

白色过膝袜

你做东方鬼畜音MAD，好吗

【日报】再谈小球与盒子

posted on 2022-06-29 12:04:24 | under [日报](#) ([.#type=日报](#)) | 20

前言

洛谷早期日报有一篇就是在讲[小球与盒子](https://www.luogu.com.cn/blog/chengni5673/dang-xiao-qiu-yu-shang-he-zi) (<https://www.luogu.com.cn/blog/chengni5673/dang-xiao-qiu-yu-shang-he-zi>)的，但是有的地方讲的不是很清楚，而且一些情形的多项式做法没讲，所以我来填坑 ()

- 本文对情形的分类主要参考了P5824 [十二重计数法](https://www.luogu.com.cn/problem/P5824) (<https://www.luogu.com.cn/problem/P5824>)。这是一道十分好的模板题，希望读者读完这篇文章后去爆切。
- 本文还是尽量从基础讲起吧，尽管会有多项式内容。
- orz EntropyIncreaser。

小球盒子模型，大致上是讲 n 个球装在 m 个盒子里，每个小球必须要放在一个盒子里，计算方案数。由于球、盒子会有不同的限制条件，因此产生了多样的计数问题。

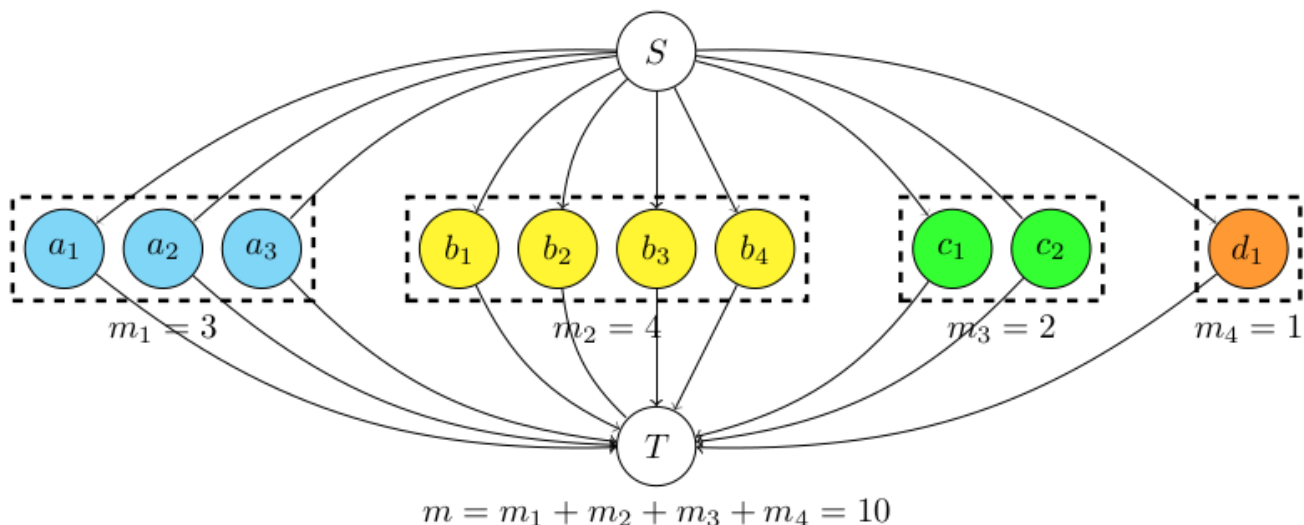
本文中，小球盒子的分类一共有 12 种。分别为**小球是否相同**、**盒子是否相同**、**装球数量无限制/最多一个球/最少一个球**，共 $2 \times 2 \times 3 = 12$ 种情况。比早期日报的那篇还多了 4 种。

正文

前置知识

- **加法原理：**

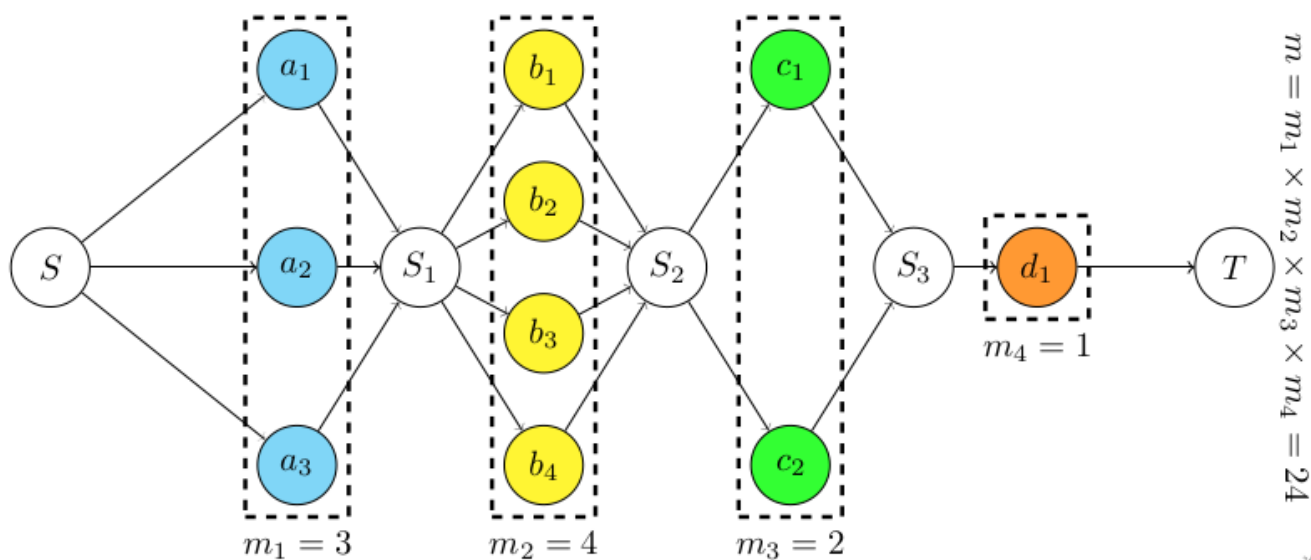
做一件事情，完成它有 n 类方式，第一类方式有 m_1 种方法，第二类方式有 m_2 种方法，.....，第 n 类方式有 m_n 种方法，那么完成这件事情共有 $m_1 + m_2 + \cdots + m_n = \sum_{i=1}^n m_i$ 种方法。



洛谷

• 乘法原理：

做一件事，完成它需要分成 n 个步骤，做第一步有 m_1 种不同的方法，做第二步有 m_2 种不同的方法，……，做第 n 步有 m_n 种不同的方法。那么完成这件事共有 $m_1 \cdot m_2 \cdots m_n = \prod_{i=1}^n m_i$ 种不同的方法。



洛谷

由此可以求出排列数 A_n^m ，它表示在 n 个物品中按顺序选出 m 个物品的方案数。显然，选出来的第一个物品有 n 种可能，第二个物品有 $n - 1$ 种可能，以此类推，第 m 个物品有 $n - m + 1$ 种可能。于是，

$$A_n^m = n \cdot (n - 1) \cdot (n - 2) \cdots (n - m + 1)$$

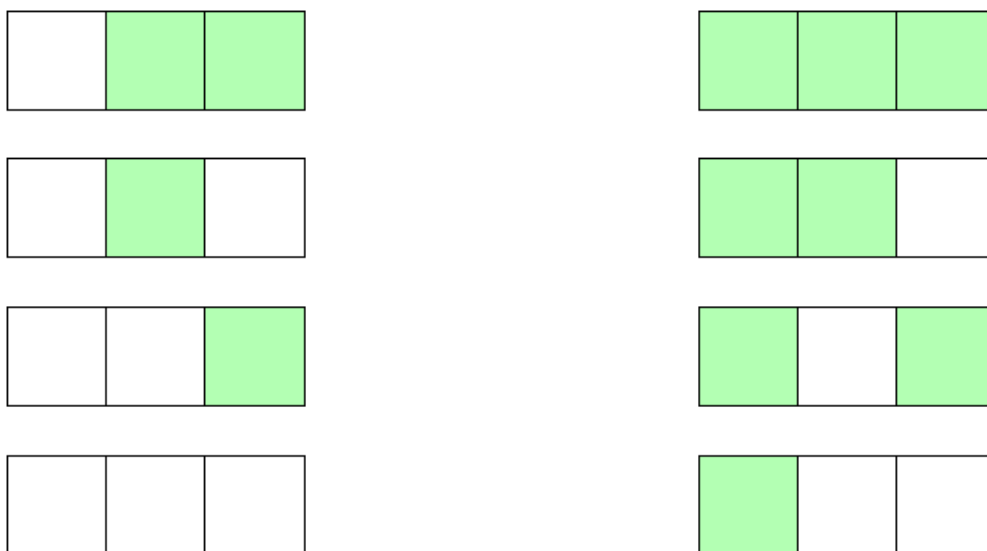
为了简便起见，

- 定义阶乘： $n! = n \cdot (n - 1) \cdot (n - 2) \cdots 1$ 。
- 定义下降幂： $n^{\underline{m}} = n \cdot (n - 1) \cdot (n - 2) \cdots (n - m + 1)$ 。

因此，

$$A_n^m = \frac{n!}{(n-m)!} = n^{\underline{m}}$$

由此引入组合数 C_n^m 。它的定义是，在 n 个不同的物品，选择 m 个数的方案数。要注意的是，这选出来的 m 个物品是没有顺序之别的。



洛谷

丢个图。这个是在 3 个数中选择若干个数的情况。可以发现， $C_3^0 = 1, C_3^1 = 3, C_3^2 = 3, C_3^3 = 1$ 。

组合数的推导也比较简单：它选出来的 m 个物品是没有顺序的，但是用排列数选出来的 m 个物品是有顺序的。显然这 m 个物品形成的排列的数量为 $A_m^m = m!$ ，而它们都等价于一种情况。因此，

$$C_n^m = \frac{A_n^m}{m!} = \frac{n!}{(n-m)!m!}$$

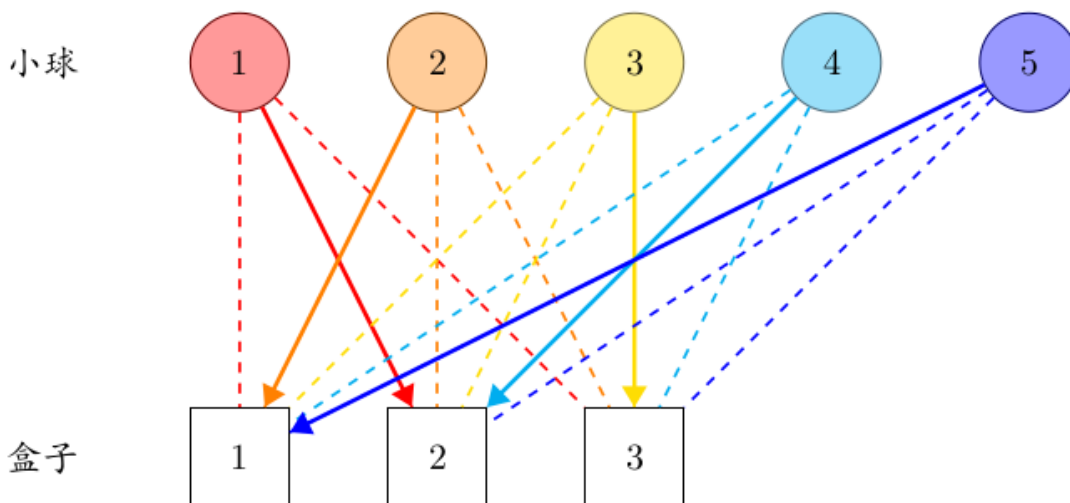
为了美观，组合数还有另外一种写法：

$$\binom{n}{m} = C_n^m$$

剩下的知识就在十二重计数法里面体现啦。

I. 球不同，盒不同，装球数量无限制

考虑从每个小球出发。每个球都有 m 个地方可以塞，根据乘法原理， n 个球的塞法共有 $\underbrace{m \cdot m \cdot m \cdots m}_{n \text{ 个}} = m^n$ 。因为每个小球进盒子后，盒子里的小球是没有顺序可言的，所以把每个球塞完后所有情况就已经统计完了。



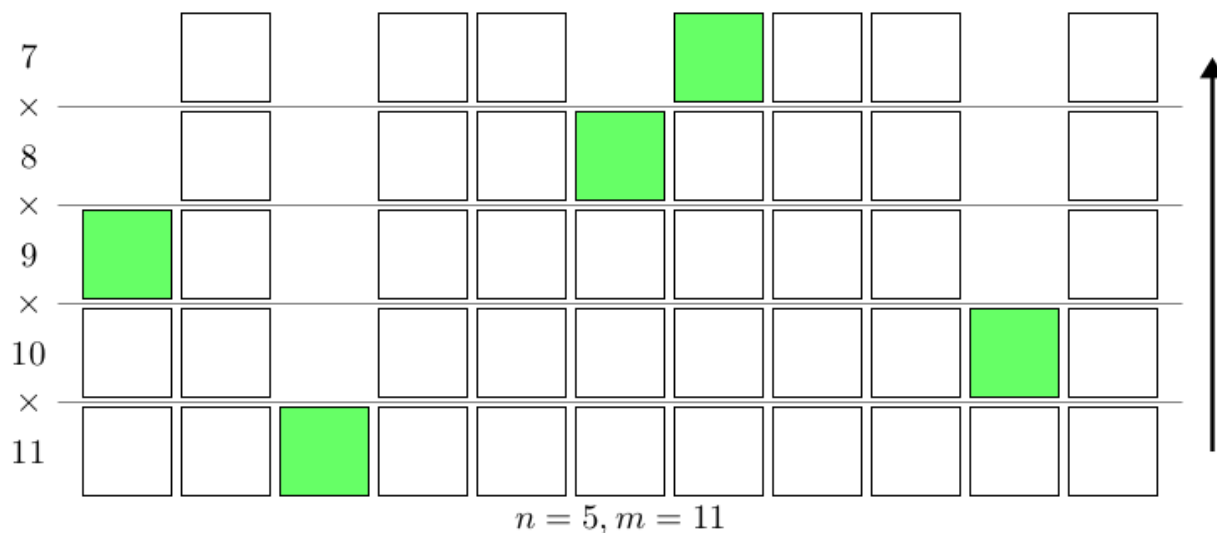
洛谷

如图所示，是 $n = 5, m = 3$ 的一种情况。五个小球分别放入了 2, 1, 3, 2, 1 号盒子里。每个小球都有 3 中选择，因此总方案数就是 3^5 。

使用快速幂，时间复杂度为 $\mathcal{O}(\log n)$ 。

II. 球不同，盒不同，盒子至多装一球

因为盒子最多只能装 1 个球，所以球和盒子是一一对应的。因此我们要按顺序从 m 个盒子选出 n 个。这就是排列数的定义，答案就是 m^n 。



洛谷

时间复杂度为 $\mathcal{O}(n)$ 。

III. 球不同，盒不同，盒子至少装一球

这里开始要引入二项式反演。

- 记「把 n 个不同球装入 m 个不同盒子，盒子非空」的方案数为 $f(n, m)$ 。

- 记「把 n 个不同球装入 m 个不同盒子，盒子可空」的方案数为 $g(n, m)$ 。

容易发现， $g(n, m)$ 就是十二重计数法的 I，于是 $g(n, m) = m^n$ 。

但是 $g(n, m)$ 还有另外一种表示方法。考虑枚举有 i 个盒子是非空的，这 i 个盒子得从 m 个盒子里选，所以要乘上系数 $\binom{m}{i}$ ；而 n 个不同小球放入 i 个盒子，由 f 的定义，就是 $f(n, i)$ 。那么：

$$g(n, m) = \sum_{i=0}^m \binom{m}{i} f(n, i)$$

首先给出二项式反演的式子：

$$f(n) = \sum_{i=0}^n \binom{n}{i} g(i) \iff g(n) = \sum_{i=0}^n \binom{n}{i} (-1)^{n-i} f(i)$$

在证明之前，我们会频繁用到二项式定理。即：

$$(a+b)^n = \sum_{i=0}^n \binom{n}{i} a^i b^{n-i}$$

二项式定理的证明并不太难。考虑右式由 n 个单项式 $(a+b)$ 相乘得到，那么若 a 的幂数为 i ，就说明这 n 个单项式中，有 i 个选择了 a 。从 n 个里面选 i 个的方案数就是 $\binom{n}{i}$ 。

$$\begin{array}{ll} (a+b)(\color{red}{a}+b)(\color{red}{a}+b)(a+b)(a+b) & (a+b)(a+b)(a+b)(\color{red}{a}+b)(\color{red}{a}+b) \\ (\color{red}{a}+b)(a+b)(a+b)(a+b)(\color{red}{a}+b) & (a+b)(a+b)(\color{red}{a}+b)(a+b)(\color{red}{a}+b) \\ (\color{red}{a}+b)(a+b)(a+b)(\color{red}{a}+b)(a+b) & (a+b)(a+b)(\color{red}{a}+b)(\color{red}{a}+b)(a+b) \\ (\color{red}{a}+b)(a+b)(\color{red}{a}+b)(a+b)(a+b) & (a+b)(\color{red}{a}+b)(a+b)(a+b)(\color{red}{a}+b) \\ (\color{red}{a}+b)(\color{red}{a}+b)(a+b)(a+b)(a+b) & (a+b)(\color{red}{a}+b)(a+b)(\color{red}{a}+b)(a+b) \end{array}$$

洛谷

例如， $n=5$ 时从 5 个单项式里找 2 个单项式选择 a ，方案数为 $\binom{5}{2}$ ，那么 a^2 的系数就是 $\binom{5}{2}$ 。

考虑二项式反演的证明：

$$\begin{aligned}
\sum_{i=0}^n \binom{n}{i} (-1)^{n-i} f(i) &= \sum_{i=0}^n \binom{n}{i} (-1)^{n-i} \sum_{j=0}^i \binom{i}{j} g(j) \\
&= \sum_{i=0}^n \sum_{j=0}^i \binom{n}{i} \binom{i}{j} (-1)^{n-i} g(j) \\
&= \sum_{i=0}^n \sum_{j=0}^i \binom{n}{j} \binom{n-j}{i-j} (-1)^{n-i} g(j) \\
&= \sum_{j=0}^n \binom{n}{j} g(j) \sum_{i=j}^n \binom{n-j}{i-j} (-1)^{n-i} \\
&= \sum_{j=0}^n \binom{n}{j} g(j) \sum_{i=0}^{n-j} \binom{n-j}{i} (-1)^{n-j-i} \\
&= \sum_{j=0}^n \binom{n}{j} g(j) \cdot 0^{n-j} \\
&= g(n)
\end{aligned}$$

现在已知：

$$g(n, m) = \sum_{i=0}^m \binom{m}{i} f(n, i)$$

于是得到，

$$\begin{aligned}
f(n, m) &= \sum_{i=0}^m (-1)^{m-i} \binom{m}{i} g(n, i) \\
&= \sum_{i=0}^m (-1)^{m-i} \binom{m}{i} i^n
\end{aligned}$$

直接暴力枚举+快速幂，时间复杂度为 $\mathcal{O}(m \log n)$ 。但是可以用筛法 $\mathcal{O}(m)$ 求出 $1^n, 2^n, \dots, m^n$ ，所以可以优化到 $\mathcal{O}(m + n)$ 。

IV. 球不同，盒相同，装球数量无限制

这里开始要引入第二类斯特林数。

第二类斯特林数 $\left\{ \begin{smallmatrix} n \\ m \end{smallmatrix} \right\}$ 的定义是，将 n 个不同的数划分为 m 个集合，**集合与集合之间没有差别**，的方案数。

考察 III 中 f 的定义。可以发现， f 所谓的盒子是互不相同的，而第二类斯特林数的集合是相同的。那么只要让 $f(n, m)$ 除去这 m 个盒子的排序关系即可。于是可以得到：

$$\begin{aligned}\left\{ \begin{matrix} n \\ m \end{matrix} \right\} &= \frac{1}{m!} f(n, m) = \frac{1}{m!} \sum_{i=0}^m (-1)^{m-i} \binom{m}{i} i^n \\ &= \sum_{i=0}^m \frac{(-1)^{m-i} \cdot i^n}{i!(m-i)!}\end{aligned}$$

回到 IV 上来。我们可以枚举有 i 个盒子至少装了 1 个球，那么这就是第二类斯特林数行的板子：

$$ans = \sum_{i=0}^m \left\{ \begin{matrix} n \\ i \end{matrix} \right\}$$

下面问题在于怎么求出 $\left\{ \begin{matrix} n \\ i \end{matrix} \right\}, i = 0, 1, \dots, m$ 。

考虑多项式乘法。对于两个多项式 $A(x) = \sum_{i=0}^n a_i x^i, B(x) = \sum_{i=0}^m b_i x^i$ ，可以得到：

$$C(x) = A(x) \cdot B(x) = \sum_{i=0}^{n+m} x^i \sum_{j=0}^i a_j b_{i-j} = \sum_{i=0}^{n+m} x^i c_i$$

那么只要构造如下多项式：

$$\begin{aligned}A(x) &= \sum_{i=0}^m a_i x^i = \sum_{i=0}^m \frac{i^n}{i!} \cdot x^i \\ B(x) &= \sum_{i=0}^m b_i x^i = \sum_{i=0}^m \frac{(-1)^i}{i!} \cdot x^i\end{aligned}$$

两者相乘，得到：

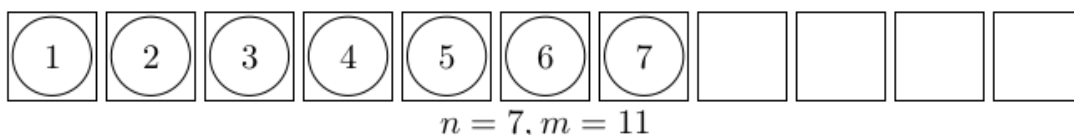
$$c_i = \sum_{j=0}^i a_j b_{i-j} = \sum_{j=0}^i \frac{(-1)^{i-j} \cdot j^n}{j!(i-j)!} = \left\{ \begin{matrix} n \\ i \end{matrix} \right\}, \quad i = 0, 1, 2 \dots m$$

- 构造 $A(x)$ 时使用筛法求出 $1^n, 2^n, \dots, m^n$ ，时间复杂度为 $\mathcal{O}(m)$ 。
- 使用 NTT 做多项式乘法，时间复杂度为 $\mathcal{O}(m \log m)$ 。

因此总的时间复杂度为 $\mathcal{O}(m \log m)$ 。

V. 球不同，盒相同，盒子至多装一球

轻松题。显然如果 $m < n$ ，无解； $m \geq n$ ，那么由于盒子相同，答案就是 1。



时间复杂度为 $\mathcal{O}(1)$ 。

VI. 球不同，盒相同，盒子至少装一球

轻松题。显然是第二类斯特林数的定义。直接拿第二类斯特林数行的做法求出一整行，再取出其中第 m 列，参照 IV。

时间复杂度为 $\mathcal{O}(m \log m)$ 。

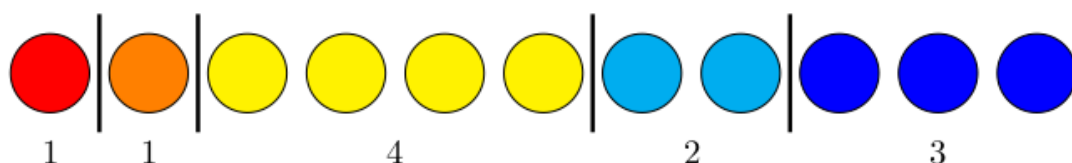
VII. 球相同，盒不同，装球数量无限制

考虑使用隔板法。



洛谷

将 n 个球依次排开。那么这 n 个球之间有 $n - 1$ 个空隙。如果我们在这 $n - 1$ 个空隙中选择 $m - 1$ 个，那么这 n 个球就会被划分为 m 堆。如下图所示，为 $n = 11, m = 5$ 的一种情况。此时五个盒子装的小球个数分别为 1, 1, 4, 2, 3。



洛谷

从左往右将每堆球丢在一个盒子里，那么我们就做到了「将 n 个相同小球，放入到 m 个不同盒子，盒子至少装一球」。容易发现插板的方案数一共有 $\binom{n-1}{m-1}$ 个，因此这个问题的方案数就是它。

但是现在盒子可以为空。做法也很简单：先往每个盒子里塞一个球，最后把这个球拿走就行了。换言之，先求「将 $n + m$ 个相同小球，放入到 m 个不同盒子，盒子至少装一球」的方案数，对于每个这个问题的划分方案，将每个盒子抽掉一个球，就是 VII 的方案。因此该问答案即为 $\binom{n+m-1}{m-1}$ 。

时间复杂度为 $\mathcal{O}(n + m)$ 。

VIII. 球相同，盒不同，盒子至多装一球

轻松题。因为每个球只能放到一个盒子里，每个盒子最多装一个球，并且每个球必须要放到一个盒子里，因此考虑 m 个盒子里哪些放了球，答案就是 $\binom{m}{n}$ 。

时间复杂度为 $\mathcal{O}(m)$ 。

IX. 球相同，盒不同，盒子至少装一球

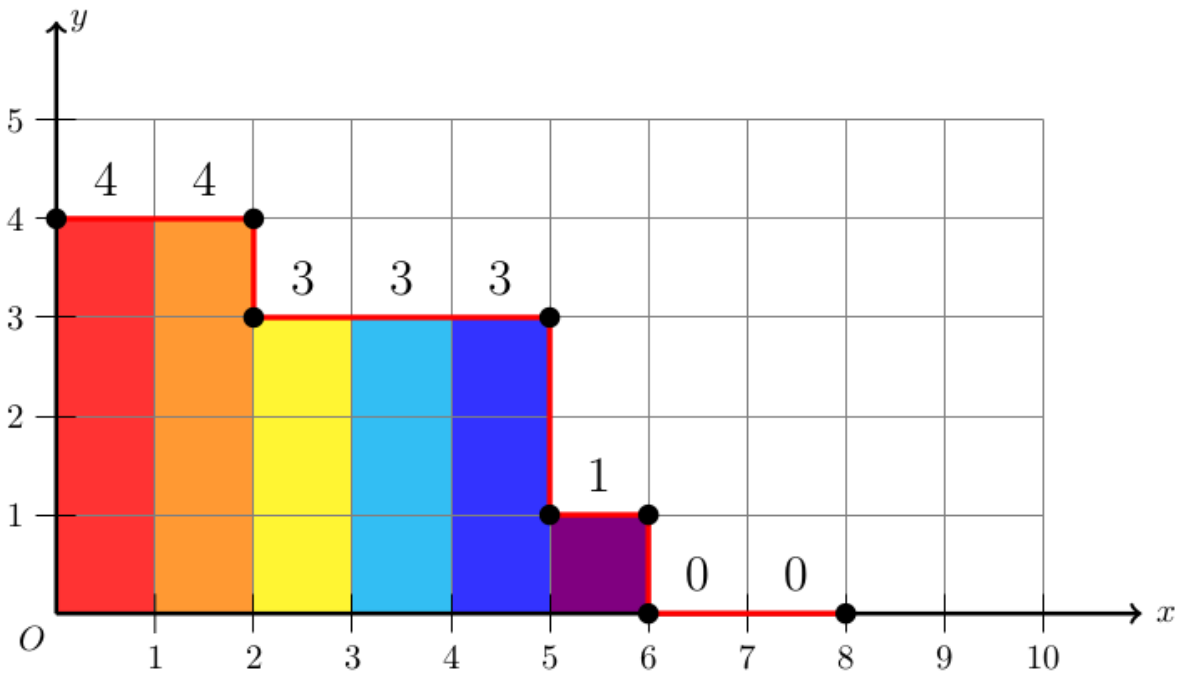
就是插板法的原版，上文已经提及。答案为 $\binom{n-1}{m-1}$ 。

时间复杂度为 $\mathcal{O}(n)$ 。

X. 球相同，盒相同，装球数量无限制

它来了。

记 $f(n, m)$ 为「 n 个相同小球，放入到 m 个相同盒子，每个盒子装球数量无限制」的方案数。观察到所有的盒子和球都是相同的，假设一个方案第 i 个盒子装的球的数量为 c_i ，那么可以将 c_i 按照从大到小的顺序排序。如果两种方案排完序后的序列完全相同，那么两者就是相同方案，否则必然是不同方案。这是本题的关键。



洛谷

如图所示，是 $n = 18, m = 8$ 的一种情形。所有在红色线段下方的面积就是 n ，每个彩色线段对应的都是排完序后每个盒子里小球的个数。 $f(n, m)$ ，本质上就是计算从 $(m, 0)$ 走到 y 轴，且红色线段下方区域面积恰好为 n 的方案数。因为盒子里小球个数按照从大到小排列，因此从 $(m, 0)$ 出发只有两种选择：

- 第一种，向上走一格。此时需要往左边填充 m 个格子，转化为了计算 $f(n - m, m)$ 。
- 第二种，向左走一格。转化为了计算 $f(n, m - 1)$ 。

容易得到状态转移方程：

$$f(n, m) = f(n, m - 1) + f(n - m, m)$$

下面考虑如何加快 $f(n, m)$ 的计算。使用生成函数。

我们记 $F_m(x) = \sum_{i=0}^{+\infty} f(i, m)x^i$, 那么根据递推式, 我们可以得到:

$$\begin{aligned} F_m(x) &= \sum_{i=0}^{+\infty} f(i, m)x^i \\ &= \sum_{i=0}^{+\infty} (f(i, m-1) + f(i-m, m))x^i \\ &= F_{m-1}(x) + x^m F_m(x) \\ F_m(x) &= \frac{1}{1-x^m} \cdot F_{m-1}(x) \end{aligned}$$

考虑到 $f(n, 0) = [n = 0]$, 于是 $F_0(x) = 1$, 那么可以得到:

$$F_m(x) = \prod_{i=1}^m \frac{1}{1-x^i}$$

两边取对数, 得到:

$$\ln F_m(x) = \sum_{i=1}^m \ln \frac{1}{1-x^i}$$

考虑一个经典结论:

$$\ln(1-x^t) = -\sum_{i=1}^{\infty} \frac{x^{ti}}{i}$$

证明如下:

$$\begin{aligned} \ln(1-x^t) &= \int \frac{-tx^{t-1}}{1-x^t} dx \\ \sum_{i=0}^{\infty} x^{ti-1} &= \frac{1}{1-x^t} \cdot \frac{1}{x} \\ \sum_{i=1}^{\infty} x^{ti-1} &= \frac{1}{1-x^t} \cdot x^{t-1} \\ \ln(1-x^t) &= \int \left(-t \cdot \sum_{i=1}^{\infty} x^{ti-1} \right) dx \\ &= -\sum_{i=1}^{\infty} \frac{x^{ti}}{i} \end{aligned}$$

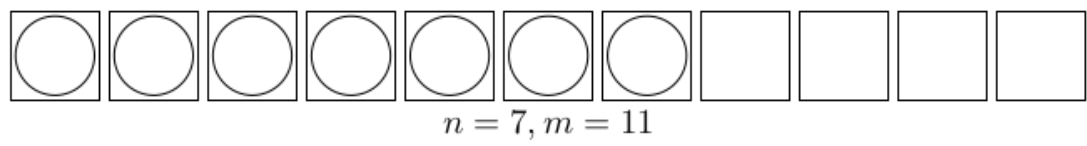
(也就是先求导再积分)

因为我们只需要 $F_m(x)$ 的前 $n+1$ 项系数, 所以只要求得 $\ln F_m(x)$ 的前 $n+1$ 项, 然后求一次多项式 \exp 即可。容易发现 $-\sum_{i=1}^{\infty} \frac{x^{ti}}{i}$ 对 $\ln F_m(x)$ 前 n 项产生贡献的项数为 $\lfloor \frac{n}{i} \rfloor$, 因此直接枚举 i , 再枚举 i 的倍数, 计入贡献即可求得 $F_m(x)$ 的前 $n+1$ 项。

算出 $\ln F_m(x)$ 的时间复杂度为 $\mathcal{O}\left(\sum_{i=1}^m \frac{n}{i}\right) = \mathcal{O}(n \log m)$ 。做一次多项式 \exp 的时间复杂度为 $\mathcal{O}(n \log n)$ 。因此总时间复杂度为 $\mathcal{O}(n \log(nm))$ 。

XI. 球相同，盒相同，盒子至多装一球

轻松题，与 V 完全相同。显然如果 $m < n$ ，无解； $m \geq n$ ，那么由于盒子相同，答案就是 1。



洛谷

时间复杂度为 $\mathcal{O}(1)$ 。

XII. 球相同，盒相同，盒子至少装一球

和 X 类似，只不过钦点了从 $(m, 0)$ 出发的点第一步必须向上走。所以答案就是 $f(n - m, m)$ 。

时间复杂度为 $\mathcal{O}(n \log(nm))$ 。

参考代码

```
#include<bits/stdc++.h>
#define up(l,r,i) for(int i=l,END##i=r;i<=END##i;++i)
#define dn(r,l,i) for(int i=r,END##i=l;i>=END##i;--i)
using namespace std;
typedef long long LL;
typedef unsigned int u32;
typedef unsigned long long u64;
const int INF =2147483647;
int qread(){
    int w=1,c,r=0;
    while((c=getchar())> '9' || c< '0') w=(c=='-'?-1:1); r=c-'0';
    while((c=getchar())>='0'&&c<='9') r=r*10+c-'0';
    return r*w;
}
const int MAX_=(1<<20)+3;
struct cplx{
    double a,b; cplx(double _a=0,double _b=0):a(_a),b(_b){}
    cplx operator +(cplx t){return cplx(a+t.a,b+t.b);}
    cplx operator -(cplx t){return cplx(a-t.a,b-t.b);}
    cplx operator *(cplx t){return cplx(a*t.a-b*t.b,a*t.b+b*t.a);}
    cplx operator *(int t){return cplx(a*t,b*t);}
};
const int MOD =998244353;
struct modint{
    unsigned w;
    modint(u32 _w=0){w=_w;}
    modint pwr(modint x,modint y){
        modint r=1; while(y.w){if(y.w&1) r=r*x; y.w>>=1,x=x*x;}
        return r;
    }
    modint pwr(modint y){
        modint x(w),r=1; while(y.w){if(y.w&1) r=r*x; y.w>>=1,x=x*x;}
        return r;
    }
    modint inv(){return modint(w).pwr(MOD-2);}
    modint dec(){if(w>=MOD) return w-MOD; return w;}
    modint operator +(modint t){return modint(w+t.w).dec();}
    modint operator -(modint t){return modint(w-t.w+MOD).dec();}
    modint operator *(modint t){return modint(1ull*w*t.w%MOD);}
    modint operator /(modint t){
        if(w%t.w==0) return modint(w/t.w); return modint(w)*t.inv();
    }
    modint operator -(){return w?MOD-w:0;}
    int to_int(){return w;}
};
struct Minv{
    modint inv(modint x){return x.inv();}
    void inv(int n,modint *T){
        T[1]=1; up(2,n,i) T[i]=modint(MOD-MOD/i)*T[MOD%i];
    }
};
```

```

}
void inv(int n,modint *A,modint *B){
    static modint P[MAX_];
    static modint Q[MAX_];
    P[0]= A[0]; up(1,n-1,i) P[i]=P[i-1]*A[i];
    Q[n-1]=inv(P[n-1]); dn(n-2,0,i) Q[i]=Q[i+1]*A[i+1];
    up(1,n-1,i) B[i]=Q[i]*P[i-1]; B[0]=Q[0];
}
void fac(int n,modint *A){
    A[0]=1; up(1,n,i) A[i]=A[i-1]*modint(i);
}
}minv;
const long double pi=acos(-1);
class Poly{
public:
void FFT(int n,cplx *Z){
    static int W[MAX_];
    int l=1; W[0]=0; while(n>=1)
        up(0,l-1,i) W[l++]=W[i]<<1|1,W[i]<=1;
    up(0,l-1,i) if(W[i]>i) swap(Z[i],Z[W[i]]);
    for(n=l>>1,l=1;n>=1,l<=1){
        cplx *S=Z,o(cos(pi/l),sin(pi/l)); up(0,n-1,i){
            cplx s(1,0); up(0,l-1,j){
                cplx x=S[j]+s*S[j+1],y=S[j]-s*S[j+1];
                S[j]=x,S[j+1]=y,s=s*o;
            }
            S+=l<<1;
        }
    }
}
void IFFT(int n,cplx *Z){
    FFT(n,Z); reverse(Z+1,Z+n);
    up(0,n-1,i) Z[i].a/=1.0*n,Z[i].b/=1.0*n;
}
void NTT(int n,modint *Z){
    static int W[MAX_];
    modint g(3); int l=1; W[0]=0; while(n>=1)
        up(0,l-1,i) W[l++]=W[i]<<1|1,W[i]<=1;
    up(0,l-1,i) if(W[i]>i) swap(Z[i],Z[W[i]]);
    for(n=l>>1,l=1;n>=1,l<=1){
        modint *S=Z,o=g.pwr((MOD-1)/l/2); up(0,n-1,i){
            modint s(1); up(0,l-1,j){
                modint x=S[j]+s*S[j+1],y=S[j]-s*S[j+1];
                S[j]=x,S[j+1]=y,s=s*o;
            }
            S+=l<<1;
        }
    }
}
void INTT(int n,modint *Z){
    NTT(n,Z); reverse(Z+1,Z+n); modint o(n); o=o.pwr(MOD-2);
    up(0,n-1,i) Z[i]=Z[i]*o;
}

```

```

}
void MUL(int n,modint *A,modint *B){
    NTT(n,A),NTT(n,B); up(0,n-1,i) A[i]=A[i]*B[i]; INTT(n,A);
}
void INV(int n,modint *Z,modint *T){
    static modint A[MAX_];
    up(0,n-1,i) T[i]=0; T[0]=Z[0].pwr(MOD-2).to_int();
    for(int l=1;l<n;l<=1){
        up(0,2*l-1,i) A[i]=Z[i]; up(2*l,4*l-1,i) A[i]=0;
        NTT(4*l,A),NTT(4*l,T);
        up(0,4*l-1,i) T[i]=modint(2)*T[i]-A[i]*T[i]*T[i];
        INTT(4*l,T);
        up(2*l,4*l-1,i) T[i]=0;
    }
}
void DIF(int n,modint *Z,modint *T){
    up(0,n-2,i) T[i]=Z[i+1]*modint(i+1); T[n-1]=0;
}
void INT(int n,int c,modint *Z,modint *T){
    up(1,n-1,i) T[i]=Z[i-1]*modint(i).pwr(MOD-2);
    T[0]=modint(c);
}
void LN(int n,modint *Z,modint *T){
    static modint A[MAX_];
    static modint B[MAX_];
    up(0,2*n-1,i) A[i]=B[i]=0;
    DIF(n,Z,A),INV(n,Z,B),MUL(2*n,A,B),INT(n,0,A,T);
}
void EXP(int n,modint *Z,modint *T){
    static modint A[MAX_];
    static modint B[MAX_];
    up(1,2*n-1,i) T[i]=0; T[0]=1;
    for(int l=1;l<n;l<=1){
        LN(2*l,T,A);
        up(0,2*l-1,i) B[i]=-A[i]+Z[i]; B[0]=B[0]+modint(1);
        up(2*l,4*l-1,i) T[i]=B[i]=0;
        MUL(4*l,T,B);
    }
}
void SQT(int n,modint *Z,modint *T){
    static modint A[MAX_];
    static modint B[MAX_];
    up(1,2*n-1,i) T[i]=0; T[0]=1; modint o=modint(2).inv();
    for(int l=1;l<n;l<=1){
        INV(2*l,T,A);
        up(0,2*l-1,i) B[i]=Z[i];
        up(2*l,4*l-1,i) A[i]=B[i]=0;
        MUL(4*l,A,B);
        up(0,2*l-1,i) T[i]=(T[i]+A[i])*o;
    }
}
void SHF(int n,modint c,modint *Z,modint *T){

```

```

static modint A[MAX_];
static modint B[MAX_];
static modint F[MAX_];
static modint G[MAX_];
modint o(1);
minv.fac(n-1,F),minv.inv(n,F,G);
up(0,n-1,i) A[i]=Z[n-1-i]*F[n-1-i];
up(0,n-1,i) B[i]=G[i]*o,o=o*c;
int l=1; while(l<2*n-1) l<<=1;
up(n,l-1,i) A[i]=B[i]=0; MUL(l,A,B);
up(0,n-1,i) T[n-1-i]=G[n-1-i]*A[i];
}

void S2L(int n,modint *T){
    static modint F[MAX_];
    static modint G[MAX_];
    static modint H[MAX_];
    static int P[MAX_]; int p=0,l=1;
    up(1,n,i) H[i]=modint(0); H[1]=1;
    up(2,n,i){
        if(H[i].w==0) P[++p]=i,H[i]=modint(i).pwr(n);
        for(int j=1;j<=p&&P[j]<=n/i;++j){
            H[i*P[j]]=H[i]*H[P[j]]; if(i%P[j]==0) break;
        }
    }
    static modint U[MAX_];
    minv.fac(n,F),minv.inv(n+1,F,G);
    up(0,n,i) T[i]=G[i]*H[i];
    up(0,n,i) U[i]=G[i]*modint(i&1?(MOD-1):1);
    while(l<2*n+1) l<<=1;
    MUL(l,T,U);
}

void DEP(int n,modint *T){ //下降幂
    if(n==1){ T[0]=0,T[1]=1; return; }
    int u=n>>1,l=1; while(l<2*u+1) l<<=1;
    modint *A=new modint[l];
    DEP(u,T), SHF(u+1,modint(MOD-u),T,A);
    up(u+1,l-1,i) T[i]=A[i]=0; MUL(l,T,A);
    if(n%2==1){ //* x-n+1
        dn(n,1,i) T[i]=T[i]*modint(MOD+1-n)+T[i-1];
        T[0]=T[0]*modint(MOD+1-n);
    }
}

void ASP(int n,modint *T){
    if(n==1){ T[0]=0,T[1]=1; return; }
    int u=n>>1,l=1; while(l<2*u+1) l<<=1;
    modint *A=new modint[l];
    ASP(u,T), SHF(u+1,u,T,A);
    up(u+1,l-1,i) T[i]=A[i]=0; MUL(l,T,A);
    if(n%2==1){ //* x+n-1
        dn(n,1,i) T[i]=T[i]*modint(n-1)+T[i-1];
        T[0]=T[0]*modint(n-1);
    }
}

```

```

}
void S2C(int n,int m,modint *T){
    static modint H[MAX_]; int l=1; while(l<n+1) l<=&=1;
    up(0,l-1,i) H[i]=0;    DEP(m+1,H);
    up(0,m,i) H[i]=H[i+1]; up(m+1,l-1,i) H[i]=0;
    reverse(H,H+m+1); INV(l,H,T);
}
void FDT(int n,modint *T){
    static modint E[MAX_];
    static modint F[MAX_];
    minv.fac(n-1,F),minv.inv(n,F,E);
    up(n,2*n-1,i) E[i]=T[i]=0;
    MUL(2*n,T,E); up(n,2*n-1,i) T[i]=0;
}
void IFDT(int n,modint *T){
    static modint E[MAX_];
    static modint F[MAX_];
    minv.fac(n-1,F),minv.inv(n,F,E);
    up(0,n-1,i) if(i&1) E[i]=-E[i];
    up(n,2*n-1,i) E[i]=T[i]=0;
    MUL(2*n,T,E); up(n,2*n-1,i) T[i]=0;
}
void PML(int n,modint *A,modint *B){
    static modint F[MAX_]; minv.fac(n-1,F);
    FDT(n,A),FDT(n,B); up(0,n-1,i) A[i]=A[i]*B[i]*F[i];
    IFDT(n,A);
}
void S1L(int n,modint *A){
    ASP(n,A);
}
}poly;
const int MAXN=(1<<20)+3;
modint M[MAXN],N[MAXN],F[MAXN],G[MAXN],O[MAXN],P[MAXN],Q[MAXN];
int main(){
    int n=qread(),m=qread(),t=n+m; modint ans=0;
    poly.S2L(n,N),poly.S2L(m,M);
    minv.fac(t,F),minv.inv(t+1,F,G),minv.inv(t,O);
    ans=modint(m).pwr(n);
    printf("%d\n",ans.to_int()),ans=0; // I
    ans=(m>=n?F[m]*G[m-n]:0);
    printf("%d\n",ans.to_int()),ans=0; // II
    ans=(n>=m?N[m]*F[m]:0);
    printf("%d\n",ans.to_int()),ans=0; // III
    up(0,m,i) if(n>=i) ans=ans+N[i];
    printf("%d\n",ans.to_int()),ans=0; // IV
    ans=(m>=n);
    printf("%d\n",ans.to_int()),ans=0; // V
    ans=(n>=m?N[m]:0);
    printf("%d\n",ans.to_int()),ans=0; // VI
    ans=F[n-1+m]*G[m-1]*G[n];
    printf("%d\n",ans.to_int()),ans=0; // VII
    ans=(m>=n?F[m]*G[n]*G[m-n]:0);

```



```

printf("%d\n",ans.to_int()),ans=0; // VIII
ans=(n>=m?F[n-1]*G[m-1]*G[n-m]:0);
printf("%d\n",ans.to_int()),ans=0; // IX
up(1,m,i){
    up(1,n/i,j) P[i*j]=P[i*j]-O[j];
}
int l=1; while(l<n+1) l<=<=1;
poly.EXP(1,P,Q);
printf("%d\n",Q[n].to_int()); // X
printf("%d\n",m>=n); // XI
printf("%d\n",n>=m?Q[n-m].to_int():0); // XII
return 0;
}

```

有什么感想?

发射评论!



(<https://www.luogu.com.cn/space/show?uid=754467>).

f hxr (<https://www.luogu.com.cn/space/show?uid=754467>) 2023-04-25 17:21:17

收藏当板子%%%



(<https://www.luogu.com.cn/space/show?uid=717781>).

ZhaoShanYang (<https://www.luogu.com.cn/space/show?uid=717781>) 2023-04-25 17:04:41

orz



(<https://www.luogu.com.cn/space/show?uid=548203>).

KK lang (<https://www.luogu.com.cn/space/show?uid=548203>) 2023-04-25 15:14:01

orz



(<https://www.luogu.com.cn/space/show?uid=549289>).

RGYZ (<https://www.luogu.com.cn/space/show?uid=549289>) 2023-04-25 08:31:03