退一步海阔天空

——“目标转化思想”的若干应用

【关键字】让步假设　割补法　应用

【摘要】本文主要讨论在算法设计中，如何将不易求解的问题转化成为一个范围较大、但容易求解的问题，然后再将所扩大部分削减，最后得出所要的答案。讨论中主要是针对计算集合和组合数学中的一些常见问题进行剖析，同时也涉及了一些公理，如容斥原理等。在一些问题中并不一味追求一个完全理想的算法，而是将所用的方法取长补短，并讨论算法的利弊及对解题的帮助。

第一章 概述

古人常说：“退一步，海阔天空。”这是在教我们在待人接物中，要学会忍让；在追求目标时，要有所取舍。的确如此。但我们除了在行为举止中遵循古人的教诲之外，是否还可以从中得到一些启发呢？比方说，在做学问受阻而停滞不前时，能否将所不能解决的问题扩大求解呢？答案是肯定的，这正如数学界对“歌德巴赫猜想”的证明，“1+1”不行，暂且退一步，而且一退就退到了“9+9”，然后再一步步地向目标逼近，直到现在陈景润的“1+2”，不管这个猜想的证明最后能不能成功，它至少给了我们一点启发：在信息学中，也可以有“让步”，退一步，同样海阔天空。

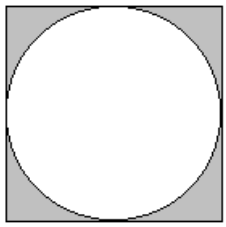
在我们平时设计算法时，经常会遇到一些难以解决的问题，有时可以先将问题求解的范围扩大（如条件放松、给以适当假设（猜想）、目标放大等），然后再对新的问题求解，最后，将扩大部分“剪掉”，就解决了问题。例如 2002NOI 分区联赛（提高组）复赛第二题，整数划分。问题问的是：将一个自然数 n 分解成为 k 部分 A1，A2，A3，……Ak（其中 A1<=A2<=A3<=……<=Ak），问有多少种不同的分法？解题时，问题不易直接求解，必须先将它进行转化。不难发现，问题等价于：将一个自然数 n 分成若干份，其中最大的一份为 k 的分法总数。这时，可以令设 G(n,k)表示这个值的大小，但这样仍然不利于解决问题，于是引入另外一个函数 F(n,k)，表示将一个自然数 n 分解成若干份，其中最大一份不超过 k 的方法总数。于是，F(n.k)便满足以下递推关系：

$$f(n,k)=\sum_{i=1}^{k}g(n,i)=\sum_{i=1}^{k}f(n-i,i)=f(n-k,k)+f(n,k-1)$$

这样，便可以用最常规的递推算法求解问题了，方法简单，每次运算也只需要进行一次加法运算，算法复杂度为O(NK)，最后 $f(n,k)-f(n,k-1)$ 就是所求。在这里，一开始先对问题进行转化，将难于求解的转化为可以求解的，但仍需进行累加，复杂度太高。于是就运用了上面所说的“让步”，将求解目标扩大到分得的允许最大一份小于k，最终得到一个二次方复杂度的算法。应当说，这里虽然也用了目标扩大的方法，但这退的只是一小步，只是达到了简化算法的目的，并不涉及解题模型的根本。

第二章 在计算几何中的应用

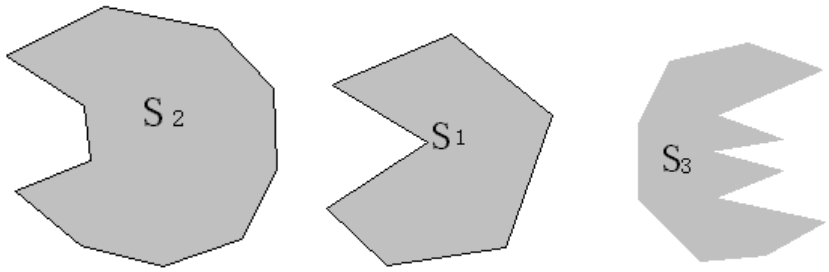
计算几何是近几年信息学竞赛经常涉及的领域，虽然考得不是很深，但一些常见的问题还是难倒了不少选手。这也许是因为现实世界与纯粹的数字世界有太大差异的缘故吧。例如遮挡问题，判断一件物体是否被另外的一件或几件物体遮挡，这在日常生活中想都不用想，看一眼就什么都清楚，但在计算几何中，就不是一件轻而易举的事情了。也正因为如此，“让步假设”才在计算几何中大有用武之地。例如许多涉及到多边形面积的问题就经常用到这一方法：求多边形面积、求多边形重心、判断点与多边形位置关系等等。对于这些问题，无论在日常计算或是实际的算法设计中都或多或少的引入了这一思想。例如小学时我们常做这样的习题：计算右图染色部分的面积。我们谁都知道要先求出矩形的面积，再将圆形面积求出，最后相减便得到结果。其实也可以将它看成这样一个过程：先假设圆形也是图形的一部分，它也染了色，于是整个图形便是一个矩形，然后将“假设”的部分拿掉，就得到了这个图形。由这种思想便产生了另外的一种更为科学的面积计算方法。



第一节　计算任意多边形的面积

学习计算几何，就不得不面对一个问题：计算平面图形面积。如果按图形的形状分，可将其分为折线图形和弧线图形，我们这里讨论的主要是折线图形的面积。折线图形又可以分为凸多边形和凹多边形（假设多边形的任意两条边都不相交，端点除外）。凸多边形面积比较好求，只须先将它分解成为若干个三角形，再用海伦公式或叉积面积计算公式就可以方便求出了，而对于凹多边形恐怕就要花费一番心思了。受到上面图形的启发，不难得到一种初学者常用的方法：先将一个凹多边形补足成为一个凸多边形，求其面积，再减多出来的部分。但这种方法显然带有瑕疵

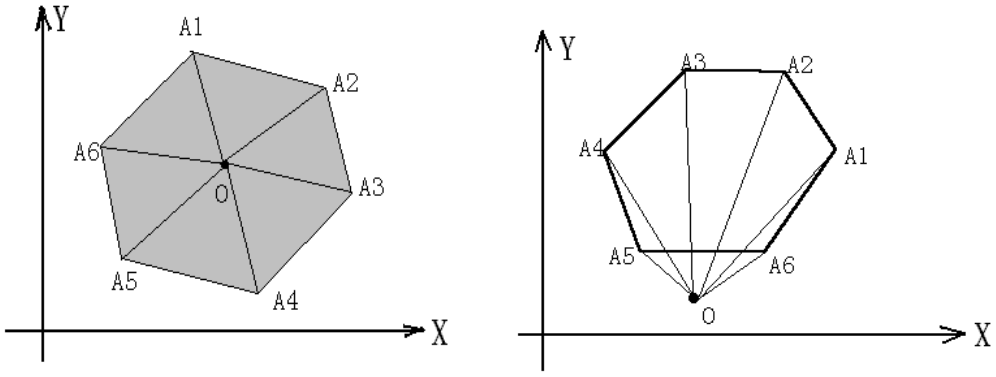
例如下面的几个图形，虽然都可以补足成为一个凸多边形，但补足部分的面积计算就大相径庭了：第一个图形的多余部分是一个三角形，第二个的是一个凸多边形，最后一个则是一个凹多边形了。这种方法正违背了运用“割补”的最重要原则：方便。“割补法”的核心正是变复杂为简单，但如果无法降低其运算复杂度，割补就失去了它的意义了。



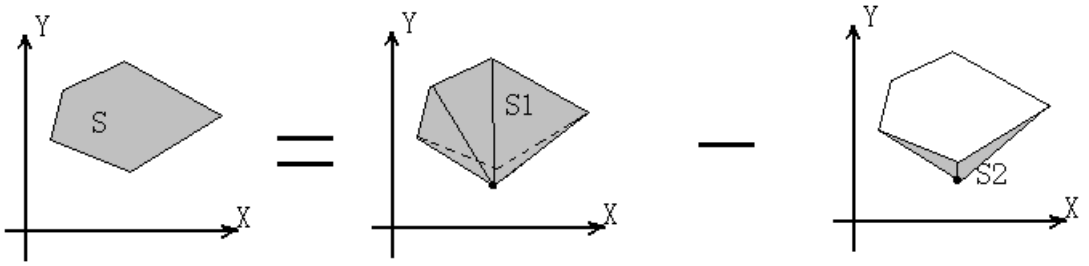
其实，求任意多边形的面积已经有一种经典的算法——叉积面积计算公式。假设笛卡儿坐标系中的一个任意多边形 A1A2A3……Ak，各点的坐标分别是 A1(X1,Y1)，A2(X2,Y2)，A3(X3,Y3)，……，Ak(Xk,Yk)，设 O(X0,Y0)任意一点，则多边形面积 S 有：

$$S=\sum_{i=1}^{k}\frac{1}{2}\begin{vmatrix}X_i-X_0 & X_{i+1}-X_0\\ Y_i-Y_0 & Y_{i+1}-Y_0\end{vmatrix}$$

这种方法中，I+1 是再 MOD 的意义下进行的，而且其中的三角形面积有正负之分。例如图一，图二所示。



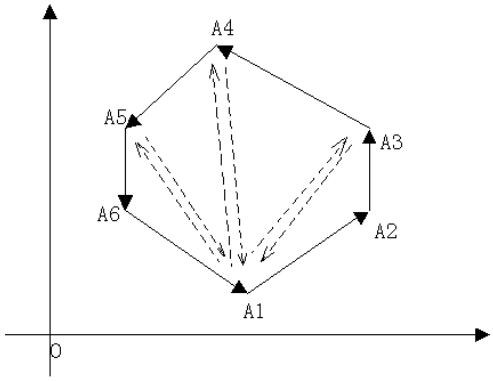
当点 O 在多边形内时，每一个三角形的面积都是正的，与第一中方法相同；当出现图二所示情形时，恰好两部分面积（如图）一正一负，相减即为所求。



其实它的正确性是比较明显的，对于刚才的图形，我们可以给出一个简略的证明：

设$S_{i,j,k,\cdots}$表示图上I，j，k，……等点构成的多边形面积（恒为正），

则

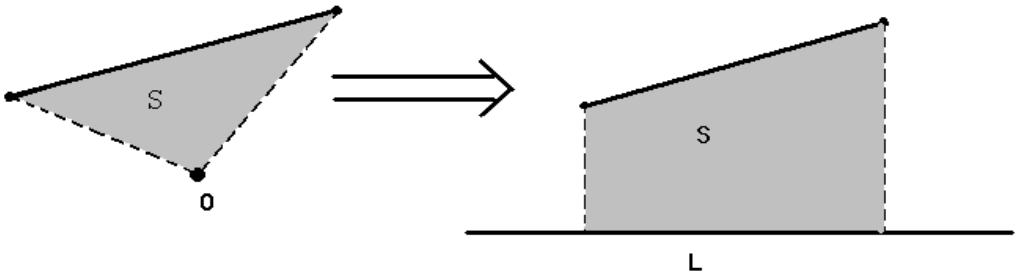


$$S=\sum_{i=1}^{6}\frac{1}{2}\begin{vmatrix}X_i-X_0 & X_{i+1}-X_0\\ Y_i-Y_0 & Y_{i+1}-Y_0\end{vmatrix}=S_{1,2,0}+S_{2,3,0}+S_{3,4,0}+S_{4,5,0}-S_{5,6,0}-S_{6,1,0}$$
$$=S_{1,2,0}+S_{2,3,0}+S_{3,4,0}+S_{4,5,0}-S_{5,6,0}-S_{6,1,0}+(S_{0,1,3}-S_{0,1,3})$$
$$+(S_{0,1,4}-S_{0,1,4})+(S_{0,1,5}-S_{0,1,5})$$
$$=(S_{1,2,0}+S_{2,3,0}-S_{3,1,0})+(S_{3,1,0}+S_{3,4,0}-S_{4,1,0})$$
$$+(S_{0,1,4}+S_{0,4,5}-S_{0,1,5})+(S_{0,1,5}-S_{0,6,5}-S_{0,1,6})$$
$$=S_{1,2,3}+S_{1,3,4}+S_{1,4,5}+S_{1,5,6}$$
$$=S_{1,2,3,4,5,6}$$

证毕。

这只是证明了图示情况的正确性而已，但由此推及一般，不难证明该公式的正确性。另外，这里采用了一个单独的点0作为“新原点”，在实际应用中，为

了简化公式，往往将0取在坐标原点上，这样，公式就简化成为 $S=\sum_{i=1}^{k}\frac{1}{2}\begin{vmatrix}X_i & X_{i+1}\\ Y_i & Y_{i+1}\end{vmatrix}$，显然方便多了。如果还不满意的话，也可以尝试将点换成一条边，即将原来的点与线段围成的面积转化成为线段与直线对应围成的面积（如下图所示）。由同样方法也不难证明这种方法对于上图同样适用。但比较这两种算法，显然后者不如前者。前者面积公式简单明了，后者就繁杂多了，此其一；其二，前者的“新原点”可以十分随便，取哪都成，但后者需要的是一条直线，为了方便起见，直线不能与图形相交，而且最好还是水平的，这就大大限制了这种方法的适用范围。但此二法就根本上讲，其实同出一家，“本是同根生”，算法复杂度也相当，均为O（N）级（N表示多边形的顶点个数）。



在这里，我们讨论了“割补法”的一个应用，也是最基础、常见的一个实例，而实际上，它远不止如此，下面我们将见到它在其他问题上的威力。

第2节　计算任意多边形的重心

“重心”在物理学上是一个使用频率极高的概念，许多信息学的“好事者”也喜欢在它上面大做文章。于是，如何求物体重心便成为一个问题摆在我们面前。其实在去年的《电脑爱好者》上就登载了一篇介绍如何求任意多边形重心的文章，文章介绍了一种以“划分三角形”为主体思想的算法。大致可以描述如下：先将多边形划分成为若干个三角形（应当讲，这一步是整个算法最复杂的，但由于不属于本文论述范围，不做深入探讨），依次求出这些三角形的重心位置和重量（即面积），然后将所有重心合并，即是所求。在这种算法中，第一步花费较多时间，其单步算法复杂度将在O（N^2）左右，整个算法复杂度较高，而且编程复杂度也很高，不易于实现。

我们知道，如果两个质点质量分别为M1，M2，横坐标为X1，X2，那么它们的合重心将有 $X=\frac{X1+\lambda X2}{1+\lambda}$(其中，$\lambda=\frac{M2}{M1}$)，$M=M1+M2$。我们不妨将这一运算为“重心加法法则”，其中（M1，X1），（M2，X2）就是两个加数，而（M，X）就是和。

再根据减法定义：“已知两个加数的和，和其中的一个加数，求另一个加数的运算叫做减法。”，于是重心可以有加法法则当然就可以有减法法则，依据减法和负数的关系我们还可以引入“负重心”这一概念来帮助解决问题。例如，已知重心位置在（1，0）处，质量10KG，它的一个分重心在（0，0），质量5KG，我们可以求出另一个分重心的位置（2，0）和质量5KG。由此可见，重心运算法则可以有加减、有正负，当然也就可以用“割补”了。只要将上面的面积计算公式迁移到这里来，就不难写出重心位置的计算式：设多边形的顶点分别为（Xi，Yi），质量分布均匀，可得：

依次处理图形的每个顶点：

令

$$\begin{cases} M_i = \dfrac{1}{2}\begin{vmatrix} X_i - X_0 & X_{i+1} - X_0 \\ Y_i - Y_0 & Y_{i+1} - Y_0 \end{vmatrix} \\ P_i = \dfrac{X_i + X_{i+1}}{3} \\ Q_i = \dfrac{Y_i + Y_{i+1}}{3} \end{cases}$$

则

$M := M_i$

$P := P_i \ (M := 0)$

$Q := Q_i$

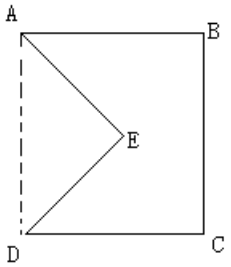
或

$$\begin{cases} k_i = \dfrac{M}{M_i} \\ P = \dfrac{P + k_i P_i}{1 + k_i} \quad (M \neq 0) \\ Q = \dfrac{Q + k_i Q_i}{1 + k_i} \\ M = M + M_i \end{cases}$$

最后，（P，Q）就是所求图形重心坐标，M即为图形质量。这里巧妙的引入了负质量和从而利用割补法简洁明了的特点方便的解决了问题。应当指出的是，该方法与上面提到的《电脑爱好者》中的方法相比，不仅仅是直观明了，更重要是降低了程序复杂度，因为该方法对于每个点至多只处理依次，每次处理也不过是几个简单的乘除运算，显然其运算复杂度只有O（N），不失为一种较为理想的方法。

第3节　判断点与任意多边形的位置关系

判断点与多边形位置关系是一个经典的计算几何问题了，也已经有许多种不错的解决方法，应当讲，在这一方面“割补法”并没有优势，只是作为一种借鉴提出来供参考。例如判断点O是否在图形K内（如图），可以先判断O是否在四边形ABCD内，如果不是，显然不会出现在K以内了，计算到此结束，如果是，则进一步判断O在不在三角形AED内，是则可以判断O在多边形K外，反之也可知O于多边形内，算法到此结束。



借鉴这种思想，针对一些较为复杂的图形可以设计一种广泛适用的算法：

FUNCTION Inside_Check（P1，P2，……PN）：BOOLEAN

1　求图形K中外凸的任意一点Pk

2　h←k

3　WHILE (Ph+1向外凸)AND(h+1<>k) THEN h←h+1

4　t←k

5　WHILE (Pt－1外凸)AND(t<>h) THEN t←t－1

6　flag←FUNCTION Inside_Check（Pt，Pt+1，……，Pk，Pk+1，……，Ph－

1，Ph）

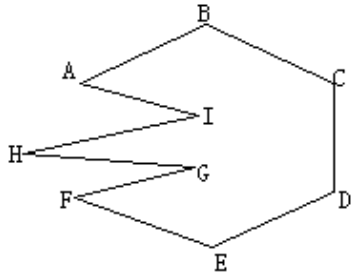
7　map←FUNCTION Inside_Check（Ph，Ph+1，……，Pt－1，Pt）

8　IF flag AND(NOT map) THEN Inside_Check:=true

ELSE Inside_Check:=false

以上过程反复递归求解，每次先将原图分解成一个标准的凸多边形和一个不规则图形，然后再分别判断点是否在这两部分中，最后依据结果判断是否O在多边形K内，一个简单的处理流程如下：

然而，不难发现这个程序是有瑕疵的，例如下图它便不能很好解决，于是又派生出它的一个改进版。



改进后的程序框架如下：

FUNCTION Inside_Check_PRO（P1，P2，……，PN）：BOOLEAN

1　求图形P1，P2，……，PN的一个凸包Pa1，Pa2，……，Pak

2　flag←FUNCTION Inside（Pa1，Pa2，……，Pak）

3　对于任何一串不属于多边形凸包的点及它们两端属于凸包的两个点构成的图形（A1，A2，……，Ap）：map:=map AND (FUNCTION Inside_Check_PRO（A1，A2，……，Ap））

4　IF flag AND(NOT map) THEN Inside_Check:=true

ELSE Inside_Check:=false

其中FUNCITON Inside(P1,P2,…,PN)表示O是否在凸多边形P1,P2,…,PN以内。这里引入凸包之后就很好的解决了原先的“出格”问题了。再观察整个算法流程，每次运算须求一次凸包（算法复杂度为O（N）），须判断一次点与凸多边形的位置关系，复杂度也为O（N），而每次循环至少可以从原图中删除一条边，整个算法递归次数不超过N次，可知算法复杂度在O（N^2）左右，还是比较理想的。但这里并没有很好的利用“割补法”化复杂为简单的特点，而是反复利用割补技术，层层递归反而增加了算法复杂度和编程难度。相比之下，几种O（N）的算法就显示出它们的优势了。其中一种较为简洁的算法可以描述成：若一个点O在图形K内，则由点O发出的任意一条射线与图形K的交点一定为奇数个，反之则一定为偶数个交点。这种方法复杂度为O（N），而且编程难度极低，应当成为这个问题的一种理想解法。

综上所述，割补法在计算几何中诞生，也在这里有极为广泛的应用，大至判断

命题，小至求面积重心等，凡是涉及图形面积的问题大都可以找到它的身影。但通过上面最后一个命题的判断可以总结使用“割补”的一个前提：补完之后的部分必须完整、方便求解，割去的部分亦必须规则或求解方便，否则就失去了方法的根本意义了。割补在计算几何的应用可以说是计算机中的“让步假设从句”——“退一步海阔天空”的最好例证，但并不是在几何中才可以有假设，在其他许多问题同样可以有假设，同样可以有“割补”思想的运用。

第三章　在组合数学中的应用

计算几何中的割补法在组合数学中即表现为计数上的“割补”：欲求解一定范围内满足条件的元素个数，不妨扩大限定范围求解，例如减法原理；抑或在统计中分别求解，再将多余部分删去，例如容斥原理。总之，退一步海阔天空，先放宽条件，再解决问题就方便多了。

第1节　减法原理

这只是一个简单的数学问题而已，可以看成是加法原理和乘法原理的一个引申：假设A地到B，C，D地分别有5条路，但到E地只有3条路，B，C，D，E地与F都有3条路相通，于是不妨假设A——E也有5条路相通，于是A——F道路总数为5×4×3=60条，其中有2×3=6条是我们“杜撰”出来的，于是实际上A——F道路总数应为60－6=54条。

第2节　有禁区的排列问题

我们先介绍有关棋盘多项式的概念。

设C是一个棋盘，$R_k(C)$表示把k个相同的棋子布到C中的方案数。在布棋时我们规定：当一个棋子放到C中的某个格以后，这个格所在的行和列就不能再放其他棋子了，并规定对任意的棋盘C有$R_0(C)=1$

不难得到以下的结果：

$R_1($ □ $)=1$

$R_1($  $)=R_1($  $)=2$

$R_2($ $)=R_2($ $)=0$

$R_2($ $)=1$

可以证明布棋方案数R_k（C）具有下面的性质：

1. 对于任意的棋盘C和正整数k，如果k大于C中的方格总数，则R（C）=0。

2. R_1（C）等于C中的方格数。

3．设 C_1 和 C_2 是两个棋盘，如果 C_1 经过旋转或者翻转变成了 C_2，则 $R_k(C_1)=R_k(C_2)$。

4．设 C_i 是从棋盘 C 中去掉指定的方格所在的行和列以后剩余的棋盘，C_1 是从棋盘 C 中去掉指定的方格以后剩余的棋盘，则有 $R_k(C)= R_{k-1}(C_i)+ R_k(C_1)$ $(k>=1)$。

5．设棋盘 C 由两个子棋盘 C_1 和 C_2 组成，如果 C_1 和 C_2 的布棋方案是互相独立的，则有

$$\mathrm{R_k(C)}=\sum_{i=0}^{k}\mathrm{R_i}(C_1)\cdot R_{k-i}(C_2)$$

定义 1：设 C 是棋盘，则

$$\mathrm{R(C)}=\sum_{k=0}^{\infty}\mathrm{R_k}(C)x^k$$

叫做棋盘多项式。

显然，在上述定义中当 k 大于棋盘的格子数时 R_k（C）=0，所以 R（C）一般只有有限项。例如：R（  ）=R_0（  ）+R_1（  ）x+ R_2（  ）$X^2=1+2X+X^2$

根据 R_k（C）的性质不难得到 R（C）的性质。

1．R（C）=x$R(C_i)$+$R(C_1)$，其中 C_i 和 C_1 的定义如前所述。

2．R（C）=$R(C_i)$×$R(C_1)$ ，其中 C_i 和 C_1 的定义如前所述。

利用这两条性质可以计算棋盘多项式。

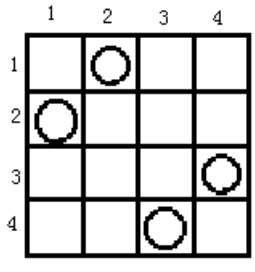
例 1 计算 R（  ）

解：R（  ）=X*R（□）+R（）

=X（1+X）+（1+2X）

=$1+3X+X^2$

下面我们就可以利用棋盘多项式来解决有禁区的排列问题。首先可以看到 X={1，2，3，……，n} 的一个排列恰好对应了 n 个棋子在 $n\times n$ 棋盘上的一种排布。在图中，我们以棋盘的 n 行表示 X 中的元素，列表示位置，则这种放置方案就对应了排列 2143。



放棋方案与排列的对应

如果在排列中限制元素 i 不能放在第 j 个位置，则相应的布棋方案中的棋盘第 i 行第 j 列就不能放置棋子。我们把所有这些不许放置棋的方格称作禁区。

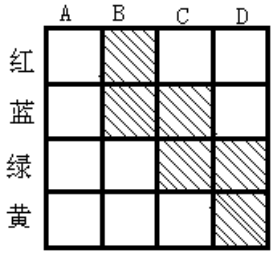
定理 1 设 C 是 $n\times n$ 的具有给定禁区的棋盘，这个禁区对应集合{1，2，……，

n}中的元素在排列中不允许出现的位置。则这种有禁区的排列数是

$$n!-r_1(n-1)!+r_2(n-2)!-\cdots+(-1)^n r_n$$

其中 r_i 是 I 个棋子放置到禁区的方案数。

证明 先不考虑禁区的限制 ，那么 n 个棋子布到 n×n 棋盘上的方案有 n！ •n!个，如果对 n 个棋子分别编号为 1，2，……，n，并且认为编号不同的棋子放入同样的方格是不同的放置方案，那么带编号的棋子布到 n×n 棋盘上的方案数是 n! • n!。我们把这些方案构成的集合记作 S。



涂色的禁区

对 j=1，2，……，令 P_j 表示第 j 个棋子落入禁区的性质，并令 A_j 是 S 中具有性质 P_j 的方案构成的子集，那么所求的排列数就是 $|\overline{A}_1 \cap \overline{A}_2 \cap \cdots \cap \overline{A}_n|$ 。

1 号棋子落入禁区的方案数为 R_1，当它落入禁区的某一格之后，2，3，……，n 号棋子可以任意布置在(n-1) ×(n-1)的棋盘上，由乘法法则得

$$|A_1| = R_1(n-1)!\times(n-1)!$$

同理，对 I=2，3，……，n 有

$$|A_i| = R_1(n-1)!\times(n-1)!,$$

对 I 求和得

$$\sum_{i=1}^{n}|A_i| = R_1(n-1)!\cdot n!$$

1 号和 2 号两个棋子落入禁区的方案数为 $2R_2$，它们落入以后，3，4，…，n 号棋子可以任意布置在（n－2）*（n-2）的棋盘上，所以

$$|A_i \cap A_j| = 2R_2(n-2)!\cdot(n-2)!,$$

对所有的 $1 \le i < j \le n$ 求和得

$$\sum_{1\le i<j\le n}|A_i \cap A_j| = 2\binom{n}{2}R_2(n-2)!(n-2)!,$$
$$= R_2(n-2)!\cdot n!$$

用类似的方法，我们可以求得

$$\sum_{1\le i<j\le n}|A_i \cap A_j \cap A_k| = R_3(n-3)!\cdot n!,$$

……

$$|A_1 \cap A_2 \cap \cdots \cap A_n| = R_n \cdot n!。$$

根据容斥原理，带编号的 n 个棋子都不落入禁区的方案数是

$$\left|\overline{A}_1 \cap \overline{A}_2 \cap \cdots \cap \overline{A}_n\right| = n! - R_1(N-1)! \cdot n! + R_2(n-2)! \cdot n! - \cdots + (-1)^n R_n \cdot n!$$。

需要说明一点，这个定理适用于$n \times n$棋盘的小禁区的布棋问题。如果是$m \times n$的棋盘或者是禁区很大的布棋问题，那么只能直接用R（C）来求解。

例　用四种颜色（红、蓝、绿、黄）涂染四台仪器A，B，C，D。规定每台仪器只能用一种颜色并且任意两台仪器都不能相同。如果B不允许用蓝色和红色，C不允许用蓝色和绿色，D不允许用绿色－和黄色，问有多少种染色方案？

解　这个问题就是图中的有禁区的布棋问题。禁区的棋盘多项式为

R（  ）$= 1 + 6x + 10x^2 + 4x^3$，

从而得到R_1=6，R_2=10，R_3=4，根据定理，所求的方案数是

N=4！－6*3！+10*2！－4*1！=24－36+20－4=4。

例　错位排列问题也可以看作是有禁区的排列问题，其禁区在主对角线上。下面使用定理来求D_n。

解　禁区的棋盘多项式是

R（  ）=R（ ）* R（ ）*……R（ ）

□　□　□　□

$= (1+x)^n$

$= 1 + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$，

从而得到$R_1 = 1$，$R_2 = \binom{n}{2}$，……，$R_n = \binom{n}{n}$，代入定理得

$$D_n = n! - n(n-1)! + \binom{n}{2} \cdot (n-2)! - \cdots + (-1)^n \binom{n}{n} \cdot 0!$$

$$= \left[1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!}\right]。$$

第4章总结

通过上面的例子可以发现，在许多直接求解有困难的问题中，“让步假设”和“割补法”有相当的威力，从某种意义上讲，它们都是将问题所求的目标放大，使

问题简单化，这种模型转化思想应当成为一种较为普遍的思路。另外，除了将目标放大，还可以将目标“反白”，即求目标集合的补集，这种思路在许多问题中也起到了事半功倍的效果，其典型应用有图论中“求最大独立集合”、组合数学中的一些计数问题等。除此之外，文章中还涉及到了许多其他的模型转化问题，在解决它们各自涉及的例题中都发挥了不可替代的作用。模型转化是信息学中的一个热门考点，也确实是一个考验数学功底的难点，文章中谈到的两种典型的转化思想恐怕只是其中的沧海一粟，只希望抛砖引玉，给大家提供一些借鉴。