

# LGV 引理

## 算法简介

一种统计有向无环图不相交路径集的算法。

## 算法原理

定义路径的权值  $w(P)$  表示路径  $P$  的所有边权之积。

定义  $e(u,v)$  表示所有  $u \rightarrow v$  的路径的权值之和，即  $e(u,v) = \sum_{P \in \{u \rightarrow v\}} w(P)$

定义起点集合为  $A$  其中第  $i$  个起点为  $a_i$  终点集合为  $B$  其中第  $i$  个终点为  $b_i$

定义路径组的集合  $S(A,B)$  中的每个元素  $c$  表示一个路径组，含有  $n$  条路径，其中第  $i$  条路径  $a_i \rightarrow b_{p_i}$   $P$  是  $1 \sim n$  的排列。

定义  $N(c)$  表示路径组  $c$  的  $P$  排列中的逆序对个数  $w(c)$  表示路径组  $c$  的所有路径权值之积。

设

$$M = \begin{bmatrix} e(a_1,b_1) & e(a_1,b_2) & \cdots & e(a_1,b_n) \\ e(a_2,b_1) & e(a_2,b_2) & \cdots & e(a_2,b_n) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_n,b_1) & e(a_n,b_2) & \cdots & e(a_n,b_n) \end{bmatrix}$$

则有

$$\det M = \sum_{c \in S(a,b)} (-1)^{N(c)} w(c)$$

特别的，令图上所有边权为  $1$ ，同时限定  $a_i$  的对应点一定是  $b_i$  即  $c$  的排列就是  $1 \sim n$

此时有  $N(c)=0, w(c)=1$  于是

$$\det M = \sum_{c \in S(a,b)} 1$$

## 算法模板

### HDU5852

给定一个二维平面，求满足如下条件的  $k$  元路径组个数：

- 第  $i$  条路径  $(1,a_i) \rightarrow (n,b_i)$
- 每次移动只能选择  $(x,y) \rightarrow (x,y+1), (x+1,y)$

显然  $e(i,j) = \binom{n-1+b_j-a_i}{n-1}$  然后直接跑高斯消元板子，时间复杂度  $O(k^3)$

```
const int mod=1e9+7,MAXN=2e5+5,MAXK=105;
```

```
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
int frac[MAXN],invf[MAXN];
int C(int n,int m){
    if(n<m)return 0;
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
int a[MAXN],b[MAXN];
int c[MAXK][MAXK];
int cal(int n){
    int ans=1;
    _rep(i,1,n){
        int pos=0;
        _rep(j,i,n){
            if(c[j][i]){
                pos=j;
                break;
            }
        }
        if(!pos)return 0;
        if(pos!=i){
            _rep(j,i,n)
            swap(c[i][j],c[pos][j]);
        }
        ans=1LL*ans*c[i][i]%mod;
        int t=quick_pow(c[i][i],mod-2);
        _rep(j,i,n)
        c[i][j]=1LL*c[i][j]*t%mod;
        _rep(j,i+1,n){
            for(int k=n;k>=i;k--)
                c[j][k]=(c[j][k]-1LL*c[j][i]*c[i][k])%mod;
        }
    }
    return (ans+mod)%mod;
}
void solve(){
    int n=read_int(),k=read_int();
    _rep(i,1,k)a[i]=read_int();
    _rep(i,1,k)b[i]=read_int();
    _rep(i,1,k)_rep(j,1,k)
    c[i][j]=C(n-1+b[j]-a[i],n-1);
    enter(cal(k));
}
```

```

int main(){
    frac[0]=1;
    _for(i,1,MAXN)
        frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
    for(int i=MAXN-1;i;i--){
        invf[i-1]=1LL*invf[i]*i%mod;
    }
    int T=read_int();
    while(T--){
        solve();
    }
    return 0;
}

```

## 算法例题

### 2021牛客暑期多校训练营9 C

#### 题意

给定一个二维平面，求满足如下条件的  $n$  元路径组个数：

1. 第  $i$  条路径  $(a_i, 0) \rightarrow (0, i)$
2. 每次移动只能选择  $(x, y) \rightarrow (x-1, y), (x, y+1)$

数据保证  $a_{i-1} \leq a_i$

#### 题解

显然有

$$M = \begin{bmatrix} \binom{a_1+1}{1} & \binom{a_1+2}{2} & \cdots & \binom{a_1+n}{n} \\ \binom{a_2+1}{1} & \binom{a_2+2}{2} & \cdots & \binom{a_2+n}{n} \\ \vdots & \vdots & \ddots & \vdots \\ \binom{a_n+1}{1} & \binom{a_n+2}{2} & \cdots & \binom{a_n+n}{n} \end{bmatrix} = \prod_{i=1}^n \frac{1}{i!} \begin{bmatrix} \frac{(a_1+1)!}{a_1!} & \frac{(a_1+2)!}{a_1!} & \cdots & \frac{(a_1+n)!}{a_1!} \\ \frac{(a_2+1)!}{a_2!} & \frac{(a_2+2)!}{a_2!} & \cdots & \frac{(a_2+n)!}{a_2!} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{(a_n+1)!}{a_n!} & \frac{(a_n+2)!}{a_n!} & \cdots & \frac{(a_n+n)!}{a_n!} \end{bmatrix}$$

设  $x_i = a_i + 1$  则

$$M = \prod_{i=1}^n \frac{1}{i!} \begin{bmatrix} x_1 & x_1(x_1+1) & \cdots & \prod_{i=0}^{n-1} (x_1+i) \\ x_2 & x_2(x_2+1) & \cdots & \prod_{i=0}^{n-1} (x_2+i) \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_n(x_n+1) & \cdots & \prod_{i=0}^{n-1} (x_n+i) \end{bmatrix}$$

从左到右用列消元，可以得到

$$M = \prod_{i=1}^n \frac{1}{i!} \begin{bmatrix} x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

$$\begin{bmatrix} x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_n^2 & \dots & x_n^n \end{bmatrix}$$

每行都提出一个  $x_i$  就可以得到一个范德蒙行列式，于是有

$$\det M = \prod_{i=1}^n \frac{a_{i+1}^{i!}}{\prod_{1 \leq i < j \leq n} (a_j - a_i)}$$

考虑  $\text{NTT}$  计算每个值在  $\prod_{1 \leq i < j \leq n} (a_j - a_i)$  出现的次数。

具体的，可以设  $f(x) = \sum_{i=1}^n x^{a_i}, g(x) = \sum_{i=1}^n x^{-a_i}$  则每个值  $k$  出现次数就是  $[x^k]f(x)g(x)$

注意还有  $i < j$  的限制，根据对称性，考虑  $\frac{1}{2}$  然后做快速幂即可，时间复杂度  $O(V \log V)$

```
const int mod=998244353,MAXN=1e6+5;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
namespace Poly{
    const int Mod=998244353,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 n=build(_n+_m-2);
    _for(i,_n,n)f[i]=0;_for(i,_m,n)g[i]=0;
    NTT(f,n,1);NTT(g,n,1);
    _for(i,0,n)f[i]=1LL*f[i]*g[i]%Mod;
    NTT(f,n,0);
}
}
int frac[MAXN],invf[MAXN],a[MAXN<<2],b[MAXN<<2];
int main(){
    frac[0]=1;
    _for(i,1,MAXN)
        frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
    for(int i=MAXN-1;i;i--)
        invf[i-1]=1LL*invf[i]*i%mod;
    int n=read_int(),base=1e6,ans=1;
    _rep(i,1,n){
        int t=read_int();
        ans=1LL*ans*(t+1)%mod*invf[i]%mod;
        a[t]++;
        b[base-t]++;
    }
    Poly::init();
    Poly::mul(a,base+1,b,base+1);
    _rep(i,base+1,base*2)
        ans=1LL*ans*quick_pow(i-base,a[i])%mod;
    enter((ans+mod)%mod);
    return 0;
}

```

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

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:lgv%E5%BC%95%E7%90%86&rev=1629010706](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:lgv%E5%BC%95%E7%90%86&rev=1629010706) 

Last update: **2021/08/15 14:58**