

多项式基础操作

by zhouzhendong

扯淡

Q: 为什么要写这个讲稿？

A: 趁没有忘记赶紧记下来.....

Q: 那你要写什么内容啊？

A: 大纲见下一页.....

大纲

以下是本文主要内容。

1. 多项式求逆
2. 牛顿迭代
3. 二次剩余
4. 多项式开根
5. 多项式对数函数
6. 多项式指数函数
7. 多项式快速幂
8. 多项式除法、取模
9. 多点求值与快速插值

前置技能

1. 快速傅里叶变换(FFT) 和 快速数论变换(NTT)

<https://www.cnblogs.com/zhouzhendong/p/Fast-Fourier-Transform.html>

2. 微积分基础

..... 凉凉，只有一篇半途弃坑的.....

<https://www.cnblogs.com/zhouzhendong/p/Calculus.html>

需要会多项式求导、多项式积分。

FFT (其实是 NTT)

```
int w[N+1], Rev[N+1], A[N+1], B[N+1];
void FFT(int a[], int n){
    for (int i=0; i<n; i++)
        if (Rev[i]<i)
            swap(a[i], a[Rev[i]]);
    for (int t=Log[n]-1, d=1; d<n; t--, d<<=1)
        for (int i=0; i<n; i+=(d<<1))
            for (int j=0; j<d; j++){
                int tmp=(LL)w[j<<t]*a[i+j+d]%mod;
                a[i+j+d]=del(a[i+j], tmp);
                Add(a[i+j], tmp);
            }
}
```

小技巧：寻址优化（卡常数）

```
void init(int _n){
    Log[1]=0,n=_n;
    for (int i=2;i<=N;i++)
        Log[i]=Log[i>>1]+1;
    for (int d=0;d<=Log[n];d++){
        w[d]=Ew,iw[d]=Ei;
        int n=1<<d;
        w[d][0]=1,w[d][1]=Pow(3,(mod-1)/n);
        for (int i=2;i<n;i++)
            w[d][i]=(LL)w[d][i-1]*w[d][1]%mod;
        iw[d][0]=1,iw[d][1]=Pow(w[d][1],mod-2);
        for (int i=2;i<n;i++)
            iw[d][i]=(LL)iw[d][i-1]*iw[d][1]%mod;
        Ew+=n,Ei+=n;
    }
    int Rev[N+1],A[N+1],B[N+1];
    void FFT(int a[],int n,int **w){
        if (!INIT_TAG)
            init(N);
        for (int i=0;i<n;i++)
            if (Rev[i]<i)
                swap(a[i],a[Rev[i]]);
        for (int t=1,d=1;d<n;t++,d<<1)
            for (int i=0;i<n;i+=(d<<1))
                for (int j=0,*W=w[t];j<d;j++){
                    int tmp=(LL)(*W++)*a[i+j+d]%mod;
                    a[i+j+d]=del(a[i+j],tmp);
                    Add(a[i+j],tmp);
                }
    }
}
```

多项式求导和积分

```
Poly Derivation(Poly A){
    for (int i=0;i<A.size()-1;i++)
        A[i]=(LL)A[i+1]*(i+1)%mod;
    A.pop_back();
    return A;
}
Poly Integral(Poly A){
    A.push_back(0);
    for (int i=A.size()-2;i>=0;i--)
        A[i+1]=(LL)A[i]*Fast :: Inv[i+1]%mod;
    A[0]=0;
    return A;
}
```

注：未知函数或变量参见最后的总模板。

一些记号

为了便于描述，这里将用一下记号来表述相应的含义。

$A(x)$: 多项式 A

$A[i]$: 多项式 A 的 i 次项

$A_2(x)$: 多项式 A 更新后的结果

多项式求逆

多项式求逆

给定多项式 $B(x)$ ，求出多项式 $A(x)$ ，使得

$$A(x)B(x) \equiv 1 \pmod{x^n}$$

其中多项式 $A(x)$ 、 $B(x)$ 只需要保留前 n 项（也就是 x^0 到 x^{n-1} ）

多项式求逆

假设

$$A(x)B(x) \equiv 1 \pmod{x^{2n}}$$

则

$$(A(x)B(x) - 1)^2 \equiv 0 \pmod{x^{2n}}$$

$$A^2(x)B^2(x) - 2A(x)B(x) + 1 \equiv 0 \pmod{x^{2n}}$$

左右同除以 $B(x)$ ，得

$$A^2(x)B(x) - 2A(x) + A_2(x) \equiv 0 \pmod{x^{2n}}$$

$$A_2(x) \equiv 2A(x) - A^2(x)B(x) \pmod{x^{2n}}$$

多项式求逆

$$A_2(x) \equiv 2A(x) - A^2(x)B(x) \pmod{x^{2n}}$$

从常数项开始迭代，直到次数超过需要的 n 即可。

初始时 $A[0] = B[0]^{-1}$

时间复杂度

$$T(n) = T(n/2) + O(n \log n)$$

$$= O(n \log n)$$

多项式求逆

```
Poly Inverse(Poly a,int n){
    static Poly A,B;
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty())
        return a;
    A.clear(),B.clear();
    B.push_back(a[0]);
    A.push_back(Pow(B[0],mod-2));
    for (int t=1;t<n;){
        for (int i=t;i<min(a.size(),(t<<1));i++)
            B.push_back(a[i]);
        t<<=1;
        A=A*2-A*A*B;
        while (A.size()>t)
            A.pop_back();
    }
    while (A.size()>n)
        A.pop_back();
    return A;
}
```

牛顿迭代

牛顿迭代

设有可导函数 $F(x)$ ，我们想要快速求其零点。

牛顿迭代：

先任选一个 x ，作为起点。接下来重复进行以下过程：

1. 求出 $F(x)$ 在 x 处的切线斜率，即 $F'(x)$ 。
2. 求出该切线与 x 轴交点的 x 坐标，即 $x' = x - \frac{F(x)}{F'(x)}$ 。

对于大多数函数，每一次牛顿迭代， x 的有效数位会增加一倍。也就是说大于只要 $O(\log n)$ 次牛顿迭代就可以得到零点。这里 n 为精度要求的数位个数。

牛顿迭代

现在我们将一个数值 x 变成一个多项式 $A(x)$ 。

在求 $F(A(x)) = 0$ 的时候，也只需要记住这个公式就好了：

$$A_2(x) = A(x) - \frac{F(A(x))}{F'(A(x))}$$

二次剩余

二次剩余

这里讲二次剩余看似很扯淡啊！

其实是为了多项式开根做准备。~~(主要是我懒得专门为这个东西写一份讲稿.....)~~

给定数 x ，在模质数 p 意义下，求 a ，使得

$$a^2 \equiv x \pmod{p}$$

或判定无解。

二次剩余

首先直接把原根搞出来然后用高次剩余的做法 BSGS 一下是可以在 $O(\sqrt{n})$ 内解决的。

但是二次剩余有更加优秀的做法
--Cipolla's algorithm

定义 勒让德符号：

$$\left(\frac{a}{p} \right)$$

当 $a \equiv 0 \pmod{p}$ 时, $\left(\frac{a}{p} \right) = 0$;

当 a 为 p 的二次剩余时, $\left(\frac{a}{p} \right) = 1$;

否则 $\left(\frac{a}{p} \right) = -1$ 。

二次剩余

首先如果 p 是 2，那么比较 simple，不去管他。
那么 p 就是一个奇质数。

定理1

模 p 意义下有 $\frac{p+1}{2}$ 个数有二次剩余。

证明：

设 $0 < a < b < p$, 如果 $a^2 \equiv b^2 \pmod{p}$, 则

$(a+b)(a-b) \equiv 0 \pmod{p}$, 由于 $a \neq b$, 所以 $a+b = p$

。可见在对于 p 取模意义下, 0 有一个根, 剩下每一个二次剩余都有 2 个平方根, 且这些平方根互不相同, 并覆盖了 $[0, p)$, 所以共有 $\frac{p-1}{2}$ 个非零二次剩余, 算上 0 就是 $\frac{p+1}{2}$ 个。

二次剩余

定理2

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

证明：

当 $a = 0$ 时，显然成立。

当 $a \neq 0$ 时，由费马小定理可得 $a^{p-1} \equiv 1 \pmod{p}$ 。

若 a 存在二次剩余 r ，则 $a^{\frac{p-1}{2}} \equiv r^{p-1} \equiv 1 \pmod{p}$ ；

若 a 没有二次剩余，由于 $(a^{\frac{p-1}{2}})^2 \equiv 1 \pmod{p}$ ，所以

$a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}$ ，设 g 为模 p 意义下的任一原根，令

$a = g^{2k+1} (k \in \mathbb{Z})$ ，则

$$a^{\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}(2k+1)} \equiv g^{\frac{p-1}{2}} \equiv -1 \pmod{p}$$

二次剩余

定理3

$$(a + b)^p \equiv a^p + b^p \pmod{p}$$

这.....这不是直接费马小定理就好了么？

NO!

这里的 a 和 b 不是整数。可能是形如 \sqrt{k} (k 是非二次剩余) 的数。

证明

直接由二项式定理得到：

$$(a + b)^p = \sum_{i=0}^p \binom{p}{i} a^i b^{p-i}$$

由于当 $i \in [1, p - 1]$ 时， $\binom{p}{i} \equiv 0 \pmod{p}$ ，所以上式等于 $a^p + b^p$ 。

二次剩余

算法

首先得到 $(\frac{a}{p})$, 判定 a 是否是二次剩余。如果是二次剩余, 那么:

随机获得一个 t , 使得 $t^2 - a$ 是非二次剩余。期望随机次数非常少。

令 $w = \sqrt{t^2 - a}$, 则 $x \equiv (t + w)^{\frac{p+1}{2}}$ 。

这个东西只要写成 $a + bw$ 的形式做快速幂就好了。

证明:

$$\begin{aligned}x^2 &\equiv (t + w)^{p+1} \equiv (t + w)(t + w)^p \equiv (t + w)(t^p + w^p) \\&\equiv (t + w)\left(t + (t^2 - a)^{\frac{p-1}{2}}w\right) \equiv (t + w)(t - w) \\&\equiv t^2 - w^2 \equiv t^2 - (t^2 - a) \equiv a\end{aligned}$$

于是同时得知 $(t + w)^{\frac{p+1}{2}}$ 写成 $a + bw$ 形式后 $b = 0$ 。

二次剩余

时间复杂度 $O(\log p)$ 。
(const int mod=998244353;)

```
namespace Rem2{
    int INIT_TAG=0,t,w;
    #define fi first
    #define se second
    void init(){
        INIT_TAG=1;
        srand('C'+'L'+'Y'+'A'+'K'+'I'+'O'+'I');
    }
    pair <int,int> Mul_pii(pair <int,int> A,pair <int,int> B){
        static int a,b;
        a=((LL)A.fi*B.fi+(LL)A.se*B.se%mod*w)%mod;
        b=((LL)A.fi*B.se+(LL)A.se*B.fi)%mod;
        return make_pair(a,b);
    }
    pair <int,int> Pow_pii(pair <int,int> x,int y){
        pair <int,int> ans=make_pair(1,0);
        for (;y;y>>=1,x=Mul_pii(x,x))
            if (y&1)
                ans=Mul_pii(ans,x);
        return ans;
    }
    int Sqrt(int x){
        if (!INIT_TAG)
            init();
        if (x==0)
            return 0;
        if (Pow(x,(mod-1)/2)!=1)
            return -1;
        do {
            t=randint()%(mod-1)+1;
            w=((LL)t*t+mod-x)%mod;
        } while (Pow(w,(mod-1)/2)==1);
        pair <int,int> res=Pow_pii(make_pair(t,1),(mod+1)/2);
        return min(res.fi,mod-res.fi);
    }
}
```

多项式开根

多项式开根

终于回归正题了。

给定多项式 $B(x)$ ，求出多项式 $A(x)$ 使得

$$A^2(x) \equiv B(x) \pmod{x^n}$$

多项式开根

直接牛顿迭代即可。

$$\text{令 } F(A(x)) = A^2(x) - B(x)$$

$$A_2(x) = A(x) - \frac{F(A(x))}{F'(A(x))}$$

$$= A(x) - \frac{A^2(x) - B(x)}{2A(x)}$$

$$= \frac{1}{2} \left(A(x) + \frac{B(x)}{A(x)} \right)$$

多项式开根

初始时

$$A[0] = \sqrt{B[0]}$$

这里需要用到二次剩余。

时间复杂度

$$T(n) = T(n/2) + O(n \log n) = O(n \log n)$$

多项式开根

```
Poly Sqrt(Poly a,int n){
    static Poly A,B;
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty())
        return a;
    A.clear(),B.clear();
    B.push_back(a[0]);
    A.push_back(Rem2 :: Sqrt(B[0]));
    for (int t=1;t<n;){
        for (int i=t;i<min(a.size(),(t<<1));i++)
            B.push_back(a[i]);
        t<<=1;
        A+=B*Inverse(A,t);
        while (A.size()>t)
            A.pop_back();
        A*=499122177;
    }
    if (A[0]>mod-A[0])
        for (int i=0;i<A.size();i++)
            A[i]=(mod-A[i])%mod;
    while (A.size()>n)
        A.pop_back();
    return A;
}
```

多项式对数函数

多项式对数函数

给定多项式 $F(x)$, 求 $\ln(F(x)) \pmod{x^n}$ 。

多项式对数函数

由于

$$\ln(x) = \int \frac{1}{x}$$

所以

$$\ln(F(x)) = \int \frac{F'(x)}{F(x)}$$

直接多项式求逆后相乘就好了。

时间复杂度

$$O(n \log n)$$

多项式对数函数

注意点：

$F(x)$ 的常数项必须是 1。

否则设 $G(x) = kF(x)$ ，则

$\ln(G(x)) = \ln(F(x)) + \ln(k)$ ，其中 $\ln(k)$ 难以用模意义下的数来表示。

容易得知 $\ln(F(x))$ 的常数项是 0。

多项式对数函数

```
Poly Ln(Poly a,int n){  
    while (!a.empty()&&!a.back())  
        a.pop_back();  
    if (a.empty()||a[0]!=1)  
        return a;  
    a=Integral(Derivation(a)*Inverse(a,n));  
    while (a.size()>n)  
        a.pop_back();  
    return a;  
}
```

多项式指数函数

多项式指数函数

给定多项式 $F(x)$ ，求 $e^{F(x)} \pmod{x^n}$ 。

多项式指数函数

首先，设 $e^{F(x)} = G(x) \pmod{x^n}$ ，则

$$F(x) - \ln(G(x)) = 0 \pmod{x^n}$$

利用设函数 $Q(x) = F(x) - \ln(G(x))$ ，利用牛顿迭代得到：

$$\begin{aligned} G_2(x) &= G(x) - \frac{Q(G(x))}{Q'(G(x))} = G(x) - \frac{F(x) - \ln(G(x))}{-\frac{1}{G(x)}} \\ &= G(x)(1 + F(x) - \ln(G(x))) \end{aligned}$$

时间复杂度

$$T(n) = T(n/2) + O(n \log n) = O(n \log n)$$

多项式指數函數

```
Poly Exp(Poly a,int n){
    static Poly A,B;
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty())
        return Poly(1);
    if (a[0]!=0)
        return a;
    A.clear(),B.clear();
    B.push_back(1);
    A.push_back(a[0]);
    for (int t=1;t<n;){
        for (int i=t;i<min(a.size(),(t<<1));i++)
            A.push_back(a[i]);
        t<<=1;
        B=B*(Poly(1)+A-Ln(B,t));
        while (B.size()>t)
            B.pop_back();
    }
    while (B.size()>n)
        B.pop_back();
    return B;
}
```

多项式快速幂

多项式快速幂

给定多项式 $F(x)$ ，求 $F^k(x) \pmod{x^n}$ 。

多项式快速幂

令 $F(x) = bx^a G(x)$ ，其中 a, b 为常数， $G(x)$ 的常数项为 1。则

$$F^k(x) = b^k x^{ak} G^k(x) = b^k x^{ak} e^{k \ln(G(x))}$$

时间复杂度

$$O(n \log n)$$

多项式快速幂

```
Poly PolyPow(Poly x,int y,int n){
    static Poly A,B;
    int k0=0,kc,ivkc;
    while (!x.empty()&&!x.back())
        x.pop_back();
    if (x.empty())
        return x;
    while (k0<x.size()&&x[k0]==0)
        k0++;
    kc=x[k0],ivkc=Pow(kc,mod-2);
    A.clear();
    for (int i=k0;i<x.size();i++)
        A.push_back((int)((LL)x[i]*ivkc%mod));
    A=Exp(Ln(A,n)*y,n);
    B.clear();
    if ((LL)k0*y>=n)
        return B;
    kc=Pow(kc,y),k0*=y;
    for (int i=0;i<k0;i++)
        B.push_back(0);
    for (int i=0;i<min(A.size(),n-k0);i++)
        B.push_back((int)((LL)A[i]*kc%mod));
    while (B.size()>n)
        B.pop_back();
    return B;
}
```

多项式除法

多项式除法

给定多项式 $F(x), G(x)$ ，求多项式 $Q(x)$ ，使得

$$F(x) = G(x)Q(x) + R(x)$$

其中 $F(x), G(x)$ 分别是 n, m 次多项式。

$Q(x), R(x)$ 分别是 $n - m + 1, m - 1$ 次多项式。

多项式除法

定义 $F^R(x)$ 表示多项式 $F(x)$ 系数翻转之后得到的结果。设 $F(x)$ 最高次项为 x^{n-1} ，则

$$F^R(x) = x^{n-1} F\left(\frac{1}{x}\right)$$

于是可得

$$F(x) = G(x)Q(x) + R(x)$$

$$F\left(\frac{1}{x}\right) = G\left(\frac{1}{x}\right)Q\left(\frac{1}{x}\right) + R\left(\frac{1}{x}\right)$$

$$x^{n-1} F\left(\frac{1}{x}\right) = x^{m-1} G\left(\frac{1}{x}\right) x^{n-m} Q\left(\frac{1}{x}\right) + x^{n-m+1} x^{m-2} R\left(\frac{1}{x}\right)$$

$$F^R(x) = G^R(x)Q^R(x) + x^{n-m+1} R^R(x)$$

多项式除法

$$F^R(x) = G^R(x)Q^R(x) + x^{n-m+1}R^R(x)$$

又由于 $Q^R(x)$ 最高次项为 x^{n-m} , 所以

$$F^R(x) = G^R(x)Q^R(x) \pmod{x^{n-m+1}}$$

$$\frac{F^R(x)}{G^R(x)} = Q^R(x) \pmod{x^{n-m+1}}$$

时间复杂度

$$O(n \log n)$$

多项式除法

这里需要注意的是 $m > n$ 要特判。

```
Poly operator / (Poly A,Poly B){//Divide
    int n=A.size(),m=B.size();
    reverse(A.v.begin(),A.v.end());
    reverse(B.v.begin(),B.v.end());
    int k=n-m+1;
    if (k<0)
        return Poly(0);
    while (A.size()>k)
        A.pop_back();
    while (B.size()>k)
        B.pop_back();
    A=A*Inverse(B,k);
    while (A.size()>k)
        A.pop_back();
    reverse(A.v.begin(),A.v.end());
    return A;
}
```

多项式取模

多项式取模

给定多项式 $F(x), G(x)$ ，求多项式 $R(x)$ ，使得

$$F(x) = G(x)Q(x) + R(x)$$

其中 $F(x), G(x)$ 分别是 n, m 次多项式。

$Q(x), R(x)$ 分别是 $n - m + 1, m - 1$ 次多项式。

多项式取模

$$R(x) = F(x) - G(x)Q(x)$$

时间复杂度

$$O(n \log n)$$

多项式取模

```
Poly operator % (Poly A,Poly B){//Modulo
    while (!A.empty()&&!A.back())
        A.pop_back();
    while (!B.empty()&&!B.back())
        B.pop_back();
    A=A-B*B;
    while (A.size()>=B.size())
        A.pop_back();
    while (!A.empty()&&!A.back())
        A.pop_back();
    return A;
}
```

多点求值

多点求值

给定最高次项为 x^{m-1} 的函数 $F(x)$ ，以及 n 个参数 $x_1 \dots n$ ，求

$$F(x_1), F(x_2), \dots, F(x_n)$$

多点求值

$$F(x_i) = (F(x) \bmod (x - x_i))[0]$$

即多项式 $F(x) \bmod (x - x_i)$ 的常数项。

设

$$L(x) = \prod_{1 \leq i \leq \lfloor \frac{n}{2} \rfloor} (x - x_i)$$

$$R(x) = \prod_{\lfloor \frac{n}{2} \rfloor < i \leq n} (x - x_i)$$

则对于 $i \leq n/2$, $F(x_i) = (F \bmod L)(x_i)$;

对于 $i > n/2$, $F(x_i) = (F \bmod R)(x_i)$;

多点求值

先预处理得到所有的 $L(x)$ 和 $R(x)$ ，然后再分治求出答案即可。

时间复杂度

$$T(n) = 2T(n/2) + O(n \log n) = O(n \log^2 n)$$

多点求值

```
namespace Qiuzhi{
    Poly P[N<<2], f[N<<2], M;
    vector <int> x, y;
    int n;
    void GetP(int rt, int L, int R){
        if (L==R){
            P[rt].clear();
            P[rt].push_back((mod-x[L])%mod);
            P[rt].push_back(1);
            return;
        }
        int mid=(L+R)>>1, ls=rt<<1, rs=ls|1;
        GetP(ls, L, mid);
        GetP(rs, mid+1, R);
        P[rt]=P[ls]*P[rs];
    }
    void qiuzhi(int rt, int L, int R){
        if (f[rt].empty())
            f[rt].push_back(0);
        if (L==R)
            return (void)(y[L]=f[rt][0]);
        int mid=(L+R)>>1, ls=rt<<1, rs=ls|1;
        f[ls]=f[rt]%P[ls];
        f[rs]=f[rt]%P[rs];
        qiuzhi(ls, L, mid);
        qiuzhi(rs, mid+1, R);
    }
    vector <int> Get_Val(vector <int> A, Poly F){
        n=A.size();
        x.clear(), y.clear();
        for (int i=0;i<n;i++){
            x.push_back(A[i]);
            y.push_back(0);
        }
        GetP(1, 0, n-1);
        f[1]=F;
        qiuzhi(1, 0, n-1);
        return y;
    }
}
```

快速插值

快速插值

给定 n 对 (x_i, y_i) ，求最高次项为 x^{n-1} 的多项式 $F(x)$ 满足

$$\forall 1 \leq i \leq n, f(x_i) = y_i$$

快速插值

考慮拉格朗日插值法

$$F(x) = \sum_{i=1}^n \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)} y_i$$

令 $M(x) = \prod_{i=1}^n (x - x_i)$

$$M_i(x) = M(x)/(x - x_i)$$

令 $t_i = \frac{y_i}{\prod_{j \neq i} (x_i - x_j)} = \frac{y_i}{M_i(x_i)}$

根据洛必达法则，当 $x \rightarrow x_i$ 时， $M_i(x) = \frac{M(x)}{x - x_i} \rightarrow M'(x)$ ，故

$$t_i = y_i / M'(x_i)$$

快速插值

设

$$L(x) = \prod_{1 \leq i \leq \lfloor \frac{n}{2} \rfloor} (x - x_i)$$

$$R(x) = \prod_{\lfloor \frac{n}{2} \rfloor < i \leq n} (x - x_i)$$

则

$$\begin{aligned} F(x) &= \sum_{i=1}^n t_i \prod_{j \neq i} (x - x_j) \\ &= \left(\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} t_i \prod_{j \neq i} (x - x_j) \right) R(x) + \left(\sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^n t_i \prod_{j \neq i} (x - x_j) \right) L(x) \end{aligned}$$

快速插值

先预处理得到所有的 $L(x)$ 和 $R(x)$ ，然后再分治求出答案即可。

时间复杂度

$$T(n) = 2T(n/2) + O(n \log n) = O(n \log^2 n)$$

快速插值

```
namespace Chazhi{
    Poly P[N<<2],M;
    vector <int> x,y;
    int n;
    void GetP(int rt,int L,int R){
        if (L==R){
            P[rt].clear();
            P[rt].push_back((mod-x[L])%mod);
            P[rt].push_back(1);
            return;
        }
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        GetP(ls,L,mid);
        GetP(rs,mid+1,R);
        P[rt]=P[ls]*P[rs];
    }
    Poly chazhi(int rt,int L,int R){
        if (L==R)
            return Poly(y[L]);
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        return chazhi(ls,L,mid)*P[rs]+chazhi(rs,mid+1,R)*P[ls];
    }
    Poly Get_Poly(vector <int> A,vector <int> B){
        n=A.size();
        x=A;
        int Product=1;
        GetP(1,0,n-1);
        M=Derivation(P[1]);
        y=QiuZhi :: Get_Val(A,M);
        for (int i=0;i<y.size();i++)
            y[i]=(LL)B[i]*Pow(y[i],mod-2)%mod;
        return chazhi(1,0,n-1);
    }
}
```

模板

模板

先放个链接～

<https://www.cnblogs.com/zhouzhendong/p/10199336.html>

模板

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
LL read(){
    LL x=0,f=0;
    char ch=getchar();
    while (!isdigit(ch))
        f|=ch=='-',ch=getchar();
    while (isdigit(ch))
        x=(x<<1)+(x<<3)+(ch^48),ch=getchar();
    return f?-x:x;
}
const int N=1<<18,mod=998244353;
void Add(int &x,int y){
    if ((x+=y)>=mod)
        x-=mod;
}
void Del(int &x,int y){
    if ((x-=y)<0)
        x+=mod;
}
int del(int x,int y){
    return x-y<0?x-y+mod:x-y;
}
int Pow(int x,int y){
    int ans=1;
    for (;y;y>>=1,x=(LL)x*x%mod)
        if (y&1)
            ans=(LL)ans*x%mod;
    return ans;
}
int randint(){
    return ((rand()&65535)<<15)^((rand()&65535));
}
```

模板

```
namespace Rem2{
    int INIT_TAG=0;
    int t,w;
#define fi first
#define se second
    void init(){
        INIT_TAG=1;
        srand('C'+'L'+'Y'+'A'+'K'+'I'+'O'+'I');
    }
    pair <int,int> Mul_pii(pair <int,int> A,pair <int,int> B){
        static int a,b;
        a=((LL)A.fi*B.fi+(LL)A.se*B.se%mod*w)%mod;
        b=((LL)A.fi*B.se+(LL)A.se*B.fi)%mod;
        return make_pair(a,b);
    }
    pair <int,int> Pow_pii(pair <int,int> x,int y){
        pair <int,int> ans=make_pair(1,0);
        for (;y;y>>=1,x=Mul_pii(x,x))
            if (y&1)
                ans=Mul_pii(ans,x);
        return ans;
    }
    int Sqrt(int x){
        if (!INIT_TAG)
            init();
        if (x==0)
            return 0;
        if (Pow(x,(mod-1)/2)!=1)
            return -1;
        do {
            t=randint()%(mod-1)+1;
            w=((LL)t*t+mod-x)%mod;
        } while (Pow(w,(mod-1)/2)==1);
        pair <int,int> res=Pow_pii(make_pair(t,1),(mod+1)/2);
        return min(res.fi,mod-res.fi);
    }
}
```

模板

```
namespace Polynomial{
    namespace Fast{
        const int N=1<<18;
        int n,Log[N+1],Fac[N+1],InvFac[N+1],Inv[N+1];
        int ww[N*2],*Ew=ww,*w[N+1];
        int iww[N*2],*Ei=iww,*iw[N+1];
        int INIT_TAG=0;
        void init(int _n){
            INIT_TAG=1;
            Log[1]=0,n=_n;
            for (int i=2;i<=N;i++)
                Log[i]=Log[i>>1]+1;
            for (int i=Fac[0]=1;i<=N;i++)
                Fac[i]=(LL)Fac[i-1]*i%mod;
            InvFac[N]=Pow(Fac[N],mod-2);
            for (int i=N;i>=1;i--)
                InvFac[i-1]=(LL)InvFac[i]*i%mod;
            for (int i=1;i<=N;i++)
                Inv[i]=(LL)InvFac[i]*Fac[i-1]%mod;
            for (int d=0;d<=Log[n];d++){
                w[d]=Ew, iw[d]=Ei;
                int n=1<<d;
                w[d][0]=1,w[d][1]=Pow(3,(mod-1)/n);
                for (int i=2;i<n;i++)
                    w[d][i]=(LL)w[d][i-1]*w[d][1]%mod;
                iw[d][0]=1,iw[d][1]=Pow(w[d][1],mod-2);
                for (int i=2;i<n;i++)
                    iw[d][i]=(LL)iw[d][i-1]*iw[d][1]%mod;
                Ew+=n,Ei+=n;
            }
        }
    }
}
```

模板

```
int Rev[N+1], A[N+1], B[N+1];
void FFT(int a[], int n, int **w){
    if (!INIT_TAG)
        init(N);
    for (int i=0; i<n; i++)
        if (Rev[i]<i)
            swap(a[i], a[Rev[i]]);
    for (int t=1, d=1; d<n; t++, d<<=1)
        for (int i=0; i<n; i+=(d<<1))
            for (int j=0, *W=w[t]; j<d; j++) {
                int tmp=(LL)(*W++)*a[i+j+d]%mod;
                a[i+j+d]=del(a[i+j], tmp);
                Add(a[i+j], tmp);
            }
}
```

模板

```
vector <int> Mul(vector <int> &a, vector <int> &b) {
    static vector <int> res;
    res.clear();
    LL Br=(LL)a.size()*b.size();
    LL FF=(a.size()+b.size())*Log[a.size()+b.size()]*10+100;
    if (Br<=FF){
        for (int i=0;i<a.size()+b.size();i++)
            res.push_back(0);
        for (int i=0;i<a.size();i++)
            for (int j=0;j<b.size();j++)
                res[i+j]=((LL)a[i]*b[j]+res[i+j])%mod;
    }
    else {
        int n=1,d=0;
        for (;n<a.size()+b.size();n<<=1,d++);
        for (int i=0;i<n;i++)
            Rev[i]=(Rev[i>>1]>>1)|((i&1)<<(d-1)),A[i]=B[i]=0;
        for (int i=0;i<a.size();i++)
            A[i]=a[i];
        for (int i=0;i<b.size();i++)
            B[i]=b[i];
        FFT(A,n,w),FFT(B,n,w);
        for (int i=0;i<n;i++)
            A[i]=(LL)A[i]*B[i]%mod;
        FFT(A,n,iw);
        int inv=Pow(n,mod-2);
        for (int i=0;i<n;i++)
            res.push_back((int)((LL)inv*A[i]%mod));
    }
    while (!res.empty()&&!res.back())
        res.pop_back();
    return res;
}
```

模板

```
vector <int> MulInv(vector <int> &a, vector <int> &b) {
    static vector <int> res;
    res.clear();
    int n=1, d=0;
    for (;n<a.size()*2+b.size();n<<=1, d++);
    for (int i=0;i<n;i++)
        Rev[i]=(Rev[i>>1]>>1)|((i&1)<<(d-1)), A[i]=B[i]=0;
    for (int i=0;i<a.size();i++)
        A[i]=a[i];
    for (int i=0;i<b.size();i++)
        B[i]=b[i];
    FFT(A,n,w),FFT(B,n,w);
    for (int i=0;i<n;i++)
        A[i]=(LL)A[i]*A[i]%mod*B[i]%mod;
    FFT(A,n,iw);
    int inv=Pow(n,mod-2);
    for (int i=0;i<n;i++)
        res.push_back((int)((LL)inv*A[i]%mod));
    while (!res.empty()&&!res.back())
        res.pop_back();
    return res;
}
```

模板

```
struct Poly{
    vector <int> v;
    Poly(){
        v.clear();
    }
    Poly(int x){
        v.clear();
        v.push_back(x);
    }
    Poly(vector <int> x){
        v=x;
    }
    int operator ()(int x){
        int ans=0,y=1;
        for (int i=0;i<v.size();i++)
            ans=((LL)v[i]*y+ans)%mod,y=(LL)y*x%mod;
        return ans;
    }
    int size(){
        return v.size();
    }
    void print(){
        for (int i=0;i<v.size();i++)
            printf("%d ",v[i]);
    }
    void print(int x){
        for (int i=0;i<x;i++)
            printf("%d ",i>=v.size()?0:v[i]);
    }
    void print(string s){
        print(),cout << s;
    }
}
```

模板

```
void clear(){
    v.clear();
}
void push_back(int x){
    v.push_back(x);
}
void pop_back(){
    v.pop_back();
}
int empty(){
    return v.empty();
}
int back(){
    return v.back();
}
int &operator [](int x){
    return v[x];
}
void operator += (Poly A){
    while (v.size()<A.size())
        v.push_back(0);
    for (int i=0;i<A.size();i++)
        Add(v[i],A[i]);
}
void operator -= (Poly &A){
    while (v.size()<A.size())
        v.push_back(0);
    for (int i=0;i<A.size();i++)
        Del(v[i],A[i]);
}
```

模板

```
void operator *= (Poly &A);
void Derivation(){
    for (int i=0;i<v.size()-1;i++)
        v[i]=(LL)v[i+1]*(i+1)%mod;
    v.pop_back();
}
void Integral(){
    v.push_back(0);
    for (int i=v.size()-2;i>=0;i--)
        v[i+1]=(LL)v[i]*Fast :: Inv[i+1]%mod;
    v[0]=0;
}
void operator *= (int x){
    for (int i=0;i<v.size();i++)
        v[i]=(LL)v[i]*x%mod;
}
}pp;
//struct Poly end-----
Poly operator + (Poly A,Poly B){
    pp.clear();
    for (int i=0;i<max(A.size(),B.size());i++)
        pp.push_back(0);
    for (int i=0;i<A.size();i++)
        Add(pp[i],A[i]);
    for (int i=0;i<B.size();i++)
        Add(pp[i],B[i]);
    return pp;
}
Poly operator - (Poly A,Poly B){
    pp.clear();
    for (int i=0;i<max(A.size(),B.size());i++)
        pp.push_back(0);
    for (int i=0;i<A.size();i++)
        Add(pp[i],A[i]);
    for (int i=0;i<B.size();i++)
        Del(pp[i],B[i]);
    return pp;
}
```

模板

```
Poly operator * (Poly A,Poly B){
    return Poly(Fast :: Mul(A.v,B.v));
}
void Poly :: operator *= (Poly &A){
    v=Fast :: Mul(v,A.v);
}
Poly operator * (Poly A,int x){
    pp=A;
    for (int i=0;i<A.size();i++)
        pp[i]=(LL)pp[i]*x%mod;
    return pp;
}
Poly Inverse(Poly a,int n);
Poly operator / (Poly A,Poly B){//Divide
    int n=A.size(),m=B.size();
    reverse(A.v.begin(),A.v.end());
    reverse(B.v.begin(),B.v.end());
    int k=n-m+1;
    if (k<0)
        return Poly(0);
    while (A.size()>k)
        A.pop_back();
    while (B.size()>k)
        B.pop_back();
    A=A*Inverse(B,k);
    while (A.size()>k)
        A.pop_back();
    reverse(A.v.begin(),A.v.end());
    return A;
}
```

模板

```
Poly operator % (Poly A,Poly B){//Modulo
    while (!A.empty()&&!A.back())
        A.pop_back();
    while (!B.empty()&&!B.back())
        B.pop_back();
    A=A-B*B;
    while (A.size()>=B.size())
        A.pop_back();
    while (!A.empty()&&!A.back())
        A.pop_back();
    return A;
}
Poly Derivation(Poly A){
    for (int i=0;i<A.size()-1;i++)
        A[i]=(LL)A[i+1]*(i+1)%mod;
    A.pop_back();
    return A;
}
Poly Integral(Poly A){
    A.push_back(0);
    for (int i=A.size()-2;i>=0;i--)
        A[i+1]=(LL)A[i]*Fast :: Inv[i+1]%mod;
    A[0]=0;
    return A;
}
```

模板

```
Poly Inverse(Poly a,int n){
    static Poly A,B;
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty())
        return a;
    A.clear(),B.clear();
    B.push_back(a[0]);
    A.push_back(Pow(B[0],mod-2));
    for (int t=1;t<n;){
        for (int i=t;i<min(a.size(),(t<<1));i++)
            B.push_back(a[i]);
        t<<=1;
        A=A*2-Poly(Fast :: MulInv(A.v,B.v));
        while (A.size()>t)
            A.pop_back();
    }
    while (A.size()>n)
        A.pop_back();
    return A;
}
Poly Sqrt(Poly a,int n){
    static Poly A,B;
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty())
        return a;
    A.clear(),B.clear();
    B.push_back(a[0]);
    A.push_back(Rem2 :: Sqrt(B[0]));
    for (int t=1;t<n;){
        for (int i=t;i<min(a.size(),(t<<1));i++)
            B.push_back(a[i]);
        t<<=1;
        A+=B*Inverse(A,t);
        while (A.size()>t)
            A.pop_back();
        A*=499122177;
    }
    if (A[0]>mod-A[0])
        for (int i=0;i<A.size();i++)
            A[i]=(mod-A[i])%mod;
    while (A.size()>n)
        A.pop_back();
    return A;
}
```

模板

```
Poly Ln(Poly a,int n){
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty()||a[0]!=1)
        return a;
    a=Integral(Derivation(a)*Inverse(a,n));
    while (a.size()>n)
        a.pop_back();
    return a;
}
Poly Exp(Poly a,int n){
    static Poly A,B;
    while (!a.empty()&&!a.back())
        a.pop_back();
    if (a.empty())
        return Poly(1);
    if (a[0]!=0)
        return a;
    A.clear(),B.clear();
    B.push_back(1);
    A.push_back(a[0]);
    for (int t=1;t<n;){
        for (int i=t;i<min(a.size(),(t<<1));i++)
            A.push_back(a[i]);
        t<<=1;
        B=B*(Poly(1)+A-Ln(B,t));
        while (B.size()>t)
            B.pop_back();
    }
    while (B.size()>n)
        B.pop_back();
    return B;
}
```

模板

```
Poly PolyPow(Poly x,int y,int n){
    static Poly A,B;
    int k0=0,kc,ivkc;
    while (!x.empty()&&!x.back())
        x.pop_back();
    if (x.empty())
        return x;
    while (k0<x.size()&&x[k0]==0)
        k0++;
    kc=x[k0],ivkc=Pow(kc,mod-2);
    A.clear();
    for (int i=k0;i<x.size();i++)
        A.push_back((int)((LL)x[i]*ivkc%mod));
    A=Exp(Ln(A,n)*y,n);
    B.clear();
    if ((LL)k0*y>=n)
        return B;
    kc=Pow(kc,y),k0*=y;
    for (int i=0;i<k0;i++)
        B.push_back(0);
    for (int i=0;i<min(A.size(),n-k0);i++)
        B.push_back((int)((LL)A[i]*kc%mod));
    while (B.size()>n)
        B.pop_back();
    return B;
}
```

模板

```
namespace Qiuzhi{
    Poly P[N<<2],f[N<<2],M;
    vector <int> x,y;
    int n;
    void GetP(int rt,int L,int R){
        if (L==R){
            P[rt].clear();
            P[rt].push_back((mod-x[L])%mod);
            P[rt].push_back(1);
            return;
        }
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        GetP(ls,L,mid);
        GetP(rs,mid+1,R);
        P[rt]=P[ls]*P[rs];
    }
    void qiuZhi(int rt,int L,int R){
        if (f[rt].empty())
            f[rt].push_back(0);
        if (L==R)
            return (void)(y[L]=f[rt][0]);
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        f[ls]=f[rt]%P[ls];
        f[rs]=f[rt]%P[rs];
        qiuZhi(ls,L,mid);
        qiuZhi(rs,mid+1,R);
    }
    vector <int> Get_Val(vector <int> A,Poly F){
        n=A.size();
        x.clear(),y.clear();
        for (int i=0;i<n;i++){
            x.push_back(A[i]);
            y.push_back(0);
        }
        GetP(1,0,n-1);
        f[1]=F;
        qiuZhi(1,0,n-1);
        return y;
    }
}
```

模板

```
namespace Chazhi{
    Poly P[N<<2],M;
    vector <int> x,y;
    int n;
    void GetP(int rt,int L,int R){
        if (L==R){
            P[rt].clear();
            P[rt].push_back((mod-x[L])%mod);
            P[rt].push_back(1);
            return;
        }
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        GetP(ls,L,mid);
        GetP(rs,mid+1,R);
        P[rt]=P[ls]*P[rs];
    }
    Poly chazhi(int rt,int L,int R){
        if (L==R)
            return Poly(y[L]);
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        return chazhi(ls,L,mid)*P[rs]+chazhi(rs,mid+1,R)*P[ls];
    }
    Poly Get_Poly(vector <int> A,vector <int> B){
        n=A.size();
        x=A;
        int Product=1;
        GetP(1,0,n-1);
        M=Derivation(P[1]);
        y=Qiuzhi :: Get_Val(A,M);
        for (int i=0;i<y.size();i++)
            y[i]=(LL)B[i]*Pow(y[i],mod-2)%mod;
        return chazhi(1,0,n-1);
    }
}
// be careful about init!!!!!
```

模板 - LOJ 挑战多项式 主程序

```
using namespace Polynomial;
Poly A,B;
vector <int> x,y;
int main(){
    int n=read()+1,k=read();
    A.clear();
    for (int i=0;i<n;i++)
        A.push_back(read());
    B=Exp(Integral(Inverse(Sqrt(A,n),n)),n);
    B=Poly(1)+Ln(Poly(2)+A-A(0)-B,n);
    B=Derivation(PolyPow(B,k,n));
    n--;
    B.print(n);
    return 0;
}
```

模板题

LOJ150 挑战多项式

<https://loj.ac/problem/150>

洛谷里的多项式模板题

<https://www.luogu.org/problemnew/lists?name=多项式>

NFLSOJ里的

<https://acm.nflsoj.com/problems/template>

有什么用？

生成函数？？

鸣谢

CMXRYNP

<https://cmxrynp.github.io/2018/10/24/多项式一些基础的操作/>

<https://cmxrynp.github.io/2018/11/27/多项式多点求值和快速插值学习笔记/>

beginend

https://blog.csdn.net/qq_33229466/article/details/79125057

以及我曾经看过的一些课件。



Thanks

--2018-12-30