

生成函数理论 5

基本应用

例 1

P2000 拯救世界

依照题意可列出 10 个条件的生成函数 $f_i(x)$ 将它们相乘再化简即可。

- $f_1(x) = (1 + x^6 + x^{12} + \dots) = \frac{1}{1-x^6}$
- $f_2(x) = (1 + x + x^2 + \dots + x^9) = \frac{1-x^{10}}{1-x}$
- $f_3(x) = (1 + x + x^2 + \dots + x^5) = \frac{1-x^6}{1-x}$
- $f_4(x) = (1 + x^4 + x^8 + \dots) = \frac{1}{1-x^4}$
- $f_5(x) = (1 + x + x^2 + \dots + x^7) = \frac{1-x^8}{1-x}$
- $f_6(x) = (1 + x^2 + x^4 + \dots) = \frac{1}{1-x^2}$
- $f_7(x) = (1 + x) = \frac{1-x^2}{1-x}$
- $f_8(x) = (1 + x^8 + x^{16} + \dots) = \frac{1}{1-x^8}$
- $f_9(x) = (1 + x^{10} + x^{20} + \dots) = \frac{1}{1-x^{10}}$
- $f_{10}(x) = (1 + x + x^2 + x^3) = \frac{1-x^4}{1-x}$

$$f(x) = \prod_{i=1}^{10} f_i(x) = \frac{1}{(1-x)^5} = \sum_{n=0}^{\infty} \binom{n+4}{4} x^n$$

$$\text{ans}_n = \binom{n+4}{4} = \frac{(n+4)(n+3)(n+2)(n+1)}{24}$$

由于 $10^{\{100000\}} \leq n < 10^{\{100001\}}$ 需要用高精度，正解是用 NTT 用 Python 会 TLE 但是竟然可以用 Ruby AC

Ruby 代码：

```
n=Integer(gets)
puts (n+4)*(n+3)*(n+2)*(n+1)/24
```

C++ 代码：

```
#include<bits/stdc++.h>
using namespace std;
const int N=5e6+5,P=998244353,G=3,Gi=332748118;
typedef long long ll;
int rev[N];
char s[N];
ll n1[N],n2[N],n3[N],n4[N];
ll fastpow(ll x,ll y){
    ll ret=1;
    for(;y;y>=1,x=x*x%P)
```

```
        if(y&1) ret=ret*x%P;
    return ret;
}

void NTT(ll *y,int len,int on){
    for(int i=0;i<len;i++)
        if(i<rev[i])
            swap(y[i],y[rev[i]]);
    for(int mid=1;mid<len;mid<=<1){
        ll wn=fastpow(on==1?G:Gi,(P-1)/(mid<=<1));
        for(int j=0;j<len;j+=(mid<=<1)){
            ll w=1;
            for(int k=0;k<mid;k++,w=w*wn%P){
                ll u=y[j+k],t=w*y[j+k+mid]%P;
                y[j+k]=(u+t)%P;
                y[j+k+mid]=(u-t+P)%P;
            }
        }
    }
}

ll c[N];
void mul(ll *a,ll *b,int &n,int &m){
    for(int i=0;i<=n;i++) a[i]%P;
    for(int i=0;i<=m;i++) b[i]%P;
    int len=1,l=0;
    while(len<=n+m) len<=<1,l++;
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));
    NTT(a,len,1);
    NTT(b,len,1);
    for(int i=0;i<len;i++) a[i]=a[i]*b[i]%P;
    NTT(a,len,-1);
    ll inv=fastpow(len,P-2);
    memset(c,0,sizeof(c));
    for(int i=0;i<=n+m;i++){
        a[i]=a[i]*inv%P;
        c[i]+=a[i];
        c[i+1]+=c[i]/10;
        c[i]%10;
    }
    if(c[n+m+1]) n++;
    for(int i=0;i<=n+m;i++) a[i]=c[i];
    n=n+m;
}

ll ans[N];
int main(){
    scanf("%s",s);
    int n=strlen(s);
    n--;
    int len1=n,len2=n,len3=n,len4=n;
    for(int i=0;i<=n;i++) n1[i]=n2[i]=n3[i]=n4[i]=s[n-i]-'0';
    n1[0]+=1;
```

```

n2[0]+=2;
n3[0]+=3;
n4[0]+=4;
for(int i=0;i<=n;i++){
    n1[i+1]+=n1[i]/10;
    n1[i]%=10;
    n2[i+1]+=n2[i]/10;
    n2[i]%=10;
    n3[i+1]+=n3[i]/10;
    n3[i]%=10;
    n4[i+1]+=n4[i]/10;
    n4[i]%=10;
}
if(n1[n+1]) len1++;
if(n2[n+1]) len2++;
if(n3[n+1]) len3++;
if(n4[n+1]) len4++;
mul(n1,n2,len1,len2);
mul(n1,n3,len1,len3);
mul(n1,n4,len1,len4);

ll p=0;
int lenans=0;
for(int i=len1;i>=0;i--){
    p=p*10+n1[i];
    ans[i]=p/24;
    p%=24;
    if(ans[i]&&!lenans) lenans=i;
}
for(int i=lenans;i>=0;i--)
    printf("%lld",ans[i]);
return 0;
}

```

例 2 (Bell 数)

令 B_n 表示 $[n]$ (n 元集合) 所有划分的个数, 试找到计算 B_n 的公式。

讨论 $[n]$ 的划分中包含元素 n 的那个分块, 设其含有 k ($1 \leq k \leq n$) 个元素, 则剩余的 $k-1$ 个元素是从 $[n-1]$ 中选取的。而剩余的那些子集为 $n-k$ 个元素的一个划分, 所以

$B_n = \sum_{k=1}^n \binom{n-1}{k-1} B_{n-k}$, $n \geq 1$. 初始值为 $B_0=1, B_1=1$ 令

$\displaystyle B(x) = \sum_{n \geq 0} \frac{B_n}{n!} x^n$ 两边求导数, 得到

$$\begin{aligned} \frac{dB(x)}{dx} &= \sum_{n=1}^{\infty} \frac{B_n}{(n-1)!} x^{n-1} \\ &= \sum_{n=1}^{\infty} \frac{1}{(n-1)!} \left(\sum_{k=1}^n \binom{n-1}{k-1} B_{n-k} \right) x^{n-1} \\ &= \sum_{n=1}^{\infty} \sum_{k=1}^n \frac{x^{k-1}}{(k-1)!} \frac{B_{n-k}}{(n-k)!} x^{n-k} \\ &= \sum_{k=1}^{\infty} \frac{x^{k-1}}{(k-1)!} \sum_{n \geq k} \frac{B_{n-k}}{(n-k)!} x^{n-k} \\ &= \sum_{i \geq 0} \frac{B_i}{i!} x^i = e^{xB(x)}. \end{aligned}$$

解微分方程 $\frac{dB(x)}{dx} = e^{xB(x)}$ 可得 $B(x) = Ce^{e^x}$, 其中 C 为待定常数。由初始条件 $B(0)=1$ 得到 $C=e^{-1}$ 即 $B(x)=e^{e^x-1}$, 从而

$$B(x) = \frac{1}{e} \sum_{k=0}^{\infty} \frac{(e^x)^k}{k!} = \frac{1}{e} \sum_{k=0}^{\infty} \frac{e^k}{k!} \\
= \frac{1}{e} \sum_{n=0}^{\infty} \left(\sum_{k=0}^{\infty} \frac{k^n}{k!} \right) \frac{x^n}{n!}.$$

因此 $B_n = \frac{1}{e} \sum_{k=0}^{\infty} \frac{k^n}{k!}$.

例 3

题意

P4451 [国家集训队]整数的lqp拆分

题解

Fibonacci 数的生成函数是这个 (基础知识) $F(x) = \frac{x}{1-x-x^2}$ 当拆分为恰好 k 个数时, 答案的生成函数是 $F(x)^k$ 那么答案的生成函数即为 $\sum_{i=0}^{\infty} F(x)^i = \frac{1}{1-F(x)} = \frac{1-x-x^2}{1-2x-x^2}$ $= 1 + \frac{x}{1-2x-x^2} = 1 + \frac{\sqrt{2}}{4} \left(\frac{1}{1-(1+\sqrt{2})x} - \frac{1}{1-(1-\sqrt{2})x} \right) = 1 + \frac{\sqrt{2}}{4} \sum_{n=0}^{\infty} [(1+\sqrt{2})^n - (1-\sqrt{2})^n] x^n$ 所以通项公式为 $a_n \equiv \frac{\sqrt{2}}{4} [(1+\sqrt{2})^n - (1-\sqrt{2})^n] \pmod{1e9+7}$ 需要用二次剩余解出 $x^2 \equiv 2 \pmod{1e9+7}$

评价

对于理解生成函数和公式推导有一定的帮助

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod=1e9+7;
const ll two=59713600;
ll ksm(ll x,ll y){
    ll ret=1;
    for(;y;y>=1,x=x*x%mod)
        if(y&1) ret=ret*x%mod;
    return ret;
}
int main(){
    string s;
    cin>>s;
    ll n=0;
    int len=s.length();
    for(int i=0;i<len;i++){
        n=(n*10+s[i]-'0')%(mod-1);
    }
```

```

    }
    n=(n+(mod-1))%mod;
    n++;
    ll ans=(ksm(1+two,n)-ksm((1-two+mod)%mod,n)+mod)%mod*two/4%mod;
    printf("%lld",ans);
    return 0;
}
// sqrt(2):59713600

```

拓展

令 a_n 表示在一个 n -集合上完成某个任务的方法数 $a_0=0$ 对于 $n \geq 1$ 令 b_n 表示将集合 $[n]$ 划分成任意的连续、非空区间段，然后在这些非空区间段上完成前面任务的方法数 $b_0=1$ 设 $\{a_n\}_{n=0}^{\infty}$ 和 $\{b_n\}_{n=0}^{\infty}$ 的普通生成函数分别为 $f(x)$ 和 $g(x)$ 则对任意固定的正整数 i 将 $[n]$ 划分成 i 个非空子区间时，所对应的方法数的普通生成函数为 $f^i(x)$ 且有 $g(x)=\frac{1}{1-f(x)}$ 。

例 4

题意

P4841 [集训队作业2013]城市规划

求出 n 个点的简单（无重边无自环）有标号无向连通图数目。

题解

令 $\{a_n\}_{n=0}^{\infty}$ 为答案 $\{b_n\}_{n=0}^{\infty}$ 为 n 个点的无向图的数目，显然 $b_n=2^{\binom{n}{2}}$ 设 1 号点所在连通块大小为 k 则有
$$b_{n+1}=\sum_{k=1}^n \binom{n}{k-1} a_k b_{n-k} \\ =\sum_{k=0}^{n-1} \binom{n-1}{k} a_{k+1} b_{n-k-1} \\ b_{n+1}=\sum_{k=0}^n \binom{n}{k} a_{k+1} b_{n-k} \quad \text{令} \\ c_k=a_{k+1} \quad A(x), B(x), C(x) \text{ 分别为} \\ \{a_n\}_{n=0}^{\infty}, \{b_n\}_{n=0}^{\infty}, \{c_n\}_{n=0}^{\infty} \text{ 的指数型生成函数, 则} \\ b_{n+1}=\sum_{k=0}^n \binom{n}{k} c_k b_{n-k} \quad \text{从而} \quad B'(x)=C(x)B(x)=A'(x)B(x) \quad \text{解得} \\ A(x)=\ln B(x) \quad \text{所以套用多项式求对数的模板即可求解 } a_n$$

代码

```

#include<bits/stdc++.h>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
using namespace std;
const int MAXN=1.3e5+5,Mod=1004535809;
int n;

```

```
int a[1<<2],b[1<<2];

int quick_pow(int x,int y){
    int ret=1;
    for(;y;y>=1,x=1ll*x*x%Mod)
        if(y&1) ret=1ll*ret*x%Mod;
    return ret;
}

namespace Poly{
    const int G=3;
    int rev[1<<2],Pool[1<<3],*Wn[30];
    void init(){
        int lg2=0,*pos=Pool,n,w;
        while((1<<lg2)<1<<2)lg2++;
        n=1<<lg2,w=quick_pow(G,(Mod-1)/(1<<lg2));
        while(~lg2){
            Wn[lg2]=pos,pos+=n;
            Wn[lg2][0]=1;
            _for(i,1,n)Wn[lg2][i]=1LL*Wn[lg2][i-1]*w%Mod;
            w=1LL*w*w%Mod;
            lg2--;n>=1;
        }
    }

    int build(int k){
        int n,pos=0;
        while((1<<pos)<=k)pos++;
        n=1<<pos;
        _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        return n;
    }

    void NTT(int *f,int n,bool type){
        _for(i,0,n)if(i<rev[i])
            swap(f[i],f[rev[i]]);
        int t1,t2;
        for(int i=1,lg2=1;i<n;i<=1,lg2++){
            for(int j=0;j<n;j+=(i<<1)){
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*Wn[lg2][k-j]*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                }
            }
        }
        if(!type){
            reverse(f+1,f+n);
            int div=quick_pow(n,Mod-2);
            _for(i,0,n)f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
        }
    }

    void Mul(int *f,int _n,int *g,int _m,int xmod=0){
        int n=build(_n+_m-2);
```

```

    _for(i,_n,n)f[i]=0;_for(i,_m,n)g[i]=0;
    NTT(f,n,true);NTT(g,n,true);
    _for(i,0,n)f[i]=1LL*f[i]*g[i]%Mod;
    NTT(f,n,false);
    if(xmod)_for(i,xmod,n)f[i]=0;
}

void Inv(const int *f,int *g,int _n){
    static int temp[MAXN<<2];
    if(_n==1)return g[0]=quick_pow(f[0],Mod-2),void();
    Inv(f,g,(_n+1)>>1);
    int n=build((_n-1)<<1);
    _for(i,(_n+1)>>1,n)g[i]=0;
    _for(i,0,_n)temp[i]=f[i];_for(i,_n,n)temp[i]=0;
    NTT(g,n,true);NTT(temp,n,true);
    _for(i,0,n)g[i]=(2-1LL*temp[i]*g[i]%Mod)*g[i]%Mod;
    NTT(g,n,false);
    _for(i,_n,n)g[i]=0;
}

void Div(const int *f,int _n,const int *g,int _m,int *q,int *r){
    static int temp[MAXN<<2];
    if(_n<_m){
        _rep(i,0,n)r[i]=f[i];
        return;
    }
    _rep(i,0,_m)temp[i]=g[_m-i];
    Inv(temp,q,_n-_m+1);
    _rep(i,0,_n)temp[i]=f[_n-i];
    Mul(q,_n-_m+1,temp,_n+1,_n-_m+1);
    for(int i=0,j=_n-_m;i<j;i++,j--)swap(q[i],q[j]);
    int __m=min(_n-_m+1,_m);
    _for(i,0,__m)r[i]=g[i];_for(i,0,__m)temp[i]=q[i];
    Mul(r,__m,temp,__m,__m);
    _for(i,0,__m)r[i]=(f[i]+Mod-r[i])%Mod;
}

void Ln(const int *f,int *g,int _n){
    static int temp[MAXN<<2];
    Inv(f,g,_n);
    _for(i,1,_n)temp[i-1]=1LL*f[i]*i%Mod;
    temp[_n-1]=0;
    Mul(g,_n,temp,_n-1,_n);
    for(int i=_n-1;i;i--)g[i]=1LL*g[i-1]*quick_pow(i,Mod-2)%Mod;
    g[0]=0;
}

void Exp(const int *f,int *g,int _n){
    static int temp[MAXN<<2];
    if(_n==1)return g[0]=1,void();
    Exp(f,g,(_n+1)>>1);
    _for(i,(_n+1)>>1,_n)g[i]=0;
    Ln(g,temp,_n);
    temp[0]=(1+f[0]-temp[0])%Mod;
    _for(i,1,_n)temp[i]=(f[i]-temp[i])%Mod;
}

```

```

        Mul(g, (_n+1)>>1, temp, _n, _n);
    }
    void Pow(const int *f, int *g, int _n, int k1, int k2){
        static int temp[MAXN<<2];
        int pos=0, posv;
        while(!f[pos]&&pos<_n)pos++;
        if(1LL*pos*k2>=_n){
            _for(i, 0, _n)g[i]=0;
            return;
        }
        posv=quick_pow(f[pos], Mod-2);
        _for(i, pos, _n)g[i-pos]=1LL*f[i]*posv%Mod;
        _for(i, _n-pos, _n)g[i]=0;
        Ln(g, temp, _n);
        _for(i, 0, _n)temp[i]=1LL*temp[i]*k1%Mod;
        Exp(temp, g, _n);
        pos=pos*k2, posv=quick_pow(posv, 1LL*k2*(Mod-2)*(Mod-1));
        for(int i=_n-1; i>=pos; i--)g[i]=1LL*g[i-pos]*posv%Mod;
        _for(i, 0, pos)g[i]=0;
    }
}

int fact[MAXN<<2];
int invfact[MAXN<<2];
int main(){
    scanf("%d", &n);
    b[0]=1;
    fact[0]=1;
    for(int i=1; i<=n; i++){
        fact[i]=1ll*fact[i-1]*i%Mod;
    }
    invfact[n]=quick_pow(fact[n], Mod-2);
    for(int i=n; i>=1; i--)
        invfact[i-1]=1ll*invfact[i]*i%Mod;
    for(int i=0; i<=n; i++)
        b[i]=1ll*quick_pow(2, 1ll*i*(i-1)/2*(Mod-1))*invfact[i]%Mod;
    Poly::init();
    Poly::Ln(b, a, n+1);
    printf("%d", 1ll*a[n]*fact[n]%Mod);
    return 0;
}

```

拓展

令 a_n 表示在一个 n -集合上完成某个任务的方法数 $a_0=0$ 对于 $n \geq 1$ 令 b_n 表示将集合 $[n]$ 划分成任意的非空子集，然后在这些非空子集上完成前面任务的方法数 $b_0=1$ 设 $\{a_n\}_{n=0}^{\infty}$ 和 $\{b_n\}_{n=0}^{\infty}$ 的指数型生成函数分别为 $f(x)$ 和 $g(x)$ 则对任意固定的正整数 i 将 $[n]$ 划分成 i 个非空子集时，所对应的方法数的指数型生成函数为 $f^i(x)/i!$ 且有 $g(x)=e^{f(x)}$ 。

例 5

题意

CF438E The Child and Binary Tree

给定一个含有 n 个互异正整数的数列 $c_1, c_2, \dots, c_n$ 。如果一棵带点权的有根二叉树满足其所有顶点的权值都在集合 $\{c_1, c_2, \dots, c_n\}$ 中，则称之为好树，一棵带点权的树的权值是其所有顶点权值的总和。给出一个正整数 m 对任意的 $s, 1 \leq s \leq m$ 计算出权值为 s 的好树个数。

题解

设 f_i 表示点权和为 i 的符合条件的二叉树的个数， g_i 表示权值 i 是否在 c_i 中。 $f_0=1, f_n=\sum_{i=1}^n g_i \sum_{j=1}^{n-i} f_j f_{n-i-j}$ 这个式子的意思是枚举根结点的权值 i ，然后枚举根结点左右子树的权值，容易发现这是三个多项式的卷积。用 $F(x), G(x)$ 分别表示数列 $\{f_n\}_{n=0}^{\infty}, \{g_n\}_{n=0}^{\infty}$ 的普通生成函数。则有 $F(x)=G(x)F^2(x)+1$ 解得 $F(x)=\frac{2}{1+\sqrt{1-4G(x)}}$ 所以套用多项式开根+求逆即可

代码

```
#include<bits/stdc++.h>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
using namespace std;
const int MAXN=1e5+5,Mod=998244353;
int n;
int g[MAXN<<2],f[MAXN<<2];
int h[MAXN<<2];
int quick_pow(int x,int y){
    int ret=1;
    for(;y;y>>=1,x=1ll*x*x%Mod)
        if(y&1) ret=1ll*ret*x%Mod;
    return ret;
}
unordered_map<int,int>H;
int bsgs(int a,int b){
    H.clear();
    int m=sqrt(Mod)+1,t=b,base;
    for(int i=1;i<=m;i++){
        t=1ll*t*a%Mod;
        H[t]=i;
    }
    t=1,base=quick_pow(a,m);
    for(int i=1;i<=m;i++){
        t=1ll*t*base%Mod;
        if(H[t]) return m*i-H[t];
    }
}
```

```
    return -1;
}

namespace Poly{
    const int G=3;
    int rev[MAXN<<2],Pool[MAXN<<3],*Wn[30];
    void init(){
        int lg2=0,*pos=Pool,n,w;
        while((1<<lg2)<MAXN*2)lg2++;
        n=1<<lg2,w=quick_pow(G,(Mod-1)/(1<<lg2));
        while(~lg2){
            Wn[lg2]=pos,pos+=n;
            Wn[lg2][0]=1;
            _for(i,1,n)Wn[lg2][i]=1LL*Wn[lg2][i-1]*w%Mod;
            w=1LL*w*w%Mod;
            lg2--;n>=>1;
        }
    }

    int build(int k){
        int n,pos=0;
        while((1<<pos)<=k)pos++;
        n=1<<pos;
        _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        return n;
    }

    void NTT(int *f,int n,bool type){
        _for(i,0,n)if(i<rev[i])
            swap(f[i],f[rev[i]]);
        int t1,t2;
        for(int i=1,lg2=1;i<n;i<=1,lg2++){
            for(int j=0;j<n;j+=(i<<1)){
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*Wn[lg2][k-j]*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                }
            }
        }
        if(!type){
            reverse(f+1,f+n);
            int div=quick_pow(n,Mod-2);
            _for(i,0,n)f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
        }
    }

    void Mul(int *f,int _n,int *g,int _m,int xmod=0){
        int n=build(_n+_m-2);
        _for(i,_n,n)f[i]=0;_for(i,_m,n)g[i]=0;
        NTT(f,n,true);NTT(g,n,true);
        _for(i,0,n)f[i]=1LL*f[i]*g[i]%Mod;
        NTT(f,n,false);
        if(xmod)_for(i,xmod,n)f[i]=0;
    }
}
```

```

void Inv(const int *f,int *g,int _n){
    static int temp[MAXN<<2];
    if(_n==1)return g[0]=quick_pow(f[0],Mod-2),void();
    Inv(f,g,(_n+1)>>1);
    int n=build((_n-1)<<1);
    _for(i,(_n+1)>>1,n)g[i]=0;
    _for(i,0,_n)temp[i]=f[i];_for(i,_n,n)temp[i]=0;
    NTT(g,n,true);NTT(temp,n,true);
    _for(i,0,n)g[i]=(2-1LL*temp[i]*g[i]%Mod)*g[i]%Mod;
    NTT(g,n,false);
    _for(i,_n,n)g[i]=0;
}

void Div(const int *f,int _n,const int *g,int _m,int *q,int *r){
    static int temp[MAXN<<2];
    if(_n<_m){
        _rep(i,0,n)r[i]=f[i];
        return;
    }
    _rep(i,0,_m)temp[i]=g[_m-i];
    Inv(temp,q,_n-_m+1);
    _rep(i,0,_n)temp[i]=f[_n-i];
    Mul(q,_n-_m+1,temp,_n+1,_n-_m+1);
    for(int i=0,j=_n-_m;i<j;i++,j--)swap(q[i],q[j]);
    int __m=min(_n-_m+1,_m);
    _for(i,0,__m)r[i]=g[i];_for(i,0,__m)temp[i]=q[i];
    Mul(r,_m,temp,__m,__m);
    _for(i,0,__m)r[i]=(f[i]+Mod-r[i])%Mod;
}

void Ln(const int *f,int *g,int _n){
    static int temp[MAXN<<2];
    Inv(f,g,_n);
    _for(i,1,_n)temp[i-1]=1LL*f[i]*i%Mod;
    temp[_n-1]=0;
    Mul(g,_n,temp,_n-1,_n);
    for(int i=_n-1;i;i--)g[i]=1LL*g[i-1]*quick_pow(i,Mod-2)%Mod;
    g[0]=0;
}

void Exp(const int *f,int *g,int _n){
    static int temp[MAXN<<2];
    if(_n==1)return g[0]=1,void();
    Exp(f,g,(_n+1)>>1);
    _for(i,(_n+1)>>1,_n)g[i]=0;
    Ln(g,temp,_n);
    temp[0]=(1+f[0]-temp[0])%Mod;
    _for(i,1,_n)temp[i]=(f[i]-temp[i])%Mod;
    Mul(g,(_n+1)>>1,temp,_n,_n);
}

void Pow(const int *f,int *g,int _n,int k1,int k2){
    static int temp[MAXN<<2];
    int pos=0,posv;
    while(!f[pos]&&pos<_n)pos++;

```

```
    if(1LL*pos*k2>=_n){
        _for(i,0,_n)g[i]=0;
        return;
    }
    posv=quick_pow(f[pos],Mod-2);
    _for(i,pos,_n)g[i-pos]=1LL*f[i]*posv%Mod;
    _for(i,_n-pos,_n)g[i]=0;
    Ln(g,temp,_n);
    _for(i,0,_n)temp[i]=1LL*temp[i]*k1%Mod;
    Exp(temp,g,_n);
    pos=pos*k2,posv=quick_pow(posv,1LL*k2*(Mod-2)*(Mod-1));
    for(int i=_n-1;i>=pos;i--)g[i]=1LL*g[i-pos]*posv%Mod;
    _for(i,0,pos)g[i]=0;
}

void Sqrt(const int *f,int *g,int _n){
    static int temp1[MAXN<<2],temp2[MAXN<<2];
    if(_n==1) return g[0]=quick_pow(G,bsgs(3,f[0])/2),void();
    Sqrt(f,g,(_n+1)>>1);
    _for(i,(_n+1)>>1,_n) g[i]=0;
    _for(i,0,_n) temp1[i]=f[i];
    Inv(g,temp2,_n);
    Mul(temp1,_n,temp2,_n);
    int div2=quick_pow(2,Mod-2);
    _for(i,0,_n) g[i]=1LL*(g[i]+temp1[i])*div2%Mod;
}

}

int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++){
        int x;
        scanf("%d",&x);
        g[x]=(Mod-4)%Mod;
    }
    g[0]=1;
    Poly::init();
    Poly::Sqrt(g,h,m+1);
    h[0]=(h[0]+1)%Mod;
    Poly::Inv(h,f,m+1);
    for(int i=1;i<=m;i++) printf("%d\n",1ll*f[i]*2%Mod);
    return 0;
}
```

