

多项式求逆

定义

对于一个多项式 $A(x)$, 如果存在另一个多项式 $B(x)$, 有 $\deg B(x) \leq \deg A(x)$, 且 $A(x)B(x) \equiv 1 \pmod{x^n}$, 那么称 $B(x)$ 为 $A(x)$ 在 $\pmod{x^n}$ 下的逆元, 记为 $A^{-1}(x)$

求法

当 $n=1$ 时, $A(x) \equiv c \pmod{x}$, 此时 $A^{-1}(x) \equiv c^{-1}$

不妨设 $A(x)B^{(1)}(x) \equiv 1 \pmod{x^{\lceil \frac{n}{2} \rceil}}$, $A(x)B(x) \equiv 1 \pmod{x^n}$

显然, 也有 $A(x)B(x) \equiv 1 \pmod{x^{\lceil \frac{n}{2} \rceil}}$

和第一个式子相减, 可得 $B(x) \equiv B^{(1)}(x) \pmod{x^{\lceil \frac{n}{2} \rceil}}$

移项, 两边平方, 有 $B^{(2)}(x) - 2B(x)B^{(1)}(x) + B^{(1)2}(x) \equiv 0 \pmod{x^n}$

两边同乘 $A(x)$, 化简可得 $B(x) \equiv 2B^{(1)}(x) - A(x)B^{(1)2}(x) \pmod{x^n}$

于是就可以根据最后一个式子递归计算了. 在计算的时候需要拿 NTT 来实现多项式乘法的过程. 总的时间复杂度为 $O(n \log n)$.

代码模板

```

IL void PolyInv(LL x[], LL y[], int len){
    int i=0;
    if (len==1) {
        y[0]=Mi(x[0], MOD-2);
        return;
    }
    PolyInv(x, y, len>>1);
    for (i=0; i<len; i++) X[i]=x[i], Y[i]=y[i];
    NTT(X, len<<1, 1);    NTT(Y, len<<1, 1);
    for (i=0; i<(len<<1); i++)
        X[i]=((X[i]*Y[i])%MOD*Y[i])%MOD;
    NTT(X, len<<1, -1);
    for (i=0; i<len; i++) y[i]=((y[i]<<1)%MOD+MOD-X[i])%MOD;
}

```

多项式除法和取模

求法

给出两个多项式 $F(x)$, $G(x)$, 求 $D(x)$, $R(x)$, 使得 $F(x) = D(x)G(x) + R(x)$. 其中, $F(x)$ 为 n 次多项式, $G(x)$ 为 m 次多项式, $m \leq n$. 要求求出的 $D(x)$ 为 $n-m$ 次多项式.

首先定义翻转操作: 对于一个 n 次多项式 $A(x)$, 它的翻转多项式为 $A^{\{r\}}(x) = x^n A(\frac{1}{x})$. 假如说 $A(x) = x^2 + 2x + 3$, 那么 $A^{\{r\}}(x) = 3x^2 + 2x + 1$, 也就是把系数翻转了一下.

定义了翻转操作后, 对上面这个多项式除法式进行一下变形, 将 $\frac{1}{x}$ 替代 x , 两边同乘 x^n .

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

化简, 有 $F^{\{r\}}(x) = G^{\{r\}}(x) D^{\{r\}}(x) + x^{n-m+1} R^{\{r\}}(x)$

两边同模 x^{n-m+1} , 则 $R^{\{r\}}(x)$ 这一项显然会被消掉, 只剩下

$F^{\{r\}}(x) \equiv G^{\{r\}}(x) D^{\{r\}}(x) \pmod{x^{n-m+1}}$

已知 $D(x)$ 的次数是 $n-m$ 次, 也就是说在上面的模意义下, $D(x)$ 的所有项都会保留下来. 进一步变形, 就有

$D^{\{r\}}(x) \equiv F^{\{r\}}(x) G^{\{-1r\}}(x) \pmod{x^{n-m+1}}$

然后利用上面的求逆元过程, 算出 $D^{\{r\}}(x)$, 翻转就可以得到 $D(x)$ 了. 然后带回到原式, 就可以算出 $R(x)$. 总的时间复杂度也是 $O(n \log n)$.

代码模板

```
IL void PolyMul(LL a[], LL b[], LL c[], int l1, int l2) {
    reg int i=0, L=Max(l1, l2), len=1;
    for (; len<=L; len<<=1);
    len<<=1;
    for (i=0; i<=l1; i++) X[i]=a[i];
    for (i=0; i<=l2; i++) Y[i]=b[i];
    NTT(X, len, 1); NTT(Y, len, 1);
    for (i=0; i<=len; i++) c[i]=(X[i]*Y[i])%MOD, X[i]=Y[i]=0;
    NTT(c, len, -1);
}
```

```
IL void PolyDiv(LL x[], LL y[], LL a[], LL b[]) {
    reg int i=0, len=1;
    reverse(x, x+1+n); reverse(y, y+1+m);
    for (; len<=(n-m); len<<=1);
    PolyInv(y, s, len);
    memset(X, 0, sizeof(X)); memset(Y, 0, sizeof(Y));
    PolyMul(x, s, a, n-m, n-m);
    reverse(a, a+n-m+1); reverse(x, x+1+n); reverse(y, y+1+m);
    PolyMul(a, y, b, n-m, m);
}
```

```
for (i=0;i<m;i++)    b[i]=(f[i]-b[i]+MOD)%MOD;
}
```

多项式开根

假如说求 $B(x)$,使得 $B(x)^2 \equiv A(x) \pmod{x^n}$.不妨设 $B'(x)^2 \equiv A(x) \pmod{x^{\lceil \frac{n}{2} \rceil}}$.同时,已知 $B(x)^2 \equiv A(x) \pmod{x^{\lceil \frac{n}{2} \rceil}}$.

两个等式相减,再平方,可得 $B'(x)^4 - 2B'(x)^2B(x)^2 + B(x)^4 \equiv 0 \pmod{x^n}$

做一下变形,有 $B'(x)^4 + 2B'(x)^2B(x)^2 + B(x)^4 \equiv 4B'(x)^2B(x)^2 \pmod{x^n}$

故有 $B'(x)^2 + B(x)^2 \equiv 2B(x)B'(x) \pmod{x^n}$

将已知条件代入,有 $B(x) \equiv \frac{A(x) + B'(x)^2}{2B'(x)}$

和多项式求逆那样递归计算即可.

有一个问题是,当递归到 $n=1$ 的时候,要求常数项在模意义下开根.洛谷上的例题规定了常数项一定为1,所以保证有解.如果不规定常数项的话,还需要通过二次剩余来判断解的存在性.

代码模板

注意数组大小要开到8倍以上.

```
IL void PolySqrt(LL a[],LL b[],int len){
    reg int i=0;
    if (len==1){
        b[0]=1; return;
    }
    PolySqrt(a,b,(len+1)>>1);
    memset(e,0,sizeof(e));
    for (i=0;i<len;i++) c[i]=(b[i]<<1)%MOD;
    PolyInv(c,e,len);
    PolyMul(b,b,d,len,len);
    for (i=0;i<len;i++) b[i]=(d[i]+a[i])%MOD;
    PolyMul(b,e,b,len,len);
}
```

多项式求导&积分

原理应该不用多讲了...直接算就行了,积分的时候除上逆元.

代码模板

```
IL void PolyDx(LL a[],LL b[],LL len){
    reg LL i=0;
    for (i=0;i<=len;i++)    b[i]=(a[i+1]*(i+1))%MOD;
}

IL void PolyInte(LL a[],LL b[],LL len){
    reg LL i=0;
    b[0]=0;
    for (i=1;i<=len;i++)
        b[i]=(a[i-1]*Mi(i,MOD-2))%MOD;
}
```

多项式Ln

给出 $A(x)$,求 $B(x)$,使得 $B(x) \equiv \ln(A(x)) \pmod{x^n}$

对原式两边求导,有 $B'(x) \equiv \frac{A'(x)}{A(x)} \pmod{x^n}$

通过多项式的求导和求逆算出 $B'(x)$ 后,再积分即可得到 $B(x)$.

代码模板

```
IL void PolyLn(LL a[],LL b[],LL len){
    PolyDx(a,d,len);
    PolyInv(a,c,len);
    PolyMul(d,c,d,len,len);
    PolyInte(d,b,len);
}
```

多项式Exp

需要一些多项式牛顿迭代的基础.

求一个多项式 $G(x)$,使得 $F(G(x)) \equiv 0 \pmod{x^n}$.如果说已经求出来了 $F(G_0(x)) \equiv 0 \pmod{x^{\lceil \frac{n}{2} \rceil}}$,将 $F(G(x))$ 在 $G_0(x)$ 处展开,因为 $G(x)-G_0(x)$ 的最低次数已经是 $\lceil \frac{n}{2} \rceil$ 了,所以这一项平方之后,在模 x^n 意义下为0.故泰勒展开只需要保留前两项,也就是:

$$F(G(x)) = F(G_0(x)) + (G(x)-G_0(x))F'(G_0(x)) \equiv 0 \pmod{x^n}$$

其中的 $F'(x)$ 表示一阶导.化简,有 $G(x) \equiv G_0(x) - \frac{F(G_0(x))}{F'(G_0(x))} \pmod{x^n}$.

然后回到这个问题上.现在是要求 $B(x) \equiv e^{A(x)} \pmod{x^n}$.两边求对数,化简,有 $\ln(B(x)) - A(x) \equiv 0 \pmod{x^n}$.

这个时候,构造 $F(G(x)) = \ln G(x) - A(x)$.利用牛顿迭代不断求解就行了.

式子的话, $(F(G(x)))' = \frac{G'(x)}{G(x)}$.把它带到迭代的式子中,化简一下就有 $G(x) = G_0(x) - \frac{G_0(x)(1 - \ln G_0(x) + A(x))}{G_0'(x)}$

最后根据上面这个式子就可以递归计算了.

因为要多次算Ln,所以中间用到的两个数组需要清空一下

不知道哪个环节有问题,这个板子的常数极大,而且是别人平均时间的两倍左右...有待优化.

```
IL void PolyLn(LL a[], LL b[], LL len){
    PolyDx(a, d, len);
    PolyInv(a, c, len);
    PolyMul(d, c, d, len, len);
    PolyInte(d, b, len);
    for (LL i=0; i<=len; i++) c[i]=d[i]=0;
}

IL void PolyExp(LL a[], LL b[], LL len){
    reg LL i=0;
    if (len==1){
        b[0]=1; return;
    }
    PolyExp(a, b, len>>1);
    for (i=0; i<=len; i++) e[i]=0;
    PolyLn(b, e, len);
    for (i=0; i<len; i++)
        e[i]=(((i==0)?1:0) - e[i] + a[i] + MOD)%MOD;
    PolyMul(b, e, b, len, len);
}
```

多项式快速幂

会了求导和求对数之后就很简单了.

$B(x) \equiv A(x)^k$,求导,有 $\ln B(x) \equiv k \ln A(x)$.

求导,乘以 k ,再exp回去就行了.

代码模板

因为同时使用了对数和指数的原因,常数极大.

```
IL void PolyMi(LL a[],LL b[],LL len,LL k){
    reg int i=0;
    PolyLn(a,p,len);
    for (i=0;i<=len;i++)    p[i]=(p[i]*k)%MOD;
    PolyExp(p,b,len);
}
```

多项式三角函数

根据著名的欧拉公式: $e^{ix}=\cos x+i\sin x$.代入正负 x ,然后解方程,有:

$$\sin x=\frac{e^{ix}-e^{-ix}}{2i}$$

$$\cos x=\frac{e^{ix}+e^{-ix}}{2}$$

然后就是一波玄学操作: $i^2=-1$, $i^2 \equiv -1 \pmod{998244353}$, $i \equiv 86583718 \pmod{998244353}$, $i^{-1}()$

分子分母上的 i 就搞定了.

没学过复变,但是感觉复变老师看到这一段可能想打人

代码模板

这里参考大佬的博客给NTT加了一个预处理的优化,但是效果不是特别理想.

```
IL void Ready(){
    reg LL i=0;
    for (i=2;i<=800000;i<=1)
        Wn[i]=Mi(3,(MOD-1)/i),WN[i]=Mi(Wn[i],MOD-2);
}
```

所以还是依靠O2,O3优化靠谱一些.慎重使用这个板子吧.

```
IL void PolySin(LL a[],LL b[],LL len){
    reg LL i=0,u=86583718,invu=Mt(u,MOD-2),inv2=Mt(2,MOD-2),inv=0;
    for (i=0;i<=len;i++)    a[i]=(a[i]*u)%MOD;
    PolyExp(a,p,len);    PolyInv(p,b,len);
    inv=(inv2*invu)%MOD;
    for (i=0;i<=len;i++)    b[i]=(p[i]-b[i]+MOD)%MOD;
    for (i=0;i<=len;i++)    b[i]=(b[i]*inv)%MOD;
}

IL void PolyCos(LL a[],LL b[],LL len){
```

```

reg LL i=0,u=86583718,invu=Mi(u,MOD-2),inv2=Mi(2,MOD-2);
for (i=0;i<=len;i++)    a[i]=(a[i]*u)%MOD;
PolyExp(a,p,len);    PolyInv(p,b,len);
for (i=0;i<=len;i++)    b[i]=(p[i]+b[i])%MOD;
for (i=0;i<=len;i++)    b[i]=(b[i]*inv2)%MOD;
}

```

多项式反三角函数

首先,有

$$\frac{d}{dx} \arcsin x = \frac{1}{\sqrt{1-x^2}}$$

$$\frac{d}{dx} \arctan x = \frac{1}{1+x^2}$$

根据这个式子,将多项式代入,再积分,就有

$$B(x) = \int \frac{A'(x)}{\sqrt{1-A'(x)^2}} dx$$

$$B(x) = \int \frac{A'(x)}{1+A'(x)^2} dx$$

所以只需要多项式求导,求逆,开根,积分就能完成反三角函数的求解.

代码模板

```


IL void PolyArcsin(LL a[],LL b[],LL len){
    reg int i=0;
    PolyDx(a,p,len);
    PolyMul(a,a,q,len,len);
    for (i=0;i<=len;i++)    q[i]=(MOD-q[i])%MOD;
    q[0]=Upd(q[0]+1-MOD);
    PolySqrt(q,w,len);
    memset(q,0,sizeof(q));
    PolyInv(w,q,len);    PolyMul(p,q,p,len,len);
    PolyInte(p,b,len);
}

IL void PolyArctan(LL a[],LL b[],LL len){
    PolyDx(a,p,len);
    PolyMul(a,a,q,len,len);
    q[0]=(q[0]+1)%MOD;
    PolyInv(q,w,len);    PolyMul(p,w,p,len,len);
    PolyInte(p,b,len);
}

```

Last update: 2020-2021:teams:t1e233:polynomial <https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:t1e233:polynomial&rev=1590239657>
2020/05/23 21:14

From:
<https://wiki.cvbbacm.com/> - **CVBB ACM Team**

Permanent link:
<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:t1e233:polynomial&rev=1590239657> 

Last update: **2020/05/23 21:14**