*x>=n)return vector<int>(n,0);
    for(int i=0;i<n-x;i++)a[i]=a[i+x];
    for(int i=n-x;i<n;i++)a[i]=0;
    int a0=a[0],y=qpow(a[0],mod-2);
    a.resize(n-x*k1);
    for(int i=0;i<a.size();i++)
        a[i]=a[i]*y%mod;
    a=get_ln(a);
    for(int i=0;i<a.size();i++)
        a[i]=a[i]*k1%mod;
    a=get_exp(a);
    y=qpow(a0,k2);
    a.resize(n);
    for(int i=n-1;i>=k1*x;i--)
        a[i]=a[i-k1*x]*y%mod;
    for(int i=0;i<k1*x;i++)a[i]=0;
    return a;
}
```

快速多点求值

```
// O(nlog^2(n)) 多点求值
struct Eval{
    #define ls (p<<1)
    #define rs ((p<<1)|1)
    VI ans,vec[4*N],t;
    void build(int p,int l,int r){
        if(l==r){
            vec[p]={1,(mod-t[l])%mod};
            return;
        }
        int mid=(l+r)>>1;
        build(ls,l,mid);
        build(rs,mid+1,r);
        vec[p]=vec[ls]*vec[rs];
    }
    VI mul(VI a,VI b){
        int n=a.size(),m=b.size();
        reverse(b.begin(), b.end());
        b=b*a;
        for(int i=0;i<n;i++)a[i]=b[i+m-1];
        return a;
    }
    void dfs(int p,int l,int r,VI f){
        f.resize(r-l+1);
        if(l==r){
            ans[l]=f[0];
            return;
        }
        int mid=(l+r)>>1;
        // 注意转置时不要写错儿子
        dfs(ls,l,mid,mul(f,vec[rs]));
        dfs(rs,mid+1,r,mul(f,vec[ls]));
    }
```

```
    // f 是多项式，g 是要求点值对应的 x 数组
    VI get_ans(VI f,VI g){
        int n=max(f.size(),g.size());
        g.resize(n);
        ans.resize(n);
        t=g;
        build(1,0,n-1);
        vec[1]=get_inv(vec[1]);
        dfs(1,0,n-1,mul(f,vec[1]));
        return ans;
    }
}eval;
```

快速插值

```
// O(nlog^2(n)) 快速插值
// 注意：N 不能过大，会 RE，(1<<18) 较合适
struct Interpolation{
    VI vec[4*N],ans[4*N],x,y,g;
    void build(int p,int l,int r){
        if(l==r){
            vec[p]={-x[l],1};
            return;
        }
        int mid=(l+r)>>1;
        build(ls,l,mid);
        build(rs,mid+1,r);
        vec[p]=vec[ls]*vec[rs];
    }
    void dfs(int p,int l,int r){
        if(l==r){
            ans[p]={y[l]*qpow(g[l],mod-2)%mod};
            return;
        }
        int mid=(l+r)>>1;
        dfs(ls,l,mid);
        dfs(rs,mid+1,r);
        ans[p]=ans[ls]*vec[rs]+ans[rs]*vec[ls];
    }
    // 点对为 (xx,yy)，返回一个 n-1 次多项式
    VI get_ans(const VI&xx,const VI&yy){
        x=xx,y=yy;
        int n=x.size();
        build(1,0,n-1);
        g=vec[1];
        for(int i=1;i<=n;i++)g[i-1]=i*g[i]%mod;
        g.resize(n);
        Eval eval;
        g=eval.get_ans(g,x);
        dfs(1,0,n-1);
        return ans[1];
    }
}inter;
```