指数生成函数(EGF)

算法简介

形如 $F(x)=\sum_{n=0}^{\infty}a_n\frac {x^n}{n!}$ 的函数 a_n 可以提供关于这个序列的信息，一般用于解决有标号组合计数问题。

基本运算

$$F(x)=\sum_{n=0}^{\infty}a_n\frac {x^n}{n!},G(x)=\sum_{n=0}^{\infty}b_n\frac {x^n}{n!}$$

$$F(x)G(x)=\sum_{n=0}^{\infty}\sum_{i=0}^n a_ib_{n-i}\frac {x^n}{i!(n-i)!}=\sum_{n=0}^{\infty}\sum_{i=0}^n {n\choose i}a_ib_{n-i}\frac {x^n}{n!}$$

$$\sum_{n=0}^{\infty}k^n\frac {x^n}{n!}=e^{kx},\sum_{n=0}^{\infty}n^{\underline k}\frac {x^n}{n!}=x^ke^x$$

算法例题

例题一

洛谷p5748

题意

定义贝尔数 w_n 表示将集合 $\{1,2\cdots n\}$ 划分为若干个不相交非空集合的方案数，求 w_n

题解

假设将 n 化分到一个大小为 i 的集合，不难得出递推式

$$w_n=[n==0]+\sum_{i=1}^{n} {n-1\choose i-1}w_{n-i}$$

构造 $F(x)=\sum_{n=0}^{\infty}w_n\frac {x^n}{n!},G(x)=\sum_{n=0}^{\infty}\frac {x^n}{n!}=e^x$ 于是有

$$F(x)=1+\int F(x)G(x)\mathrm{d}x=1+\int F(x)e^x\mathrm{d}x$$

接下来考虑求解 $F(x)$

$$\mathrm{d}F(x)=F(x)e^x\mathrm{d}x$$

$$\frac {\mathrm{d}F(x)}{F(x)}=e^x\mathrm{d}x$$

$$\ln F(x)=e^x+C$$

将 $F(0)=1$ 代入，得 $F(x)=\exp (e^x-1)$于是 $w_n=[\frac {x^n}{n!}]F(x)$

```
const int MAXN=1e5+5,Mod=998244353;
int quick_pow(int a,int b){
    int ans=1;
    while(b){
        if(b&1)
        ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        b>>=1;
    }
    return ans;
}
namespace Poly{
    const int G=3;
    int rev[MAXN<<2],Wn[30][2];
    void init(){
        int m=Mod-1,lg2=0;
        while(m%2==0)m>>=1,lg2++;
        Wn[lg2][1]=quick_pow(G,m);
        Wn[lg2][0]=quick_pow(Wn[lg2][1],Mod-2);
        while(lg2){
            m<<=1,lg2--;
            Wn[lg2][0]=1LL*Wn[lg2+1][0]*Wn[lg2+1][0]%Mod;
            Wn[lg2][1]=1LL*Wn[lg2+1][1]*Wn[lg2+1][1]%Mod;
        }
    }
    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=0;i<n;i<<=1,lg2++){
            int w=Wn[lg2+1][type];
            for(int j=0;j<n;j+=(i<<1)){
                int cur=1;
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*cur*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                    cur=1LL*cur*w%Mod;
                }
```

```
            }
        }
        if(!type){
            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 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);
    }
}
int a[MAXN<<2],b[MAXN<<2],frac[MAXN];
int main()
{
    Poly::init();
```

```
    frac[0]=1;
    _for(i,1,MAXN)frac[i]=1LL*frac[i-1]*i%Mod;
    a[1]=1;
    Poly::Exp(a,b,1e5+1);
    b[0]--;
    Poly::Exp(b,a,1e5+1);
    int T=read_int();
    while(T--){
        int n=read_int();
        enter(1LL*a[n]*frac[n]%Mod);
    }
    return 0;
}
```

拓展

洛谷p4841

考虑对结果的解释$e^x-1=\sum_{n=1}^{\infty}a_n\frac {x^n}{n!}(a_n=1)$ 可以理解为将所有 n 个元素化为为一个集合的方案数 a_n 的 EGF

$\exp (e^x-1)=\sum_{i=0}^{\infty} \cfrac {A^i(x)}{i!}$ 式子中 $\sum_{i=0}^{\infty}$ 可以理解为枚举最终划分的集合数 i

$A^i(x)$ 可以理解为将所有元素划分为 i 个非空集合$i!$ 可以理解为划分的集合之间无序所以除以全排列数。

类似的，设 n 个点带标号生成树的 EGF 为 $F(x)$则 n 个点带标号生成森林的 EGF 为 $\exp F(x)$

其中 n 个点带标号生成树的 EGF 容易求得为 $\sum_{n=0}^{\infty} n^{n-2}\frac {x^n}{n!}$所以取 $\exp$ 即可快速求得 n 个点带标号生成森林的 EGF

设 n 个点带标号无向连通图的 EGF 为 $F(x)$则 n 个点带标号无向图的 EGF 为 $\exp F(x)$

其中 n 个点带标号无向图的 EGF 容易求得为 $\sum_{n=0}^{\infty} 2^{n(n-1)/2}\frac {x^n}{n!}$所以取 $\ln$ 即可快速求得 n 个点带标号无向连通图的 EGF

```
const int MAXN=13e4+5,Mod=1004535809;
int quick_pow(int a,int b){
    int ans=1;
    while(b){
        if(b&1)
        ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        b>>=1;
    }
    return ans;
```

```
}
namespace Poly{
    const int G=3;
    int rev[MAXN<<2],Wn[30][2];
    void init(){
        int m=Mod-1,lg2=0;
        while(m%2==0)m>>=1,lg2++;
        Wn[lg2][1]=quick_pow(G,m);
        Wn[lg2][0]=quick_pow(Wn[lg2][1],Mod-2);
        while(lg2){
            m<<=1,lg2--;
            Wn[lg2][0]=1LL*Wn[lg2+1][0]*Wn[lg2+1][0]%Mod;
            Wn[lg2][1]=1LL*Wn[lg2+1][1]*Wn[lg2+1][1]%Mod;
        }
    }
    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=0;i<n;i<<=1,lg2++){
            int w=Wn[lg2+1][type];
            for(int j=0;j<n;j+=(i<<1)){
                int cur=1;
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*cur*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                    cur=1LL*cur*w%Mod;
                }
            }
        }
        if(!type){
            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 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;
    }
}
int a[MAXN<<2],b[MAXN<<2],frac[MAXN],invfrac[MAXN];
int main()
{
    Poly::init();
    int n=read_int();
    frac[0]=a[0]=1;
    _rep(i,1,n){
        frac[i]=1LL*frac[i-1]*i%Mod;
        a[i]=quick_pow(2,1LL*i*(i-1)/2%(Mod-1));
    }
    invfrac[n]=quick_pow(frac[n],Mod-2);
    for(int i=n;i;i--)
    invfrac[i-1]=1LL*invfrac[i]*i%Mod;
    _rep(i,0,n)a[i]=1LL*a[i]*invfrac[i]%Mod;
    Poly::Ln(a,b,n+1);
    enter(1LL*b[n]*frac[n]%Mod);
    return 0;
}
```

例题二

题意

定义错排数 a_i 等于长度为 n 且 $p_i\neq i$ 的置换个数，求错排数 a_n

题解

$p_i\neq i$ 等价于不含长度为 1 的置换环，考虑所有元素处于同一个置换环时的 EGF。

$$\sum_{n=2}^{\infty}(n-1)!\frac {x^n}{n!}=\sum_{n=2}^{\infty}\frac {x^n}n=-\ln (1-x)-x$$

错排等价于将置换划分为若干个长度不为 1 的置换环，于是错排数的 EGF 即为 $\exp (-\ln (1-x)-x)$。

例题三

题意

求满足 $\underbrace{f\circ f\circ\cdots\circ f}_{k}=\underbrace{f\circ f\circ\cdots\circ f}_{k-1}$ 的长度为 n 的置换个数。

题解

建图，连有向边 $i\to f(i)$。由于 $\underbrace{f\circ f\circ\cdots\circ f}_{k}=\underbrace{f\circ f\circ\cdots\circ f}_{k-1}$。该图显然存在自环，且所有点 $k-1$ 步内一定到达自环点。

而该图只有 n 条边，于是该图显然只存在一个环，于是只存在一个自环点，设其为根节点，于是忽略自环则该图恰好为一个内向树。

于是问题转化为求 n 个点带标号的深度不超过 k (假设根节点深度为 1)的生成树的 EGF。设其为 $F_k(x)$。

显然这等价于选择一个点作为根节点然后取 $n-1$ 个点带标号的深度不超过 $k-1$ 的生成森林向其连边。

于是有 $[\frac {x^n}{n!}]F_k(x)=n[\frac {x^{n-1}}{(n-1)!}]\exp F_{k-1}(x)=[\frac {x^n}{n!}]\exp xF_{k-1}(x)$。

即 $F_k(x)=\exp xF_{k-1}(x)$。边界条件 $F_1(x)=x$。时间复杂度 $O(nk\log n)$。

例题四

CF891E

题意

给定 n 个数 $a_1,a_2\cdots a_n$。

接下来 k 次操作。每次随机选取其中一个数 $x\in [1,n]$。将 a_x 减一，同时得到 $\prod_{i\neq x}a_i$ 的分数。

问最终得分的期望值，答案对 10^9+7 取模。

题解

设势能函数为 $\prod_{i=1}^n a_i$，发现每次操作得分恰好等于势能函数减少量。

不妨设 c_i 表示 k 次操作中 $x=i$ 的次数，则最终得分为 $\prod_{i=1}^n a_i-\prod_{i=1}^n (a_i-c_i)$。

考虑计算每个方案的 $\prod_{i=1}^n (a_i-c_i)$ 的和，最后除以总方案数 n^k。

对每种方案 $(c_1,c_2\cdots c_n)$，显然有 $\frac {k!}{c_1!c_2!\cdots c_n!}$ 个，发现这与 EGF 卷积相似。

构造生成函数 $F_i(x)=\sum_{j=0}^{\infty} (a_i-j)\frac {x^j}{j!}=\sum_{j=0}^{\infty} a_i\frac {x^j}{j!}-j\frac {x^j}{j!}=(a_i-x)e^x$，$F(x)=\prod_{i=1}^n F_i(x)=e^{nx}\prod_{i=1}^n (a_i-x)$。

不难发现所有方案对应答案为 $[\frac {x^k}{k!}]F(x)$。

考虑分治 FFT 处理 $\prod_{i=1}^n (a_i-x)$，假设计算结果为 $b_0+b_1x+b_2x^2+\cdots +b_nx^n$。

$$F(x)=\left(\sum_{i=0}^{\infty}\frac {n^ix^i}{i!}\right)\left(\sum_{i=0}^n b_ix^i\right)=\sum_{i=0}^{\infty} x^i\sum_{j=0}^i \frac{n^{i-j}b_j}{(i-j)!}=\sum_{i=0}^{\infty} \frac{x^i}{i!}\sum_{j=0}^{\min (i,n)} n^{i-j}i^{\underline j}b_j$$

于是 $[\frac {x^k}{k!}]F(x)=n^k\sum_{i=0}^{\min (k,n)}\cfrac {k^{\underline i}b_i}{n^i}$。

```
const int MAXN=1e5+5,mod=1e9+7,m=sqrt(mod)+0.5;
const long double pi=acos(-1.0);
int quick_pow(int x,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*x%mod;
        x=1LL*x*x%mod;
        k>>=1;
    }
    return ans;
}
struct complex{
    long double x,y;
    complex(long double x=0.0,long double y=0.0):x(x),y(y){}
    complex operator + (const complex &b){
        return complex(x+b.x,y+b.y);
    }
    complex operator - (const complex &b){
        return complex(x-b.x,y-b.y);
    }
    complex operator * (const complex &b){
        return complex(x*b.x-y*b.y,x*b.y+y*b.x);
    }
};
```

```
int rev[MAXN<<2];
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 FFT(complex *f,int n,int type){
    _for(i,0,n)if(i<rev[i])
    swap(f[i],f[rev[i]]);
    complex t1,t2;
    for(int i=1;i<n;i<<=1){
        complex w(cos(pi/i),type*sin(pi/i));
        for(int j=0;j<n;j+=(i<<1)){
            complex cur(1.0,0.0);
            _for(k,j,j+i){
                t1=f[k],t2=cur*f[k+i];
                f[k]=t1+t2,f[k+i]=t1-t2;
                cur=cur*w;
            }
        }
    }
    if(type==-1)_for(i,0,n)
    f[i].x/=n,f[i].y/=n;
}
void FFT2(complex *f1,complex *f2,int n){
    FFT(f1,n,1);
    f2[0].x=f1[0].x,f2[0].y=-f1[0].y;
    _for(i,1,n)
    f2[i].x=f1[n-i].x,f2[i].y=-f1[n-i].y;
    complex t1,t2;
    _for(i,0,n){
        t1=f1[i],t2=f2[i];
        f1[i]=complex((t1.x+t2.x)*0.5,(t1.y+t2.y)*0.5);
        f2[i]=complex((t1.y-t2.y)*0.5,(t2.x-t1.x)*0.5);
    }
}
LL getv(double x){
    if(x>-0.5)return (LL)(x+0.5);
    return (LL)(x-0.5);
}
complex f1[MAXN<<2],f2[MAXN<<2],g1[MAXN<<2],g2[MAXN<<2],temp[2][MAXN<<2];
void MTT(int *f,int n1,int *g,int n2){
    int n=build(n1+n2);
_rep(i,0,n1)f1[i].x=f[i]/m,f1[i].y=f[i]%m;_for(i,n1+1,n)f1[i].x=f1[i].y=0.0
;
_rep(i,0,n2)g1[i].x=g[i]/m,g1[i].y=g[i]%m;_for(i,n2+1,n)g1[i].x=g1[i].y=0.0
;
    FFT2(f1,f2,n);FFT2(g1,g2,n);
    complex I(0.0,1.0);
```

```
    _for(i,0,n){
        temp[0][i]=f1[i]*g1[i]+I*f2[i]*g1[i];
        temp[1][i]=f1[i]*g2[i]+I*f2[i]*g2[i];
    }
    FFT(temp[0],n,-1);FFT(temp[1],n,-1);
    LL a,b,c;
    _rep(i,0,n1+n2){
a=getv(temp[0][i].x);b=getv(temp[0][i].y+temp[1][i].x);c=getv(temp[1][i].y)
;
        f[i]=((a%mod*m%mod*m%mod+b%mod*m%mod+c%mod)%mod+mod)%mod;
    }
}
int a[MAXN],b[MAXN<<3],c[MAXN];
void solve(int *f,int lef,int rig){
    int mid=lef+rig>>1;
    if(lef==rig){
        f[0]=a[mid],f[1]=-1;
        return;
    }
    solve(f,lef,mid);solve(f+mid-lef+2,mid+1,rig);
    MTT(f,mid-lef+1,f+mid-lef+2,rig-mid);
}
int main()
{
    int n=read_int(),k=read_int(),ans=1;
    _rep(i,1,n)a[i]=read_int(),ans=1LL*ans*a[i]%mod;
    solve(b,1,n);
    int _n=min(n,k),divn=quick_pow(n,mod-2);
    c[0]=1;
    _for(i,0,_n)c[i+1]=1LL*c[i]*(k-i)%mod*divn%mod;
    _rep(i,0,_n)ans=(ans-1LL*b[i]*c[i])%mod;
    enter((ans+mod)%mod);
    return 0;
}
```