斯特林数

第一类斯特林数

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 个圆排列的方案。

性质一

$$\begin{bmatrix}n\\ k\end{bmatrix}=\begin{bmatrix}n-1\\ k-1\end{bmatrix}+(n-1)\begin{bmatrix}n-1\\ k\end{bmatrix}$$

考虑新加入的数 n□要么单独成环，要么插入到其他环中，其中插入方式有 $n-1$ 种。

性质二

$$x^{\overline n}=\sum_{i=0}^n \begin{bmatrix}n\\ i\end{bmatrix}x^i$$

其中 $x^{\overline n}$ 表示上升幂。

考虑归纳证明，有

$$x^{\overline {n+1}}=x^{\overline n}(x+n)=(x+n)\sum_{i=0}^n \begin{bmatrix}n\\ i\end{bmatrix}x^i=\sum_{i=0}^{n+1} \left(\begin{bmatrix}n\\ i-1\end{bmatrix}+n\begin{bmatrix}n\\ i\end{bmatrix}\right)x^i=\sum_{i=0}^{n+1} \begin{bmatrix}n+1\\ i\end{bmatrix}x^i$$

性质三

$$x^{\underline n}=\sum_{i=0}^n \begin{bmatrix}n\\ i\end{bmatrix}(-1)^{n-i}x^i$$

其中 $x^{\underline n}$ 表示下降幂。

考虑归纳证明，有

$$x^{\underline {n+1}}=x^{\underline n}(x-n)=(x-n)\sum_{i=0}^n \begin{bmatrix}n\\ i\end{bmatrix}(-1)^{n-i}x^i=\sum_{i=0}^{n+1} \left(-\begin{bmatrix}n\\ i-1\end{bmatrix}-n\begin{bmatrix}n\\ i\end{bmatrix}\right)(-1)^{n-i}x^i=\sum_{i=0}^{n+1} \begin{bmatrix}n+1\\ i\end{bmatrix}(-1)^{n+1-i}x^i$$

运算

第一类斯特林数$\cdot$行

洛谷p5408

$$\begin{bmatrix}n\\ i\end{bmatrix}(0\le i\le n)$$

考虑倍增法，假设已知 $x^{\overline n}$□显然可以 $O(n)$ 计算出 $x^{\overline {n+1}}=(x+n)x^{\overline n}$□

接下来考虑求解 $x^{\overline {2n}}$□有 $x^{\overline {2n}}=x^{\overline n}(x+n)^{\overline n}$□设 $x^{\overline n}=\sum_{i=0}^n a_ix^i$□于是有

$$(x+n)^{\overline n}=\sum_{i=0}^n a_i(x+n)^i=\sum_{i=0}^n a_i\sum_{j=0}^i{i\choose j}n^{i-j}x^j=\sum_{j=0}^n x^j\sum_{i=j}^n a_i{i\choose j}n^{i-j}$$

展开组合数，有

$$\sum_{j=0}^n x^j\sum_{i=j}^n a_i{i\choose j}n^{i-j}x^j=\sum_{j=0}^n \frac {x^j}{j!}\sum_{i=j}^n a_ii!\frac {n^{i-j}}{(i-j)!}=\sum_{j=0}^n \frac {x^j}{j!}\sum_{i=0}^{n-j} a_{i+j}(i+j)!\frac {n^i}{i!}$$

设 $b_i=a_ii!,c_i=\cfrac {n^{n-i}}{(n-i)!}$□于是有

$$\sum_{j=0}^n \frac {x^j}{j!}\sum_{i=0}^{n-j} a_{i+j}(i+j)!\frac {n^i}{i!}=\sum_{j=0}^n \frac {x^j}{j!}\sum_{i=0}^{n-j} b_{i+j}c_{n-i}$$

于是可以 $O(n\log n)$ 计算出 $(x+n)^{\overline n}$□最后与 $x^{\overline n}$ 卷积即可 $O(n\log n)$ 计算出 $x^{\overline {2n}}$□

总时间复杂度 $T(n)=T(\frac n2)+O(n\log n)$□于是 $T(n)=O(n\log n)$□

```
const int MAXN=1<<18,Mod=167772161,G=3;
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;
    }
}
int f[MAXN],g[MAXN],h[MAXN],frac[MAXN],invfrac[MAXN],pown[MAXN];
void Add(int n){
    for(int i=n;i>=0;i--)
    f[i+1]=(f[i]+1LL*f[i+1]*n)%Mod;
}
void Mul(int n){
    _rep(i,0,n)g[i]=1LL*f[i]*frac[i]%Mod;
    pown[0]=1;
    _rep(i,1,n)pown[i]=1LL*pown[i-1]*n%Mod;
```

```
    _rep(i,0,n)h[i]=1LL*pown[n-i]*invfrac[n-i]%Mod;
    Poly::Mul(g,n+1,h,n+1);
    _rep(i,0,n)g[i]=1LL*g[n+i]*invfrac[i]%Mod;
    Poly::Mul(f,n+1,g,n+1);
}
int main()
{
    Poly::init();
    int n=read_int(),pos=18,cur=1;
    while(n<(1<<pos))pos--;
    f[0]=0,f[1]=1,frac[0]=1;
    _rep(i,1,n)frac[i]=1LL*frac[i-1]*i%Mod;
    invfrac[n]=quick_pow(frac[n],Mod-2);
    for(int i=n;i;i--)invfrac[i-1]=1LL*invfrac[i]*i%Mod;
    while(pos--){
        Mul(cur);
        cur<<=1;
        if((n>>pos)&1){
            Add(cur);
            cur|=1;
        }
    }
    _rep(i,0,n)
    space(f[i]);
    return 0;
}
```

第一类斯特林数$\cdot$列

洛谷p5409

$$\begin{bmatrix}i\\ k\end{bmatrix}(0\le i\le n)$$

考虑只有一个圆排列的方案的 EGF，有 $F(x)=\sum_{i=1}^{\infty} (i-1)!\cfrac {x^i}{i!}$。

于是 k 个圆排列的 EGF 为 $\cfrac {F^k(x)}{k!}$，利用 Exp 和 Ln 可以 $O(n\log n)$ 计算。

```
const int MAXN=1<<16,Mod=167772161,G=3;
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);
    }
    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 f[MAXN<<2],g[MAXN<<2],frac[MAXN<<1],invfrac[MAXN<<1];
```

```
int main()
{
    Poly::init();
    int n=read_int(),k=read_int();
    frac[0]=1;
    _rep(i,1,n)frac[i]=1LL*frac[i-1]*i%Mod;
    invfrac[n]=quick_pow(frac[n],Mod-2);
    for(int i=n;i;i--)invfrac[i-1]=1LL*invfrac[i]*i%Mod;
    _rep(i,1,n)f[i]=1LL*frac[i-1]*invfrac[i]%Mod;
    Poly::Pow(f,g,n+1,k,k);
    _rep(i,0,n)g[i]=1LL*g[i]*frac[i]%Mod*invfrac[k]%Mod;
    _rep(i,0,n)space(g[i]);
    return 0;
}
```

第二类斯特林数

定义

第二类斯特林数 $\begin{Bmatrix}n\\ k\end{Bmatrix}$ 表示将 n 个不同元素划分到 m 个非空集的方案数。

性质一

$$\begin{Bmatrix}n\\ k\end{Bmatrix}=\begin{Bmatrix}n-1\\ k-1\end{Bmatrix}+k\begin{Bmatrix}n-1\\ k\end{Bmatrix}$$

考虑新加入的数 n要么单独划分成一个集合，要么加入到其他集合中，其中加入方式有 k 种。

性质二

$$x^n=\sum_{i=0}^n \begin{Bmatrix}n\\ i\end{Bmatrix}(-1)^{n-i}x^{\overline i}$$

其中 $x^{\overline i}$ 表示上升幂。

考虑归纳证明，有

$$x^{n+1}=x\sum_{i=0}^n \begin{Bmatrix}n\\ i\end{Bmatrix}(-1)^{n-i}x^{\overline i}=\sum_{i=0}^n \begin{Bmatrix}n\\ i\end{Bmatrix}(-1)^{n-i}\left(x^{\overline {i+1}}-ix^{\overline i}\right)=\sum_{i=0}^{n+1} \begin{Bmatrix}n+1\\ i\end{Bmatrix}(-1)^{n+1-i}x^{\overline i}$$

性质三

$$x^n=\sum_{i=0}^n \begin{Bmatrix}n\\ i\end{Bmatrix}x^{\underline i}$$

其中 $x^{\underline i}$ 表示下降幂。

考虑归纳证明，有

$$x^{n+1}=x\sum_{i=0}^n \begin{Bmatrix}n\\ i\end{Bmatrix}x^{\underline i}=\sum_{i=0}^n \begin{Bmatrix}n\\ i\end{Bmatrix}\left(x^{\underline {i+1}}+ix^{\underline i}\right)=\sum_{i=0}^{n+1} \begin{Bmatrix}n+1\\ i\end{Bmatrix}x^{\underline i}$$

运算

第二类斯特林数$\cdot$行

洛谷p5395

```

```

第二类斯特林数$\cdot$列

洛谷p5396

```

```