

[\[关闭\]](#)

@Scarlet 2017-03-27 19:07 字数 12189 阅读 75

辣鸡多项式毁我青春

多项式 数学

感觉自己听得风就是雨就下海了。

- [辣鸡多项式毁我青春](#)
 - [前置技能](#)
 - [Fast Fourier Transform](#)
 - [引入](#)
 - [多项式乘法](#)
 - [前置技能的前置技能：单位根](#)
 - [DFT](#)
 - [IDFT](#)
 - [蝴蝶优化](#)
 - [板子](#)
 - [Number Theory Transform](#)
 - [引入](#)
 - [快速数论变换](#)
 - [板子](#)
 - [附NTT常用模数原根表](#)
 - [Inverse Element of Polynomial](#)
 - [引入](#)
 - [多项式求逆](#)
 - [倍增](#)
 - [板子](#)
 - [Polynomial Division](#)
 - [引入](#)
 - [多项式除法](#)
 - [板子](#)
 - [Square Root of Polynomial](#)
 - [引入](#)
 - [多项式开根](#)
 - [板子](#)
 - [Simple模型](#)
 - [卷积](#)
 - [反卷积](#)
 - [可重集内n元（有序）数对和](#)
 - [可重集内n元（有序）数对积取质数模](#)
 - [可重集内数对差](#)
 - [多项式公因式](#)
 - [多项式求逆EXT](#)
 - [齐次线性递推数列](#)
 - [非齐次线性递推数列](#)
 - [卡特兰数列（自卷积数列）](#)
 - [辣鸡例题](#)
 - [UOJ#34. 多项式乘法](#)
 - [BZOJ3527: \[Zjoi2014\]力](#)
 - [BZOJ3160: 万径人踪灭](#)
 - [BZOJ3992: \[SDOI2015\]序列统计](#)
 - [codeforces #250E The Child and Binary Tree](#)

- [BZOJ3456: 城市规划](#)
- [BZOJ3684: 大朋友和多叉树](#)
- [codeforces #568B Symmetric and Transitive](#)
- [UOJ#50. 【UR #3】链式反应](#)
- [UOJ#23. 【UR #1】跳蚤国王下江南](#)
- [UOJ#86. mx的组合数](#)
- [UOJ#102. 【集训队互测2015】ydc的奖金](#)
- [UOJ#182. 【UR #12】 \$a^{-1} + b\$ problem](#)
- [References](#)

前置技能

Fast Fourier Transform

引入

没有人不知道它是用来干什么的吧

多项式乘法

两个多项式 $A(x), B(x)$, 求 $C(x) = A(x) * B(x)$

简单的说, 我们可以考虑先在 $O(n \log n)$ 时间内把系数表示的 $A(x), B(x)$ 转成点值表达, 再用 $O(n)$ 时间计算出 $C(x)$ 的点值表达, 最后用 $O(n \log n)$ 时间内把 $C(x)$ 转换为系数表达。

系数表示转点值表示称为 DFT , 点值表示转系数表示称为 $IDFT$

前置技能的前置技能: 单位根

满足 $x^n = 1$ 的复数 n 有个, 分别为 $\omega_n^k = e^{i \frac{2k\pi}{n}}$, 其中称 ω_n^1 位主 n 次单位根.

单位根有些很好的性质:

1. $\omega_n^i = \omega_n^{i-1} * \omega_n^1$ (复平面旋转)
2. $\omega_n^j \omega_n^k = \omega_n^{j+k} = \omega_n^{j+k \bmod n}$ (群的性质)
3. $\omega_{dn}^{dk} = \omega_n^k$ (互质性质)

(为了方便进行 DFT 和 $IDFT$, 下文中的 n 均是不小于多项式次数界的最小2的幂次)

DFT

先记 $A(x) = \sum_{i=0}^{n-1} a_i x^i$. 计算 $A(x)$ 在 n 次单位根处的取值. 定义新多项式

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{\frac{n}{2}-1}$$

$$A^{[1]}(x) = a_1 + a_3 x + \cdots + a_{n-1} x^{\frac{n}{2}-1}$$

开心地发现:

$$A(x) = A^{[0]}(x^2) + x A^{[1]}(x^2)$$

亦即

$$A(\omega_n^k) = A^{[0]}(\omega_n^{2k}) + \omega_n^k A^{[1]}(\omega_n^{2k})$$
$$A(\omega_n^{k+\frac{n}{2}}) = A^{[0]}(\omega_n^{2k}) - \omega_n^k A^{[1]}(\omega_n^{2k})$$

转化问题：
 $A(x)$ 在 $\omega_n^0, \dots, \omega_n^{n-1}$ 的值，变为 $A^{[0]}(x), A^{[1]}(x)$ 在 $A(\omega_n^0)^2, \dots, (\omega_n^{n-1})^2$ 的值。
次数减半，递归硬++。

IDFT

IDFT实质上就是一个矩阵乘法

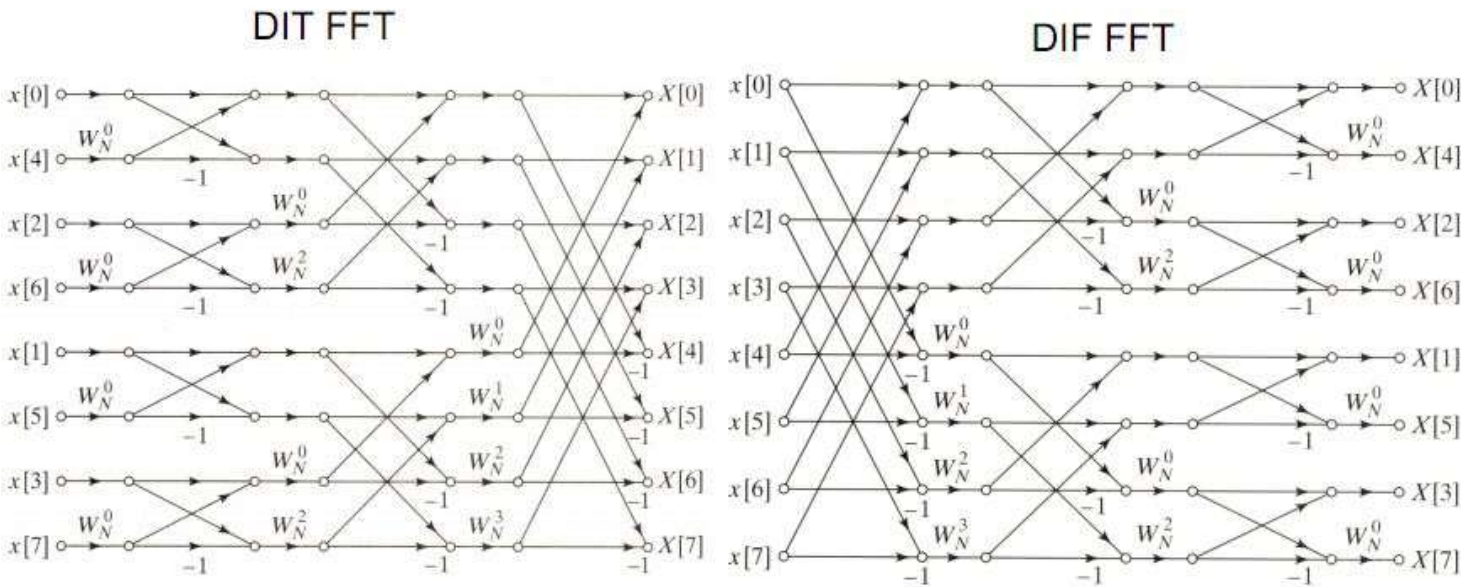
$$\begin{pmatrix} \omega_n^0 & \omega_n^0 & \omega_n^0 & \dots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \omega_n^2 & \dots & \omega_n^n \\ \omega_n^0 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2n} \\ \dots & \dots & \dots & \dots & \dots \\ \omega_n^0 & \omega_n^{n-1} & \omega_n^{2n-2} & \dots & \omega_n^{n^2-n} \end{pmatrix} * \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \dots \\ b_{n-1} \end{pmatrix}$$

关键是找到左边矩阵的逆矩阵。可以构造出来：每个元素取共轭复数，再除以 n 。正确性易证。
到此我们已经可以写出一个常数较大的递归FFT了。

蝴蝶优化

观察到我们的分治是将奇偶分类，即对于每一层我取这层对应位数的0放在一起，1放在一起。
那么最后元素*i*所处的位置，就是将*i*的二进制表示翻转的位置。

引入Cooley – Tukey算法的图：



板子

```
1.  /*
2.   Author:Scarlet
3.  */
4.  #define pi M_PI
5.  typedef complex<double>C;
6.  int N,g[maxn];
7.  void DFT(C *a,int f)
```

```

8.  {
9.    for(int i=0;i<N;i++) if(g[i]>i) swap(a[i], a[g[i]]);
10.   for(int i=1;i<N;i<=1)
11.   {
12.       C e(cos(pi/i), f*sin(pi/i));
13.       for(int j=0;j<N;j+=i<<1)
14.       {
15.           C w(1,0);
16.           for(int k=0;k<i;k++, w*=e)
17.           {
18.               C x=a[j+k], y=w*a[j+k+i];
19.               a[j+k]=x+y; a[j+k+i]=x-y;
20.           }
21.       }
22.   }
23.   if(f-1) for(int i=0;i<N;i++) a[i]/=N;
24. }
25. void mul(C *a, C *b, int n)
26. {
27.     int t=-1;
28.     for(N=1;N<=n;N<=1, t++);
29.     for(int i=1;i<N;i++) g[i]=(g[i>>1]>>1) | ((i&1)<<t);
30.     DFT(a, 1); DFT(b, 1);
31.     for(int i=0;i<N;i++)
32.         a[i]=a[i]*b[i];
33.     DFT(a, -1);
34. }

```

Number Theory Transform

引入

复数计算常数大，误差大，*NTT*可以做到多项式乘法的过程中取模。

快速数论变换

当模数十分特殊时可以用数论变换加速，基于一个极强的等价：

$$g^{\frac{P-1}{m}} \equiv \omega_m \pmod{P}$$

其中 g 是 P 的原根。

条件是 $2^k | P - 1$ ，其中 $2^k > n$ 。

板子

```

1.  /*
2.     Author:Scarlet
3.  */
4.  void NTT(LL *a, int f)
5.  {
6.      for(int i=0;i<N;i++) if(g[i]>i) swap(a[i], a[g[i]]);
7.      for(int b=0, i=1; i<N; i<=1)
8.      {
9.          b++;
10.         LL e=Pow(3, (1<<23-b)*119);
11.         if(f<0) e=Pow(e, mod-2);
12.         for(int j=0; j<N; j+=i<<1)
13.         {
14.             LL w=1;
15.             for(int k=0; k<i; k++, w=w*e%mod)
16.             {
17.                 LL x=a[j+k], y=w*a[j+k+i]%mod;
18.                 a[j+k]=x+y; a[j+k+i]=x-y;
19.                 if(a[j+k+i]<0) a[j+k+i]+=mod;
20.                 if(a[j+k]>=mod) a[j+k]-=mod;
21.             }
22.         }

```

```

23.     }
24.     if (f < 0)
25.     {
26.         LL v = Pow(N, mod - 2);
27.         for (int i = 0; i < N; i++) a[i] = a[i] * v % mod;
28.     }
29. }
```

附NTT常用模数原根表

<i>Prime</i>	<i>deg</i>	<i>g</i>
469762049	2	3
998244353	23	3
1004535809	21	3
1107296257	24	10
10000093151233	26	5
1000000523862017	26	3
1000000000949747713	26	2

选自[CodeForces Submission #19950343](https://codeforces.com/contest/1000/submission/19950343)

Inverse Element of Polynomial

引入

该说啥我也不知道辣

多项式求逆

给出多项式 $A(x)$ ，求多项式 $B(x)$ 使得

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

倍增

这里我们将使用一种倍增的思想来完成。

首先，当 $n = 1$ 时， $B[0] = A[0]^{-1}$ 。

假如我们已经求出了在模 $x^{\lceil \frac{t}{2} \rceil}$ 意义下的答案 $B_0(x)$ ，要求在 x^t 意义下的答案。那么：

$$A(x) * B_0(x) \equiv 1 \pmod{x^{\lceil \frac{t}{2} \rceil}}$$

$$A(x) * B(x) \equiv 1 \pmod{x^{\lceil \frac{t}{2} \rceil}}$$

由于 $A(x)$ 逆元 $B_0(x)$ 存在，故得：

$$B(x) - B_0(x) \equiv 0 \pmod{x^{\lceil \frac{t}{2} \rceil}}$$

那么我们将等式各部分平方，展开后得：

$$B^2(x) - 2 * B(x) * B_0(x) + B_0^2(x) \equiv 0 \pmod{x^t}$$

两边同乘 $A(x)$ ，利用逆元的性质移项便可得：

$$B(x) \equiv B_0(x) * (2 - A(x) * B_0(x)) \pmod{x^t}$$

上式便可在 $O(t \log t)$ 时间内求出了。

由于每次是倍增的，所以总复杂度可得为 $O(n \log n)$ 。

板子

1.

Polynomial Division

引入

多项式除法...

多项式除法

给出多项式 $A(x), B(x)$ ，求出多项式 $D(x), R(x)$ ，满足：

$$\begin{aligned} A(x) &= B(x) * D(x) + R(x) \\ \deg D &= \deg A - \deg B \\ \deg R &\leq \deg B - 1 \end{aligned}$$

其中： $\deg F$ 表示 $F(x)$ 的最高次项的次数。

不妨令：

$$\begin{aligned} n &= \deg A \\ m &= \deg B \end{aligned}$$

假如我们可以把多项式放到模 x^k 意义下，那很多操作都会很好做了。

这样，我们就要排除掉 $R(x)$ 的影响。

怎么办呢？定义：

$$F^R(x) = x^{\deg F} * F\left(\frac{1}{x}\right)$$

考虑下这个变换的形式理解，即将多项式的系数翻转，高次项的到低次项来，低次项的到高次项去。

那么不妨将最初的等式两边的 $F(x)$ 替换为 $F(\frac{1}{x})$ ，再都乘以 x^n ，看能得到什么。

$$\begin{aligned} x^n * A\left(\frac{1}{x}\right) &= [x^m B\left(\frac{1}{x}\right)][x^{n-m} D\left(\frac{1}{x}\right)] + x^{n-m+1} [x^{m-1} R\left(\frac{1}{x}\right)] \\ A^R(x) &= B^R(x) * D^R(x) + x^{n-m+1} R^R(x) \end{aligned}$$

放至模意义下！就消去了 $R(x)$ 带来的影响！

$$A^R(x) \equiv B^R(x) * D^R(x) \pmod{x^{n-m+1}} \quad (1)$$

$$D^R(x) \equiv A^R(x) * [B^R(x)]^{-1} \pmod{x^{n-m+1}} \quad (2)$$

注意到 $\deg D = n - m$ ，故 $D^R(x)$ 不会受到模意义的影响！

我们可以利用多项式求逆算出 $[B^R(x)]^{-1}$ ，用FFT算出 $D^R(x)$ ，然后代入最开始的式子中，就可以解得 $R(x)$ 了。

注意到由于 $B(x)$ 的 $\deg B$ 项必然不为0, 故 $B^R(x)$ 的第0项必然不为0。则逆元一定存在。
复杂度: $O(n \log n)$

板子

1.

Square Root of Polynomial

引入

Picks好神啊

多项式开根

给出多项式 $A_n(x)^2$. 求 $A(x)$.

显然 $A_1(x) \equiv \sqrt{A[0]} \pmod{x}$. 注意这里的开方可以开出负数。

考虑如何从 $A_n(x)$ 推到 $A_{2n}(x)$.

$$\begin{aligned} (A_n(x)^2 - A(x))^2 &\equiv 0 & (\text{mod } x^{2n}) \\ (A_n(x)^2 + A(x))^2 &\equiv 4 * A_n(x)^2 * A(x) & (\text{mod } x^{2n}) \\ \left(\frac{A_n(x)^2 + A(x)}{2 * A_n(x)}\right)^2 &\equiv A(x) & (\text{mod } x^{2n}) \\ (2^{-1} * A_n(x) + 2^{-1} * A(x) * A_n^{-1}(x))^2 &\equiv A(x) & (\text{mod } x^{2n}) \end{aligned}$$

注意到实际上 $A_{2n}(x)$ 的末 n 位与 $A_n(x)$ 一致。故可以同时维护 $A_n^{-1}(x)$
与多项式求逆的复杂度分析一致, 此时我们的复杂度依然是 $O(n \log n)$

板子

1.

Simple模型

前置技能讲完了是不是讲模型了啊

卷积

$$A_k = \sum_{i=0}^k B_i * C_{k-i}$$

好像和多项式乘法没什么区别啊...

反卷积

$$A_k = \sum_{i=k}^N B_i * C_{i-k}$$

构造 B^R , 指标变换就变回卷积了

$$A_k = G_{N-k} = \sum_{i=0}^{N-k} B_{N-k-i}^R * C_i$$

其中 $G(x) = B^R(x) * C(x)$

可重集内 n 元（有序）数对和

构造 $A_n = S.count(n)$ ，自乘 n 次，快速幂。

可重集内 n 元（有序）数对积取质数模

XJB找个原根对 $A_n = S.count(n)$ 的每个下标取指标，然后同上

可重集内数对差

构造 $A_n = S.count(n)$ ，然后 A 和 A^R 卷积，所得答案要除以2

咦，好像就是反卷积嘛..

多项式公因式

类比与欧几里德算法，我们可以证明欧几里德算法需要的结论在多项式上均成立。

考虑使用欧几里德算法。

每次相当于求两个多项式的商与余数。

正好可以利用多项式除法解决。

不过每次取模均只能将最高次减小1。

复杂度 $O(N^2 \log N)$

多项式求逆EXT

类比扩展欧几里得

齐次线性递推数列

$$h_n = \begin{cases} h_n & n < k \\ a_1 h_{n-1} + a_2 h_{n-2} + \dots + a_k h_{n-k} & n \geq k \end{cases}$$

对于递推式

$$h_n = a_1 h_{n-1} + a_2 h_{n-2} + \dots + a_k h_{n-k}$$

的生成函数

$$F(x) = h_0 + h_1 x + h_2 x^2 + \dots$$

构造函数：

$$H(x) = h_0 + h_1 x + h_2 x^2 + \dots + h_{k-1} x^{k-1}$$

$$A(x) = 1 - a_1 x - a_2 x^2 - \dots - a_k x^k$$

则：

$$F(x)A(x) = H(x)A(x) \pmod{x^k}$$

$$F(x) = \frac{H(x)A(x) \pmod{x^k}}{A(x)}$$

非齐次线性递推数列

$$h_n = \begin{cases} h_n & n < k \\ a_1 h_{n-1} + a_2 h_{n-2} + \dots + a_k h_{n-k} + g(n) & n \geq k \end{cases}$$

其中 $g(n)$ 是关于 n 的函数，可以是常函数

和之前一样，构造多项式：

$$H(x) = h_0 + h_1x + h_2x^2 + \dots + h_{k-1}x^{k-1}$$

$$A(x) = 1 - a_1x - a_2x^2 - \dots - a_kx^k$$

$$G(x) = g(0) + g(1)x + g(2)x^2 + \dots$$

则：

$$F(x)A(x) = H(x)A(x) \bmod x^k + (G(x) - G(x) \bmod x^k)$$

$$F(x) = \frac{[H(x)A(x) - G(x)] \bmod x^k + G(x)}{A(x)}$$

所以如果 $G(x)$ 能被有限项多项式表示的话就做完了。。。

卡特兰数列（自卷积数列）

$$h_n = \begin{cases} 1 & n = 0 \\ h_0h_{n-1} + h_1h_{n-2} + \dots + h_{n-1}h_0 & n \geq 1 \end{cases}$$

构造 $F(x) = h_0 + h_1x + h_2x^2 + \dots$

容易发现这个多项式就是自身的卷积，所以我们直接把这个多项式自乘一下

$$F^2(x) = h_1 + h_2x + h_3x^2 + \dots = \frac{F(x)-1}{x}$$

解得

$$F(x) = \frac{1 - \sqrt{1 - 4x}}{2x}$$

辣鸡例题

UOJ#34. 多项式乘法

模板！

```

1.  /*
2.     Author:Scarlet
3.  */
4.  #include<bits/stdc++.h>
5.  #define maxn 262144
6.  using namespace std;
7.  typedef long long LL;
8.  #define G c=getchar()
9.  inline int read()
10. {
11.     int x=0,f=1;char G;
12.     while(c>57||c<48){if(c=='-')f=-1;G;}
13.     while(c>47&& c<58){x=x*10+c-48;G;}
14.     return x*f;
15. }
16. #define pi M_PI
17. typedef complex<double>C;
18. int N,g[maxn];
19. C a[maxn],b[maxn];
20. void DFT(C *a,int f)
21. {
22.     for(int i=0;i<N;i++)if(g[i]>i)swap(a[i],a[g[i]]);
23.     for(int i=1;i<N;i<=1)
24.     {
25.         C e(cos(pi/i),f*sin(pi/i));
26.         for(int j=0;j<N;j+=i<1)
27.         {
28.             C w(1,0);
29.             for(int k=0;k<i;k++,w*=e)
30.             {
31.                 C x=a[j+k],y=w*a[j+k+i];
32.                 a[j+k]=x+y;a[j+k+i]=x-y;
33.             }

```

```

34.     }
35.   }
36.   if(f-1)for(int i=0;i<N;i++)a[i]/=N;
37. }
38. void mul(C *a,C *b,int n)
39. {
40.     int t=-1;
41.     for(N=1;N<=n;N<=1,t++);
42.     for(int i=1;i<N;i++)g[i]=(g[i>>1]>>1)|((i&1)<<t);
43.     DFT(a,1);DFT(b,1);
44.     for(int i=0;i<N;i++)
45.         a[i]=a[i]*b[i];
46.     DFT(a,-1);
47. }
48. int main()
49. {
50.     int n=read(),m=read();
51.     for(int i=0;i<=n;i++)a[i]=read();
52.     for(int i=0;i<=m;i++)b[i]=read();
53.     mul(a,b,n+m+1);
54.     for(int i=0;i<=n+m;i++)
55.         printf("%d ",(int)(a[i].real()+0.5));
56. }

```

BZOJ3527: [Zjoi2014]力

感觉挺傻逼的啊...

$$E_j = \sum_{i<j} \frac{q_j}{(i-j)^2} - \sum_{i>j} \frac{q_j}{(i-j)^2}$$

可以拆成左右两边算，发现左边就是一个卷积，右边则是一个反卷积。
不就做完了？

```

1.  /*
2.     Author:Scarlet
3.  */
4.  #include<bits/stdc++.h>
5.  #define maxn 262144
6.  using namespace std;
7.  typedef long long LL;
8.  #define G c=getchar()
9.  inline int read()
10. {
11.     int x=0,f=1;char G;
12.     while(c>57||c<48){if(c=='-')f=-1;G;}
13.     while(c>47&& c<58){x=x*10+c-48;G;}
14.     return x*f;
15. }
16. typedef complex<long double> C;
17. #define pi M_PI
18. int N,g[maxn];
19. C q[maxn],qq[maxn],f1[maxn],f2[maxn];
20. void DFT(C *a,int f)
21. {
22.     for(int i=0;i<N;i++)if(g[i]<i)swap(a[i],a[g[i]]);
23.     for(int i=1;i<N;i<=1)
24.     {
25.         C e(cos(pi/i),f*sin(pi/i));
26.         for(int j=0;j<N;j+=i<<1)
27.         {
28.             C w(1,0);
29.             for(int k=0;k<i;k++,w*=e)
30.             {
31.                 C x=a[j+k],y=w*a[j+k+i];
32.                 a[j+k]=x+y,a[j+k+i]=x-y;
33.             }
34.         }
35.     }
36.     if(f-1)for(int i=0;i<N;i++)a[i]/=N;
37. }
38. void mul(C *a,C *b,int n)
39. {
40.     int t=-1;

```

```

41.     for(N=1;N<=n;N<=1, t++);
42.     for(int i=1; i<N; i++) g[i]=g[i>>1]>>1|((i&1)<<t);
43.     DFT(a, 1); DFT(b, 1);
44.     for(int i=0; i<N; i++) a[i]*=b[i];
45.     DFT(a, -1);
46. }
47. int main()
48. {
49.     int n=read(); double _;
50.     for(int i=0; i<n; i++) scanf("%lf", &_), qq[i]=q[i]=_;
51.     for(int i=1; i<n; i++) f1[i]=1.0/i/i;
52.     mul(f1, q, n+n);
53.     for(int i=0; i<n-1; i++) f2[i]=1.0/(n-1-i)/(n-1-i);
54.     mul(f2, qq, n+n);
55.     for(int i=0; i<n; i++)
56.         printf("%.10lf\n", (double)(f1[i].real()-f2[n-1+i].real()));
57.     return 0;
58. }

```

多DFT了一次 q ，有点慢...

BZOJ3160: 万径人踪灭

Manacher求一遍回文子串的数量

可以用数对和模型求一遍回文子序列

答案就是两个相减

```

1.  /*
2.     Author:Scarlet
3.  */
4.  #include<bits/stdc++.h>
5.  #define maxn 524288
6.  #define mod 1000000007
7.  using namespace std;
8.  typedef long long LL;
9.  #define G c=getchar()
10. inline int read()
11. {
12.     int x=0, f=1; char G;
13.     while(c>57||c<48){if(c=='-')f=-1;G;}
14.     while(c>47&& c<58){x=x*10+c-48;G;}
15.     return x*f;
16. }
17. #define pi M_PI
18. int g[maxn], N;
19. typedef complex<double> C;
20. C b[maxn], a[maxn];
21. char s[maxn];
22. void DFT(C *a, int f)
23. {
24.     for(int i=0; i<N; i++) if(g[i]<i) swap(a[i], a[g[i]]);
25.     for(int i=1; i<N; i<=1)
26.     {
27.         C e(cos(pi/i), f*sin(pi/i));
28.         for(int j=0; j<N; j+=i<<1)
29.         {
30.             C w(1, 0);
31.             for(int k=0; k<i; k++, w*=e)
32.             {
33.                 C x=a[j+k], y=w*a[j+k+i];
34.                 a[j+k]=x+y, a[j+k+i]=x-y;
35.             }
36.         }
37.     }
38.     if(f-1) for(int i=0; i<N; i++) a[i]/=N;
39. }
40. int manacher(char str[], int n)
41. {
42.     static char s[maxn];
43.     static int f[maxn];
44.     s[0]='$', s[1]='#';
45.     for(int i=1; i<=n; i++)

```

```

46.         s[i<<1]=str[i], s[i<<1|1]='#';
47.     n=n<<1|1;
48.     int mx=1, id=1, re=0;
49.     for(int i=1; i<=n; i++)
50.     {
51.         f[i]=min(f[id*2-i], mx-i);
52.         for(; s[i+f[i]]==s[i-f[i]]; f[i]++);
53.         if(f[i]+i>mx) mx=f[i]+i, id=i;
54.         re+=f[i]>>1, re%=mod;
55.     }
56.     return re;
57. }
58. LL gg[maxn];
59. int P[maxn];
60. int main()
61. {
62.     P[0]=1; for(int i=1; i<maxn; i++) P[i]=(P[i-1]<<1)%mod;
63.     scanf("%s", s+1); int n=strlen(s+1);
64.     int ans=-manacher(s, n), t=-1;
65.     for(N=1; N<=n*2; N<=1) t++;
66.     for(int i=1; i<N; i++) g[i]=g[i>>1]>>1|((i&1)<<t);
67.     for(int i=1; i<=n; i++) a[i]=(int)(s[i]=='a');
68.     DFT(a, 1);
69.     for(int i=0; i<N; i++) b[i]=a[i]*a[i];
70.     memset(a, 0, sizeof(a));
71.     for(int i=1; i<=n; i++) a[i]=(int)(s[i]=='b');
72.     DFT(a, 1);
73.     for(int i=0; i<N; i++) b[i]+=a[i]*a[i];
74.     DFT(b, -1);
75.     for(int i=1; i<N; i++) gg[i]+=LL(b[i].real()+0.5);
76.     for(int i=1; i<N; i++) (ans+=P[gg[i]+1>>1]-1)%=mod;
77.     printf("%d", ans<0?mod+ans:ans);
78.     return 0;
79. }

```

BZOJ3992: [SDOI2015]序列统计

辣鸡指标变换

```

1.  /*
2.     Author:Scarlet
3.  */
4.  #include<bits/stdc++.h>
5.  #define maxn 16384
6.  #define mod 1004535809
7.  using namespace std;
8.  typedef long long LL;
9.  #define _ c=getchar()
10. inline LL read()
11. {
12.     LL x=0, f=1; char _;
13.     while(c>57||c<48){if(c=='-') f=-1; _;}
14.     while(c>47&& c<58){x=x*10+c-48; _;}
15.     return x*f;
16. }
17. int n, m, x, S, G, N, g[maxn];
18. LL Pow(LL x, LL y)
19. {
20.     LL a=1;
21.     for(x%=mod; y>=1, x=x*x%mod)
22.         if(y&1) a=a*x%mod;
23.     return a;
24. }
25. bool check(int p, int m)
26. {
27.     int w=p;
28.     for(int i=1; i<m-2; i++, (w*=p)%=m)
29.         if(w==1) return 0;
30.     return 1;
31. }
32. int gpr(int m)
33. {
34.     for(int i=2; i++)

```

```

35.         if(check(i,m))return i;
36.     }
37. void NTT(LL *a, int f)
38. {
39.     int b=0;
40.     for(int i=0;i<N;i++)if(g[i]>i)swap(a[i],a[g[i]]);
41.     for(int i=1;i<N;i<=1)
42.     {
43.         b++;
44.         LL e=Pow(3, (1<<(21-b))*479);
45.         if(f<0)e=Pow(e,mod-2);
46.         for(int j=0;j<N;j+=i<<1)
47.         {
48.             LL w=1;
49.             for(int k=0;k<i;k++,w=w*e%mod)
50.             {
51.                 LL x=a[j+k],y=w*a[j+k+i]%mod;
52.                 a[j+k]=x+y;a[j+k+i]=x-y;
53.                 if(a[j+k]>=mod)a[j+k]-=mod;
54.                 if(a[j+k+i]<0)a[j+k+i]+=mod;
55.             }
56.         }
57.     }
58.     if(f<0)
59.     {
60.         LL v=Pow(N,mod-2);
61.         for(int i=0;i<N;i++)a[i]=a[i]*v%mod;
62.     }
63. }
64. LL c[maxn];
65. void mul(LL *a, LL *b)
66. {
67.     if(a==b)
68.     {
69.         NTT(a, 1);
70.         for(int i=0;i<N;i++)
71.             a[i]=a[i]*a[i]%mod;
72.         NTT(a, -1);
73.         for(int i=0;i<m-1;i++)
74.             (a[i]+=a[i+m-1])%=mod, a[i+m-1]=0;
75.         return;
76.     }
77.     memcpy(c, b, sizeof(c));
78.     NTT(a, 1);NTT(b, 1);
79.     for(int i=0;i<N;i++)
80.         a[i]=a[i]*b[i]%mod;
81.     NTT(a, -1);
82.     for(int i=0;i<m-1;i++)
83.         (a[i]+=a[i+m-1])%=mod, a[i+m-1]=0;
84.     memcpy(b, c, sizeof(c));
85. }
86. int A[maxn], B[maxn];
87. LL a[maxn], b[maxn];
88. int main()
89. {
90.     n=read(), m=read(), x=read(), S=read();
91.     int t=-1;
92.     for(N=1;N<=2*m;N<=1)t++;
93.     for(int i=1;i<N;i++)g[i]=g[i>>1]>>1|((i&1)<<t);
94.     G=gpr(m);
95.     for(LL i=0,w=1;i<m-1;i++,w=w*G%mod)B[i]=w;
96.     for(int i=0;i<m-1;i++)A[B[i]]=i;
97.     for(int i=1;i<=S;i++)
98.     {
99.         int x=read();
100.        if(x)b[A[x]]++;
101.    }
102.    a[0]=1;
103.    for(;n>=1,mul(b,b))
104.        if(n&1)mul(a,b);
105.    printf("%11d",a[A[x]]);
106.    return 0;
107. }

```

codeforces #250E The Child and Binary Tree

BZOJ3456: 城市规划

BZOJ3684: 大朋友和多叉树

codeforces #568B Symmetric and Transitive

UOJ#50. 【UR #3】链式反应

UOJ#23. 【UR #1】跳蚤国王下江南

UOJ#86. mx的组合数

UOJ#102. 【集训队互测2015】ydc的奖金

UOJ#182. 【UR #12】 $a^{-1} + b$ problem

References

说是References实际上是Copies&Rearrangements啦

- [1]Picks.[Fast Fourier Transform](#)[EB/OL].logdown,2014
- [2]Picks.[Inverse Element of Polynomial](#)[EB/OL].logdown,2014
- [3]Picks.[Polynomial Division](#)[EB/OL].logdown,2014
- [4]Picks.[Square Root of Polynomial](#)[EB/OL].logdown,2014
- [5]PoPoQQQ.[一些常见数列的生成函数推导](#)[EB/OL].csdn,2016

• 内容目录

-
- [辣鸡多项式毁我青春](#)
 - [前置技能](#)
 - [Fast Fourier Transform](#)
 - [引入](#)
 - [多项式乘法](#)
 - [前置技能的前置技能：单位根](#)
 - [DFT](#)
 - [IDFT](#)
 - [蝴蝶优化](#)
 - [板子](#)
 - [Number Theory Transform](#)
 - [引入](#)
 - [快速数论变换](#)
 - [板子](#)
 - [附NTT常用模数原根表](#)
 - [Inverse Element of Polynomial](#)
 - [引入](#)
 - [多项式求逆](#)
 - [倍增](#)
 - [板子](#)
 - [Polynomial Division](#)
 - [引入](#)
 - [多项式除法](#)
 - [板子](#)
 - [Square Root of Polynomial](#)
 - [引入](#)

- [多项式开根](#)
 - [板子](#)
- [Simple模型](#)
 - [卷积](#)
 - [反卷积](#)
 - [可重集内n元\(有序\)数对和](#)
 - [可重集内n元\(有序\)数对积取质数模](#)
 - [可重集内数对差](#)
 - [多项式公因式](#)
 - [多项式求逆EXT](#)
 - [齐次线性递推数列](#)
 - [非齐次线性递推数列](#)
 - [卡特兰数列\(自卷积数列\)](#)
- [辣鸡例题](#)
 - [UOJ#34. 多项式乘法](#)
 - [BZOJ3527: \[Zjoi2014\]力](#)
 - [BZOJ3160: 万径人踪灭](#)
 - [BZOJ3992: \[SDOI2015\]序列统计](#)
 - [codeforces #250E The Child and Binary Tree](#)
 - [BZOJ3456: 城市规划](#)
 - [BZOJ3684: 大朋友和多叉树](#)
 - [codeforces #568B Symmetric and Transitive](#)
 - [UOJ#50. 【UR #3】链式反应](#)
 - [UOJ#23. 【UR #1】跳蚤国王下江南](#)
 - [UOJ#86. mx的组合数](#)
 - [UOJ#102. 【集训队互测2015】ydc的奖金](#)
 - [UOJ#182. 【UR #12】 \$a^{-1} + b\$ problem](#)
- [References](#)

•

- - ■ 1997 1
 - ■ 2003 1
 - ■ 2004 1
 - ■ 2005 1
 - ■ 2006 1
 - ■ 2016 1
 - ■ 51nod 1
 - ■ BZOJ 2
 - ■ CF 1
 - ■ NOIP 1
 - ■ OI 5
 - ■ POI 5
 - ■ 多项式 1
 - [辣鸡多项式毁我青春](#)
 - ■ 字符串 1
 - ■ 我好菜啊 1
 - ■ 数学 1
 - ■ 数据结构 1
 - ■ 算法 1
 - ■ 综合 1
 - ■ 解题报告 6
 - ■ 未分类 1

搜索 Scarlet 的文稿标题, *

•

-
-
- - [下载客户端](#)
 - [关注开发者](#)
 - [报告问题，建议](#)
 - [联系我们](#)
-

添加新批注



- 私有
- 公开
- 删除

回复批注



通知

-
-