第三章　问题求解

第 1 节　组合数学初步

1、排列与组合

历史

1772 年，旺德蒙德以[n]p 表示由 n 个不同的元素中每次取 p 个的排列数。而欧拉则于 1771 年以及于 1778 年以表示由 n 个不同元素中每次取出 p 个元素的组合数。至 1872 年，埃汀肖森引入了以表相同之意，这组合符号（SignsofCombinations）一直沿用至今。

1830 年，皮科克引入符号 nCr 以表示由 n 个元素中每次取出 r 个元素的组合数；1869 年或稍早些，剑桥的古德文以符号 nPr 表示由 n 个元素中每次取 r 个元素的排列数，这用法亦延用至今。按此法，nPn 便相当於现在的 n!。

1880 年，鲍茨以 nCr 及 nPr 分别表示由 n 个元素取出 r 个的组合数与排列数；至 1899 年，克里斯托尔以 nPr 及 nCr 分别表示由 n 个不同元素中每次取出 r 个不重复元素的排列数与组合数，并以 nHr 表示相同意义下之可重复的排列数，这三种符号也通用至今。

两个基本原理是排列和组合的基础

(1)加法原理：做一件事，完成它可以有 n 类办法，在第一类办法中有 m1 种不同的方法，在第二类办法中有 m2 种不同的方法，……，在第 n 类办法中有 mn 种不同的方法，那么完成这件事共有 N＝m1＋m2＋m3＋…＋mn 种不同方法。

(2)乘法原理：做一件事，完成它需要分成 n 个步骤，做第一步有 m1 种不同的方法，做第二步有 m2 种不同的方法，……，做第 n 步有 mn 种不同的方法，那么完成这件事共有 N＝m1×m2×m3×…×mn 种不同的方法。

这里要注意区分两个原理，要做一件事，完成它若是有 n 类办法，是分类问题，第一类中的方法都是独立的，因此用加法原理；做一件事，需要分 n 个步骤，步与步之间是连续的，只有将分成的若干个互相联系的步骤，依次相继完成，这件事才算完成，因此用乘法原理。

这样完成一件事的分“类”和“步”是有本质区别的，因此也将两个原理区分开来。

排列和排列数

(1)排列：从 n 个不同元素中，任取 m(m≤n)个元素，按照一定的顺序排成一列，叫做从 n 个不同元素中取出 m 个元素的一个排列。

从排列的意义可知，如果两个排列相同，不仅这两个排列的元素必须完全相同，而且排列的顺序必须完全相同，这就告诉了我们如何判断两个排列是否相同的方法。

(2)排列数公式：从 n 个不同元素中取出 m(m≤n)个元素的所有排列

排列的记号和公式为 $P_n^m = \dfrac{n!}{(n-m)!}$

当 m＝n 时，为全排列 nPn=n*(n－1)*(n－2)…3*2*1＝n!

组合和组合数

(1)组合：从 n 个不同元素中，任取 m(m≤n)个元素并成一组，叫做从 n 个不同元素中取出 m 个元素的一个组合。

从组合的定义知，如果两个组合中的元素完全相同，不管元素的顺序如何，都是相同的组合；只有当两个组合中的元素不完全相同时，才是不同的组合。

(2) 组合数：从 n 个不同元素中取出 m(m≤n)个元素的所有组合的个数

(3) 这里要注意排列和组合的区别和联系，从 n 个不同元素中，任取 m(m≤n)个元素，“按照一定的顺序排成一列”与“不管怎样的顺序并成一组”这是有本质区别的。

组合的记号为 C(n,k)或$\binom{n}{k}$

$$\binom{n}{k} = \frac{P(n,k)}{P(k,k)}$$

根据 P(*n*,*k*)的定义：

$$P(n,k) = \frac{n!}{(n-k)!},$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}。$$

常见计数方法

1. 特殊优先

特殊元素，优先处理；特殊位置，优先考虑

例．六人站成一排，求

(1)甲不在排头，乙不在排尾的排列数

(2)甲不在排头，乙不在排尾，且甲乙不相邻的排法数

分析一、

先考虑排头，排尾，但这两个要求相互有影响，因而考虑分类。

第一类：乙在排头，有 P(5,5)种站法；

第二类：乙不在排头，当然他也不能在排尾，有 4*4*P(4,4)种站法；

共 P(5,5)+4*4*P(4,4)种站法。

分析二、

第一类：甲在排尾，乙在排头，有 P(4,4)种方法。

第二类：甲在排尾，乙不在排头，有 3*P(4,4)种方法。

第三类：乙在排头，甲不在排头，有 4*P(4,4)种方法。

第四类：甲不在排尾，乙不在排头，有 P(3,3)*P(4,4)种方法。

共 P(4,4)+3*P(4,4)+4*P(4,4)+P(3,3)*P(4,4)=312 种。

2. 捆绑与插空

例．8 人排成一队

(1) 甲乙必须相邻

(2)甲乙不相邻

(3)甲乙必须相邻且与丙不相邻

(4)甲乙必须相邻，丙丁必须相邻

(5)甲乙不相邻，丙丁不相邻

分析：（1）甲乙必须相邻，就是把甲乙捆绑(甲乙可交换)和 7 人排列 P(7,7)*2

（2）甲乙不相邻，P(8,8)-P(7,7)*2。

（3）甲乙必须相邻且与丙不相邻，先求甲乙必须相邻且与丙相邻 P(6,6)*2*2；甲乙必须相邻且与丙不相邻 P(7,7)*2-P(6,6)*2*2

（4）甲乙必须相邻，丙丁必须相邻 P(6,6)*2*2

（5）甲乙不相邻，丙丁不相邻，P(8,8)-P(7,7)*2*2+P(6,6)*2*2

例．某人射击 8 枪，命中 4 枪，恰好有三枪连续命中，有多少种不同的情况?

分析：因为连续命中的三枪与单独命中的一枪不能相邻，因而这是一个插空问题。另外没有命中的之间没有区别，不必计数。即在四发空枪之间形成的 5 个空中选出 2 个的排列，即 P(5,2)。

例．马路上有编号为 1，2，3，……，10 十个路灯，为节约用电又看清路面，可以把其中的三只灯关掉，但不能同时关掉相邻的两只或三只，在两端的灯也不能关掉的情况下，求满足条件的关灯方法共有多少种?

分析：即关掉的灯不能相邻，也不能在两端。又因为灯与灯之间没有区别，因而问题为在 7 盏亮着的灯形成的不包含两端的 6 个空中选出 3 个空放置熄灭的灯。

所以共 C(6,3)=20 种方法。

不定方程的正整数解个数

例：求方程$x_1+x_2+x_3+\cdots+x_N=M$的正整数解的个数

分析：想象现在有 M 个球一字排列，要把它们分成 N 组，我们用 N-1 块木板将它们分割成 N 段，N-1 块木板放在中间的 M-1 个空格中，由于每个空格中只能放一块木板，所以这是一个组合问题，答案是

$$\binom{M-1}{N-1}$$

又因为每一种放置木板的方法对应了一组不定方程的解，所以原问题的答案也是

$$\binom{M-1}{N-1}$$

例：求方程$x_1+x_2+x_3+\cdots+x_N=M$的非负整数解的个数。

分析：设$y_i=x_i+1$，则$y_1+y_2+y_3+\cdots+y_N=M+N$，问题转化为上一种情况，于是答案为

$$\binom{M+N-1}{N-1}$$

2、几个特殊的数列

Catalan 数

在一个有 n+2 条边的凸多边形中，我们可以画出 n-1 条不相交的对角线将多边形分为 n 个三角形，设所有满足条件的方案数是h_n，定义$h_0=1$，求h_3、h_4。

分析：

我们试图找出h_n的递推关系和通项公式。

考虑凸 n+2 边形的任意一条边，我们设为基边，考虑这条边所在的三角形，这个三角形会把凸 n+2 边形剖成两块，一块有 k+2 条边（可以剖分成 k 个三角形），那么另一块有 n+1-k 条边（可以剖分成 n-1-k 个三角形），根据乘法原理，这样的剖分方案数是$h_k * h_{n-1-k}$，又因为 k 显然可以从 0 一直变化到 n-1（考虑边界情况，并注意到对角线不相交，所以这里不会出现重复计数），故

$$h_n = \sum_{k=0}^{n-1} h_k * h_{n-1-k} \qquad (1)$$

数列$\{h_n\}$就是著名的 Catalan 数列，在组合数学的许多问题上都有很重要的应用。Catalan 数列的递推关系正如上式，再加上一点生成函数的知识就可以推导出它的通项公式，但是生成函数已远远超出本书所介绍的知识范畴，有兴趣的读者可以参考《组合数学》（《Introductory Combinatorics》，Richard A. Brualdi 著）中第 7 章和第 8 章的相关内容。笔者在此仅仅给出 Catalan 数列的通项公式，并不要求读者明白它是怎么来的：

$$h_n = \frac{1}{n+1}\binom{2n}{n}$$

我们会在以后看到这个数列非常有用，希望读者认真记住这个公式。

从另一种模型推导 Catalan 数

定理：n 个+1 和 n 个-1 构成的 2n 项$a_1, a_2, a_3, \ldots\ldots, a_{2n}$，并且其部分和满足

$$a_1 + a_2 + a_3 + \cdots + a_k \geq 0 \quad (k = 1,2,3,\ldots\ldots,2n)$$

对所有 k 都成立的数列个数等于第 n 个 Catalan 数

$$h_n = \frac{1}{n+1}\binom{2n}{n} \qquad (2)$$

证明：如果 n 个+1 和 n 个-1 的序列满足部分和都不小于 0，则称其为可接受的，否则为不可接受的，令A_n为 n 个+1 和 n 个-1 形成的可接受序列的个数，令U_n为不可接受的序列个数。我们尝试计算出$A_n + U_n$的值和U_n的值以得到A_n的值。

任意一个有 n 个+1 和-1 的构成的 2n 项必然属于且仅属于可接受序列和不可接受序列之一，于是有

$$A_n + U_n = \binom{2n}{n}$$

下面我们试图计算出U_n的值。

对于任意一个不可接受的数列中一定存在一个最小的 k，使得$a_1+a_2+a_3+\cdots+a_k<0$，于是我们可以得到一些显然的结论：k 是一个奇数，前 k-1 个数中+1 和-1 恰好各占一半且$a_k=-1$。我们将这 k 个数取相反数，后面的数不变，由此得到了一个有 n+1 个+1 和 n-1 个-1 的数列。注意这个操作是可逆的：对于一个有 n+1 个+1 和 n-1 个-1 的数列，我们只需要找到最小的 k，满足$a_1+a_2+a_3+\cdots+a_k>0$，然后将前 k 个数取反即可得到原数列。所以不可接受的序列和有 n+1 个+1 和 n-1 个-1 的数列是一一对应的，故

$$U_n=\binom{2n}{n+1}=\frac{n}{n+1}\binom{2n}{n}$$

再结合

$$A_n+U_n=\binom{2n}{n}$$

可知定理成立。

Catalan 数列的应用

因为 Catalan 数列满足递推关系（1）和定理（2），所以它在许多看似没有联系的模型中都有重要的应用。

1．括号序列：给出一个有 n 个运算符 n+1 个运算数的算式，要求在算式中任意添加括号，那么本质不同的运算顺序有多少种？

分析：考虑最后一个运算的符号，它的左边有 k 个运算符，右边则有 n-1-k 个运算符，由乘法原理可知这个问题满足递推关系（1），于是答案是 Catalan 数列。

2．有 2n 个人排队进入剧场，入场费 50 元，每人都带着一张 50 元或 100 元的钞票，剧院售票处没有任何零钱，有多少种情况满足无论什么时候售票处都能找的开零钱。

3．一个 n*n 的网格图，从左下走到右上，每次只能向上或向右走，不能走到左下到右上的对角线的上方，一共有多少种走法？

分析：2、3 两个问题本质是由 n 个+1 和 n 个-1 组成的序列中任意部分和都非负的序列总数（模型的转换希望读者认真思考），于是这两个问题的答案也是 Catalan 数列。

4．有 n 个结点的形态不同的二叉树一共有多少棵？

分析请读者自行完成

这个问题的答案也满足递推关系（1）

5．出栈序列统计，有 n 个数和一个栈，本质不同的合法序列一共有多少种？

提示：将一次入栈操作视为+1，一次出栈操作视为-1，剩余的分析请读者自行完成。

第 2 类 Stirling 数

请读者注意，第 1 类 Stirling 数和第 2 类 Stirling 数的出现都是基于对差分数列的研究，但是笔者无意给读者介绍这些在信息学初赛中毫无用处的数列，所以差分数列和第一类 Stirling 数的相关知识完全略去，请有兴趣的读者参考《组合数学》（《Introductory Combinatorics》，Richard A. Brualdi 著）中第 8 章第 2 节的相关内容。下面仅仅从第 2

类 Stirling 数的一个小应用引入第 2 类 Stirling 数。

定理：第二类 Stirling 数 S(n,k)是将 n 个元素的集合划分成 k 个不可辨认的非空盒子的划分的个数。注意，这里的不可辨认是指不能把一个盒子与另一个盒子分辨开，它们看起来都一样。

下面给出第二类 Stirling 数的递推关系和证明：

S(n,k) = k*S(n-1,k) + S(n-1,k-1)；S(n,1)=1（n≥1），S(n,n)=1。

上面的递推式可以用组合证明：一方面，如果将元素 n 单独拿出来划分成 1 个集合，那么方法数是 S(n-1,k-1)；另一方面，如果元素 n 所在的集合不止一个元素，那么可以先将剩下的 n-1 个元素划分好了以后再选一个集合把 n 放进去，方法数是 k*S(n-1,k)。

例题：盒子与球

有 3 个一模一样的盒子和 6 个不同颜色的球，将这些球放到 3 个盒子里并且保证每个盒子里至少有 1 个球，求放置的方案总数。

答案：第二类 Stirling 数 S(6,3)

3、容斥原理与错位排列问题

容斥原理

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n}|A_i| - \sum_{i,j:\,i\neq j}|A_i \cap A_j| + \sum_{i,j,k:\,i\neq j\neq k}|A_i \cap A_j \cap A_k| - \cdots\cdots \pm |A_1 \cap \cdots \cap A_n|$$

在计数时，必须注意无一重复，无一遗漏。为了使重叠部分不被重复计算，人们研究出一种新的计数方法，这种方法的基本思想是：先不考虑重叠的情况，把包含于某内容中的所有对象的数目先计算出来，然后再把计数时重复计算的数目排斥出去，使得计算的结果既无遗漏又无重复，这种计数的方法称为容斥原理。

例题：有多少能被 3 或 5 或 7 整除的小于 1000 的正整数？

分析：设A_1、A_2、A_3分别表示被 3 整除、被 5 整除、被 7 整除的数构成的集合，根据公式，

$$\left|\bigcup_{i=1}^{3} A_i\right| = \sum_{i=1}^{3}|A_i| - \sum_{i=1}|A_i \cap A_j| + \sum |A_1 \cap A_2 \cap A_3|$$

于是答案是

$$\left\lfloor\frac{1000}{3}\right\rfloor + \left\lfloor\frac{1000}{5}\right\rfloor + \left\lfloor\frac{1000}{7}\right\rfloor - \left\lfloor\frac{1000}{15}\right\rfloor - \left\lfloor\frac{1000}{21}\right\rfloor - \left\lfloor\frac{1000}{35}\right\rfloor + \left\lfloor\frac{1000}{105}\right\rfloor$$

$$= 543$$

错位排列问题

对于一个 1……n 的任意排列，有一种情况是第 i 个位置上的数不是 i 对任意 i 都成立，例如 n=3 时，有且仅有排列 231 和 312 是满足要求的，这样的排列被称为错位排列。现在的

问题是，能否对于任意 n，给出错位排列的个数？

分析：利用容斥原理。

设A_i表示 i 这个数在原来位置上的排列的集合，则错位排列的个数

$$D_n = n! - \left|\bigcup_{i=1}^{n} A_i\right|$$

现在我们来算$|\cup_{i=1}^{n} A_i|$，

先算$\sum|A_i|$，因为有 1 个数在固定的位置上，有 C(n,1)种情况，对于每一种情况，剩下的 n-1 个数任意排列，于是有 C(n,1)*(n-1)!=n!种情况。

再算$\sum|A_i \cap A_j|$，因为有 2 个数在固定的位置上，有 C(n,2)种情况，对于每一种情况，剩下 n-2 个数任意排列，于是有 C(n,2)*(n-2)!=n!/2 种情况。

……

……

一般地，因为有 k 个数在固定的位置上，另外(n-k)个数任意排列，所以

$$\sum|A_{i_1} \cap A_{i_2} \cap A_{i_3} \cap \ldots\ldots \cap A_{i_k}| = \binom{n}{k} * (n-k)! = \frac{n!}{k! * (n-k)!} * (n-k)! = \frac{n!}{k!}$$

综上，由容斥原理，我们可以得到错位排列的公式

$$D_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n * \frac{1}{n!}\right)$$

错位排列的递推关系

考虑 1 号位置的那个数，不妨设为 k，k>1

以下有两种情况：

1．k 号位置的那个数是 1，于是剩下 n-2 个数也是一个错位排列，有D_{n-2}种情况；

2．k 号位置的那个数不是 1，那么不妨设 1 原来所在的位置应该在 k，这样就是除 k 以外的 n-1 个数的一个错位排列，方案数为D_{n-1}。

最后，因为 k 的取值可以是 2 到 n 中的任意一个，于是我们得到了错位排列的递推关系

$$D_n = (n-1) * (D_{n-1} + D_{n-2})$$

初始值$D_1 = 0,\ D_2 = 1$

事实上在 n 较小的时候，计算D_n更多的是通过递推关系而不是通项公式，但是通项公式和递推关系以及它们的推导过程希望读者能牢牢记住。

两个计数技巧——算两次和补集转化

算两次

在对一个集合进行计数的时候，我们往往可以通过不止一条途径得到答案，那么用两种不同的方法计算得到答案应该是相同的，这样的技巧称为算两次，事实上算两次的应用并不仅仅局限于验证答案的正确性，请看下面的例题：

一棵二叉树中有 7 个结点有 2 个儿子，求二叉树中有多少叶子结点(没有儿子的结点)？

分析：设 S 表示二叉树的结点总数，S_i表示有 i 个儿子的结点的个数。

一方面 S 可以这样计算：所有结点的个数总和，

即$S = S_1 + S_2 + S_0$

另一方面，S 可以这样计算，根结点+所有有 1 个儿子的结点的儿子个数+所有有 2 个儿子的结点的儿子个数，

即$S = 1 + S_1 + 2 * S_2$

两次计算结果相互对照可以得出$S_0 = S_2 + 1$，于是例题的答案是 8。

这样的技巧我们可以推广到 k 叉树中得到更一般的结论：

$$S_0 = S_2 + S_3 * 2 + S_4 * 3 + \cdots + S_k * (k-1) + 1$$

补集转化

在计算一个集合 A 中元素的个数的时候，往往会发现直接求解很困难，但是如果我们换一种思路，利用全集 C 和 A 在 C 中的补集中元素的个数迂回求解一般可以达到更好的效果，这样求解的技巧称为补集转化，多用于集合本身元素个数不好分析但是全集和补集都很容易求解的情况，由于这样的情况大量存在，所以补集转化思想是一个很重要的计数技巧。

计算中用到补集转化的例子比比皆是，我们甚至不需要刻意去找这样的例题，稍微留心的读者大概就已经注意到了，我们在上文求解错位排列的通项公式的时候就用了补集转化的思想。另一个稍远的例子在前面第 2 部分“从另一种模型推导 Catalan 数”部分，可接受序列A_n的个数不好求，但是$A_n + U_n$的值和U_n的值都可以很方便地求到，于是我们利用补集转化的技巧得到了A_n的值。

4、鸽巢原理

鸽巢原理

鸽巢原理，又名狄利克雷抽屉原理、鸽笼原理。

其中一种简单的表述法为：若有 n 个笼子和 n+1 只鸽子，所有的鸽子都被关在鸽笼里，那么至少有一个笼子有至少 2 只鸽子。

另一种为：若有 n 个笼子和 kn+1 只鸽子，所有的鸽子都被关在鸽笼里，那么至少有一个笼子有至少 k+1 只鸽子。

鸽巢原理的加强形式

令$q_1, q_2, \cdots\cdots, q_n$**为正整数**，如果将

$$q_1 + q_2 + q_3 + \cdots + q_n - n + 1$$

个鸽子放入 n 个笼子里，那么，或者第 1 个笼子至少有q_1个鸽子，或者第 2 个笼子至少有q_2个鸽子，……，或者第 n 个笼子至少有q_n个鸽子。

若有 n 个笼子和 kn-1 只鸽子，所有的鸽子都被关在鸽笼里，那么至少有一个笼子至多只有 k-1 只鸽子。

鸽巢原理的简单应用

一位国际象棋大师有 11 周的时间备战一场锦标赛，他决定每天至少下一盘棋，但为了不使自己过于疲劳他还决定每周不能下棋超过 12 盘，证明存在连续的若干天，期间这位大师恰好下了 21 盘棋。

证明：令a_1表示第 1 天所下的盘数，a_2表示前 2 天所下的盘数，a_3表示前 3 天所下的盘数……由于每天都要下至少一盘棋，故数列$a_1, a_2, a_3, \cdots\cdots, a_{77}$是一个严格递增的序列。此外，$a_1 \geq 1$，而且由于每周下棋最多是 12 盘，即$a_{77} \leq 12 * 11 = 132$。因此，我们有

$$1 \leq a_1 < a_2 < a_3 < \cdots < a_{77} \leq 132$$

此外，序列$a_1 + 21, a_2 + 21, a_3 + 21, \cdots\cdots, a_{77} + 21$也是一个严格递增的序列：

$$22 \leq a_1 + 21 < a_2 + 21 < a_3 + 21 < \cdots < a_{77} + 21 \leq 153$$

于是，这 154 个数

$$a_1, a_2, a_3, \cdots\cdots, a_{77}, a_1 + 21, a_2 + 21, a_3 + 21, \cdots\cdots, a_{77} + 21$$

中的每一个都是 1 到 153 之间的一个整数。根据鸽巢原理克制，它们中间有两个是相等的。既然$a_1, a_2, a_3, \cdots\cdots, a_{77}$中没有相等的数，并且$a_1 + 21, a_2 + 21, a_3 + 21, \cdots\cdots, a_{77} + 21$中也没有相等的数，因此必然存在一个 i 和一个 j 使得$a_i - a_j + 21$。从而，这位国际象棋大师在第 j+1，j+2，j+3，……，i 天这段时间内一共下了 21 盘棋。

例题【NOIP2010 提高组】记 T 为一队列，初始时为空，现有 n 个总和不超过 32 的正整数依次入列。如果无论这些数具体为何值，都能找到一种出队的方式，使得存在某个时刻队列 T 中的数之和恰好为 9，那么 n 的最小值是_____。

分析：组合数学题目的一种常见思路是先猜测答案再证明。我们试图通过鸽巢原理猜出答案。

设S_i表示前 i 个数的和，并规定$S_0=0$，事实上原题要求出最小的 n，使得存在$0\le i<j\le n$满足$S_j-S_i=9$。于是我们可以把 S 数组看做鸽子，用不能同时取的一组（即差为9)的集合构造笼子,构造方法如下:一共有18个集合按如下方式选取，{0,9}、{1,10}、{2,11}、{3，12}、{4,13}、{5,14}、{6,15}、{7,16}、{8,17}、{18,27}、{19,28}、{20,29}、{21,30}、{22,31}、{23,32}、{24}、{25}、{26}，根据题意，我们一旦在某个集合中取了两个元素，那么一定存在某个时刻队列 T 中数之和恰好为9，于是由鸽巢原理我们可以知道 n=18（S 数组有 19 个元素）一定满足条件。

下面我们来证明 n=17 不可行，事实上只要构造出一组不合法的序列即可。可以选择如下不合法的序列：1 1 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1。

综上：原问题所求的 n 的最小值为 18。

5、Nim 取石子游戏

最简单的取石子游戏

考虑这样一个简单的问题：

有一堆石子，共 n 个，两个人轮流取石子，每次最少取 1 个，最多取 m 个，取完最后一个石子的人获胜，问先手有没有必胜方案。

这是一个非常经典的题目，相信大部分读者都做过这样的题目。这里直接给出答案：当 n mod (m+1) = 0 时,先手必败;否则先手必胜,必胜策略是每一次都取走当前石子数 mod (m+1) 的值。

SG 函数

现在我们来考察上面所说的策略为什么是对的。

首先说明必败态和必胜态的概念。必败态是说当前这个状态,先手无论如何操作都必败;必胜态则是说当前这个状态，先手无论如何操作都必胜。注意，在一般确定操作状态（例如不会有每次给一个随机数根据随机数操作的情况）的组合游戏中，只会存在这两种状态，如果先手和后手都足够聪明，不会出现介于必胜态和必败态之间的状态。

一个重要的性质：从必败态走到的每一个状态都是必胜态，从必胜态操作一步走到的所有状态中至少有一个是必败态。规定无法操作的状态为必败态。

在最简单的取石子游戏中，设 f[N]表示还剩 N 个石子的状态，那么显然 f[N]可以转移到 f[N-1]到 f[N-M]，这里面只要有一个必败态 f[N]就必胜否则必败。

那么我们是否可以不通过递推的方式判断必胜态还是必败态呢？

下面引入 SG 函数。

SG 函数的规定如下：最终状态（不可操作状态）的 SG 函数为 0，其余状态的 SG 函数规定为所有不等于它的某一个后继状态的SG函数的最小非负整数,例如一个状态有 2 个后继状态，它们的 SG 函数分别为 2 和 3，则当前状态的 SG 函数为 0；2 个后继状态的 SG 函数分别为 0 和 2，则当前状态的 SG 函数为 1。

SG 函数判断状态是否必胜的规则是如果当前状态的 SG 函数为 0，则当前状态必败，否则当前状态必胜。容易发现引入 SG 函数后我们之前递推判断必胜必败的规则也可用于 SG

函数的递推上。往往我们可以找到一些状态和 SG 函数之间的规律，从而避免递推。

例题【NOIP2010 提高组】

```
#include <iostream>
using namespace std;
const int NUM = 5;
int r(int n){
    int i;
    if (n <= NUM)  return n;
    for (i = 1; i <= NUM; i++)
        if (r(n - i) < 0) return i;
    return -1;
}
int main(){
    int n;   cin>>n;
    cout<<r(n)<<endl;
    return 0;
}
```

输入：16

输出：_____

分析：这个程序实际上就是通过递推的方式来求最少取 1 个石子，最多取 5 个石子的博弈游戏的 SG 函数，但是我们可以分析出规律：状态 f[N]的 SG 函数=n mod 6，于是应该输出 4。

Nim 取石子游戏

现在有若干堆石子，两人轮流操作，每人每次可以从任意一堆中取任意多个，但是不能不取，取完最后一个石子的人获胜（即无法再取的人就输了）。

这个问题就是最经典的 Nim 取石子问题。

我们在此不加证明地直接给出这个游戏状态的 SG 函数。

对于一个状态，设有 N 堆石子，每堆个数分别为a_1、a_2、a_3、……、a_N，那么这个状态的 SG 函数为 $a_1 \text{ xor } a_2 \text{ xor } a_3 \text{ xor } \ldots\ldots \text{ xor } a_N$。根据 SG 函数的定义，$a_1 \text{ xor } a_2 \text{ xor } a_3 \text{ xor } \ldots\ldots \text{ xor } a_N > 0$的状态为必胜态，否则为必败态。

在这里笔者仅仅给出一个每个状态的SG函数不会等于其任意一个后继状态的SG函数的证明，更详细的关于 SG 函数为什么等于所有数 xor 值的证明读者可以参照下面一部分《Nim 取石子游戏的必胜态策略分析》中的分析自行完成。

xor 函数有一个相当优美的性质，它是一个可逆函数（所谓可逆函数，就是存在反函数的函数），并且它的反函数就是本身。

举一个简单的例子，a xor b = c，那么一定有 b xor c = a 和 a xor c = b。

下面我们用反证法进行证明，假设存在一个状态的 SG 函数和它的一个后继状态的 SG 函数相等，设$k = a_1 \text{ xor } a_2 \text{ xor } a_3 \text{ xor } \ldots\ldots \text{ xor } a_N$，当前的操作是在第 M 堆中取若干石子，设

$s = k \text{ xor } a_M$，即除a_M以外其他所有数的 xor 值，设在a_M这一堆中取走了 x 个，因为当前状态和这个后继状态 SG 函数相等，则有$s \text{ xor } a_M = s \text{ xor } (a_M - x) = k$，由于 xor 函数的反函数就是它本身，于是可知$a_M = a_M - x$，从而 x=0，与游戏者不能不取的条件矛盾，故原命题成立。

Nim 取石子游戏的必胜态策略分析

最后我们来分析一下 Nim 取石子游戏中的必胜态策略。

由上一部分的证明可以看出，如果有一个状态的 SG 函数为 0，那么它的任何后继状态的 SG 函数一定不为 0，也就是说它的任何后继状态都是必胜态，现在我们只需要找出一个方案，使得任何一个必胜态依此操作后都会形成一个必败态。

考虑任意一个必胜态的 SG 函数值 s，将它写成二进制的表示，找出最高位的一个 1。因为这一位是 1，又由 xor 函数的运算规则可知，至少有一堆的石子数写成二进制之后这一位也是 1，不妨设是第 k 堆，有 x 个石子。设$x' = x \text{ xor } s$，由 xor 函数的性质可知，用 x’代替 x 作为第 k 堆的石子数之后，所有石子堆的石子数 xor 的值为 0，于是我们只需证明$x' < x$。事实上因为 s 的最高位的 1，不妨设为第 c 位，由 xor 函数的运算规则和 x 这一位也是 1 的约定可知：x 和 x’在比 c 更高的数位上数字完全相同，且 x 第 c 位为 1，x’的第 c 位为 0，所以我们甚至无需继续比较就可以得出结论：$x' < x$。

综上，按照如上的构造方法，在第 k 堆中取走 x-x’个石子就可以使一个必胜状态变成必败状态，于是必胜策略的可行性得证。

例题【NOIP2006 普及组】现有 5 堆石子,石子数依次为 3，5，7，19，50，甲乙两人轮流从任一堆中任取（每次只能取自一堆，不能不取），取最后一颗石子的一方获胜。甲先取，问甲有没有获胜策略（即无论乙怎样取，甲只要不失误，都能获胜）？如果有，甲第一步应该在哪一堆里取多少？请写出你的结果：_____。

分析：首先我们来看第一个问题，判断当前这个状态是否是必胜态。

因为 3 ^ 5 ^ 7 ^ 19 ^ 50 = 32 > 0，所以当前的状态是必胜态，先手有必胜策略。

下面我们来回答第二个问题，根据上一部分的策略，先看 32 的最高位，最高位为 6，而石子的个数中，50 的第 6 位为 1，于是我们应该让那一堆剩下 50 xor 32 = 18 个石子，即从 50 那一堆中取走 32 个。

6、图论浅谈——二分图理论与 Ramsey 理论

二分图

二分图又称作二部图，是图论中的一种特殊模型。 设 G=(V,E)是一个无向图，如果顶点 V 可分割为两个互不相交的子集(A,B)，并且图中的每条边（i，j）所关联的两个顶点 i 和 j 分别属于这两个不同的顶点集(i in A,j in B)，则称图 G 为一个二分图。

二分图最著名也是最重要的一个应用就是最大匹配，但是在这里，最大匹配与初赛无关，我们来看另一个也很重要但是往往容易被忽视的结论：

一个图是二分图的充分必要条件是这个图中不含奇环

充分性是说如果一个图不含奇环，那么这个图是二分图；必要性是说如果一个图是二分图那么这个图一定不含奇环。证明显然。

另一个显然的事实是，如果二分图的某一类点有 x 个，另一类点有 y 个，那么这个二分图中最多有 xy 条边。

例题【NOIP2010 提高组】无向图 G 有 7 个顶点，若不存在由奇数条边构成的简单回路，则它至多有_____条边。

分析，由条件可知这个图是一个二分图，设某一类点有 x 个，那么另一类点有(7-x)个，原题即求 x(7-x)的最大值，其中 x 是正整数。可以用一元二次方程的性质或者一些经典的不等式，很快就能得到答案为 12。

我们将这个题目推广开去，如果一个图有 N 个点，不存在由奇数条边构成的简单回路，则它至多有$\left\lfloor \frac{N^2}{4} \right\rfloor$条边。

注意，如果我们用一个看似更松一点的约束条件：这个图中不存在三角形，那么这个问题的答案并不会增加，取等号的条件也不变，仍为一个一类点有$\left\lfloor \frac{N}{2} \right\rfloor$个、另一类点有$\left\lceil \frac{N}{2} \right\rceil$个的完全二分图。

7、从两个形似的问题看数学模型的构建

在这一节里，我们将讨论最小任意交换排序和最小相邻交换排序。笔者将在最后给出一些分析数学问题和建立数学模型的建议，希望能对读者有一点启发，起到一点抛砖引玉的作用。

最少任意交换排序

【NOIP2008 普及组】将数组{8，23，4，16，77，-5，53，100}中的元素按从小到大的顺序排列，每次可以交换任意两个元素，最少需要交换（ ）次。

A．4 B．5 C．6 D．7

分析：初看此题，毫无头绪，我们不妨从最简单的贪心和模拟开始，看看能不能找到一点思路。

首先第一个数是 8，排好序后第一个数应该是-5，于是做一次交换；第二个数是 23，排好序后应该是 4，于是做一次交换。此时序列变成{-5，4，23，16，77，8，53，100}。然后再把 23 和 8 交换，77 和 23 交换，77 和 53 交换即可排好序，一共用了 5 次交换。我们用了贪心的思想做到了正确答案 5 但是很可惜目前我们还不知道这个答案为什么是对的。

为此需要引入置换的概念。

设 X 是一个有限集。不失一般性，取 X={1,2,3，……，N}

X 的一个置换

$$i_1, i_2, i_3, \ldots\ldots, i_N$$

是一个 1 到 N 的排列，这个置换可以视为 X 到其自身定义的一个一对一的函数

$$f: X \to X$$

其中$f(1)=i_1, f(2)=i_2, \ldots\ldots, f(N)=i_N$，用行列式可以表示为

$$\begin{pmatrix} 1 & 2 & \ldots & N \\ i_1 & i_2 & \ldots & i_N \end{pmatrix}$$

事实上如果我们把 k 和i_k之间连一条有向边，那么一个置换可以用一个有向图唯一地表示，特别地，这个有向图是由一些有向环组成的（包括只有一个点的自环），注意到我们每次交换两个元素只有当这两个元素在一个环上时才有意义，并且一个有 k 个点的环至少需要交换 k-1 次才能完成排序，于是最少任意交换排序的答案就是 N-置换环的个数。

最少相邻交换排序

现在我们把上面一个问题的要求变一下，每次只能交换两个相邻元素，那么这时的最少交换次数又会是多少呢？

分析：这个问题和上面那个问题看上去很相似，然而很可惜，它们只是形似，本质的数学模型没有一点相近的地方。

为解决这个问题，我们先考察相邻交换的性质。注意到相邻交换只会改变相邻两个数的相对大小而不会影响到别的数，因此我们考虑能不能从改变相对大小这个方面入手。在数列中，有一个值和相对大小密切相关，那就是逆序对数。所谓逆序对，就是指一个数列中的两个数 i 和 j，i 在数列中的位置在 j 的前面并且 i>j。显而易见的是，一个排好序的数列的逆序对数为 0。现在我们看看相邻交换对逆序对数有什么影响，如果前一个数小于后一个数，交换后逆序对数加 1；否则逆序对数减 1。因为排好序的数组不含逆序对，所以我们希望逆序对数减得尽量快，这样可以尽可能减少交换次数。而另一方面，对于一个没有排好序的数组，我们一定可以找到两个相邻的数是逆序对，这样当逆序对数大于 0 的时候，我们总可以通过一次交换使逆序对数减 1。于是最小相邻交换排序的交换次数就等于原数列的逆序对数。

浅谈组合数学问题分析中数学模型的构建

1．尝试，或者说探索，这是一个很重要的步骤，在这一阶段我们可以用模拟、贪心等各种手段来探索问题的解，很可能在这一步我们就得到的答案甚至不需要进行后续的分析。

2．从小数据入手，分而治之。很多题目把数据规模改小之后和原题的处理方法是差不多的，这一点可以在第 1 步探索时试出来。

【NOIP2006 普及组】

现有 80 枚硬币，其中有一枚是假币，其重量稍轻，所有真币的重量都相同，如果使用不带砝码的天平称重，最少需要称几次，就可以找出假币？你还要指出第 1 次的称重方法。请写出你的结果：_____。

80 这个数据规模很大，我们不妨从小数据入手。如果有 2 枚硬币，只有 1 枚假币，那么很显然一次称重就可以；3 枚硬币也是 1 次就可以（天平的两边各放一枚）。3 枚以上的可以分成 3 堆，其中两堆数目相同，然后把两堆相同数目的放到天平上称重，这样一次我们就可以知道假币在哪一堆中，然后这个问题就被用一次称重转化到数据规模为原来 1/3 的问题了，于是原问题的答案是$\lceil \log_3 N \rceil = 4$。

3．分析题目中的有效信息，准确地说，是找到题目中的“特点”，例如计数时的一些限制，最少交换排序的交换方式，这些题目特有的性质是我们分析的要点。

4．构建有效的数学模型。这是最重要的一步，但事实上在一般的考试题目是没有必要进行严格的分析的，第一步和第二步往往已经可以解决问题。模型的构建是和题目密切相关的，也就是根据题目的特点构建的。比如最小相邻排序问题中，我们需要寻找一个量，使得交换相邻两个数时，这个量的变化是确定的，这样逆序对数就给我提供了一个恰当的帮助。

最后，数学问题是千变万化层出不穷的，我们也无法给读者提供一个能解决所有问题的方法，希望这一部分《组合数学初步》能给读者一些思维方法和分析问题上的启迪，起到一点抛砖引玉的作用，但是要想真正做好数学题，仅靠这一点知识基础是不够的，希望读者能寻找到一些合适的题目多加练习。