[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。

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$O(nlogn)$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$O(nlogn)$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$\omega_n^1$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. $\omega_n^j \omega_n^k = \omega_n^{j+k} = \omega_n^{j+k \mod n}$（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：

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

亦即

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

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

IDFT

$IDFT$实质上就是一个矩阵乘法

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

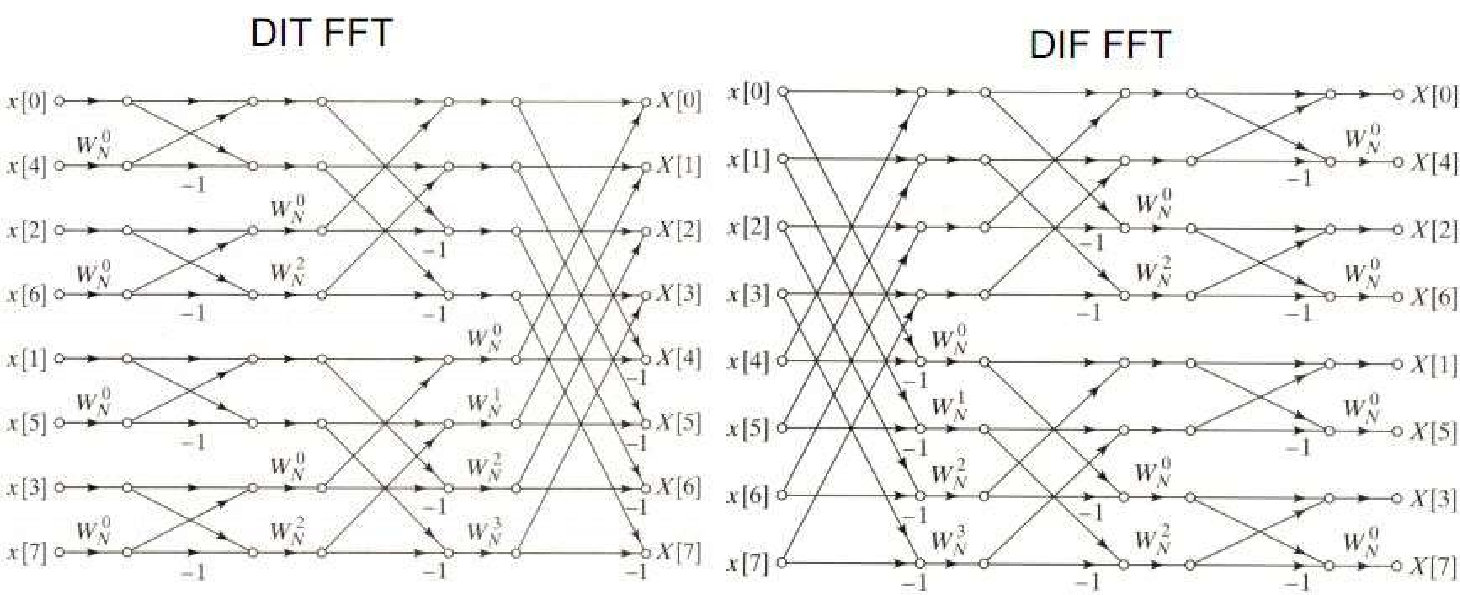
关键是找到左边矩阵的逆矩阵。可以构造出来：每个元素取共轭复数，再除以n。正确性易证。

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

```
/*
    Author:Scarlet
*/
#define pi M_PI
typedef complex<double>C;
int N,g[maxn];
void DFT(C *a,int f)
```

```
{
    for(int i=0;i<N;i++)if(g[i]>i)swap(a[i],a[g[i]]);
    for(int i=1;i<N;i<<=1)
    {
        C e(cos(pi/i),f*sin(pi/i));
        for(int j=0;j<N;j+=i<<1)
        {
            C w(1,0);
            for(int k=0;k<i;k++,w*=e)
            {
                C x=a[j+k],y=w*a[j+k+i];
                a[j+k]=x+y;a[j+k+i]=x-y;
            }
        }
    }
    if(f-1)for(int i=0;i<N;i++)a[i]/=N;
}
void mul(C *a,C *b,int n)
{
    int t=-1;
    for(N=1;N<=n;N<<=1,t++);
    for(int i=1;i<N;i++)g[i]=(g[i>>1]>>1)|((i&1)<<t);
    DFT(a,1);DFT(b,1);
    for(int i=0;i<N;i++)
        a[i]=a[i]*b[i];
    DFT(a,-1);
}
```

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

```
/*
    Author:Scarlet
*/
void NTT(LL *a,int f)
{
    for(int i=0;i<N;i++)if(g[i]>i)swap(a[i],a[g[i]]);
    for(int b=0,i=1;i<N;i<<=1)
    {
        b++;
        LL e=Pow(3,(1<<23-b)*119);
        if(f<0)e=Pow(e,mod-2);
        for(int j=0;j<N;j+=i<<1)
        {
            LL w=1;
            for(int k=0;k<i;k++,w=w*e%mod)
            {
                LL x=a[j+k],y=w*a[j+k+i]%mod;
                a[j+k]=x+y;a[j+k+i]=x-y;
                if(a[j+k+i]<0)a[j+k+i]+=mod;
                if(a[j+k]>=mod)a[j+k]-=mod;
            }
        }
```

```
        }
        if(f<0)
        {
            LL v=Pow(N,mod-2);
            for(int i=0;i<N;i++)a[i]=a[i]*v%mod;
        }
    }
```

附NTT常用模数原根表

$Prime$	deg	g
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

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：

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

其中：$\deg F$表示$F(x)$的最高次项的次数。

不妨令：

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

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。
怎么做呢？定义：

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

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。

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

放至模意义下！就消去了$R(x)$带来的影响！

$$
\begin{aligned}
A^R(x)&\equiv B^R(x)*D^R(x) &\quad (mod\ x^{n-m+1}) \qquad (1)\\
D^R(x)&\equiv A^R(x)*[B^R(x)]^{-1} &\quad (mod\ x^{n-m+1}) \qquad (2)
\end{aligned}
$$

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$A_{2n}(x)$.

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

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

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

对于递推式
$h_n = a_1 h_{n-1} + a_2 h_{n-2} + \ldots + a_k h_{n-k}$
的生成函数
$F(x) = h_0 + h_1 x + h_2 x^2 + \ldots$
构造函数：
$H(x) = h_0 + h_1 x + h_2 x^2 + \ldots + h_{k-1} x^{k-1}$
$A(x) = 1 - a_1 x - a_2 x^2 - \ldots - a_k x^k$
则：
$F(x)A(x) = H(x)A(x) \mod x^k$
$F(x) = \frac{H(x)A(x) \mod x^k}{A(x)}$

非齐次线性递推数列

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

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：
$H(x)=h_0+h_1x+h_2x^2+\ldots+h_{k-1}x^{k-1}$
$A(x)=1-a_1x-a_2x^2-\ldots-a_kx^k$
$G(x)=g(0)+g(1)x+g(2)x^2+\ldots$
则：
$F(x)A(x)=H(x)A(x) \mod x^k+(G(x)-G(x) \mod x^k)$
$F(x)=\frac{[H(x)A(x)-G(x)]modx^k+G(x)}{A(x)}$
所以如果$G(x)$能被有限项多项式表示的话就做完了。。。

卡塔兰数列（自卷积数列）

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

构造$F(x)=h_0+h_1x+h_2x^2+\ldots$
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
$F^2(x)=h_1+h_2x+h_3x^2+\ldots=\frac{F(x)-1}{x}$
解得

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

辣鸡例题

UOJ#34. 多项式乘法

模板！

```
/*
    Author:Scarlet
*/
#include<bits/stdc++.h>
#define maxn 262144
using namespace std;
typedef long long LL;
#define G c=getchar()
inline int read()
{
    int x=0,f=1;char G;
    while(c>57||c<48){if(c=='-')f=-1;G;}
    while(c>47&&c<58){x=x*10+c-48;G;}
    return x*f;
}
#define pi M_PI
typedef complex<double>C;
int N,g[maxn];
C a[maxn],b[maxn];
void DFT(C *a,int f)
{
    for(int i=0;i<N;i++)if(g[i]>i)swap(a[i],a[g[i]]);
    for(int i=1;i<N;i<<=1)
    {
        C e(cos(pi/i),f*sin(pi/i));
        for(int j=0;j<N;j+=i<<1)
        {
            C w(1,0);
            for(int k=0;k<i;k++,w*=e)
            {
                C x=a[j+k],y=w*a[j+k+i];
                a[j+k]=x+y;a[j+k+i]=x-y;
            }
```

```
        }
    }
    if(f-1)for(int i=0;i<N;i++)a[i]/=N;
}
void mul(C *a,C *b,int n)
{
    int t=-1;
    for(N=1;N<=n;N<<=1,t++);
    for(int i=1;i<N;i++)g[i]=(g[i>>1]>>1)|((i&1)<<t);
    DFT(a,1);DFT(b,1);
    for(int i=0;i<N;i++)
        a[i]=a[i]*b[i];
    DFT(a,-1);
}
int main()
{
    int n=read(),m=read();
    for(int i=0;i<=n;i++)a[i]=read();
    for(int i=0;i<=m;i++)b[i]=read();
    mul(a,b,n+m+1);
    for(int i=0;i<=n+m;i++)
        printf("%d ",(int)(a[i].real()+0.5));
}
```

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？

```
/*
    Author:Scarlet
*/
#include<bits/stdc++.h>
#define maxn 262144
using namespace std;
typedef long long LL;
#define G c=getchar()
inline int read()
{
    int x=0,f=1;char G;
    while(c>57||c<48){if(c=='-')f=-1;G;}
    while(c>47&&c<58){x=x*10+c-48;G;}
    return x*f;
}
typedef complex<long double> C;
#define pi M_PI
int N,g[maxn];
C q[maxn],qq[maxn],f1[maxn],f2[maxn];
void DFT(C *a,int f)
{
    for(int i=0;i<N;i++)if(g[i]<i)swap(a[i],a[g[i]]);
    for(int i=1;i<N;i<<=1)
    {
        C e(cos(pi/i),f*sin(pi/i));
        for(int j=0;j<N;j+=i<<1)
        {
            C w(1,0);
            for(int k=0;k<i;k++,w*=e)
            {
                C x=a[j+k],y=w*a[j+k+i];
                a[j+k]=x+y,a[j+k+i]=x-y;
            }
        }
    }
    if(f-1)for(int i=0;i<N;i++)a[i]/=N;
}
void mul(C *a,C *b,int n)
{
    int t=-1;
```

```
    for(N=1;N<=n;N<<=1,t++);
    for(int i=1;i<N;i++)g[i]=g[i>>1]>>1|((i&1)<<t);
    DFT(a,1);DFT(b,1);
    for(int i=0;i<N;i++)a[i]*=b[i];
    DFT(a,-1);
}
int main()
{
    int n=read();double _;
    for(int i=0;i<n;i++)scanf("%lf",&_),qq[i]=q[i]=_;
    for(int i=1;i<n;i++)f1[i]=1.0/i/i;
    mul(f1,q,n+n);
    for(int i=0;i<n-1;i++)f2[i]=1.0/(n-1-i)/(n-1-i);
    mul(f2,qq,n+n);
    for(int i=0;i<n;i++)
        printf("%.10lf\n",(double)(f1[i].real()-f2[n-1+i].real()));
    return 0;
}
```

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

```
/*
    Author:Scarlet
*/
#include<bits/stdc++.h>
#define maxn 524288
#define mod 1000000007
using namespace std;
typedef long long LL;
#define G c=getchar()
inline int read()
{
    int x=0,f=1;char G;
    while(c>57||c<48){if(c=='-')f=-1;G;}
    while(c>47&&c<58){x=x*10+c-48;G;}
    return x*f;
}
#define pi M_PI
int g[maxn],N;
typedef complex<double> C;
C b[maxn],a[maxn];
char s[maxn];
void DFT(C *a,int f)
{
    for(int i=0;i<N;i++)if(g[i]<i)swap(a[i],a[g[i]]);
    for(int i=1;i<N;i<<=1)
    {
        C e(cos(pi/i),f*sin(pi/i));
        for(int j=0;j<N;j+=i<<1)
        {
            C w(1,0);
            for(int k=0;k<i;k++,w*=e)
            {
                C x=a[j+k],y=w*a[j+k+i];
                a[j+k]=x+y,a[j+k+i]=x-y;
            }
        }
    }
    if(f-1)for(int i=0;i<N;i++)a[i]/=N;
}
int manacher(char str[],int n)
{
    static char s[maxn];
    static int f[maxn];
    s[0]='$',s[1]='#';
    for(int i=1;i<=n;i++)
```

```
        s[i<<1]=str[i],s[i<<1|1]='#';
    n=n<<1|1;
    int mx=1,id=1,re=0;
    for(int i=1;i<=n;i++)
    {
        f[i]=min(f[id*2-i],mx-i);
        for(;s[i+f[i]]==s[i-f[i]];f[i]++);
        if(f[i]+i>mx)mx=f[i]+i,id=i;
        re+=f[i]>>1,re%=mod;
    }
    return re;
}
LL gg[maxn];
int P[maxn];
int main()
{
    P[0]=1;for(int i=1;i<maxn;i++)P[i]=(P[i-1]<<1)%mod;
    scanf("%s",s+1);int n=strlen(s+1);
    int ans=-manacher(s,n),t=-1;
    for(N=1;N<=n*2;N<<=1)t++;
    for(int i=1;i<N;i++)g[i]=g[i>>1]>>1|((i&1)<<t);
    for(int i=1;i<=n;i++)a[i]=(int)(s[i]=='a');
    DFT(a,1);
    for(int i=0;i<N;i++)b[i]=a[i]*a[i];
    memset(a,0,sizeof(a));
    for(int i=1;i<=n;i++)a[i]=(int)(s[i]=='b');
    DFT(a,1);
    for(int i=0;i<N;i++)b[i]+=a[i]*a[i];
    DFT(b,-1);
    for(int i=1;i<N;i++)gg[i]+=LL(b[i].real()+0.5);
    for(int i=1;i<N;i++)(ans+=P[gg[i]+1>>1]-1)%=mod;
    printf("%d",ans<0?mod+ans:ans);
    return 0;
}
```

BZOJ3992: [SDOI2015]序列统计

辣鸡指标变换

```
/*
    Author:Scarlet
*/
#include<bits/stdc++.h>
#define maxn 16384
#define mod 1004535809
using namespace std;
typedef long long LL;
#define _ c=getchar()
inline LL read()
{
    LL x=0,f=1;char _;
    while(c>57||c<48){if(c=='-')f=-1;_;}
    while(c>47&&c<58){x=x*10+c-48;_;}
    return x*f;
}
int n,m,x,S,G,N,g[maxn];
LL Pow(LL x,LL y)
{
    LL a=1;
    for(x%=mod;y;y>>=1,x=x*x%mod)
        if(y&1)a=a*x%mod;
    return a;
}
bool check(int p,int m)
{
    int w=p;
    for(int i=1;i<m-2;i++,(w*=p)%=m)
        if(w==1)return 0;
    return 1;
}
int gpr(int m)
{
    for(int i=2;;i++)
```

```
        if(check(i,m))return i;
}
void NTT(LL *a,int f)
{
    int b=0;
    for(int i=0;i<N;i++)if(g[i]>i)swap(a[i],a[g[i]]);
    for(int i=1;i<N;i<<=1)
    {
        b++;
        LL e=Pow(3,(1<<21-b)*479);
        if(f<0)e=Pow(e,mod-2);
        for(int j=0;j<N;j+=i<<1)
        {
            LL w=1;
            for(int k=0;k<i;k++,w=w*e%mod)
            {
                LL x=a[j+k],y=w*a[j+k+i]%mod;
                a[j+k]=x+y;a[j+k+i]=x-y;
                if(a[j+k]>=mod)a[j+k]-=mod;
                if(a[j+k+i]<0)a[j+k+i]+=mod;
            }
        }
    }
    if(f<0)
    {
        LL v=Pow(N,mod-2);
        for(int i=0;i<N;i++)a[i]=a[i]*v%mod;
    }
}
LL c[maxn];
void mul(LL *a,LL *b)
{
    if(a==b)
    {
        NTT(a,1);
        for(int i=0;i<N;i++)
            a[i]=a[i]*a[i]%mod;
        NTT(a,-1);
        for(int i=0;i<m-1;i++)
            (a[i]+=a[i+m-1])%=mod,a[i+m-1]=0;
        return;
    }
    memcpy(c,b,sizeof(c));
    NTT(a,1);NTT(b,1);
    for(int i=0;i<N;i++)
        a[i]=a[i]*b[i]%mod;
    NTT(a,-1);
    for(int i=0;i<m-1;i++)
        (a[i]+=a[i+m-1])%=mod,a[i+m-1]=0;
    memcpy(b,c,sizeof(c));
}
int A[maxn],B[maxn];
LL a[maxn],b[maxn];
int main()
{
    n=read(),m=read(),x=read(),S=read();
    int t=-1;
    for(N=1;N<=2*m;N<<=1)t++;
    for(int i=1;i<N;i++)g[i]=g[i>>1]>>1|((i&1)<<t);
    G=gpr(m);
    for(LL i=0,w=1;i<m-1;i++,w=w*G%m)B[i]=w;
    for(int i=0;i<m-1;i++)A[B[i]]=i;
    for(int i=1;i<=S;i++)
    {
        int x=read();
        if(x)b[A[x]]++;
    }
    a[0]=1;
    for(;n;n>>=1,mul(b,b))
        if(n&1)mul(a,b);
    printf("%lld",a[A[x]]);
    return 0;
}
```

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**
