1993

1996

1993—1996 美国计算机程序设计竞赛试题与解析

吴文虎 主编
赵 鹏 编著

USACO

清华大学出版社
http://www.tup.tsinghua.edu.cn

1993—1996
美国计算机程序设计竞赛
试题与解析

吴文虎　主编
赵　鹏　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书收集了美国中学生程序设计竞赛(USACO)近年来的试题。这些题目趣味性强,许多题目有实际背景,同国际信息学奥林匹克竞赛(IOI)的题目一样,旨在考核选手的智力与利用计算机编程解题的能力。本书的作者是中国参加 IOI 的选手,也是代表清华大学参加 ACM 世界大学生计算机程序设计竞赛的选手。通过阅读本书可以学到用计算机解题的思路、方法与技巧,提高编程解题的能力。本书可供大学生、中学生和程序设计竞赛的爱好者使用。

图书在版编目(CIP)数据

1993—1996 美国计算机程序设计竞赛试题与解析/吴文虎主编;越鹏编著. —北京:清华大学出版社,1998.12
ISBN 7-302-03211-4

Ⅰ.19… Ⅱ.①吴… ②赵… Ⅲ.程序设计-解题 Ⅳ.TP311

中国版本图书馆 CIP 数据核字(98)第 33080 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者:北京市通州区大中印刷厂
发行者:新华书店总店北京发行所
开 本:787×1092 1/16 印张:9 字数:212 千字
版 次:1999 年 1 月第 1 版 1999 年 1 月第 1 次印刷
书 号:ISBN 7-302-03211-4/TP·1715
印 数:0001～5000
定 价:12.00 元

序

程序设计竞赛是青少年学习计算机的一种很好的活动形式，它对开发智力和提高程序设计能力都非常有益，因此，受到世界各国的重视，像美国、俄罗斯和东欧等国家几乎每年都组织这方面的竞赛。为了借鉴国外的好经验，也为了开阔眼界，我们特将近年来美国中学生程序设计竞赛(USACO)的试题及其分析与解法介绍给广大青少年读者。

大家知道，计算机与数学关系最为密切。数学是研究现实世界的数量关系和空间形式的科学。一门科学只有成功地运用数学时才算达到了完善的地步，有了数学模型才有可能被计算机计算。在我们做这套试题解析时，重点是讨论解题的思路，尽力找出可用的数学模型。美国人出的题目有它特有的幽默感，有的看似简单，实则蕴含着较高深的数学问题。分析研究这些题目并给出相应解法，不仅可以提高分析问题和解决问题的能力，还可以提高怎样把书本上学来的数学知识应用于解决实际问题的能力。

程序设计竞赛过程是一种高强度的创造性思维过程，许多题目没有固定的解法，要想获得高效精确的解，往往需要选手根据题意和自己的想象力、判断力，拿出异乎寻常的办法。在这一过程中可以锻炼观察能力、想象能力、判断和决策能力。参加这类高水平的竞赛，除了具备熟练的计算机高级语言编程能力外，还要具备组合数学、图论集合论、基本算法与数据结构的知识，特别是要具有能将实际问题抽象为数学模型的能力。因此，我们可以说：能够参加程序设计竞赛的选手，一定是一些“高手”，他们不仅计算机应用能力强，而且数学等基础知识学得扎实，用得灵活。

这本书的解法是原信息学奥林匹克竞赛中国队的金牌得主做出的，可能会对读者的思路有所启发。但从编写此书的宗旨看，是想起到抛砖引玉的作用。很多赛题的解法可能不是最好的，希望你比我们做得更好。

吴文虎

1998年8月

前　言

USACO 美国中学生程序设计竞赛是美国为了培养中学生的逻辑思考能力、数学推理能力和编程能力并为每年的 IOI(国际青少年信息学奥林匹克竞赛)选拔参赛选手而举办的。

USACO 共包含以下三部分：

1. 资格赛：在每年的二月中旬举行。竞赛委员会将试题以及测试数据寄给各个地区的组织者，这些组织者再将试题发给所有愿意参加竞赛的学生。每次资格赛大约有 5 天的时间，学生可以利用课余的时间在任何地点独立地完成，然后将程序交给竞赛的组织者进行评审。资格赛的题目是三部分中最简单的，要求也是最低的。每次大约有四道题，一般完成两道以上的就有资格进入下一轮竞赛——选拔赛；

2. 选拔赛：在每年的三月中旬举行。仍由竞赛委员会将试题以及测试数据寄给各个地区的组织者，不过参赛的是所有通过资格赛的学生。各地的组织者要选一个合适的时间，将学生组织起来进行竞赛。竞赛的时间大约为 5 个小时，竞赛的题目为难度不同的两道题，希望学生能完整的做出其中的一道题，当然两道都完成是最好的。竞赛后，各地的组织者要将学生的程序以及测试结果寄给竞赛委员会，由他们来决定哪 16 个学生来参加选拔 IOI 参赛选手的总决赛；

3. 总决赛：在每年的四月上旬举行。通过选拔赛的全国各地的学生将被组织在一起进行最后一轮总决赛，选拔参加 IOI 竞赛的国家队队员。总决赛的题目在三部分中是最难的，对选手各个方面能力的要求很高。

总的来说，USACO 的题目趣味性很强，很多题目有它的实际背景。三部分竞赛的题目由易到难，适合各个层次的学生钻研。

本书第一章是一个竞赛的讲座“全面地考虑问题”。这是编程的基础，同时也是最容易被忽视的部分。希望读者通过阅读这个讲座能对考虑问题的全面性加以重视。在之后的两章中，精选了近年来 USACO 程序设计竞赛的题目，并编写了相应的解题思路和参考程序。

本书的编写是在中国计算机学会普及工作委员会主任、清华大学计算机系吴文虎教授的精心指导下完成的。他亲自参与了本书的选题与规划，并且仔细审阅和修改了本书的初稿，确保了本书编写任务的高质量完成。在编写的过程中，得到了 IOI 中国队教练组全体成员的大力支持与协助，在此向他们表示由衷的感谢。

由于作者水平有限，书中难免有不足和疏漏之处，恳请广大读者提出宝贵意见。

编著者

1998 年 5 月于清华园

目　录

第1章

专题讲座——全面地考虑问题

在编程序时常常会遇到这样的问题：一道很简单的题目，编出的程序却错了很多测试点。这其中的主要原因是由于考虑问题不全面，只想到了一些普通的情况，而遗漏了很多特殊的地方。

下面通过几个例子来进行讨论。

1. 项链(IOI'93 第一题)

由 n(n≤100)个珠子组成一个项链，珠子有红、蓝、白三种颜色，各种颜色的珠子的安排顺序由输入文件任意给定。

图 1.1 可看作由字符 b(代表蓝色珠子)和字符 r(代表红色珠子)所组成的字符串。假定从项链的某处将其剪断，把它摆成一直线，从一端收集同种颜色珠子(直到遇到另一种颜色的珠子时停止)。然后再从另一端重复上述过程(请注意，这一端珠子的颜色不一定和另一端珠子的颜色相同)。

brbrrrbbbrrrrbrrbbrbbbbrrrrb

图 1.1

请选择项链被剪断的位置，目标是使两端各自颜色相同的珠子数目之和最大。例如，对于上图(只有红蓝两种颜色)，最大值 M 是 8，断点位置在珠子 9 和珠子 10 之间，或珠子 24 和珠子 25 之间。

项链中可以有三种颜色用 b(蓝)、r(红)和 w(白)表示。白色既可看成是红色，又可看成蓝色。

(1) 一个 ASCII 文件 NECKLACE. DAT 中的内容：该文件中每一行代表某个项链中各种颜色珠子的配置。把输出内容写入 ASCII 输出文件 NECKLACE. SOL 中。

作为举例，输入文件的内容可以是：

brbrrrbbbbrrrrrbbrbbbbrrrrb
bbwbrrrwbrbrrrrb

(2) 对于给定的每个项链的配置，求出收集到的珠子数的最大值 M 及相应的断点位置(注意可能存在多个位置)。

(3) 在输出文件 NECKLACE. SOL 中写入收集到的珠子数的最大值 M 及断点位置。

例如：

brbrrrbbbrrrrrbbrbbbbrrrrb

8 between 9 and 10

bbwbrrrwbrbrrrrrb

10 between 16 and 17

作为竞赛的第一题，这道题目显然是比较简单的题目。它只包含两个步骤：剪断项链和收集同颜色的珠子。例如下面的一条项链(a)从 N＝3 处断开变为项链(b)。这个操作只需要将前 N 个珠子移到后边即可。

brb| rrwb ⟶ rrwbbrb

(a) (b)

现在只剩下收集同颜色的珠子这一步，根据上面的例子我们很容易写出下面的程序。

```
用变量 c 来记录最左边珠子的颜色;
Left:=0;
  FOR i←1 TO 项链长度 DO
    IF 左数第 i 个珠子的颜色与 c 相同  OR  此珠子为白色
      THEN Inc(Left)
      ELSE Break;
```

这样变量 Left 中存放的就是从左边收集到的珠子的数目，同理可求得从右边收集到的珠子的数目 Right，则所求的值为 Left＋Right。这个程序显然能通过上面的例子，由于这是一道简单的题目，谁也不想在它上面多费时间，往往做到此为止。可是如果仔细想想，再举几个例子，就会发现错误。上面的那条项链断开后左右两个珠子为红色和蓝色，在题目中这两种颜色的珠子都比较“普通”，只有白色的珠子比较“特殊”。所以应举一个断开后左右两端有白色珠子的例子。还是上面那条项链从 N＝6 处断开。

brbrrw| b ⟶ bbrbrrw

正确答案应是收集到 5 个珠子：左边 2 个，右边 3 个。而上面的程序得到的结果却是 3 个：左边 2 个，右边 1 个。错误就在于没有考虑到左右两端有白色珠子的情况。一种较容易的解决方案是先将左右两端的白色珠子均取下，记其数目为 Other，再用上面的程序来求，结果为 Left＋Right＋Other。我们解决了左右两端出现白色珠子的情况，还有没有别的特殊情况呢？一个真正“特殊”的项链不应包含所有颜色的珠子，最好只包含一种颜色。如下面的项链是由 10 个红色的珠子组成。

rrrrrrrrr

用我们的程序得出的结果是 20 个，显然是不对的。因为题目中要求是收集珠子而不是数珠子，所以最后得到的总数不应超过珠子的总数。这只是一个字眼的问题，却使当年中国队的选手失了不少分。一个简单的改正措施是判断最后的结果是否大于珠子总数，如果是则输出珠子的总数即可。

虽然项链这道题比较简单，却很难“简单”地得到满分，最容易出的错误就是考虑的不全面。

2. 多项式加法

由文件输入两个多项式的各项系数和指数，试编程求出它们的和，并以手写的习惯输出此多项式。

要求：

(1) 多项式的每一项 ax^b 用 axb 的格式输出。

(2) 两个多项式在文件中各占一行，每行有 2m 个数，依次为第一项的系数，第一项的指数，第二项的系数，第二项的指数……

例如输入文件为：

1 2 3 0

−1 1

输出：x2−x+3

此题是一道大学生竞赛的题目，很多人只用了很短的时间就编出程序。但最后测试的结果却令他们很惊讶：通过的数据还不到一半！他们犯的错误归根结底就是考虑得不够全面。

此题对于多项式相加的过程很简单，只要找出幂次相同的项相加即可。关键在于题目中要求用符合手写的习惯输出结果。何为手写的习惯呢？例如多项式 $3x^2-x$ 中就有很多手写的习惯。我们不会将其写成 $3x^2-1x^1+0$。因为首先当某项系数为 1 时，我们习惯于不写系数；其次对于一次项我们也要省略指数；还有我们从来不写出系数为 0 的项。一个简单的多项式就有这么多的手写习惯，我们已经感觉到了要把这题全面地做出很不容易。虽然我们平时总在写多项式，但是谁也不会留心我们写多项式时的习惯。我们写多项式的习惯究竟有哪些呢？

(1) 首先我们考虑对于多项式中的任一项 ax^b 它有多少手写习惯：

◇ 当 a＝0 时，此项省去不写。

◇ 当 a＝1 时，省去 a。

◇ 当 a＝−1 时，系数只写一个负号‘−’。

◇ 当 b＝0 时，省去 x 和 b。

◇ 当 b＝1 时，省去 b。

◇ 当 a＜−1 时，省去此项前面的加号(首项除外)。

我们一口气写了这么多条规则，每一条看起来都很正确，但合在一起是否还正确呢？当 a＝1 或−1 时要省去其中的数字 1，这是针对一般情况而言。如果 b＝0，则数字 1 就不应当省去。所以我们不仅要单独考虑 a 和 b，而且要将其和起来考虑。

(2) 其次对于整个多项式有哪些规则呢？

◇ 多项式的首项系数前不应有加号‘＋’。

◇ 如果一个多项式为零多项式，则应写出数字‘0’。

现在看起来这道题并不是一道很容易的题目。它需要一个人在很短的时间内能全面地总结出上述那么多规则。这对一个人的全面考虑问题的能力是一个很好的检验。

3. 求最长的公共子串(NOI'93 第一题)

求 N 个字符串的最长公共子串,N<20,字符串长度不超过 255。例如 N=3,由键盘依次输入 3 个字符串为

What is local bus ?
Name some local buses.
local bus is a high speed I/O bus close to the processor.

则最长公共子串为"local bus"。

此题也是作为第一题出现,同样有很多人在此题上失分。我们都做过求 n 个数最大公约数的问题,在那道题中求 3 个数的最大公约数时,可以先求两个数的最大公约数,再将此数与第三个数求一次最大公约数。有人从那道题中得到"启发",设 s(p,q)为字符串 p 和 q 的最长公共子串,则 p、q、r 的最长公共子串为 s(s(p,q),r)。这样只需编写一个求两个字符串的最长公共子串的过程即可。但这种方法对不对呢?看看下面的例子。

三个字符串分别为'abc'、'cab'、'c',则 s(p,q)='ab',s(s(p,q),r)=''。事实上这三个字符串有公共子串'c'。显然上面的算法是错误的,原因在于没有考虑到本题与求最大公约数那道题在性质上的不同之处。最大公约数可以由局部解得到全局解,而本题却不能。正确的做法是列举出其中一个字符串的所有子串,找出其中最长的而且是公共的子串。

```
FOR i:=1 TO 第一个字符串的长度 DO
  FOR j:=i TO 第一个字符串的长度 DO
    IF (第 i 个字符到第 j 个字符的子串为公共子串
      AND (j-i+1>当前找到的最长公共子串的长度 max)
    THEN BEGIN
           max←j-i+1;
           最长公共子串←此子串;
         END;
```

为了提高效率,我们可以将最短的字符串作为第一个字符串。此题需要考虑的并不像多项式加法那道题那么多,但是它提醒我们在不清楚题目的性质之前,不能滥用以前的方法。

4. 可重复排列(NOI'94 第一题)

键盘输入一个仅由小写字母组成的字符串,输出以该串中任取 M 个字母的所有排列及排列总数(输入数据均不需判错)。

此题是由全排列问题转变而来,不同之处在于:一个字符串中可能有相同的字符,导致可能出现重复的排列。例如从字符串'aab'中取 2 个字符的排列只有三种:'aa'、'ab'、'ba'。如何去掉那些可能重复的排列呢?一种想法就是每产生一种不同的排列就记录下来,以便让以后产生的排列进行比较判重。这种想法显然没有考虑到随着字符串长度的增加,排列将会多得无法记录,而且这种判重方法在效率上也会很低。最好有一种方法能在

产生排列的过程中就能将重复的去掉。先看一看全排列的递归过程：

```
PROCEDURE Work(k);
  BEGIN
    IF k=m+1
      THEN 打印结果
      ELSE FOR i←1 TO 字符串长度 n DO
             IF i<>e[1],e[2]…e[k-1]
               THEN
                 BEGIN
                   e[k]←i;
                   Work(k+1);
                 END;
END;
```

让我们来分析产生重复的原因。考虑从字符串'aab'中取 2 个字符的排列。当 e[1]从 1 变到 2 时，字符串中的字符却没有变，都是'a'。这样我们只要在改变 e[k]时，判断其对应的字符是否也改变。即在上面的过程的循环中加一句判断(设字符串为 s)：IF s[i]<> s[e[k]] THEN …；这当然只是一个粗略的想法，我们仅仅用上面的例子就能发现问题：程序在对 e[k]的每一次赋值之前都要进行一次判断，而不是我们预想的在改变 e[k]时才进行判重。我们用一个布尔型的局部变量 First 来记录是否是对 e[k]进行第一次赋值。修改后的程序如下：

```
PROCEDURE Work(k);
  BEGIN
    First←True;
    IF k=m+1
      THEN 打印结果
      ELSE FOR i←1 TO 字符串长度 n DO
             IF (i<>e[1],e[2]…e[k-1]) AND
               (First OR s[i]<>s[e[k]])
               THEN
                 BEGIN
                   e[k]←i;
                   First←False;
                   Work(k+1);
                 END;
  END;
```

很多选手的程序到此就为止了，可是它还有一个致命的错误：我们在判重时假定这个字符串中的字符已经排好顺序，即相同的字符已经连在一起。事实上并不是这样，输入的字符串中的字符排列是任意的，需要我们在递归之前作一次排序的初始化才能保证程序运行得正确。可见虽然本题的难点是在递归过程中，但是那些简单的初始化却是程序最容

易忽略，最容易出错的部分。

5. 剔除多余括号(IOI'94 国家队选拔赛第一题)

键盘输入一个含有括号的四则运算表达式，可能含有多余的括号，编程整理该表达式，去掉所有多余的括号，原表达式中所有变量和运算符相对位置保持不变，并保持与原表达式等价。

例：输入表达式　　应输出表达式

a+(b+c)　　a+b+c

(a*b)+c/d　　a*b+c/d

a+b/(c-d)　　a+b/(c-d)

注意输入 a+b 时不能输出 b+a。

表达式以字符串输入，长度不超过 255。输入不要判错。

所有变量为单个小写字母。只是要求去掉所有多余括号，不要求对表达式化简。

同多项式加法一样，此题要求的也是一个我们平时很习惯但却没有注意的规则：去掉多余的括号。哪些括号是多余的呢？这要根据紧挨着括号前后的运算符号和括号中最后一个运算符号来决定。例如表达式 a+(b*c+d)-e 中，括号前的符号为"+"，括号后的符号为"-"，括号中最后一个运算符号为"+"，所以括号是多余的。总结括号中最后一个运算符号为"+"、"-"、"*"、"/"时，括号前、后为何运算符号时括号是多余的，我们很容易得出表 1.1。

表 1.1　多余的括号满足的条件

括号前的符号	括号中的符号	括号后的符号
+	+	+、-
+	-	+、-
+、-、*	*	+、-、*、/
+、-、*	/	+、-、*、/

上表只是对一般情况的分析，显然是很不全面的。首先括号前，括号中和括号后都可能无运算符号，例如表达式((a))+b；其次变量前还可能带有负号，例如表达式(-a)+(-b)中的括号一个是多余的，一个不是多余的。还有一些其它的情况如表达式只有括号无任何变量等需要考虑。此题留给大家自己去完成。注意多用一些特殊的数据来测试自己的程序。大家也可以试着用表达式树的方法来做此题。

6. 合并表格(NOI'93 第二题)

在两个文本文件中各存有一个西文制表符制成的未填入任何表项的表结构，分别称之为表 1 和表 2，要求编程将表 1 和表 2 按下述规则合并成表 3。

规则：表 1 在上表 2 在下，表 1 与表 2 在左边框对齐，将表 1 的最底行与表 2 的最顶行合并。

例如,3 张表见图 1.2。

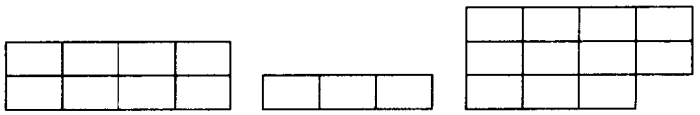


图 1.2

编程要求:

(1) 程序应能从给定的文本文件中读入两个源表并显示。

(2) 若源表有错,应能指出其错(错误只出在表 1 的最底行和表 2 的第一行)。

(3) 将表 1 和表 2 按规则合并成表 3,并显示之。

(4) 所有制表符的 ASCII 码应由选手自己从给出的示例文件中截取。

我们把此题的分析留给读者去独立完成。

"千里之堤,毁于蚁穴"。一些程序往往不是错在算法上,而是错在考虑得不全面上。希望通过以上几个例子,能使大家对此引起重视,在以后的编程中多加注意,避免不应有的遗憾。

第2章

1993—1996
美国计算机程序设计竞赛试题

2.1 1993年美国计算机程序设计资格赛试题

第一题：分数变小数

写出一个程序，接受一个以N/D的形式输入的分数，其中N为分子，D为分母，输出它的小数形式。如果它的小数形式存在循环节，要将其用括号括起来。例如：1/3=.33333...表示为.(3)，又如41/333=.123123123...表示为.(123)。

一些转化的例子：

1/3=.(3)

22/5=4.4

1/7=.(142857)

3/8=.375

45/56=.803(571428)

用上面的分数和11/59来测试你的程序。

运行举例：

```
ENTER N,D : 1 7
1/7 = .(142857)
```

本题中，0≤N≤65535，0≤D≤65535，设运算结果小数点后最多保留100位。

第二题：质数竖式

下面的竖式是一个乘法运算问题，它的每个*号可以代入一个数字，这个数字属于一个特定的由N个数字组成的集合。如果这个集合是{2,3,5,7}那么这个竖式称作“质数竖式”。

```
    * * *
  ×   * *
-----------
  * * * *
```

```
  *   *   *   *
 ...................
  *   *   *   *   *
```

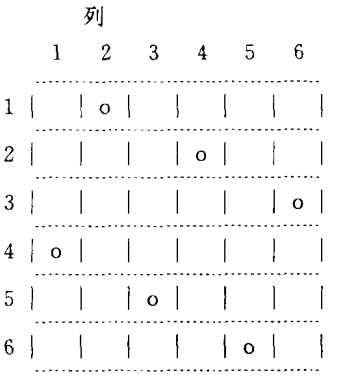
写一个程序找出对应于集合{1,2,3,4,5,6,7,8,9}的任意一个子集的所有竖式。用集合{2,3,4,6,8}和质数集合{2,3,5,7}来测试你的程序。

运行举例：

```
ENTER A SET OF DIGITS:23468
    2   2   2
X       2   2
 ...................
    4   4   4        <还有 3 个竖式未显示>
4   4   4
 ...................
4   8   8   4
The number of unique solutions = 4
```

第三题：6×6 棋子挑战

观察下面的 6×6 棋盘，你可以看到 6 个棋子放在了这个棋盘上，使得每行每列和每个对角线上只有一个棋子(对角线不仅仅是两个主对角线，它是指从东南到西北和从西南到东北的所有对角线)。



上面那组解被描述为：2 4 6 1 3 5，即第 1 至第 6 行每行中棋子所在的列。

```
ROW       1 2 3 4 5 6
COLUMN    2 4 6 1 3 5
```

这是 6×6 棋子挑战的一组解。写一个程序找出所有不同的解，以上面所列的形式打印出来，并统计总数(对称和旋转后相同的解视为不同的解)。

运行举例：

```
2 4 6 1 3 5
```

3 6 2 5 1 4
? ? ? ? ? ?
? ? ? ? ? ?

THERE ARE ? SOLUTIONS TO THE
6×6 CHECKER CHALLENGE.

第四题：跳棋难题

大小为 3 的“跳棋难题”包括 3 个白石子、3 个黑石子和一个有 7 个洞的长条木头。同样颜色的石子分别放在长条木头的每一端，中间的洞为空。

初始状态：WWW　B B B

目标状态：B B B　WWW

为了解决这个难题，采用两种移动方法。将一个石子移入相邻的空洞或者跳过一个相反颜色的石子进入空洞(朝目标方向)。你不可以后退或者跳过两个石子。一个大小为 N 的“跳棋难题”包括 N 个白石子、N 个黑石子和 2N+1 个洞。

写一个程序，解决大小为 N(N≤10)的跳棋难题，并显示出每步移动后的状态。用 W 代表白石子，B 代表黑石子，空格代表空洞。用 N=3 和 N=4 来测试你的程序。

运行举例：

N = 3

WWW　B B B
WWWB　B B
WW　BWB B
W　WBWB B
WBW　WB B
WBWBW　B
WBWBWB
WBWB　BW
WB　BWBW
　BWBWBW
B　WBWBW
B BW　WBW
B BWBW　W
B BWB　WW
B B　BWWW
B B B　WWW

第五题：全部拉丁方阵

下面的数字方阵

1 2 3 4 5

2 1 4 5 3
3 4 5 1 2
4 5 2 3 1
5 3 1 2 4

是一个 5×5 的拉丁方阵，其中数字 1 到 5 在每行每列中只出现一次。

写一个程序，计算第一行是：

1 2 3 4 5 … N

的 N×N 拉丁方阵的个数。

你的程序应该对 2 到 6 中的任意一个 N 都有效。

用 N＝4 和 N＝5 来测试你的程序。

运行举例：

```
ENTER A WHOLE NUMBER (2－6)：4
THE NUMBER OF 4 ×4 LATIN
SQUARES IS 24.
```

2.2 1993 年美国计算机程序设计总决赛试题

第一题：分数排序

考虑在 0 和 1 之间的所有分母不大于 N 的最简分数。下面是 N＝5 时的情况：

0/1 1/5 1/4 1/3 2/5 1/2 3/5 2/3 3/4 4/5 1/1

写出一个程序对于一个给定的整数 N(1≤N≤100)，按从小到大的顺序打印出这些分数。你还应该打印出它们的总数。在每个分数后面打印一个制表符，使他们在显示的时候一行不会很长。

要求：当 N＜1 或 N＞100 时，程序应判错。

运行举例：

```
Enter the maximum denominator：5
0/1  1/5  1/4  1/3  2/5  1/2  3/5  2/3  3/4  4/5  1/1
There were 11 fractions.
```

第二题：最大的谷仓

一个农夫想在自己的农场上建一个尽可能大的谷仓。但是他的农场上有树木及其他建筑物，他不想移动它们中任何一个。为了简单，农场用一个 M×N 的网格代表，包含占用整数格的树木及其他建筑物。长方形的谷仓不能与它们中的任何一个接触。如果农场的大小为 5×6：

```
  1  2  3  4  5  6
1 XX
2
```