二项式反演

算法简介

一种特殊反演，主要应用于组合计数。

算法思想

$$f(n)=\sum_{i=m}^n{n\choose i}g(i)\iff g(n)=\sum_{i=m}^n(-1)^{n-i}{n\choose i}f(i)$$

$$f(n)=\sum_{i=n}^m{i\choose n}g(i)\iff g(n)=\sum_{i=n}^m(-1)^{i-n}{i\choose n}f(i)$$

算法例题

例题一

题意

求错排数 D_n

题解

考虑任意排列，有 $n!$ 种，其中恰有 i 封信错排的方案数为 ${n\choose i}D_i$于是 $n!=\sum_{i=0}^n{n\choose i}D_i$

根据二项式反演，有 $D_n=\sum_{i=0}^n(-1)^{n-i}{n\choose i}i!$

例题二

题意

一个有 n 个元素的集合，现在要从他的所有子集中取出至少一个集合，使得所选子集的交集的元素个数为 k求方案数。

题解

定义 $f(k)$ 表示所有钦定 k 个元素属于交集(其余元素任意)的方案数之和，于是有 $f(k)={n\choose k}2^{2^{n-k}}$

定义 $g(k)$ 表示恰好有 k 个元素属于交集的方案数，对于 $f(k)$ 的每个方案$g(i)(i\ge k)$ 被计算

${i\choose k}$ 次，于是 $f(k)=\sum_{i=k}^n{i\choose k}g(i)$。

根据二项式反演，有 $g(k)=\sum_{i=k}^n(-1)^{i-k}{i\choose k}f(i)$。题目所求即为 $g(k)$。时间复杂度 $O(n)$。

算法练习

习题一

洛谷p5505

题意

有 n 个人和 m 种物品，第 i 种物品有 a_i 个，同种物品之间没有区别。现在要将这些物品分给这些人，使得每个人至少分到一个物品，求方案数。

题解

定义 $f(k)$ 表示所有钦定 k 个人没有分到物品(其余人任意)的方案数之和，于是有 $f(k)={n\choose k}\prod_{i=1}^m{n-k+a_i-1\choose n-k-1}$。

定义 $g(k)$ 表示恰好有 k 个人没有分到物品的方案数，对于 $f(k)$ 的每个方案，$g(i)(i\ge k)$ 被计算 ${i\choose k}$ 次，于是 $f(k)=\sum_{i=k}^n{i\choose k}g(i)$。

根据二项式反演，有 $g(k)=\sum_{i=k}^n(-1)^{i-k}{i\choose k}f(i)$。题目所求即为 $g(0)$。时间复杂度 $O(nm)$。

```
const int MAXN=2005,Mod=1e9+7;
int a[MAXN],C[MAXN][MAXN];
int main()
{
    int n=read_int(),m=read_int();
    _for(i,0,m)a[i]=read_int();
    C[0][0]=1;
    _for(i,1,MAXN){
        C[i][0]=1;
        _rep(j,1,i)
        C[i][j]=(C[i-1][j-1]+C[i-1][j])%Mod;
    }
    int ans=0;
    _rep(i,0,n){
        int t=C[n][i];
        _for(j,0,m)t=1LL*t*C[n-i+a[j]-1][n-i-1]%Mod;
        if(i&1)ans=(ans-t)%Mod;
        else ans=(ans+t)%Mod;
```

```
    }
    enter((ans+Mod)%Mod);
    return 0;
}
```

习题二

洛谷p4859

题意

给出两个长度均为 n 的序列 A 和 B，保证所有数互异。

现要将 A 序列中的数与 B 序列中的数两两配对，求 $a_i\gt b_i$ 的对数比 $a_i\lt b_i$ 的对数多 k 的配对方案数。

题解

首先如果 $n+k$ 为奇数，则方案数为 0，否则 $a_i\gt b_i$ 的对数恰好为 $\frac {n+k}2$。

将序列 A 和 B 都按从小到大排序，设 $\text{dp}(i,j)$ 表示序列 A 的前 i 个数中钦定 j 对数满足 $a_i\gt b_i$ (其余数暂时无视)的方案数之和。

设有 c 个数小于 a_i，不难得出状态转移方程 $\text{dp}(i,j)=\text{dp}(i-1,j)+(c-j+1)\text{dp}(i-1,j-1)$，边界条件 $\text{dp}(0,0)=0$。

定义 $f(k)$ 表示所有钦定 k 对数满足 $a_i\gt b_i$ (其余数对任意)的方案数之和，于是有 $f(k)=(n-k)!\text{dp}(n,k)$。

定义 $g(k)$ 表示恰好有 k 对数满足 $a_i\gt b_i$ 的方案数，对于 $f(k)$ 的每个方案，$g(i)(i\ge k)$ 被计算 ${i\choose k}$ 次，于是 $f(k)=\sum_{i=k}^n{i\choose k}g(i)$。

根据二项式反演，有 $g(k)=\sum_{i=k}^n(-1)^{i-k}{i\choose k}f(i)$，题目所求即为 $g(\frac {n+k}2)$，时间复杂度 $O(n^2)$。

```
const int MAXN=2005,Mod=1e9+9;
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 a[MAXN],b[MAXN],dp[MAXN][MAXN],frac[MAXN],invfrac[MAXN];
int C(int a,int b){return 1LL*frac[a]*invfrac[b]%Mod*invfrac[a-b]%Mod;}
```

```
int main()
{
    int n=read_int(),k=read_int(),m;
    if((n-k)&1){
        puts("0");
        return 0;
    }
    m=n+k>>1;
    _rep(i,1,n)a[i]=read_int();
    _for(i,0,n)b[i]=read_int();
    sort(a+1,a+n+1);sort(b,b+n);
    int pos=0;
    dp[0][0]=1;
    _rep(i,1,n){
        while(b[pos]<a[i]&&pos<n)pos++;
        dp[i][0]=1;
        _rep(j,1,i){
            if(j>pos)break;
            dp[i][j]=(dp[i-1][j]+1LL*dp[i-1][j-1]*(pos+1-j))%Mod;
        }
    }
    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;
    int ans=0,t;
    _rep(i,m,n){
        t=1LL*C(i,m)*frac[n-i]%Mod*dp[n][i]%Mod;
        if((i-m)&1)ans=(ans-t)%Mod;
        else ans=(ans+t)%Mod;
    }
    enter((ans+Mod)%Mod);
    return 0;
}
```