放小球问题

从排列组合说起

组合数学 Combinatorics

清华大学 马昱春

小球放盒子

- 编号的小球：
  - 4个小球：1号，2号，3号，4号
  - 取出其中的3个
  - 如果**考虑**顺序，则称之为排列数 P(4,3) 无重排列
    - P(4,3)=4×3×2 = 24
  - 如果**不考虑**顺序，则称之为组合数 C(4,3)无重组合
    - C(4,3)=24/3！=4

排列与组合

定义 [排列 Permutation]从n个不同的元素中，取r个不重复的元素，按次序排列，称为从n个中取r个的无重排列。排列的个数用$P(n,r)$表示，或者 P_n^r 。当$r=n$时称为全排列。一般不说可重即无重。$(n \geq r)$

定义 [组合 Combination]从n个不同元素中取r个不重复的元素组成一个子集，而不考虑其元素的顺序，称为从n个中取r个的无重组合。$(n \geq r)$

组合的个数用$\boldsymbol{C(n,r)}$表示或者C_n^r

排列的模型

[排列 Permutation]从n个中取r个的排列的典型例子是从n个不同的球中,取出r个,放入r个不同的盒子里,每盒1个。$(n \geq r)$

- 第1个盒子有n种选择，第2个有n-1种选择， ……，第r个有n-r+1种选择。

故有 $P(n,r)=n(n-1)\cdots(n-r+1)=\dfrac{n!}{(n-r)!}$

有时也用$[n]_r$记$n(n-1)\cdots(n-r+1)$

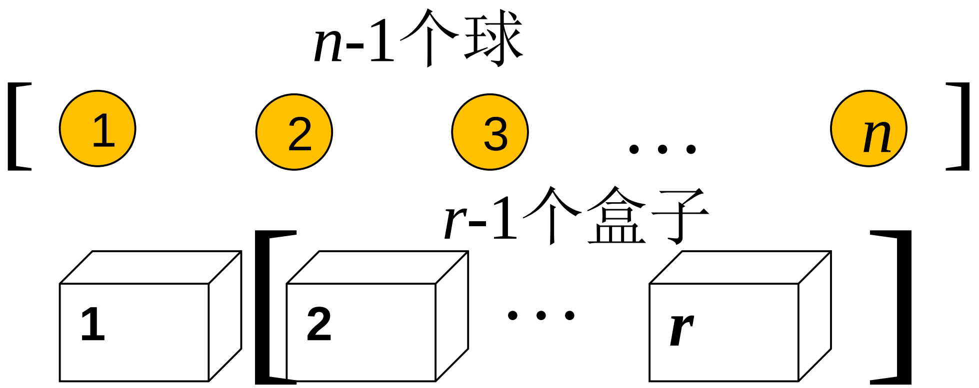
全排列：$P(n,n) = n!$

排列$P(n, r)$的递推关系

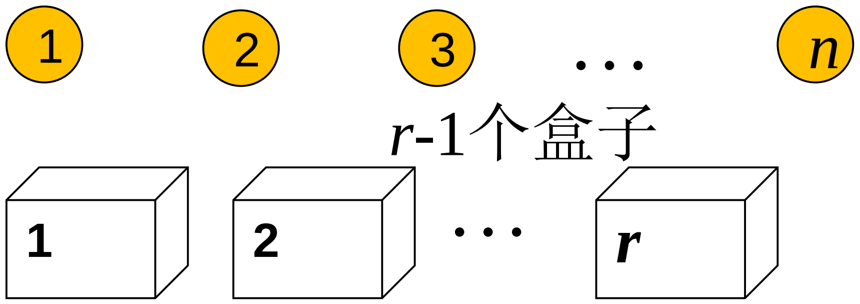
$P(n,r) = nP(n\text{-}1,r\text{-}1)$

$P(n,r) = P(n\text{-}1,r) + rP(n\text{-}1,r\text{-}1)$

- 分步递推
  - 选No.1盒子内的乒乓球
    - ***n*种选择**
  - 从n-1个球中选出r-1个放入r-1个盒子内的排列
    - **$P(n\text{-}1,r\text{-}1)$**



- 分类递推
  - 不选第一个球？
    - **$P(n\text{-}1,r)$**
  - 选择第一个球？
    - **$rP(n\text{-}1,r\text{-}1)$**



组合的模型

若球不同，盒子**相同**，则是从n个中取r个的**组合**的模型。

若放入盒子后再将盒子标号区别，则又回到排列模型。每一个组合可有$r!$个标号方案。

故有 $C(n,r)\,r!=P(n,r)=\dfrac{n!}{(n-r)!}$ $\quad \boldsymbol{C(n,r)=\dfrac{n!}{r!(n-r)!}}$

$C(n,r)=C(n,n\text{-}r)$

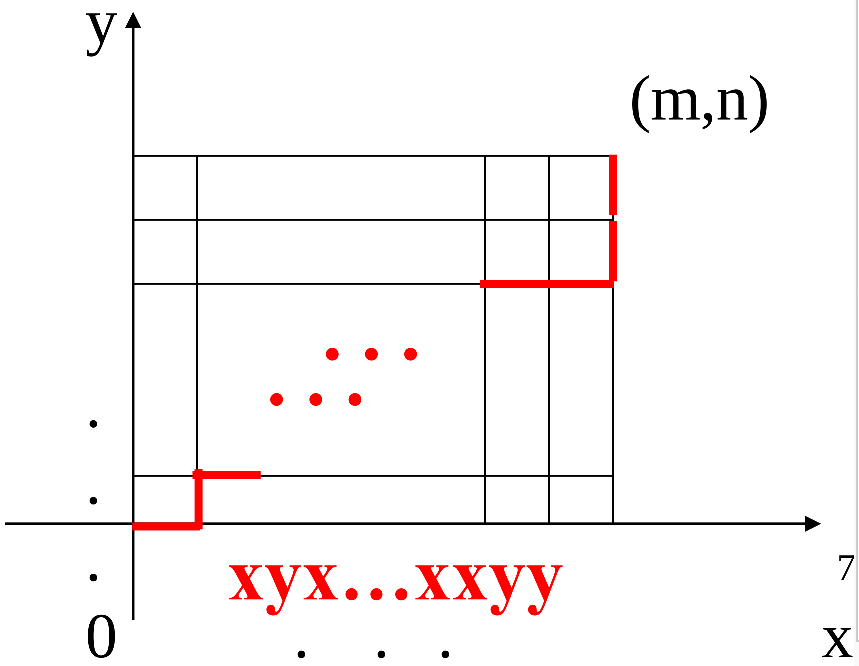
n个乒乓球中选出r个的方法自然等于剩下n-r个的方法。

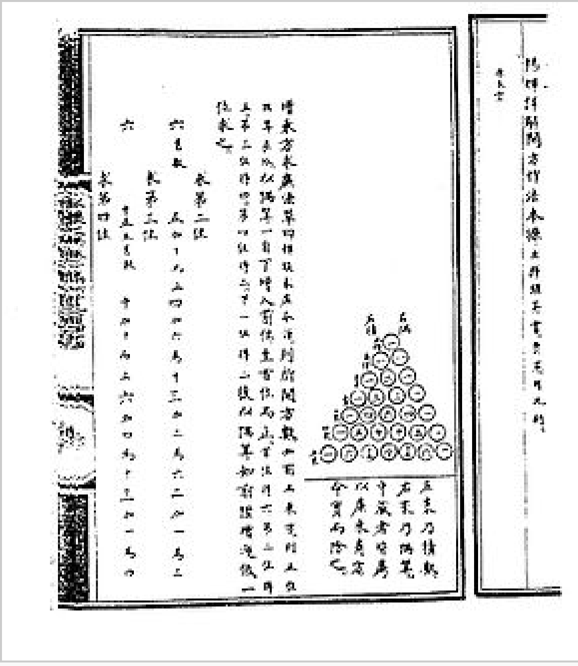
$\boldsymbol{C}(n,l)C(l,r)=C(n,r)C(n\text{-}r,l\text{-}r)$

非文科班级共n位同学，选出l位班委，班委中选出r位为核心；先从n个同学中选出r个核心，再从剩下的n-r中选剩下的l-r班委

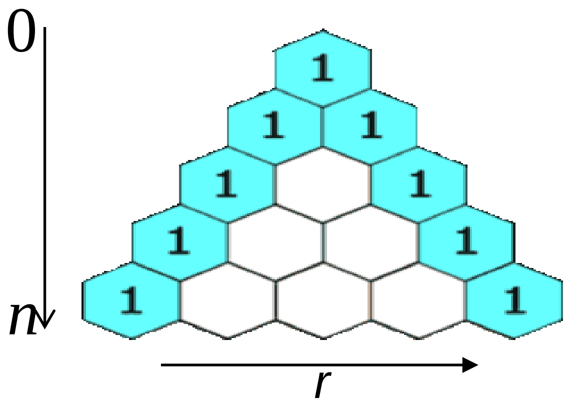
组合模型

- 格路模型：
- 从 (0,0)点出发沿x轴或y轴的正方向每步走一个单位，最终走到(m,n)点，其路径数是$C(m+n,n)$.





永乐大典一页
杨辉引用贾宪《释锁算书》中的贾宪三角形



第n行第r列的数值是$C(n,k)$
第n行对应$(a+b)^n$的系数？
组合数和多项式展开系数？

二项式定理

- $(a+b)^n = \mathrm{C}(n,0)a^n + \mathrm{C}(n,1)a^{n-1}b + \ldots + \mathrm{C}(n, n)b^n$
- (a+b)(a+b)…(a+b)　n个$(a+b)$

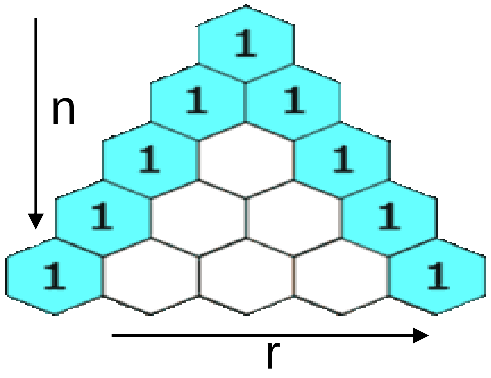
  a　　b　　…　a

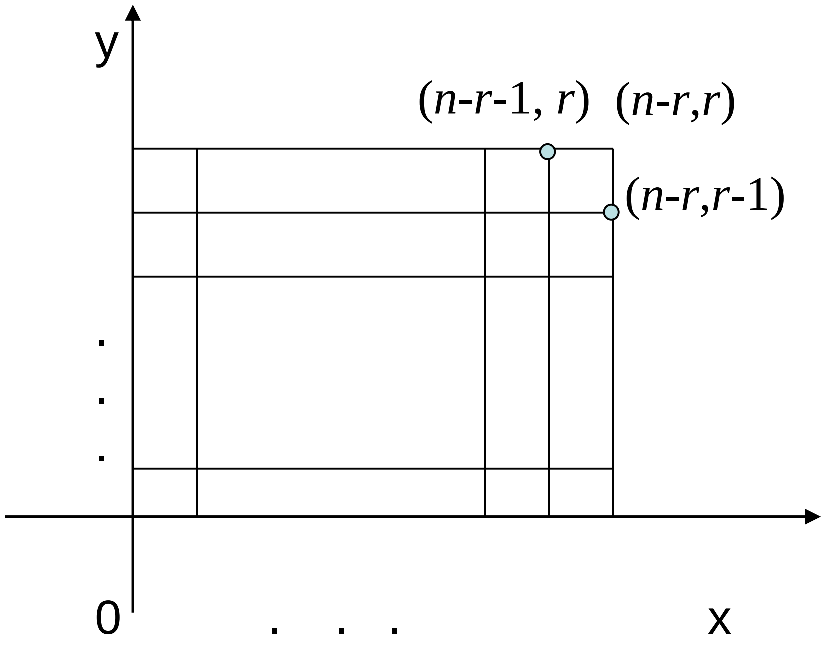
  从n个位置中选出r个b, 构成了$a^{n-r}b^r$

  其个数$\mathrm{C}(n, r)$即$a^{n-r}b^r$
- 若$a=b=1$, 则有
  - $2^n = C(n,0) + C(n,1) + \ldots + C(n, n)$
- 若$a=1$, $b=-1$, 则有
  - $0 = C(n,0) - C(n,1) + \ldots \pm C(n, n)$

组合恒等式
Combinatorial Identities

- $C(n, r)=C(n-1,r)+C(n-1,r-1)$





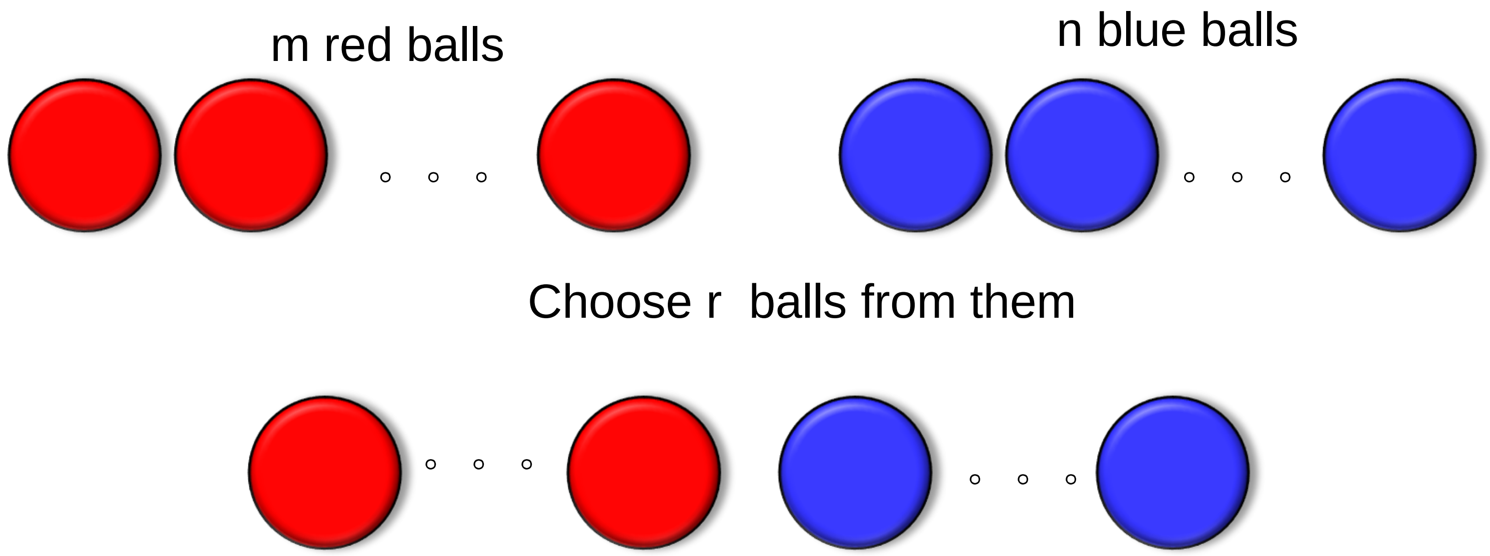
- 等式左侧: (0,0) 至 $(n\text{-}r,r)$所有格路
- 等式右侧:
  - (0,0) 至 $(n\text{-}r\text{-}1,r)$
  - (0,0) 至 $(n\text{-}r,r\text{-}1)$

恒等式

- **C**$(m+n, r)=C(m,0)C(n, r)+C(m, 1)C(n,r-1)+ \ldots +C(m, r)C(n, 0)$
- 即Vandermonde恒等式**Vandermonde's identity**
- Alexandre-Théophile Vandermonde (28 February 1735 – 1 January 1796) 法国音乐家，数学家
- Donald E. Knuth (高德納) 在The Art of Computer Programming Vol.ii (1998) 中, 讲到实际上这个恒等式的变形体早在**1303 由朱世杰在**《四元玉鉴》介绍了.
- **Chu-** Vandermonde恒等式

小球来证明恒等式

- **C**$(m+n,r)=C(m,0)C(n,r)+C(m,1)C(n,r-1)+ \ldots +C(m,r)C(n,0)$
- $C(m,0)C(n,r)+C(m,1)C(n,r-1)+\ldots+C(m,r)C(n,0)$

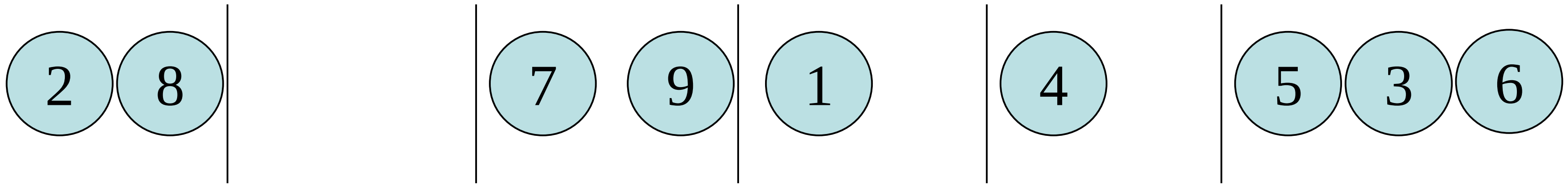


小球入洞

例　　小球入洞游戏：共有6个洞，洞口每次每次只能进入一个小球，一组编号为1-9的9个小球滚入洞口的方案有多少？

[解]一进站方案表示成：XX11 XX 1X1X1XXX

其中“X”表示某人，“1”表示门框，其中“X”是不同元，“1”是相同元。任意进站方案可表示成上面14个元素的一个排列。



XX11 XX 1X1X1XXX

[解法1]给每个方案的门标号可产生5!个14个元的全排列。故若设x为所求方案数，则

$$x\ 5!=14!$$

$$\therefore x=14!/5!=726485760$$

[解法2]在14个元的排列中先确定“1”的位置，有C(14,5)种选择，再确定球的位置，有9!种选择。

故 C(14,5) 9! 即所求

[解法3]把全部选择分解成若干步，使每步宜于计算。

- 1号有6种选择；
- 2号除可有1号的所有选择外，还可（也必须）选择当与1号同一门时在1号的前面还是后面，故2号有7种选择；
- 3号的选择方法同2号，故共有8种
- 。。。。
- 以此类推，9号有14种选择。

故所求方案为 6*7*8*....*14 =14!/5!=726485760

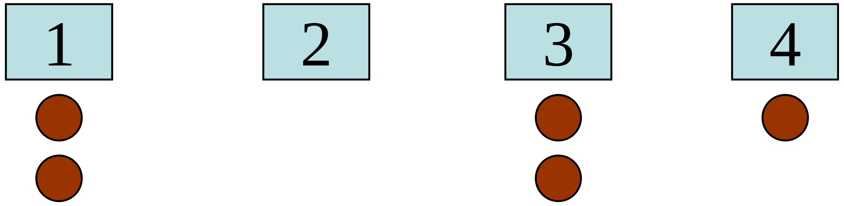
准备果篮

苹果	梨
1	3
2	2
3	1
4	0

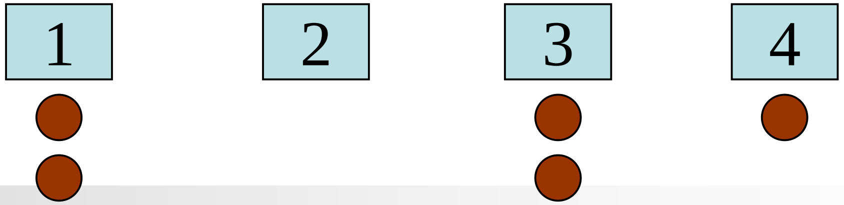
- 苹果和梨两种水果，要选4个水果做果篮
- 多重集：元素可以多次出现的集合。t_i=0, 1, …∞表示元素a_i可以出现的次数，含有n个不同元素的多重集可以记为$\{t_1 \cdot a_1, t_2 \cdot a_2 \ldots, t_n \cdot a_n\}$
- $t_i = \infty$多重集
- **可重组合：**从$A=\{1,2,3\ldots.n\}$中取r个元素$\{a_1,a_2,\ldots a_r\}$,$a_i \in A$，i=1,2,..r,且允许$a_i=a_j$, $i\neq j$，记为$\overline{C}(n,r)$。

可重组合

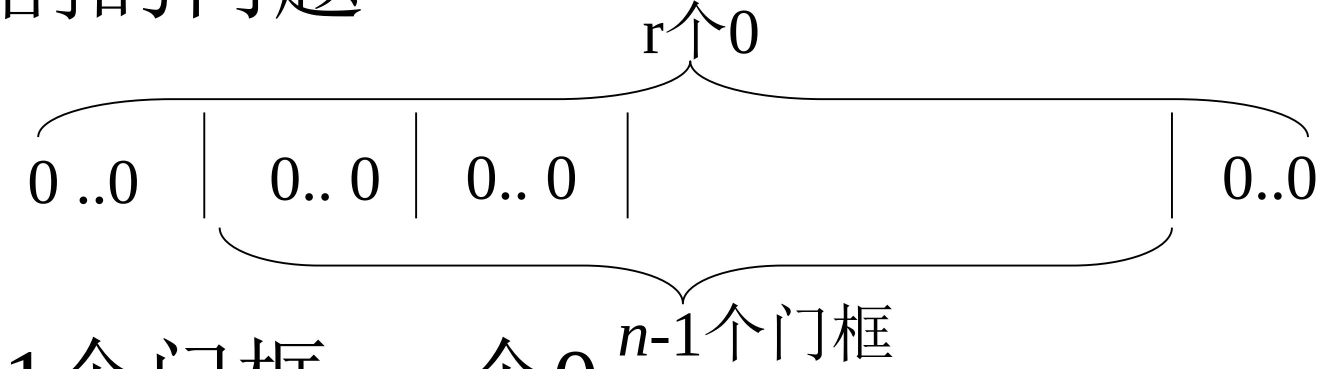
- A={1,2,3,4}中取5个元素构成组合，元素可以重复
- **可重组合：**
- {1 1 3 3 4}



- 可重组合模型：取r个**无标志**的球，n个**有区别**的盒子，每个盒子**允许放多于一个球，或者空盒**。
- 无重组合的模型：n个球是**有区别**的，r个盒子是**无区别**的，取r个球放入盒子，每个盒子**一个球**。



- <u>**在*n*个不同的元素中取*r*个进行组合，若允许重复的组合数为C(*n*+*r*-1,*r*)**</u>
- **证明：**可以转化为门框分割的问题



- 总共n+r-1个元素，其中n-1个门框，r个0
- 若门框和0都分别标号的话，构成n+r-1个元素的全排列
- 故所求组合数为 $\dfrac{(n+r-1)!}{r!(n-1)!} = C(n+r-1, r)$

线性方程整数解

- 线性方程$x_1+x_2\ldots+x_n=b$的非负整数解的个数是C(n+b-1,b)

$$\overbrace{\underset{x_1\text{个}1}{1\ ..1}\ \underbrace{\Big|\ 1..\ 1\ \Big|\ 1..\ 1\ \Big|\qquad\qquad\qquad\Big|}_{\text{n-1个门框}}\ 1..1}^{\text{b个1}}$$

- 其方程解的个数是C(n+b-1,b)

Type	Sample	Order counts?	Repetition allowed?	Number of ways
无重组合	从n个球中取r个	No	No	$C(n, r)$
无重排列	从n个人中找r个排队	Yes	No	$P(n, r)$
可重组合	从n种水果中选r个拼果篮	No	Yes	$C(n+r-1, r)$
可重排列	n个字母组成的r位串	Yes	Yes	n^r
多重全排列	r_1个a, r_2个b组成的n位串	Yes	Yes	$n!/(r_1!\ r_2!)$

放球问题

n个球放到m个盒子里，依球和盒子是否有区别？是否允许空盒？共有 $2^3=8$种状态。。

- n个球有区别，m个盒子有区别，有空盒 m^n
- n个球有区别，m个盒子有区别，无空盒
- n个球有区别，m个盒子无区别，有空盒
- n个球有区别，m个盒子无区别，无空盒
- n个球无区别，m个盒子有区别，有空盒
- n个球无区别，m个盒子有区别，无空盒
- n个球无区别，m个盒子无区别，有空盒
- n个球无区别，m个盒子无区别，无空盒

【任务1.1】下楼

从楼上走到楼下共有 h 个台阶，每一步有三种走法

- 走一个台阶；
- 走二个台阶；
- 走三个台阶。

把n个球放入若干个篮子，每个篮子可以放1-3个球

问：一共可以走出多少种方案？即共要多少步？每一步走几级台阶？

母函数和计数法则

且：乘法法则
或：加法法则

- 例：两个色子掷出6点，有多少种可能？

$$\boxed{} + \boxed{} = 6$$

- 解法1：第一位数 **1-5**，且第二位数由第一位来确定　　　　$5 \times 1 = 5$
- 解法2：1+5 = 5+1；或 2+4 = 4+2；或 3+3　　　　$2 + 2 + 1 = 5$

雅各布 伯努利
瑞士数学家1654年－1705年

- **投掷*m*粒色子时，加起来点数总和等于*n*的可能方式的数目？**

母函数和计数法则

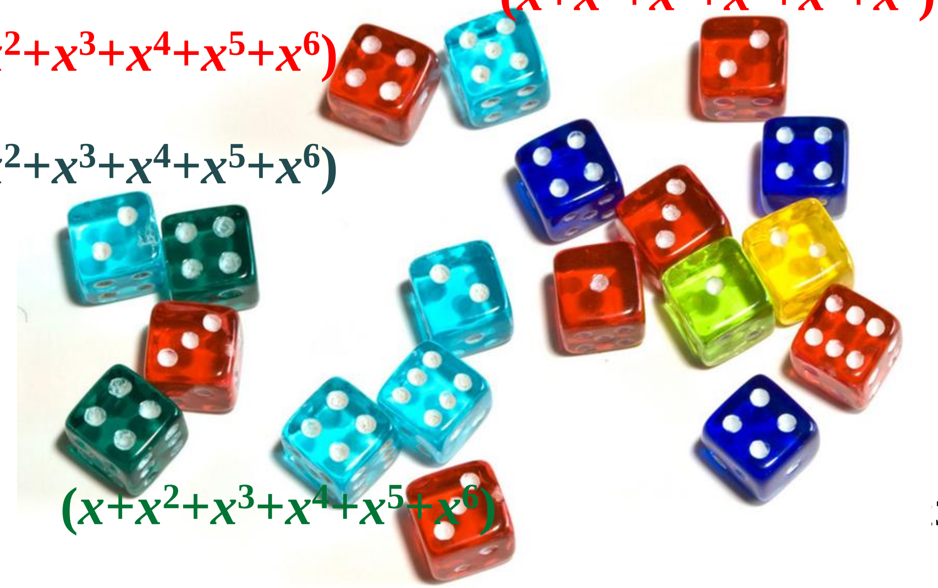
母函数是母亲，计数序列是孩子。

雅各布 ·伯努利
Jakob I. Bernoulli
瑞士数学家1654年－1705年

- **投掷m粒色子时，加起来点数总和等于n的可能方式的数目？**

$$G(x)=(x+x^2+x^3+x^4+x^5+x^6)^m$$

展开式中x^n项的系数



母函数和计数法则

且：乘法法则
或：加法法则

- 例：两个色子掷出n点，有多少种可能？

$$\square + \square = n$$

○ x○ (○)^2 x^2  $x^6 \leftarrow x^2$ ×指数对应点数 x^4  x^4

(⚀或⚁或⚂或⚃或⚄或⚅)(⚀或⚁或⚂或⚃或⚄或⚅)

$$(x + x^2 + x^3 + x^4 + x^5 + x^6)\times(x + x^2 + x^3 + x^4 + x^5 + x^6)$$

x^6: $\boxed{x^1 x^5 + x^2 x^4 + x^3 x^3 + x^4 x^2 + x^5 x^1}$ $=5x^6$

x^6的系数为5，表示两个色子掷出6点的可能方法有5种。

$$G(x)=(x+x^2+\ldots+x^6)^2= x^2+2x^3+3x^4+4x^5+5x^6+6x^7+5x^8+4x^9+3x^{10}+2x^{11}+x^{12}$$

两个色子掷出n点的可能方法数即为求$G(x)=(x+x^2+\ldots+x^6)^2$中$x^n$的系数。

函数中的系数对应计数序列。

【任务1.1】下楼

从楼上走到楼下共有 h 个台阶，每一步有三种走法

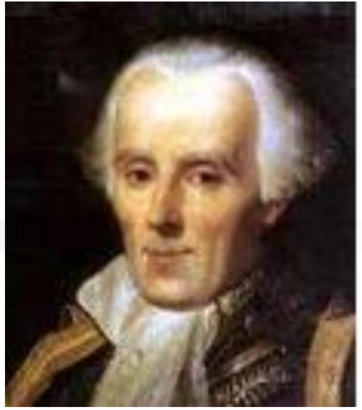
- 走一个台阶；
- 走二个台阶；
- 走三个台阶。

$G(x)=(x+x^2+x^3)^m$

展开式中x^n项的系数

问：一共可以走出多少种方案？即共要多少步？每一步走几级台阶？

母函数和计数法则



拉普拉斯

- **定义2-1** 对于计数序列$c_0, c_1, c_2, \ldots$,

$$G(x) = c_0 + c_1 x + c_2 x^2 + \ldots\ldots$$

函数$G(x)$是序列$c_0, c_1, c_2 \ldots$ 的母函数。

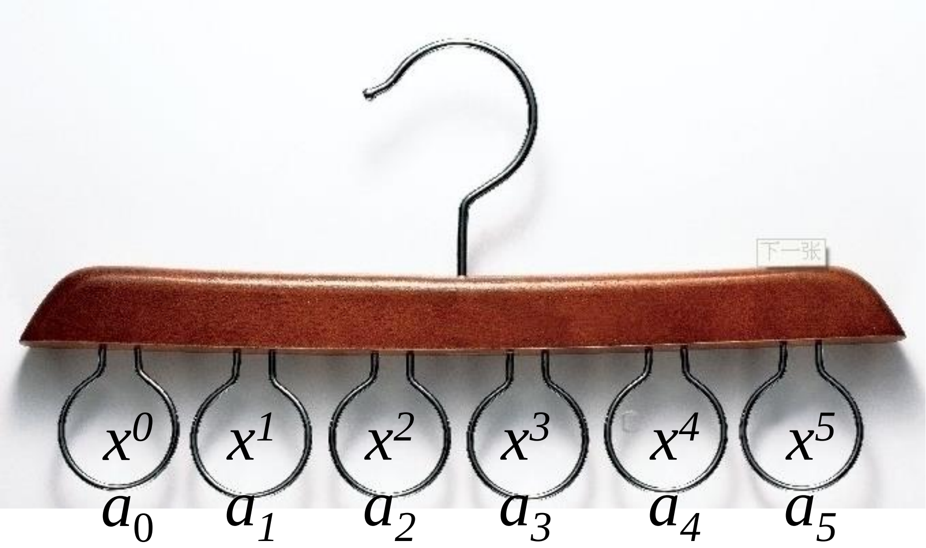
- 1812年，法国数学家拉普拉斯在著作《概率的分析理论》的第一卷中系统地研究了母函数方法及与之有关的理论
  - 计数工具
  - 不考虑收敛性
  - 不考虑实际上的数值
  - 形式幂级数(Formal power series)

母函数就是一列用来展示一串数字序列的挂衣架。

—— 赫伯特·维尔夫

$$G(x)=x^2+2x^3+3x^4+4x^5+5x^6+6x^7+5x^8+4x^9+3x^{10}+2x^{11}+x^{12}$$

函数: $\boxed{f(x)}=\sum_{n=0}^{\infty}a_n\boxed{x^n}$ $\qquad G(x)=\sum_{n=0}^{\infty}\boxed{a_n}x^n$



整数的拆分

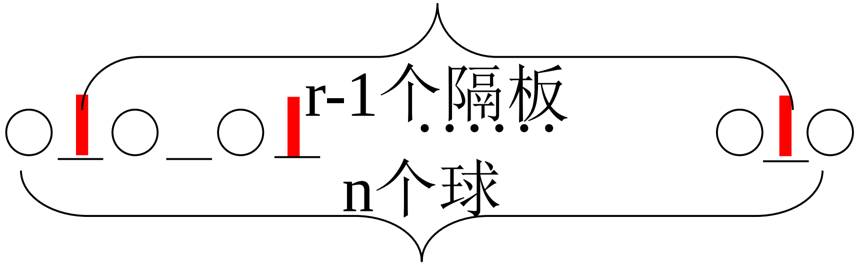
所谓自然数(正整数)分拆,就是将一个正整数表达成若干个正整数之和:
各部分之间考虑顺序的叫有序分拆(Composition) ;
否则叫无序分拆(Partition).

3的有序2-拆分：3=2+1=1+2

n的有序r−拆分的个数是C(n-1,r-1)

n个球，要分成r份，

用r−1个隔板插入到球之间的n−1个空隙，方案数C(n−1, r−1)



n-1个空隙

放球模型：　n的一个r-分拆相当于把n个无区别的球放到r个有标志的盒子，盒子不允许空着

有序拆分的放球模型：n的一个r-分拆相当于把n个无区别的球放到r个**有标志**的盒子，盒子不允许空着

- <u>无序分拆</u>
- 3的无序2-拆分：3=2+1
- 3的所有无序拆分3=3+0+0=2+1+0=1+1+1
- $x_1+x_2+\ldots+x_r=n$的非负整数解个数？$C(n+r-1,n)$

n个1

1 ..1 | 1.. 1 | 1.. 1 | | 1..1

x_1个1

r-1个门框

相当于把n个无区别的球放到r个**有标志**的盒子，盒子允许空着

0+3+0

3+0+0

与无序拆分不同

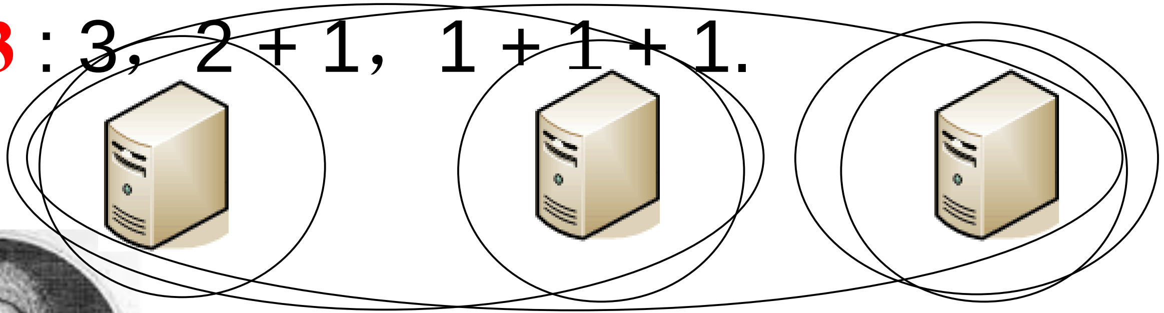
无序拆分

n个相同的球，
分成若干堆，
有几种分法？

所谓整数拆分(**partition** of a positive integer *n*)即把整数分解成若干整数的和，相当于把n个无区别的球放到n个无标志的盒子，盒子允许空着，也允许放多于一个球。整数拆分成若干整数的和，办法不一，不同拆分法的总数叫做拆分数。

母函数的应用：整数拆分数

- **正整数的无序拆分**：将一个正整数n拆分成若干正整数的和，数字之间顺序无关并允许重复，其不同的拆分数即$p(n)$。
  - 密码学，统计学，生物学……
- **$p(3)=3$**：3，2 + 1，1 + 1 + 1.



诺地

1740年诺地在信中问欧拉：

整数的拆分方案数？

指数对应点数

欧拉

$G(x) = (1+x+x^2+\ldots)(1+x^2+x^4+\ldots)\ldots(1+x^m+x^{2m}+\ldots)\ldots$ 中x^n的系数

“1”的母函数　“2”的母函数　“m”的母函数

Weil A．Number Theory：An Approach Thought History [M]．Boston：Birkhouser，1984．284

放球问题总结

n个球放到m个盒子里，依球和盒子是否有区别？

是否允许空盒？共有 $2^3=8$ 种状态。。

- n个球有区别，m个盒子有区别，有空盒 m^n
- n个球有区别，m个盒子有区别，无空盒
- n个球有区别，m个盒子无区别，有空盒
- n个球有区别，m个盒子无区别，无空盒
- n个球无区别，m个盒子有区别，有空盒 $C(n+m-1,n)$
- n个球无区别，m个盒子有区别，无空盒 $C(n-1,n-m)=C(n-1,m-1)$
- n个球无区别，m个盒子无区别，有空盒 $G(x)=\dfrac{1}{(1-x)(1-x^2)\cdots(1-x^m)}$ 的 x^n 项系数。
- n个球无区别，m个盒子无区别，无空盒 $G(x)=\dfrac{x^m}{(1-x)(1-x^2)\cdots(1-x^m)}$ 的 x^n 项系数。

- n个球无区别，m个盒子无区别，无空盒 $G(x)=\dfrac{x^m}{(1-x)(1-x^2)\cdots(1-x^m)}$ 的x^n 项系数。

无空盒相当于每个盒子先投一个，剩下的**n-m**做有空盒的放球。

James Stirling(1692-1770)

- Stirling's approximation (Stirling估计式)
- Stirling Permutation

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

  - 多重集 (1, 1, 2, 2,…,k, k)的排列
  - 要求两个重复数字之间的数字都要大于该数
  - (1, 1, 2, 2)
  - 1221, 1122, 2211

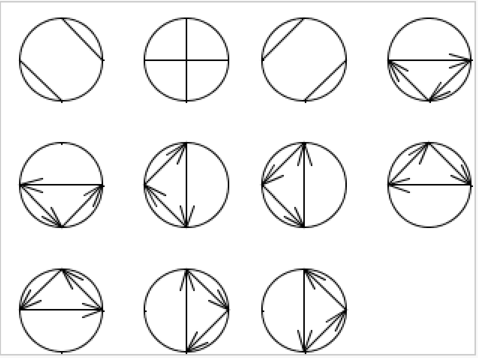
Stirling Numbers

[Stirling numbers of the first kind](#) 第一类Stirling数

[Stirling numbers of the second kind](#) 第二类Stirling数

第一类Stirling数

- n个人跳集体舞，分成m个圆环的方法数目。
  - *A*, *B*, *C*, *D*四个人跳舞，组成2个圆排列的方法
    - {A,B},{C,D}　{A,C},{B,D}　{A,D},{B,C}
    - {A},{B,C,D}　{A},{B,D,C}　{B},{A,C,D}　{B},{A,D,C}　{C},{A,B,D}　{C},{A,D,B}　{D},{A,B,C}　{D},{A,C,B}
  - 第一类Stirling数$s(n,k)$
  - $s(4,2)=11$



s(4,2)=11

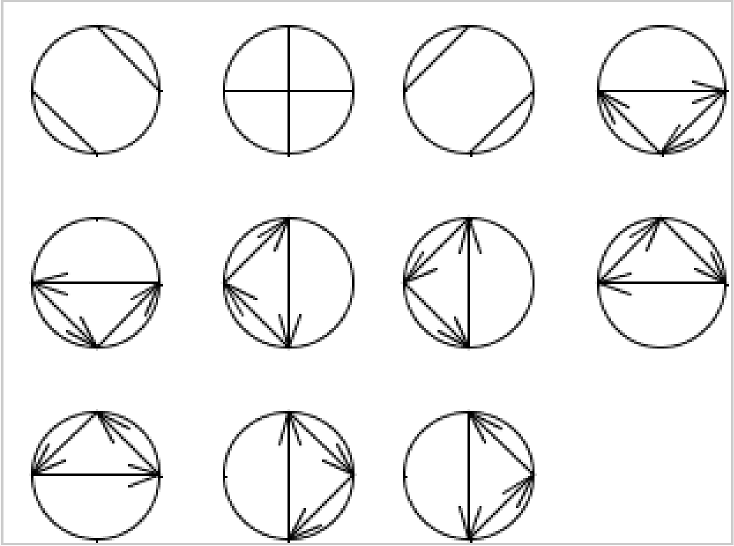
第一类Stirling数

- 第一类Stirling数$s(n,m)$
  - $s(n,0)=0$, $s(1,1)=1$
  - **$s(n+1,m)=s(n, m-1)+n\ s(n, m)$**
    - 第n+1个人可以单独自己跳舞，其他n个人构成m-1个圈
    - 第n+1个人加入到别的队伍中，可以选择第i个的左边，所以有n个不同的位置，而其他n个人有$s(n, m)$种不同的组圈方法

$s(n,k)$和$S(n,k)$

- $s(n,k)$表示把n个人分成k组，每组内再按特定顺序围圈。

1. {A, B}, {C, D}
2. {A, C}, {B, D}
3. {A, D}, {B, C}
4. {A}, {B, C, D}
5. {A}, {B, D, C}
6. {B}, {A, C, D}
7. {B}, {A, D, C}
8. {C}, {A, B, D}
9. {C}, {A, D, B}
10. {D}, {A, B, C}
11. {D}, {A, C, B}



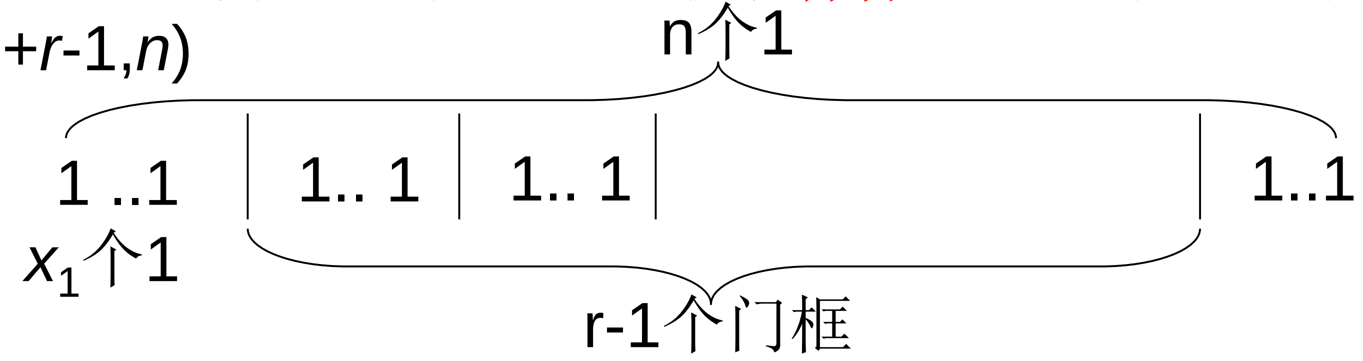
s(4, 2)=11

- $S(n,k)$表示把n个不同的球放入k个相同的盒子

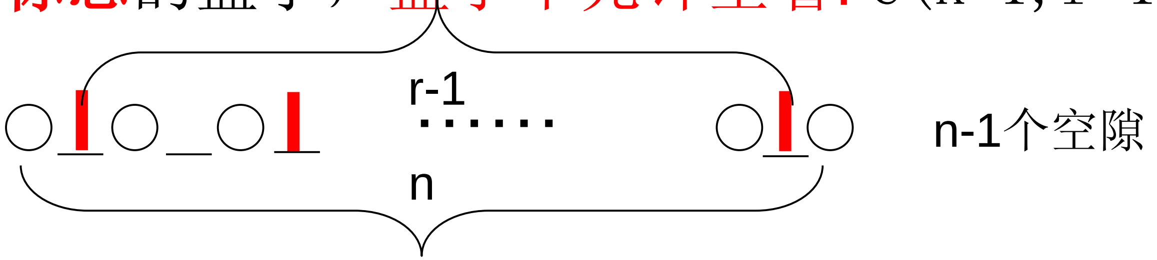
1. {A,B},{C,D} 2. {A,C},{B,D} 3. {A,D},{B,C}
4.{A},{B,C,D} 5. {B},{A,C,D} 6.{C},{A,B,D}
7.{D},{A,B,C}

n个有区别的球放到m个相同的盒子中,要求无一空盒?

- 相当于把n个无区别的球放到r个**有标志**的盒子，盒子允许空着: C(*n*+*r*-1,*n*)



- 有序拆分的放球模型： n的一个r-分拆相当于把n个无区别的球放到r个**有标志**的盒子，盒子不允许空着: C(k−1, r−1)



- 无序拆分的放球模型：相当于把n个无区别的球放到n个无标志的盒子，盒子允许空着，也允许放多于一个球。

$$G(x)=\frac{1}{(1-x)(1-x^2)\cdots(1-x^m)}$$ 的x^n项系数。

- **例 第二类Stirling数的展开式** $S(n,m)=\frac{1}{m!}\sum_{k=1}^{m}C(m,k)(-1)^k(m-k)^n$
- $S(n,m)$的组合意义: 将n个有标志的球放入m个无区别的盒子,而且无一空盒的方案数.
- 先考虑n个有标志的球,放入m个有区别的盒子,无一空盒的方案数.

解: n个有标志的球放入m个有区别的盒子的事件全体为S, $|S|=m^n$

A_i表示第i个盒子为空, $i=1,2\ldots m$;

$|A_i| = (m-1)^n$　　共有 C(m,1)个

$|A_i \cap A_j| = (m-2)^n$　　共有 C(m,2)个

…………………

求无空盒的方案数

§ 3. 7 容斥原理应用举例

m个有区别盒子,无空盒的方案数:

$$N=|\overline{A1}\cap\overline{A2}\ldots\ldots\cap\overline{An}|$$

$$=m^n-C(m,1)(m-1)^n+C(m,2)(m-2)^n+\ldots+(-1)^m C(m,m)(m-m)^n$$

$$=\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^n$$

而第二类Stirling数要求盒子无区别,则:

$$S(n,m)=\frac{1}{m!}\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^n$$

推论: 因为S(m,m) = 1,

$$m!=\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^m$$

放球问题总结

n个球放到m个盒子里，依球和盒子是否有区别？是否允许空盒？共有 $2^3=8$种状态。。

- n个球有区别，m个盒子有区别，有空盒　m^n
- n个球有区别，m个盒子有区别，无空盒　$m!S(n,m)$
- n个球有区别，m个盒子无区别，有空盒　$S(n,1)+S(n,2)+\cdots+S(n,m),\ n\ge m$；$S(n,1)+S(n,2)+\cdots+S(n,n),\ n\le m$
- n个球有区别，m个盒子无区别，无空盒　$S(n,m)$
- n个球无区别，m个盒子有区别，有空盒　$C(n+m-1,n)$
- n个球无区别，m个盒子有区别，无空盒　$C(n-1,n-m)=C(n-1,m-1)$
- n个球无区别，m个盒子无区别，有空盒　$G(x)=\dfrac{1}{(1-x)(1-x^2)\cdots(1-x^m)}$ 的 x^n项系数。
- n个球无区别，m个盒子无区别，无空盒　$G(x)=\dfrac{x^m}{(1-x)(1-x^2)\cdots(1-x^m)}$ 的 x^n项系数。

程序求解组合数

苏 凯

组合数

- 组合数的定义：

从n个不同的物品中选出m个有C_n^m种方法

- 用程序计算组合数C_n^m？

递推公式：$C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$

通项公式：$C_n^m = \frac{n!}{m!(n-m)!}$

基于递推公式

- $C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$
- 一个比较基础的思路：定义一个函数C_n^m，返回$C_{n-1}^{m-1} + C_{n-1}^m$

```
int C(int n,int m){
    return C(n-1,m-1)+C(n-1,m);
}
int main(){
    cout<<C(5,2)<<endl;
}
```

Stack Overflow

基于递推公式

- 打表

C	m=0	m=1	m=2	m=3	m=4	m=5	m=6	m=7
n=0	1							
n=1	1	1						
n=2	1	2	1					
n=3	1	3	3	1				
n=4	1	4	6	4	1			
n=5	1	5	10	10	5	1		
n=6	1	6	15	20	15	6	1	
n=7	1	7	21	35	35	21	7	1

```
if(n<m){
  return 0;
}
```

m=0

```
if(m==0){
  return 1;
}
```

m=n

基于递推公式

- $C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$
- 考虑C_n^m的组合意义，就能在极端情况（如n=0,m=0,n<m等）下保证答案正确。

```
int C(int n, int m) {
  if (n < m) {
    return 0;
  }
  if (m == 0) {
    return 1;
  }
  return C(n - 1, m - 1) + C(n - 1, m);
}
```

基于递推公式

```cpp
#include <iostream>
using namespace std;
int C(int n, int m) {
  if (n < m) {
    return 0;
  }
  if (m == 0) {
    return 1;
  }
  return C(n - 1, m - 1) + C(n - 1, m);
}
int main()
{
  cout << C(5, 2) << endl;
}
```

- 计算C_5^2
- 输出：10
- 这个数量级太小了，我能手算！

基于递推公式

```
#include <iostream>
using namespace std;
int C(int n, int m) {
  if (n < m) {
    return 0;
  }
  if (m == 0) {
    return 1;
  }
  return C(n - 1, m - 1) + C(n - 1, m);
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- 计算C_{30}^{15}
- 输出：155117520
- 运行得很慢？
- n每多1，调用C的次数就多一倍
- 个人计算机1秒只能进行10^9量级的计算
- $2^{31} = 2147483648 \approx 2 \times 10^9$

基于递推公式

```cpp
#include <iostream>
using namespace std;
int C(int n, int m) {
  if (n < m) {
    return 0;
  }
  if (m == 0) {
    return 1;
  }
  return C(n - 1, m - 1) + C(n - 1, m);
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- 慢在哪？
- 重复计算很多！
- 函数C的返回值只与n, m有关

计算过的值可以保存下来，直接返回

基于递推公式

```cpp
#include <iostream>
using namespace std;
bool visited[31][31];
int a[31][31];
int C(int n, int m)
{
  if (visited[n][m])
    return a[n][m];
  int ans;
  if (n < m)
    ans = 0;
  else if (m == 0)
    ans = 1;
  else
    ans = C(n - 1, m - 1) + C(n - 1, m);
  visited[n][m] = true;
  a[n][m] = ans;
  return ans;
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- bool数组visited[n][m]表示C(n,m)是否计算过
- int数组a[n][m]表示C_n^m如果计算过，它的计算结果是多少
- 我们只需要计算n≤30,m≤15的组合数，数组下标从0开始，所以大小应设为[31][31]
- 如果C_n^m计算过了，可以直接得到结果
- 不要忘记在这次计算之后保存一下结果

这种储存搜索计算结果的方法叫“记忆化搜索”

基于递推公式

```
#include <iostream>
using namespace std;
bool visited[31][31];
int a[31][31];
int C(int n, int m)
{
  if (visited[n][m])
    return a[n][m];
  int ans;
  if (n < m)
    ans = 0;
  else if (m == 0)
    ans = 1;
  else
    ans = C(n - 1, m - 1) + C(n - 1, m);
  visited[n][m] = true;
  a[n][m] = ans;
  return ans;
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- 现在C_{30}^{15}一下子就跑出来了
- 计算的实际是a数组

能不能直接把a数组计算出来？

基于递推公式

- 再看一遍

C	m=0	m=1	m=2	m=3	m=4	m=5	m=6	m=7
n=0	1							
n=1	1	1						
n=2	1	2	1					
n=3	1	3	3	1				
n=4	1	4	6	4	1			
n=5	1	5	10	10	5	1		
n=6	1	6	15	20	15	6	1	
n=7	1	7	21	35	35	21	7	1

基于递推公式

- 倒着来似乎更顺一点

C	m=0	m=1	m=2	m=3	m=4	m=5	m=6	m=7
n=0	1							
n=1	1	1						
n=2	1	2	1					
n=3	1	3	3	1				
n=4	1	4	6	4	1			
n=5	1	5	10	10	5	1		
n=6	1	6	15	20	15	6	1	
n=7	1	7	21	35	35	21	7	1

杨辉三角！

基于递推公式

```cpp
#include <iostream>
using namespace std;
int a[31][31];
int C(int n, int m)
{
  for(int i=0;i<=n;i++){
    a[i][0]=1;
    for(int j=1;j<=i;j++){
      a[i][j]=a[i-1][j-1]+a[i-1][j];
    }
  }
  return a[n][m];
}
int main()
{
  cout << C(30,15) << endl;
}
```

- 明确递推顺序：先算n,m较小的位置
- 回顾边界情况
  - n<m:在大多数C++编译器中，全局数组会默认初始化为0，所以没有遍历到的位置自动为0
  - m=0:需要手动设置

基于递推公式

- 对于a[i][*]，好像只依赖a[i-1][*]

C	m=0	m=1	m=2	m=3	m=4	m=5	m=6	m=7
n=0	1							
n=1	1	1						
n=2	1	2	1					
n=3	1	3	3	1				
n=4	1	4	6	4	1			
n=5	1	5	10	10	5	1		
n=6	1	6	15	20	15	6	1	
n=7	1	7	21	35	35	21	7	1

只要维护红框内的信息就可以了！

基于递推公式

```
#include <iostream>
using namespace std;
int b[2][31];
int C(int n, int m)
{
  bool cur = 1, last = 0;
  for (int i = 0; i <= n; i++)
  {
    swap(cur, last);
    b[cur][0] = 1;
    for (int j = 1; j <= i; j++)
    {
      b[cur][j] = b[last][j - 1] + b[last][j];
    }
  }
  return b[cur][m];
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- b[cur]代表当前行a[i]
- b[last]代表上一行a[i-1]
- 进行下一行计算的时候，b[cur]成了上一行
- 现在的b[last]就没有价值了，新的一行计算结果可以直接覆盖保存在上面
- 这样只要交换cur和last即可

这种实现方式被形象地称作“滚动数组”

基于递推公式

```
#include <iostream>
using namespace std;
int b[2][31];
int C(int n, int m)
{
  bool cur = 1, last = 0;
  for (int i = 0; i <= n; i++)
  {
    swap(cur, last);
    b[cur][0] = 1;
    for (int j = 1; j <= i; j++)
    {
      b[cur][j] = b[last][j - 1] + b[last][j];
    }
  }
  return b[cur][m];
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- 本来需要大小为31×31=961的数组
- 现在只需要2×31=62
- 空间复杂度从$O(n^2)$降到了$O(n)$

基于递推公式

把[2]省掉？

	m=0	m=1	m=2	m=3	m=4	m=5	m=6	m=7
n=6	1	6	15	35	15	6	1	

这里怎么办？

基于递推公式

从后面开始倒回来算就可以了！

	m=0	m=1	m=2	m=3	m=4	m=5	m=6	m=7
n=6	1	7	21	35	35	21	7	1

基于递推公式

```
#include <iostream>
using namespace std;
int b[31];
int C(int n, int m)
{
  b[0] = 1;
  for (int i = 1; i <= n; i++)
  {
    for (int j = i; j >= 1; j--)
    {
      b[j] = b[j - 1] + b[j];
    }
  }
  return b[m];
}
int main()
{
  cout << C(30, 15) << endl;
}
```

- j从大到小枚举，保证在覆盖上一层的值之前就把当前层的值算出来

基于递推公式

```cpp
#include <iostream>
using namespace std;
int b[31];
int S(int n, int m)
{
  b[0] = 1;
  for (int i = 1; i <= n; i++)
  {
    for (int j = i; j >= 1; j--)
    {
      b[j] = b[j] * j + b[j - 1];
    }
    b[0] = 0;
  }
  return b[m];
}
int main()
{
  cout << S(20, 15) << endl;
}
```

- 以上做法还能解决类似形式的递推
- E.g. 第一类斯特林数、第二类斯特林数
- 以第二类斯特林数为例
- S(n,k)=k*S(n-1,k)+S(n-1,k-1)
- 边界情况：
  - S(0,0)=1
  - S(n,0)=0
  - S(n,n+1)=0

基于……

- 利用递推公式我们可以得到一个$O(n^2)$的算法，即计算量大概是n^2级别的
- 1s大约能运行$n \le 5000$的$O(n^2)$算法
- 想要更快的速度，必须利用通项公式

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

基于通项公式

```
int main()
{
  //计算n!
  int n=10;
  int ans=1;
  for(int i=1;i<=n;i++){
    ans*=i;
  }
}
```

- $C_n^m = \frac{n!}{m!(n-m)!}$
- 只需计算n!、m!、(n-m)!
- $n! = 1 \times 2 \times 3 \times 4 \times \cdots \times n$
- 可以直接循环计算

基于通项公式

```
int fac[11];
int main()
{
  int n=10;
  fac[0]=1;
  for(int i=1;i<=n;i++){
    fac[i]=i*fac[i-1];
  }
}
```

- 只需计算n!、m!、(n-m)!
- 考虑递推算出fac[n]=n!=n*(n-1)!
- 约定0!=1，这样1=1!=1*0! 才成立
- 和刚才直接循环计算的区别在于保存了中间结果

基于通项公式

```
#include <iostream>
using namespace std;
int fac[11];
inline int C(int n,int m){
  return fac[n]/fac[m]/fac[n-m];
}
int main()
{
  int n=10;
  fac[0]=1;
  for(int i=1;i<=n;i++){
    fac[i]=i*fac[i-1];
  }
  cout<<C(5,3)<<endl;
  cout<<C(7,2)<<endl;
}
```

- 和刚才直接循环计算的区别在于保存了中间结果
- 结合通项公式$C_n^m = \frac{n!}{m!(n-m)!}$
- 可以O(1)快速算出组合数

基于通项公式

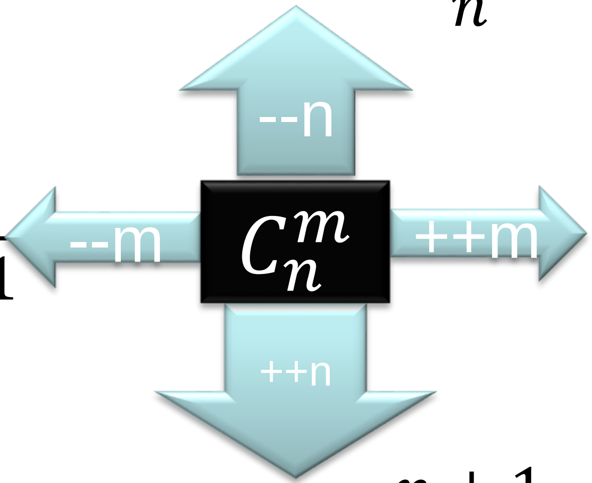
- $C_n^m = \frac{n!}{m!(n-m)!}$
- 手算C_{10}^3
- $\frac{10\times9\times8}{3\times2\times1}$
- 不需要把10的阶乘完整地算出来！
- $C_n^m = \frac{n(n-1)\cdots(n-m+1)}{m(m-1)\cdots\times2\times1}$

基于通项公式

- 如果知道了C(n,m)的值，可以立刻得到相邻项的值
- 代入通项公式立即得证
- 我们还知道$C_n^0 = C_n^n = 1$

$$C_{n-1}^{m} = C_n^m \cdot \frac{n-m}{n}$$

$$C_n^{m-1} = C_n^m \cdot \frac{m}{n-\mathrm{m}+1}$$



$$C_n^{m+1} = C_n^m \cdot \frac{n-m}{\mathrm{m}+1}$$

$$C_{n+1}^{m} = C_n^m \cdot \frac{n+1}{n-m+1}$$

基于通项公式

```
#include <iostream>
using namespace std;
int C(int n,int m){
  int ans=1;
  for(int i=1;i<=m;i++){
    ans=ans*(n-i+1)/i;
  }
  return ans;
}
int main(){
  cout<<C(10,3)<<endl;
}
```

- $C_n^m = C_n^{m\text{-}1} \cdot \frac{n-m+1}{\mathrm{m}}$
- 在m足够小的时候，即使n很大，也能快速算出组合数的值

简单比较

	基于递推公式	基于通项公式
公式	$C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$	$C_n^m = \frac{n!}{m!\,(n-m)!}$
计算时间效率	$O(n^2)$或$O(nm)$	$O(n)$或$O(m)$
运行内存空间	$O(n^2)$或$O(n)$或$O(m)$	$O(n)$或$O(m)$或$O(1)$
运算类型	只有加法	有乘法和除法

组合数

- 实战：使用递推公式计算C(60,30)的结果

```
[redacted] ~
$ ./test.exe
-1515254800
```

- 大部分平台的int存储的范围为$[-2^{31},2^{31}-1]$，整数越界
- 即使用long long代替int，最多也只能计算n≤66的组合数

组合数

- 如何选择最合适的办法？
  - 我想要一个精确解→**用比long long更强的类型储存下这个大整数：高精度**
  - 对于一个具体问题，我已得到答案，只想检验答案对不对→**将答案取模，如保留后9位即为对模10^9取模**

    身份证号码也使用了取模来校验
  - 我只需知道大概的数值，用于概率估算→**利用浮点数仅保留前若干位，进行数值估计**

取模校验

- 求C_{1000}^{500}，输出后九位即可。
- 即模10^9意义下的值

a%p为a除以p的余数
模可加：$(a\%p + b\%p)\%p = (a + b)\%p$
模可乘：$((a\%p) \cdot (b\%p))\%p = (a \cdot b)\%p$

```
for(int i=0;i<=n;i++){
    c[i][0]=1;
    for(int j=1;j<=i;j++){
      c[i][j]=(c[i-1][j-1]+c[i-1][j])%p;
    }
}
```

加入了红色字

使用基于通项公式的做法要做除法

估算组合数

- 例题：n个球中有r个红球，有b个蓝球；从中均匀随机选择100个球，问不超过40个是蓝球的概率是多少？$100 \le n \le 500$，答案误差不能超过10^{-6}。

要求不超过一半是蓝球的概率，可以先枚举恰好k个是蓝球的概率，再使用加法法则把每种情况的概率加起来。

估算组合数

- 问题转为：n个球中有b个蓝球，r个红球，摸出100个，恰好有k个是蓝球的概率是多少？
- 摸出100个球的总方案数：C_n^{100}
- 摸出恰好k个蓝球的方案数：
  - 蓝球中摸出k个的方案数：C_b^k
  - 红球中摸出100-k个的方案数：C_r^{100-k}
  - 共$C_b^k \cdot C_r^{100-k}$种
- 由于每种情况概率相等，所以恰有k个是蓝球的概率是$\frac{C_b^k \cdot C_r^{100-k}}{C_n^{100}}$

这是“超几何分布”

估算组合数

```cpp
#include <iostream>
using namespace std;
const int N=501;
int c[N][N];
int main()
{
  int n=500,r=300,b=200;//代入了一组测试数据
  for(int i=0;i<=n;i++){
    c[i][0]=1;
    for(int j=1;j<=i;j++){
      c[i][j]=c[i-1][j-1]+c[i-1][j];
    }
  }
  double ans=0;
  for(int k=0;k<=40;k++){
    ans+=double(c[b][k]*c[r][100-k])/c[n][100];
  }
  cout<<ans<<endl;
}
```

- 将刚才的思路实现，可以预料会遇到整数越界问题
- 我们需要求组合数，但是范围太大
- 我们只需要答案的前6位，不需要进行准确计算

估算组合数

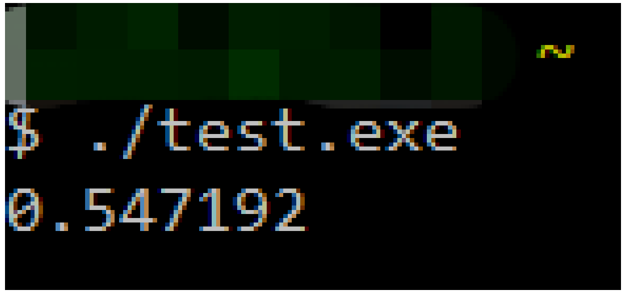
- 浮点运算（IEEE 754）使用二进制科学计数法存储和计算小数
- $3.5*10^3 = (1.10110101100)_2*2^{11}$
- 对于比较大的数，浮点数只保留前面若干位有效数字
- 可以利用浮点数进行估算

float和double都属于浮点数类型

估算组合数

```cpp
#include <iostream>
using namespace std;
const int N=501;
double c[N][N];
int main()
{
  int n=500,r=300,b=200;//代入了一组测试数据
  for(int i=0;i<=n;i++){
    c[i][0]=1;
    for(int j=1;j<=i;j++){
      c[i][j]=c[i-1][j-1]+c[i-1][j];
    }
  }
  double ans=0;
  for(int k=0;k<=40;k++){
    ans+=c[b][k]*c[r][100-k]/c[n][100];
  }
  cout<<ans<<endl;
}
```

- 使用double类型存储组合数即可



估算组合数

- double能表示的数也有上界

$$3.5*10^3 = (1.10110101100)_2*2^{11}$$ （指数）

浮点类型	最大指数*	最大能表示的数
float	+127	$\approx 1.70*10^{38}$
double	+1023	$\approx 8.99*10^{307}$
long double	+16383	$\approx 5.95*10^{4931}$

阶乘的增长速度比组合数快得多，可以在存储的时候取个对数。

*数据参考来源：Wikipedia IEEE 754词条 https://en.wikipedia.org/wiki/IEEE_754

小结

- 基于递推公式
  - 递归
  - 记忆化搜索
  - 递推
  - 滚动数组
  - “原地滚动”
- 基于通项公式
  - 预处理阶乘
  - 相邻项推出
- 大整数的处理
  - 高精度
  - 取模
  - 浮点数估计