

普通生成函数(OGF)

算法简介

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

算法例题

例题一

[洛谷p2000](#)

题意

已知如下约束条件：

- $6 \mid a$
- $b \leq 9$
- $c \leq 5$
- $4 \mid d$
- $e \leq 7$
- $2 \mid f$
- $0 \leq g \leq 1$
- $8 \mid h$
- $10 \mid i$
- $j \leq 3$
- $a+b+c+d+e+f+g+h+i+j=n$
- 所有数非负整数

问满足以上约束的所有情况数。

题解

根据上述条件，可以得到如下生成函数：

- $1+x^6+x^{12}+\dots=\frac{1}{1-x^6}$
- $1+x+x^2+\dots+x^9=\frac{1-x^{10}}{1-x}$
- $1+x+x^2+\dots+x^5=\frac{1-x^6}{1-x}$
- $1+x^4+x^8+\dots=\frac{1}{1-x^4}$
- $1+x+x^2+\dots+x^7=\frac{1-x^8}{1-x}$
- $1+x^2+x^4+\dots=\frac{1}{1-x^2}$
- $1+x=\frac{1-x^2}{1-x}$
- $1+x^8+x^{16}+\dots=\frac{1}{1-x^8}$
- $1+x^{10}+x^{20}+\dots=\frac{1}{1-x^{10}}$

$$\bullet \quad 1+x+x^2+x^3=\frac{1-x^4}{1-x}$$

全部相乘得到 $\frac{1}{(1-x)^5}=\sum_{n=0}^{\infty} \binom{n+4}{n}x^n=\sum_{n=0}^{\infty} \binom{n+4}{4}x^n$

于是答案为 $\binom{n+4}{4}$

例题二

题意

已知卡特兰数定义

$$c_n=\begin{cases} 1, & n=0 \\ \sum_{i=0}^{n-1}c_i c_{n-i-1}, & n>0 \end{cases}$$

求 c_n 通项公式。

题解

解得 $F(x)=\frac{1-\sqrt{1-4x}}{2x}$ 又有 $F(0)=c_0=1$ 于是发现只有 $F(x)=\frac{1-\sqrt{1-4x}}{2x}$ 一种可能。

所以有
$$c_n=\frac{(2n)!}{n!(n+1)!}=\frac{2n}{n+1}\binom{2n-1}{n}$$

例题三

题意

求满足下列条件的序列数：

1. $a_1+a_2+\dots+a_k=n$
2. $a_1\geq a_2\geq\dots\geq a_k\geq 1$

题解

先考虑 $a_1 \le t$ 的情况。

只选择 1 的方案生成函数为 $1+x+x^2+\dots=\frac{1}{1-x}$

只选择 2 的方案生成函数为 $1+x^2+x^4+\dots=\frac{1}{1-x^2}$

以此类推，只选择 t 的方案生成函数为 $1+x^t+x^{2t}+\dots=\frac{1}{1-x^t}$

于是 $a_1 \le t$ 的生成函数为 $P_t(x)=\prod_{i=1}^t \frac{1}{1-x^i}$ 。无限制时生成函数为 $P(x)=\prod_{i=1}^{\infty} \frac{1}{1-x^i}$

答案即为 $[x^n]P(x)$ 。接下来考虑如何展开 $P(x)$ 。

这里给出五边形数定理和证明。

$$\prod_{i=1}^{\infty} (1-x^i) = 1 + \sum_{k=1}^{\infty} (-1)^k x^{k(3k-1)/2} = 1 - x - x^2 + x^5 - x^7 + x^{12} - x^{15} + \dots$$

于是有 $P(x) \prod_{i=1}^{\infty} (1-x^i) = 1$ 。展开

$$\begin{matrix} P(x) & p_0 & p_1x & p_2x^2 & p_3x^3 & p_4x^4 & p_5x^5 & \dots \\ -xP(x) & & -p_0x & -p_1x^2 & -p_2x^3 & -p_3x^4 & -p_4x^5 & \dots \\ x^2P(x) & & & p_0x^2 & p_1x^3 & p_2x^4 & p_3x^5 & \dots \\ \dots & & & & & & & \dots \\ \prod_{i=1}^{\infty} (1-x^i)P(x) & 1 & 0x & 0x^2 & 0x^3 & 0x^4 & 0x^5 & \dots \end{matrix}$$

于是 $p_n = [n=0] + p_{n-1} + p_{n-2} - p_{n-5} - p_{n-7} + \dots$

例题四

[洛谷p4389](#)

题意

给定 n 种物品，每种物品体积为 v_i ，有无限个。问物品恰好装满体积为 $i(1 \le i \le m)$ 的背包的方案数。

题解

首先只选每个物品 i 的生成函数为 $1+x^{v_i}+x^{2v_i}+\dots=\frac{1}{1-x^{v_i}}$

于是最终方案的生成函数为 $F(x)=\prod_{i=1}^n \frac{1}{1-x^{v_i}}=\prod_{i=1}^n \sum_{j=1}^{\infty} x^{jv_i}$

记体积为 i 的物品有 c_i 个，同时考虑取 \ln 加速乘法，有 $F(x)=\exp(\ln F(x))=\exp(\sum_{i=1}^n \sum_{j=1}^{\infty} \frac{x^{jv_i}}{j})=\exp(\sum_{i=1}^n \sum_{j=1}^{\infty} \frac{x^{ij}}{j})$

由于只需要计算 $\sum_{i=1}^n x^i F(x) \pmod m$ 于是只需要计算出 $\sum_{j=1}^n x^j \pmod m$ 即可。

时间复杂度 $O(m \log m)$

```
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],c[MAXN],inv[MAXN];
void get_inv(){
    inv[1]=1;

```

```
_for(i, 2, MAXN)
inv[i]=1LL*(Mod-Mod/i)*inv[Mod%i]%Mod;
}
int main()
{
    Poly::init();
    get_inv();
    int n=read_int(),m=read_int();
    _for(i,0,n)c[read_int()]++;
    _rep(i,1,m){
        for(int j=1;i*j<=m;j++)
            a[i*j]=(a[i*j]+1LL*c[i]*inv[j])%Mod;
    }
    Poly::Exp(a,b,m+1);
    _rep(i,1,m)enter(b[i]);
    return 0;
}
```

例题五

[洛谷p4451](#)

题意

求 $\sum \prod_{i=1}^m F_{a_i}$ 其中

1. $m > 0$
2. $a_i > 0$
3. $\sum_{i=1}^m a_i = n$
4. F 为斐波那契数列，且 $F_0=0, F_1=1$

答案对 10^9+7 取模。

题解

先考虑 $m=1$ 的生成函数，有 $G(x) = \sum_{i=0}^{\infty} F_{ix}^i$

于是答案的生成函数为 $H(x) = \sum_{m=1}^{\infty} G^m(x) = \frac{G(x)}{1-G(x)}$ 接下来考虑求解 $G(x)$

有 $G(x) - xG(x) - x^2G(x) = F_0 + F_1x - F_0 = x$ 于是 $G(x) = \frac{x}{1-x-x^2}$ 代入可得 $H(x) = \frac{x}{1-2x-x^2}$

于是有 $(1-2x-x^2)H(x) = x$ 于是 $h_n = 2h_{n-1} + h_{n-2} + [n=1]$

通过特征根求解得 $h_n = \frac{\sqrt{2}}{4}(1+\sqrt{2})^n - \frac{\sqrt{2}}{4}(1-\sqrt{2})^n$

```

const int MAXL=1e4+5,Mod=1e9+7,sqrt2=59713600,inv4=250000002;
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;
}
char buf[MAXL];
int main()
{
    scanf("%s",buf);
    int n=0,len=strlen(buf);
    _for(i,0,len)
        n=(10LL*n+buf[i]-'0')%(Mod-1);
    int ans=1LL*(quick_pow(1+sqrt2,n)-quick_pow(1-
sqrt2,n))*sqrt2%Mod*inv4%Mod;
    enter((ans+Mod)%Mod);
    return 0;
}

```

例题六

牛客暑期多校(第五场) C 题

题意

已知 $\sum_{i=1}^k a_i = n, \sum_{i=1}^k b_i = m, P = \prod_{i=1}^k \min(a_i, b_i)$ 求 $\sum_{a,b} P$

题解

令 $F(x) = \sum_{i=1, j=1}^{\infty} \min(a_i, b_j) x^i y^j$ 于是答案为 $[x^n y^m] F^k(x)$

考虑求出 $F(x)$ 的封闭式。

$$\begin{aligned} F(x) &= xy + xy^2 + xy^3 + xy^4 + \dots \\ &+ 2x^2y^2 + 2x^2y^3 + 2x^2y^4 + \dots \\ &+ x^3y + 2x^3y^2 + 3x^3y^3 \\ &+ 3x^3y^4 + \dots \end{aligned}$$

先想办法将系数化为 1，考虑相邻行之间错位相减，有

$$\begin{aligned} (1-x)F(x) &= xy + xy^2 + xy^3 + xy^4 + \dots \\ &- x^2y^2 - x^2y^3 - x^2y^4 - \dots \\ &+ x^3y + x^3y^2 + x^3y^3 + x^3y^4 + \dots \end{aligned}$$

然后发现每行均成为等比数列，直接求和得 $(1-x)F(x) = \frac{xy}{1-y} + \frac{x^2y^2}{1-y} + \frac{x^3y^3}{1-y} + \dots$ 发现结果仍然为等比数列，继续求和得 $(1-x)F(x) = \frac{xy}{(1-x)(1-xy)}$ 于是有
$$F^k(x) = \frac{x^k y^k}{(1-x)^k (1-xy)^k} = \sum_{a=0, b=0, c=0}^{\infty} \binom{k+a-1}{a} x^a \binom{k+b-1}{b} x^b$$

$y^b \{k+c-1 \choose c\} x^c y^c \end{split} \end{equation}$ 考虑枚举 c 的同时计算出 a, b 即可, 于是 $[x^n y^m] F^k(x) = \sum_{i=0}^{\min(n,m)-k} \{n-i-1 \choose n-k-i\} \{m-i-1 \choose m-k-i\} \{k+i-1 \choose i\}$

预处理阶乘和阶乘的逆即可, 时间复杂度 $O(n)$

```
const int MAXN=1e6+5,Mod=998244353;
int frac[MAXN],invfrac[MAXN];
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;
}
int C(int n,int m){return 1LL*frac[n]*invfrac[n-m]%Mod*invfrac[m]%Mod;}
int main()
{
    frac[0]=1;
    _for(i,1,MAXN) frac[i]=1LL*frac[i-1]*i%Mod;
    invfrac[MAXN-1]=quick_pow(frac[MAXN-1],Mod-2);
    for(int i=MAXN-1;i;i--)
        invfrac[i-1]=1LL*invfrac[i]*i%Mod;
    int T=read_int();
    while(T--){
        int n=read_int(),m=read_int(),k=read_int();
        int K=min(n,m)-k,ans=0;
        _rep(i,0,K)
            ans=(ans+1LL*C(n-i-1,n-k-i)*C(m-i-1,m-k-i)%Mod*C(k+i-1,i))%Mod;
        enter(ans);
    }
    return 0;
}
```

例题七

CF923E

题意

给定一个数 $x \in [0, n]$ 其中 $x=i$ 的概率为 p_i 每次操作将 x 等概率变成 $[0, x]$ 中的某个数。问 m 轮操作后 $x=i$ 的概率。

题解

设 $f_{k,i}$ 表示 k 轮操作后 $x=k$ 的概率，显然有 $f_{0,i}=p_i$ 并可以得到递推式

$$f_{k,i} = \sum_{j=i}^n \frac{f_{k-1,j}}{j+1}$$

考虑构造生成函数优化递推过程

$$\begin{aligned} F_k(x) &= \sum_{i=0}^n f_{k,i} x^i &&= \sum_{i=0}^n \sum_{j=i}^n \frac{f_{k-1,j}}{j+1} x^i \\ &= \sum_{j=0}^n \frac{f_{k-1,j}}{j+1} \sum_{i=0}^j x^i &&= \frac{1-x^{-1}}{1-x} \sum_{j=0}^n f_{k-1,j} \frac{x^{j+1}-1}{j+1} \\ &= \frac{1-x^{-1}}{1-x} \int_1^x F_{k-1}(t) dt \end{aligned}$$

由于 $\frac{1-x^{-1}}{1-x}$ 和 \int_1^x 不利于更进一步处理，于是考虑构造辅助函数 $G_k(x)=F_k(x+1)$

$$\begin{aligned} G_k(x) &= \frac{1}{1-x} \int_1^{x+1} F_{k-1}(t) dt &&= \frac{1}{1-x} \int_0^x F_{k-1}(t+1) dt \\ &= \sum_{i=0}^n \frac{g_{k-1,i}}{i+1} x^i \end{aligned}$$

于是有 $g_{k,i} = \frac{g_{k-1,i}}{i+1}$ 所以 $g_{m,i} = \frac{g_{0,i}}{(i+1)^m}$

接下来考虑 $g_{k,i}$ 与 $f_{k,i}$ 之间的转化，简记 $g_i = g_{k,i}, f_i = f_{k,i}$ 于是有

$$g_i = \sum_{j=i}^n \binom{j}{i} f_j = \sum_{j=i}^n \frac{j!}{i!(j-i)!} f_j = \frac{1}{i!} \sum_{j=0}^{n-i} \binom{n-i}{j} f_{i+j}$$

记 $a_i = \frac{1}{i!}, b_i = \binom{n-i}{n-i} f_{n-i}$ 于是有 $g_i = \sum_{j=0}^{n-i} a_j b_{n-i-j}$

直接 NTT 可以由 $F_0(x)$ 求出 $G_0(x)$ 进而求出 $G_m(x)$ 考虑对 $\frac{1}{i!} x^i$ 求逆再与 $G_m(x)$ 卷积即可求出 $F_m(x)$ 总时间复杂度 $O(n \log n)$

```
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;
}
```

```

}
}
int a[MAXN<<2],b[MAXN<<2],c[MAXN<<2],frac[MAXN],invfrac[MAXN];
int main()
{
    Poly::init();
    frac[0]=1;
    _for(i,1,MAXN)frac[i]=1LL*frac[i-1]*i%Mod;
    invfrac[MAXN-1]=quick_pow(frac[MAXN-1],Mod-2);
    for(int i=MAXN-1;i;i--)invfrac[i-1]=1LL*invfrac[i]*i%Mod;
    int n=read_int(),m=read_LL()%(Mod-1)*(Mod-2)%(Mod-1);
    _rep(i,0,n)
    a[n-i]=1LL*read_int()*frac[i]%Mod;
    _rep(i,0,n)
    b[i]=invfrac[i];
    Poly::Mul(a,n+1,b,n+1);
    _rep(i,0,n)
    a[i]=1LL*a[i]*quick_pow(n-i+1,m)%Mod,b[i]=invfrac[i];
    Poly::Inv(b,c,n+1);
    Poly::Mul(a,n+1,c,n+1);
    _rep(i,0,n)a[i]=1LL*a[i]*invfrac[n-i]%Mod;
    reverse(a,a+n+1);
    _rep(i,0,n)space(a[i]);
    return 0;
}

```

例题八

[洛谷p5408](#)

题意

定义第一类斯特林数 $\begin{bmatrix} n \\ k \end{bmatrix}$ 表示将 n 个不同元素构成 m 个圆排列的数目。求 $\begin{bmatrix} n \\ i \end{bmatrix} (0 \leq i \leq n)$

题解

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

Permanent link: https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E7%94%9F%E6%88%90%E5%87%BD%E6%95%B0_1&rev=1598007161

Last update: 2020/08/21 18:52