

# 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  $u \to v$  的路径的权值之和，即  $e(u,v) = \sum_{P \in \{u \to v\}} w(P)$

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  $a_i \to b_{p_i}$   $P$  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

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  $(1,a_i) \to (n,b_i)$
- 每次移动只能选择  $(x,y) \to (x,y+1), (x+1,y)$

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。

```
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;
}
```

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=1628946748](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:lgv%E5%BC%95%E7%90%86&rev=1628946748) 

Last update: 2021/08/14 21:12