

# 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);
}
```

From:

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

Permanent link:

<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:tle233:polynomial&rev=1590239567>

Last update: 2020/05/23 21:12