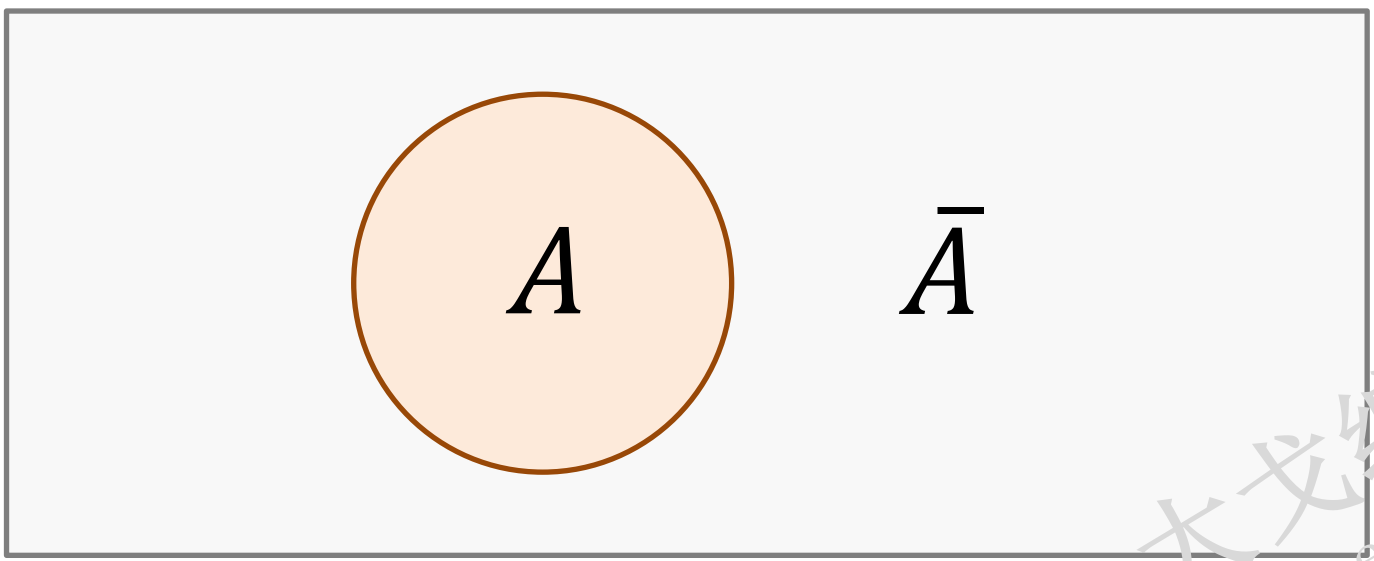

太戈编程
etiger.vip
信奥算法

集合

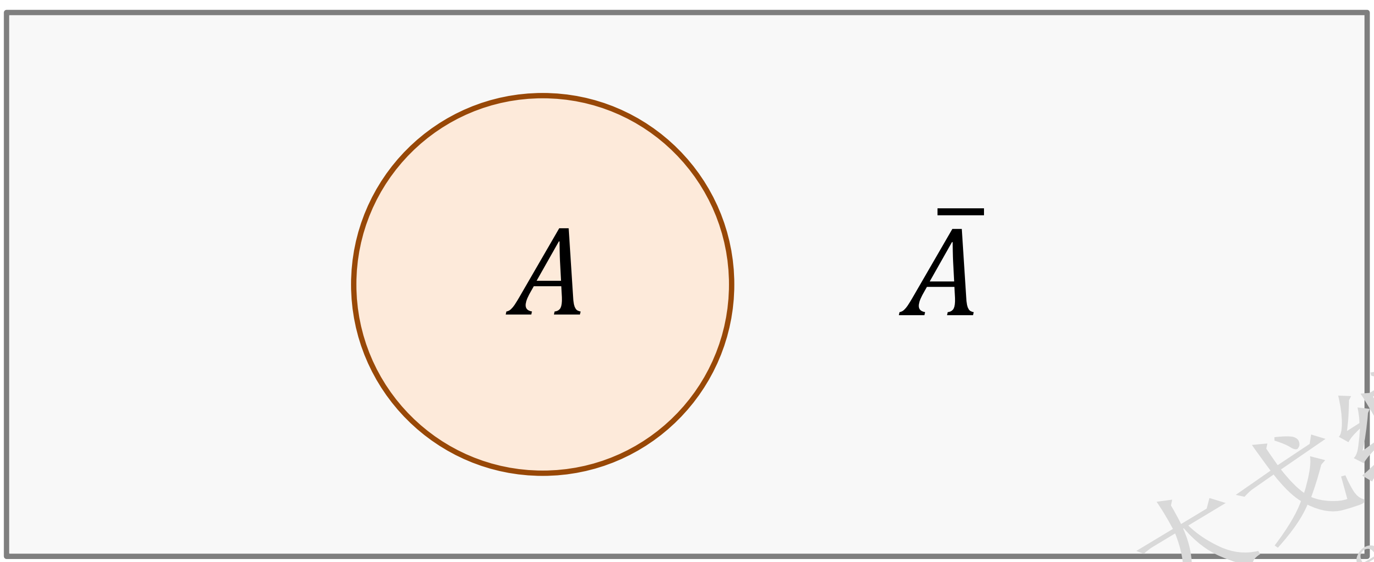
	集合A：喜欢大蒜的人
补集	集合$\bar{A}$：不喜欢大蒜的人

集合大小	$\lvert A\rvert$：集合A里有几个元素



总人数 $= |A| + |\bar{A}|$

补集转换 $|A| =$ 总人数 $- |\bar{A}|$



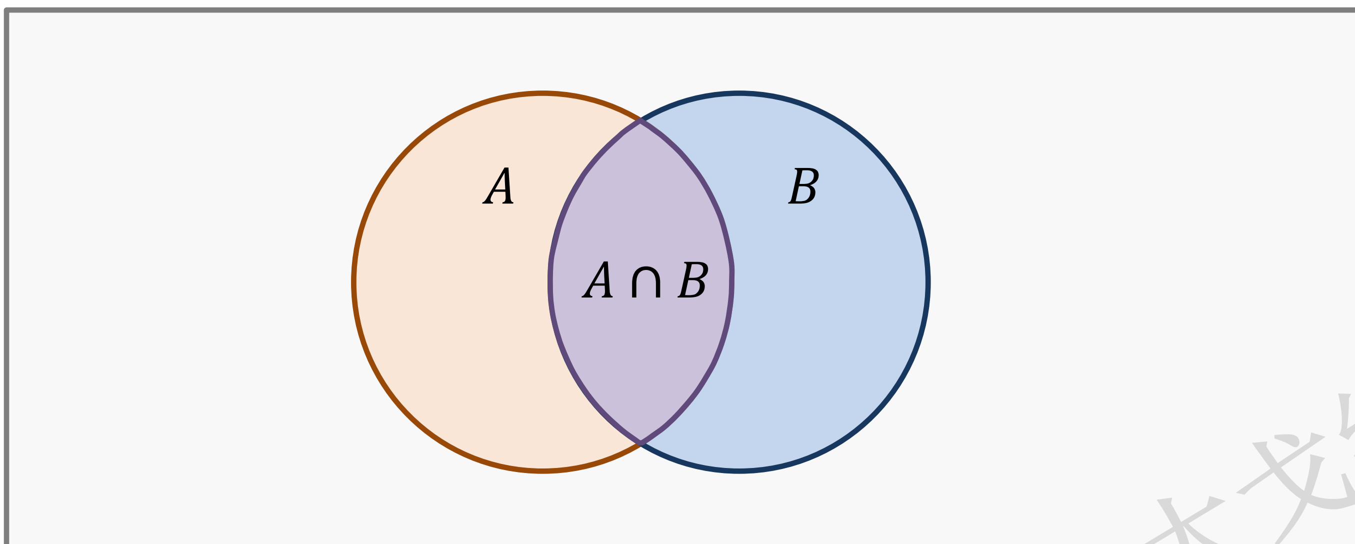
容斥原理

Inclusion-Exclusion Principle

2种约束

有2种食物：大蒜，榴莲

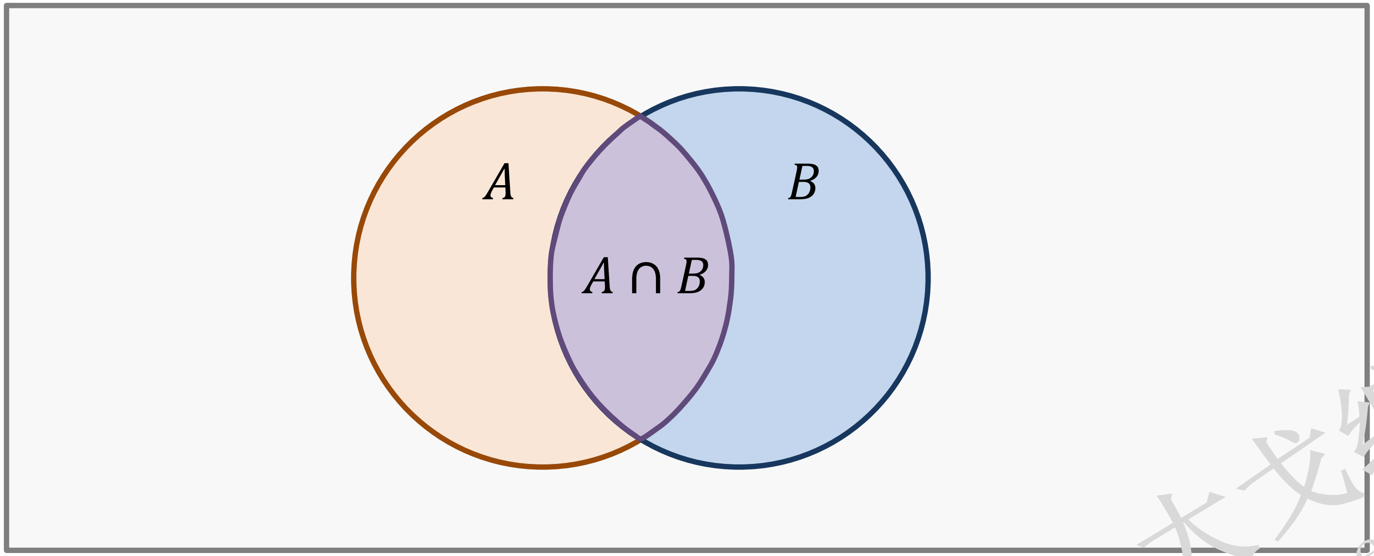
	集合A：喜欢大蒜的人
	集合B：喜欢榴莲的人
交集	集合$A \cap B$：同时喜欢大蒜和榴莲的人
并集	集合$A \cup B$：至少喜欢大蒜和榴莲之一的人



2种约束

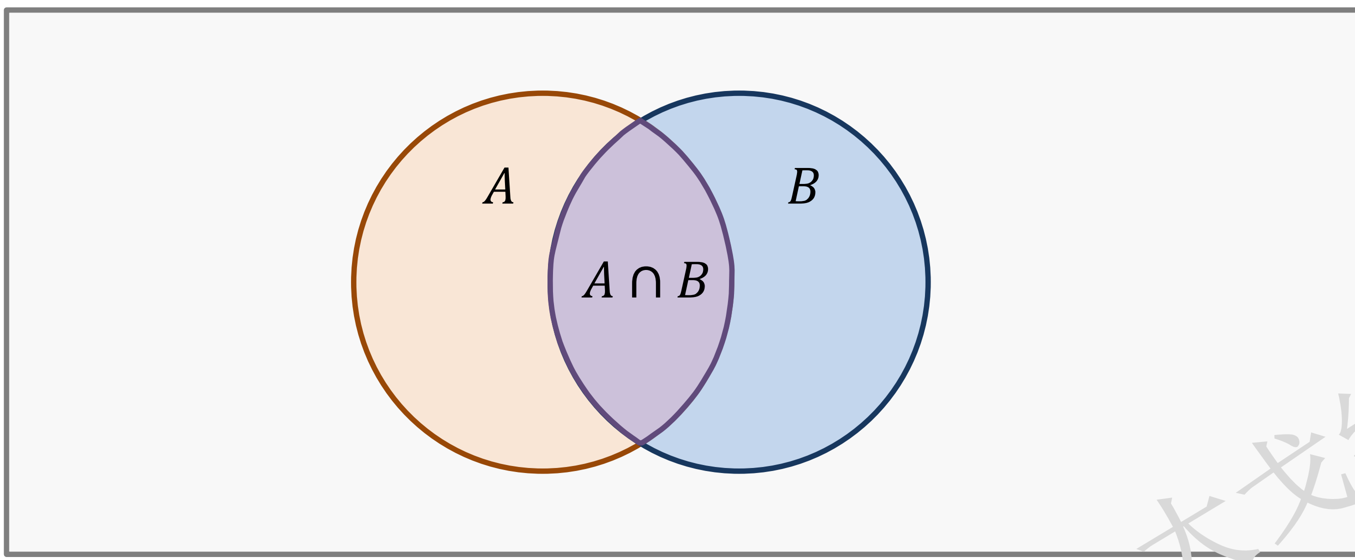
至少喜欢大蒜和榴莲之一的人数

$$|A \cup B| = |A| + |B| - |A \cap B|$$



2种约束

两种食物都不喜欢的人数
=总人数−至少喜欢大蒜和榴莲之一的人数
$= 总人数 - |A \cup B|$
$= 总人数 - |A| - |B| + |A \cap B|$



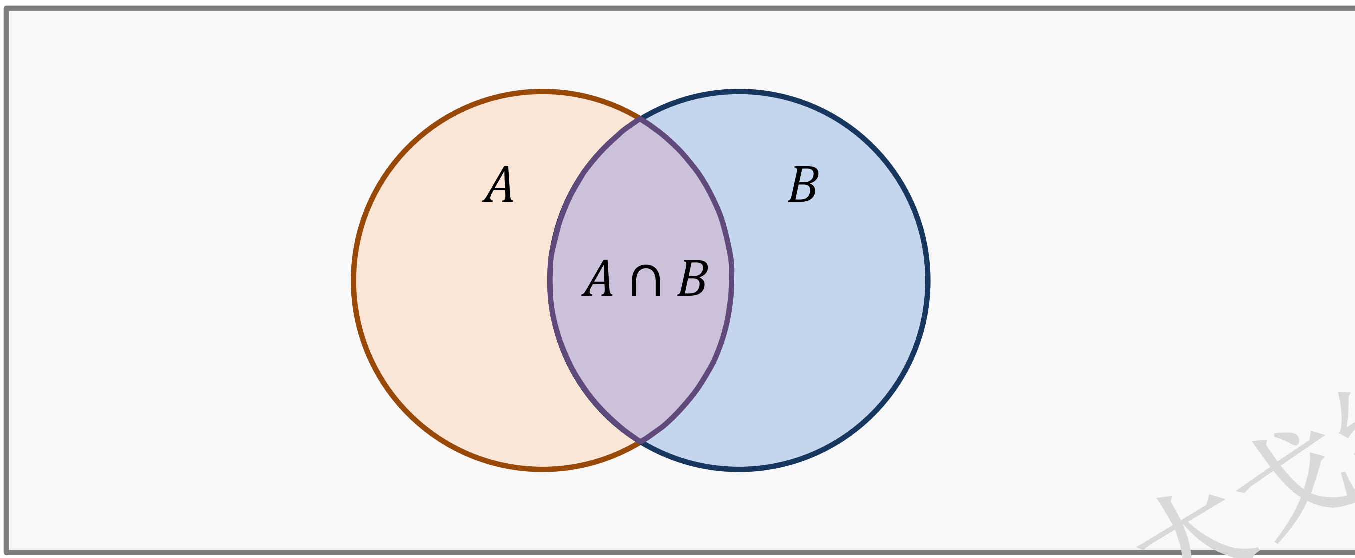
2种约束

两种食物都不喜欢的人数

$= \text{总人数} - |A| - |B| + |A \cap B|$

0种约束	1种约束	2种约束

被彩色覆盖层数对应约束个数



3种约束

有3种食物:大蒜,榴莲,香菜

集合A:喜欢大蒜的人
集合B:喜欢榴莲的人
集合C:喜欢香菜的人

至少喜欢一种食物的人数

$$|A \cup B \cup C|$$
$$= |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$$

3种食物都不喜欢的人数

$$= \text{总人数} - |A| - |B| - |C| + |A \cap B| + |B \cap C| + |C \cap A| - |A \cap B \cap C|$$

0种约束	1种约束	2种约束	3种约束
1项	3项	3项	1项

4种约束

A, B, C, D

1项	0种约束
4项	1种约束
6项	2种约束
4项	3种约束
1项	4种约束

4种食物都不喜欢的人数

$= \text{总人数}$

$-|A| - |B| - |C| - |D|$

$+|A \cap B| + |A \cap C| + |A \cap D| + |B \cap C| + |B \cap D| + |C \cap D|$

$-|A \cap B \cap C| - |A \cap B \cap D| - |A \cap C \cap D| - |B \cap C \cap D|$

$+|A \cap B \cap C \cap D|$

根据约束数量的奇偶性
切换加减符号

每一项只涉及
交集

n种约束

$A, B, C, D, E, F, G, \ldots$

0种约束	C(n,0)项
1种约束	C(n,1)项
2种约束	C(n,2)项
3种约束	C(n,3)项
4种约束	C(n,4)项
...	... ...

快快编程2616

请同学写出题目大意

算法分析

背包模型

计数问题

多重背包

朴素算法：跑T次多重背包

难点处理

识别难点	价值vi货币的使用数量不超过ci
	共4种货币,对应4种约束
简化为0个约束	每种货币使用数量无限: 完全背包问题(凑数方案数)
	f[j]计算凑出数字j共几种方案 (假设每个数字使用次数没有限制)

完全背包问题
(凑数方案数)

```
typedef long long ll;
const ll V=100009;
ll v[5],c[5],f[V];

    for(int i=1;i<=4;i++)cin>>v[i];
    f[0]=1;
    for(int i=1;i<=4;i++)
        for(int j=v[i];j<V;j++)
            
```

容斥原理

共4种货币,对应4种约束

集合A:价值v1货币使用数量超过c1的方案
集合B:价值v2货币使用数量超过c2的方案
集合C:价值v3货币使用数量超过c3的方案
集合D:价值v4货币使用数量超过c4的方案

答案描述	不在$A\cup B\cup C\cup D$内的方案数
0个约束	在不在ABCD某个集合内都没关系 f[p]计算凑出数字p共几种方案
1个约束	求|A| =价值v1货币使用数量超过c1方案数
	f[p-v1*(c1+1)]

容斥原理

0个约束	在不在ABCD某个集合内都没关系 f[p]计算凑出数字p共几种方案
1个约束	求|A| =价值v1货币使用数量超过c1方案数
	f[p-v1*(c1+1)]
2个约束	求|A∩B| =v1使用量超c1且v2使用量超c2方案数
	f[p-v1*(c1+1)-v2*(c2+1)]
3个约束	
4个约束	

容斥原理

答案描述	不在$A \cup B \cup C \cup D$内的方案数

0种约束的方案数
-|A|-|B|-|C|-|D|
+ABCD里所有2个集合交集的方案数
-ABCD里所有3个集合交集的方案数
+ABCD里所有4个集合交集的方案数

4种约束,每种约束可能被满足或不满足
共16种可能性
可以用{0,1,2,...,15}的二进制编码

根据约束数量的奇偶性
切换加减符号

```
    cin>>nQ;
    for(int q=1;q<=nQ;q++){
        for(int i=1;i<=4;i++)cin>>c[i];
        ll price;
        cin>>price;
        ll ans=f[price];
        for(int ptn=1;ptn<=15;ptn++){
            ll cnt=0;
            ll sV=0;
            for(int i=1;i<=4;i++){
                if(                    ) continue;
                cnt++;
                sV+=
            }
            if(sV>price) continue;


        }
        cout<<ans<<endl;
    }
```

快快编程2617

请同学写出题目大意

算法分析

	球盒模型	
	n个不同盒子	
难点1	m种不同球,第i种a[i]个相同	不同球相同球出现两个概念
难点2	不允许空盒	
简化1	删除难点1删除难点2	
简化2	保留难点1删除难点2	
简化3	保留难点2简化难点1	

算法分析

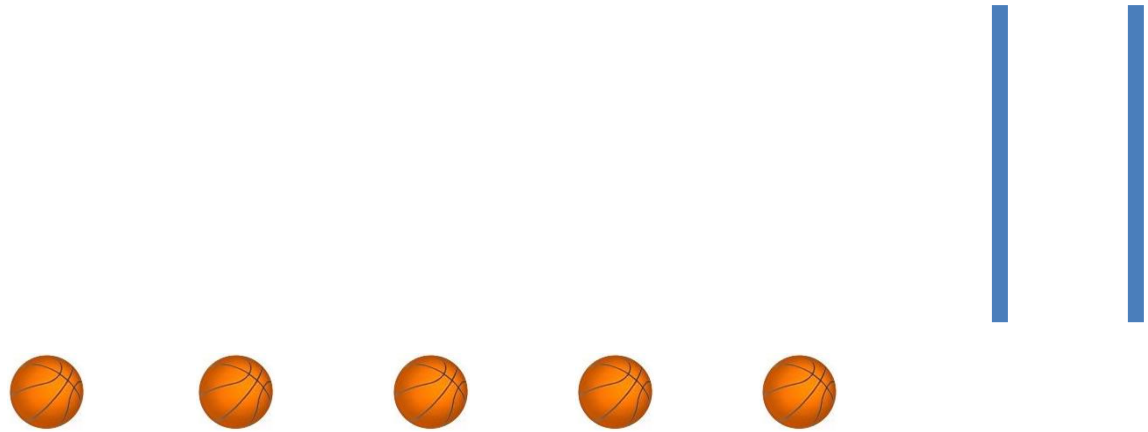
球盒模型

n个不同盒子

难点1	m种不同球,第i种a[i]个相同
难点2	不允许空盒
简化1	p个相同球,允许空盒
简化2	m种不同球,第i种a[i]个相同,允许空盒
简化3	t个不同球,不允许空盒

简化1

n个不同盒子
p个相同球,允许空盒



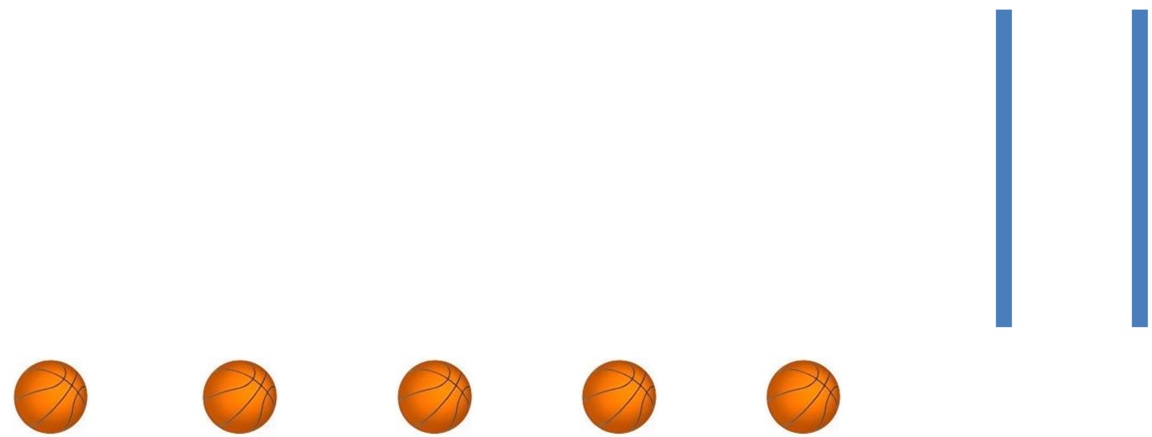
插板法

3个不同盒子放5个相同的球

7个位置选2个位置放隔板5个位置放球	C(7,2)=21

简化1

n个不同盒子
p个相同球,允许空盒



插板法

p+n-1个位置 选n-1个位置放隔板 p个位置放球
C(p+n-1,n-1) = C(p+n-1,p)

简化2

n个不同盒子

m种不同球,第i种a[i]个相同,允许空盒

分步摆放乘法原理

第1种共a[1]个相同球,方案数=C(a[1]+n-1,a[1])

第2种共a[2]个相同球,方案数=C(a[2]+n-1,a[2])

... ...

答案=C(a[1]+n-1,a[1])*C(a[2]+n-1,a[2])*...*C(a[m]+n-1,a[m])

简化3

n个不同盒子

t个不同球,不允许空盒

解法1

利用斯特林数的二维递推求出
S[t][n]表示t个不同球放入n个相同盒子
答案=S[t][n]*n!

因为t个球完全不同
所以n!对应n个盒子全排列不会产生重复方案

但原题里会有相同球出现,此解法较难应用于原题

解法2

容斥原理: 识别出 不允许空盒 对应 n个约束

$n^t - C(n,1)*(n-1)^t + C(n,2)*(n-2)^t - ...$

0约束 1约束 2约束

原题

容斥原理

答案描述	没有盒子空着的方案数

至少0个盒子空着的方案数
-至少1个盒子空着的方案数
+至少2个盒子空着的方案数
-...

f[i]表示n-i个盒子参与放球的方案数
(不要求n-i个盒子都有球)

C[n][i]*f[i]
表示n个盒子里选出i个后
使这i个盒子一定空着的方案数
(其他盒子也可能空着)

```
C[0][0]=1;
for(int i=1;i<N;++i){
    C[i][0]=C[i][i]=1;
    for(int j=1;j<i;++j)
        C[i][j]=(C[i-1][j]+C[i-1][j-1])%MOD;
}
```

f[i]表示n-i个盒子参与放球的方案数
(不要求n-i个盒子都有球)

```
    cin>>n>>m;
    for(int i=1;i<=m;++i) cin>>a[i];
    ll ans=0;
    for(int i=0;i<=n;++i){
        f[i]=1;
        for(int j=1;j<=m;++j){
            f[i]*=
            f[i]%=MOD;
        }
        if(i&1) ans-=
        else ans+=
        ans=(ans%MOD+MOD)%MOD;
    }
```

快快编程1702

请同学写出题目大意

题意抽象

nR*nC的二维表格

第i行第j列有a[i][j]个菜

要求1	每行最多选1个菜
要求2	每列选的菜量不超过总量一半

简化版：删除要求2

容斥原理

答案描述	任何列都没选出超过一半的菜的方案数

无约束的方案数
-有1列选出超过一半的菜的方案数

枚举第c列	计算：此列选出超过一半的菜的方案数

f[r][d]表示只考虑前r行时
第c列选出总数减其他列总数恰为d的方案数

因为差值d可能为负数
d范围[-nR,nR]
所以储存时平移:非负数d+nR对应d

```
ll id(ll d){return d+nR;}
```

0约束

```
cin>>nR>>nC;
for(ll r=1;r<=nR;++r)
    for(ll c=1;c<=nC;++c){
        cin>>a[r][c];
        sR[r]=(sR[r]+a[r][c])%MOD;
    }
ll ans=1;
for(ll r=1;r<=nR;++r){
    ans*=
    ans%=MOD;
}
ans=(ans-1+MOD)%MOD;
```

1个约束

```
ll id(ll d){return d+nR+1;}
```

```
    for(ll c=1;c<=nC;++c){
        memset(f,0,sizeof(f));
        f[0][id(0)]=1;
        for(ll r=1;r<=nR;++r)
            for(ll d=-r;d<=r;++d){
                f[r][id(d)]=f[r-1][id(d)];
                f[r][id(d)]+=
                f[r][id(d)]+=
                f[r][id(d)]%=MOD;
            }
        for(ll d=1;d<=nR;++d)
            ans=(ans-                +MOD)%MOD;
    }
```

3种转移的决策

第r行不取

第r行取在第c列

第r行取在其他列

太戈编程

2616,2617,1702

拓展题	1628,1629,1630,1631