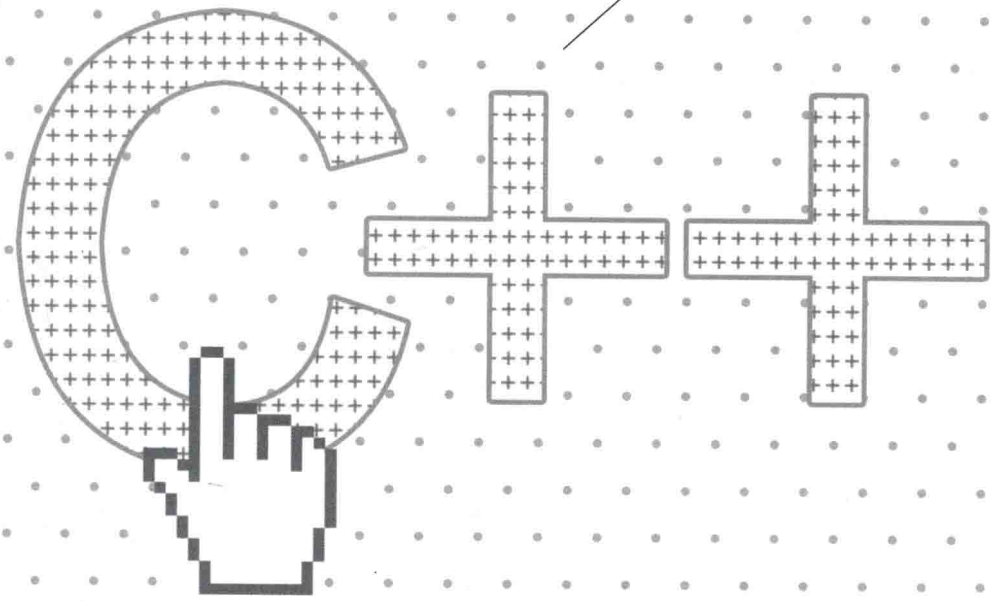
异步图书
www.epubit.com.cn

92 个饶有兴趣的算法，让读者轻松阅读中学通算法，
每个算法都给出完整的 C++ 解决方案
6 个核心视频帮助读者深入学习算法



趣题学算法

徐子珊◎著

涵盖计数问题、信息查找问题、组合优化问题、
图中搜索问题和数论问题
讲解了算法的构思和设计，如渐增策略、分治策略、回溯策略、
动态规划和贪婪策略、广度优先搜索策略、深度优先搜索策略等

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

图书在版编目（CIP）数据

趣题学算法 / 徐子珊著. -- 北京 : 人民邮电出版社, 2017.4
ISBN 978-7-115-44287-1

Ⅰ. ①趣… Ⅱ. ①徐… Ⅲ. ①计算机算法 Ⅳ. ①TP301.6

中国版本图书馆CIP数据核字(2017)第020971号

内 容 提 要

本书共分 10 章。第 0 章讲解了算法的概念及体例说明。第 1～7 章分别就计数问题、信息查找问题、组合优化问题、图中搜索问题和数论问题展开，讨论了算法的构思和设计，详尽介绍了解决这些问题的渐增策略、分治策略、回溯策略、动态规划和贪婪策略、广度优先搜索策略、深度优先搜索策略等。第 8 章提供了 10 个让读者自解的计算问题，让读者有机会小试牛刀。第 9 章用书中给出的各问题的 C++解决方案作为例子，讨论了 C++语言的强大编程功能。书中一共收录了 92 个饶有兴趣的计算问题，每个问题（包括第 8 章留给读者自解的题目）都给出了完整的 C++解决方案。

本书适于作为程序员的参考书，高校各专业学生学习“数据结构”“算法设计分析”“程序设计”等课程的扩展读物，也可以作为上述课程的实验或课程设计的材料，还可以作为准备参加国内或国际程序设计赛事的读者的赛前训练材料。

◆ 著　　徐子珊
责任编辑　张　涛
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：800×1000　1/16
印张：25.75
字数：579 千字　　2017 年 4 月第 1 版
印数：1 – 2 500 册　　2017 年 4 月河北第 1 次印刷

定价：69.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第 8052 号

前 言

念大学的时候曾经读过一本名字叫作 *Problem-Solving Through Problems*（by Loren C. Larson 1983）的数学书，我对这本书的印象非常深刻。除了文字清新、易读易懂之外，这本书还让我明白了数学是学习科学、认识世界的有力工具，更重要的是它给了我一些很基本的也是应用极广的解决现实问题的数学方法和思想，例如，构造等价问题、绘制图形、利用对称性质、讨论限制条件、考虑极端情形等。直至今天，我还经常受惠于这些在我头脑中沉淀的思想。

Problem-Solving Through Problems 之所以让读者感到易读易懂且印象深刻，应归功于它的一个写作特点，用今天的话来说就是“问题驱动”。它的每一章总是先提出一个引人入胜或有着广泛生活背景的问题，通过对问题的分析、理解引入相关的数学知识与解题思路，并用讨论得到的思想和方法解决一系列问题，使读者在阅读过程中产生兴趣，带动读者进行深入思考，通过解决问题激发读者进一步探索的好奇心及继续阅读的热情。

在机械和电子技术占主导地位的时代，数学思想扮演着科学与工程技术基石的角色。一本好的传播数学思想的书使学习科学与技术的年轻人受益终身。在信息时代，生活中几乎所有的活动都与计算有关。向朋友发的短信中的每个字符，行驶在高速路上的汽车中发动机的工作状态，载人航天器与空间站对接时连接口的接驳，都是某种计算的结果。计算有简单的也有复杂的。学习和掌握计算思想——集中体现在解决计算问题的算法及其计算机程序实现上——是现代人认识世界的最基本的也是最重要的思想工具。写一本问题驱动的算法、编程的书，为致力于学习信息技术的年轻朋友培养自身的计算思想素养助一臂之力，是我编著本书的动因。

本书将计算问题分成若干类，并对一类问题提出解决这类问题的若干经典算法思想，通过对若干个这类问题的算法设计与分析，与读者一起认识与体会这些经典算法设计的思想方法。第 1～7 章分别就计数问题、数据集合与信息查找、现实模拟、组合优化问题、动态规划与贪婪策略、图的搜索算法及数论问题展开讨论。每一类问题都是通过提出和解决十几个饶有兴趣的题目来展开的，对每一个题目均详尽地探讨了数据输入/输出的规范，以及解决一个测试案例的算法构思、算法描述和算法运行效率的分析。涉及的算法设计思想包括渐增策略、分治策略、回溯策略、动态规划、贪婪策略、深度与广度优先搜索等。

书中各章讨论、解决的问题很多有着实际的应用背景，例如问题 1-7 的糟糕的公交调度、问题 2-9 的通信系统、问题 3-10 的符号导数、问题 5-5 的人类基因功能、问题 5-12 的最短

路及问题 6-10 的电网等。有一些问题则是计算机科学的基本问题的简化或雏形，例如问题 2-10 的计算机调度、问题 3-8 的内存分配、问题 4-6 的命题逻辑和问题 5-2 的形式语言等。还有一些问题则是当前一些热门课题的缩影，如问题 3-5 的稳定婚姻问题（实际上是经济理论中稳定匹配的简化模型）、问题 4-4 的一步致胜（棋类游戏的搜索算法）以及问题 7-12 的 RSA 因数分解（RSA 密钥问题）等。通过对这些问题的研讨，读者能得到一些解决实际问题的灵感，在遇到计算机专业领域课题时会有某种亲切感，在科研领域中多一件计算思维工具。

计算机科学是教学理论与实践结合最紧密的学科。几乎所有的理论算法都可以立即在计算机上用程序加以验证。对书中的每一个计算问题，都给出了完整的 C++解决方案，代码文件分别存储于各自的文件夹内。这些程序均在 Microsoft Visual C++ 2010 上调试通过。所有的代码和视频都放在百度空间中，下载地址为：http://pan.baidu.com/s/1c22U7PA。为防云盘可能受政策、服务变动等不可测因素的影响，各位可以到地址为 https://github.com/xuzishan/Algorithm-learning-through-Problems/tree/master 的 github 仓库提取源代码。

书中各章内容相对独立，且按先易后难的顺序编排，即使在同一章内，题目也是按先易后难的顺序编排的，所以，对于算法和编程初入门的读者而言，可以先读懂前几章，每章中先读懂前 3 个题目，有了兴趣，再往后研读。喜欢追根溯源（笔者非常赞赏）的读者，如果对算法构思的理论背景在本书的文本介绍中仍感到不满足，可以在百度云中下载（地址同前）深入讨论相关课题的视频资料做进一步探究。对于高校相关课程的教师读者而言，第 1～2 章内容能满足程序设计课程的课程设计材料的需求，第 2～3 章的内容能满足数据结构课程实验材料需求，第 4～5 章适于算法设计与分析课程的基本实验材料需求，第 6～7 章可作为算法课程实验扩展材料。厨师都知道“众口难调”。笔者虽非厨师，但对此道理有着深切的体会，在写作中，尽量满足大家的需求。这本书若能让不同层次、不同需求的读者从中获得各自之需，实在是笔者的衷心所愿。“智者千虑，必有一失”，因而书中必会存留拙笔，望读者不吝赐教，让本书的后续版本不断完善。

笔者创建了读者 QQ 群“算法与程序”（210847302），欢迎加入参加讨论（新加入时请附言“读者”）。本书编辑联系和投稿邮箱为 zhangtao@ptpress.com.cn。

徐子珊

记于依林书斋

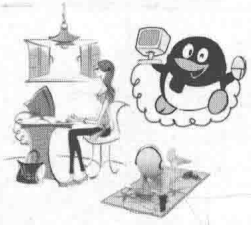
目 录

Chapter 0

从这里开始

0.1 App 程序与算法

信息时代，人们时刻都在利用各种 App 解决生活、工作中的问题，或获取各种服务。早晨，手机里设定的闹钟铃声（或你喜欢的音乐）将你唤醒。来到餐厅，你用手中的 IC 卡到取餐处的刷卡机上支付美味早餐的费用。上班途中，打开手机上的音乐播放器，用美妙的乐声，打发掉挤在公交车上的乏味的时间。上班时，利用计算机中的各种办公软件处理繁忙的业务。闲暇时，你用平板电脑里的视频程序看一部热映的大片，或在淘宝网上选购你喜欢的宝贝。晚间，你用 QQ 或 Facetime 与远方的朋友聊天、交流情感……凡此种种，不一而是。



五彩缤纷的 App 后面是什么？这些神奇的体验是怎么创造出来的？如果你对这样的问题感兴趣的话，我们就成为朋友了。从这里开始，我们将探索创造 App——计算机（及网络）应用程序的基本原理和基本技能。

其实，能用计算机（包括各种平板电脑、智能手机）解决的是所谓的**“计算问题”**，也就是有明确的**输入**与**输出**数据的问题。解决计算问题就是用一系列基本的数据操作（算术运算、逻辑运算、数据存储等）将输入数据转换成正确的输出数据。能达到这一目标的操作序列称为解决这一计算问题的**“算法”**。App 就是在一定的计算机平台（计算机设备及其配备的操作系统）上，用这个计算机系统能识别的语言来实现算法的程序。

因此，对上述的第一个问题，我们的答案是：App 背后的是算法。算法是解决计算问题的方案。要用计算机来解决应用问题，首先要能将该问题描述成计算问题——即明确该问题的输入与输出数据。只有正确、明白地描述出计算问题，才有可能给出解决该问题的算法。作为起点，本书并不着眼于如何将一个实际的应用形式化地描述为一个计算问题，而是向你描述一些有趣的计算问题，并研究、讨论如何正确、有效地解决这些问题。通过对这些问题的解决，使我们对日常面对的那些 App 有着更清醒、更理智的认识。可能的话，也许哪一天你也能为你自己，或者朋友、爱人创造出你和他们喜欢的 App。

0.2 计算问题

上面已经说到什么是计算问题，下面就来看一个有趣的计算问题。

问题 0-1 计算逆序数

问题描述

这个学期 Amy 开始学习一门重要课程——线性代数。学到行列式的时候，每次遇到对给定的序列计算其逆序数，她都觉得是个很闹心的事。所以，她央求她的好朋友 Ray 为她写一段程序，用来解决这样的问题。作为回报，她答应在周末舞会上让 Ray 成为她的伦巴舞舞伴。所谓序列 A 的逆序数，指的是序列中满足 $i<j$，$A[i]>A[j]$的所有二元组$<i, j>$的个数。

输入

输入文件包含若干个测试案例。每个案例的第一行仅含一个表示序列中元素个数的整数 N（$1 \leqslant N \leqslant 500000$）。第二行含有 N 个用空格隔开的整数，表示序列中的 N 个元素。每个元素的值不超过 1 000 000 000。N=0 是输入数据结束的标志。

输出

每个案例仅输出一行，其中只有一个表示给定序列的逆序数整数。

输入样例

```
3
1 2 3
2
2 1
0
```

输出样例

```
0
1
```

这是本书要讨论，研究的一个典型的计算问题。理解问题是解决问题的最基本的要求，理解计算问题要抓住三个要素：输入、输出和两者的逻辑关系。这三个要素中，输入、输出数据虽然是问题本身明确给定的，如果输入包含若干个案例则要理清每个案例的数据构成。

例如，问题 0-1 的输入文件 *inputdata*（本书所有计算问题的输入假设均存于文件中，统一记为 *inputdata*）中含有若干个测试案例，每个案例有两行输入数据。第 1 行中的一个整数 N 表示案例中给定序列的元素个数。第二行含有表示序列中 N 个元素的 N 个整数。当读取到的 N=0 时意味着输入结束。

所谓输入、输出数据之间的逻辑关系，实质上指的是一个测试案例的输入、输出数据之间的对应关系。为把握这一关系，往往需要认真、仔细地阅读题面，在欣赏题面阐述的故事背景之余，应琢磨、玩味其中所交代的反应事物特征属性的数据意义，以及由事物变化、发

展所引发的数据变化规律，由此理顺各数据间的关系，这是设计解决问题的算法的关键所在。

例如，如果我们把问题 0-1 的一个案例的输入数据组织成一个数组 $A[1..N]$，我们就要计算出序列中使得 $i<j$，$A[i]>A[j]$成立的所有二元组$<i, j>$，统计出这些二元组的数目，作为该案例的输出数据加以输出——作为一行写入输出文件 *outputdata*（本书所有计算问题的输出假设均存于文件中，统一记为 *outputdata*）。

对问题有了正确的理解之后，就需要根据数据间的逻辑关系，找出如何将输入数据对应为正确的输出数据的转换过程。这个过程就是通常所称的“算法”。通俗地说，算法就是计算问题的解决之道。

例如，对问题 0-1 的一个案例数据 $A[1..N]$，为计算出它的逆序数，我们设置一个计数器变量 *count*（初始化为 0）。从 $j=N$，$A[j]$开始，依次计算各元素与其前面的元素（$A[1..j-1]$）构成的逆序个数，累加到 *count* 中。当 $j<2$ 时，结束计算返回 *count* 即为所求。

0.3 算法的伪代码描述

上一节最后一段使用自然语言（汉语）描述了解决“计算逆序数”问题的算法。即如何将输入数据转换为输出数据的过程。在需要解决的问题很简单的情况下（例如“计算逆序数”问题），用自然语言描述解决这个问题的算法是不错的选择。但是，自然语言有一个重要特色——语义二岐性。语义二岐性在文学艺术方面有着非凡的作用：正话反说、双关语……。由此引起的误会、感情冲突……带给我们多少故事、小说、戏剧……。但是，在算法描述方面，语义二岐性却是我们必须避免的。因为，如果对数据的某一处理操作的表述上有二岐性，会使不同的读者做出不同的操作。对同一输入，两个貌似相同的算法的运行，将可能得出不同的结果。这样的情况对问题的解决可能是灾难性的。所以，自然语言不是最好的描述算法的工具。

在计算机上，算法过程是由一系列有序的基本操作描述的。不同的计算机系统，同样的操作，指令的表达形式不必相同。本书并不针对特殊的计算机平台描述解决计算问题的算法，我们需要一个通用的、简洁的形式描述算法，并且能方便地转换成各种计算机系统上特殊表达形式（计算机程序设计语言）所描述的程序。描述算法的通用工具之一叫**伪代码**。例如，解决上述问题数据输入/输出的伪代码过程可描述如下。

```
1 打开输入文件 inputdata
2 创建输出文件 outputdata
3 从 inputdata 中读取案例数据 N
4 while N>0
5   do 创建数组 A[1..N]
6      for i←1 to N
```

```
7          do 从 inputdata 中读取 A[i]
8       result←GET-THE-INVERSION(A)
9       将 result 作为一行写入 outputdata
10      从 inputdata 中读取案例数据 N
11 关闭 inputdata
12 关闭 outputdata
```

其中，第 8 行调用计算序列 $A[1..N]$的逆序数过程 GET-THE-INVERSION(A)是解决一个案例的关键，其**伪代码**过程如下。

```
GET-THE-INVERSION(A)                    ▷A[1..N]表示一个序列
1 N←length[A]
2 count←0
3 for j←N downto 2
4   do for i←1 to j-1
5         do if A[i]>A[j]               ▷检测到一个逆序
6               then count←count+1      ▷累加到计数器
7 return count
```

算法 0-1　解决“计算逆序数”问题的一个案例的算法伪代码过程

伪代码是一种有着类似于程序设计语言的严格外部语法（用 **if-then-else** 表示分支结构，用 **for-do**、**while-do** 或 **repeat-until** 表示循环结构），且有着内部宽松的数学语言表述方式的代码表示方法。它既没有二歧性的缺陷（严格的外部语法），又能用高度抽象的数学语言简练地描述操作细节。

本书所使用的伪代码书写规则如下。

① 用分层缩进来指示块结构。例如，从第 3 行开始的 **for** 循环的循环体由第 4～6 行的 3 行组成，分层缩进风格也应用于 **if-then-else** 语句，如第 5～6 行的 **if-then** 语句。

② 对 **for** 循环，循环计数器在退出循环后仍然保留。因此，一个 **for** 循环刚结束时，循环计数器的值首次超过 **for** 循环上界。例如在算法 0-1 中，当第 3～6 行的 **for** 循环结束时，$j=N+1$；而第 4～6 行的 **for** 循环结束时，$i=1-1=0$。

③ 符号“▷”表示本行**的注释部分**。例如，算法 0-1 的开头对参数 A 的意义进行了解释，第 5 行说明检测到一个逆序（$i<j$，$A[i]>A[j]$），而第 6 行说明将此逆序累加到逆序数 *count*（*count* 自增 1）。

④ 多重赋值形式 $i \leftarrow j \leftarrow e$ 对变量 i 和 j 同赋予表达式 e 的值；它应当被理解为在赋值操作 $j \leftarrow e$ 之后紧接着赋值操作 $i \leftarrow j$。

⑤ 变量（如 i，j，及 *count*）都局部于给定的过程。除非特别需求，我们将避免使用全局变量。

⑥ 数组元素是通过数组名后跟括在方括号内的下标来访问。例如，$A[i]$表示数组 A 的第 i 个元素。记号“…”用来表示数组中取值的范围。因此，$A[1...i]$表示数组 A 的取值由 $A[1]$

到 $A[i]$，i 个元素构成的子序列。

⑦ 组合数据通常组织在**对象**中，其中组合了若干个**属性**。用域名[对象名]的形式来访问一个具体的域。例如，我们把一个数组 A 当成一个对象，它具有说明其所包含的元素个数的属性 *length*。为访问数组 A 的元素个数，我们写 $length[A]$。

表示数组或对象的变量被当成一个指向表示数组或对象的指针。对一个对象 x 的所有域 f，设 $y \leftarrow x$ 将导致 $f[y] = f[x]$。此外，若设 $f[x] \leftarrow 3$，则不仅有 $f[x] = 3$，且有 $f[y] = 3$。换句话说，赋值 $y \leftarrow x$ 后，x 和 y 指向同一个对象。

有时，一个指针不指向任何**对象**，此时，我们给它一个特殊的值 NIL。

⑧ 过程的参数是**按值**传递的：被调用的过程以复制的方式接收参数，若对参数赋值，则主调过程不能看到这一变化。

⑨ 布尔运算符“and”和“or”都是**短回路**的。也就是说，当我们计算表达式“x and y”时，先计算 x。若 x 为 FALSE，则整个表达式不可能为 TRUE，所以我们不再计算 y。另一方面，若 x 为 TRUE，我们必须计算 y 以确定整个表达式的值。相仿地，在表达式“x or y”中，我们计算表达式 y 当且仅当 x 为 FALSE。短回路操作符使得我们能够写出诸如“$x \neq$ NIL and $f[x] = y$”这样的布尔表达式而不必担心当 x 为 NIL 时去计算 $f[x]$。

0.4 算法的正确性

解决一个计算问题的算法是正确的，指的是对问题中任意合法的输入均应得到对应的正确输出。大多数情况下，问题的合法输入无法穷尽，当然就无法穷尽输出是否正确的验证。即使合法输入是有限的，穷尽所有输出正确的验证，在实践中也许是得不偿失的。但是，无论如何，我们需要保证设计出来的算法的正确性。否则，算法设计就是去了它的应用意义。因此，对设计出来的算法在提交应用之前，应当说明它的正确性。这就需要借助我们对问题的认识与理解，利用数学、科学及逻辑推理来证实算法是正确的。例如，对于解决“计算逆序数”问题的算法 0-1，其正确性可以表述为如下命题：

当第 3～7 行的 **for** 循环结束时，*count* 已记录下了序列 $A[1..N]$中的逆序数。

如果我们能说明上述命题是真的，那就说明了算法 0-1 是正确的。由于数组 $A[1..N]$的长度 N 是任意正整数，所以这是一个与正整数相关的命题。数学中要证明一个与正整数相关的命题有一个有力的工具——数学归纳法。下面我们对本命题中的 N 进行归纳。

当 N=1 时第 3～7 行的 **for** 循环重复 0 次。*count* 保持初始值 0，这与 $A[1..N]$=$A[1]$没有任何逆序相符，结论显然为真。

设 N>1 且可用算法计算出 $A[1..N-1]$的逆序数 *count*。在此假设下，我们来证明对 $A[1..N]$

利用算法 0-1 也能得到正确的逆序数 *count*。

考虑算法中第 3～7 行的 **for** 循环在 $j=N$ 时的第一次重复的操作：第 4～6 行内嵌的 **for** 循环从 $i=1$ 开始到 $j-1$ 为止，逐一检测是否 $A[i]>A[j]$。若是，意味着找到一个关于 $A[N]$的逆序，第 6 行 *count* 自增 1。当此循环结束时 *count* 中累积了关于 $A[N]$的逆序数。由于 $N>1$，故第 3～6 行的外围 **for** 循环必定会继续对 $A[1..N-1]$做同样的操作。根据归纳假设，我们知道第 3～6 行的 **for** 循环接下来的重复操作能将 $A[1..N-1]$中个元素的逆序数累加到 *count* 中。所以第 3～6 行 **for** 循环结束时，*count* 已记录下了序列 $A[1..N]$中的逆序数。

这样，我们就从逻辑上证明了算法 0-1 能正确地解决“计算逆序数”问题的一个案例，即算法 0-1 是正确的。

应当指出，解决一个计算问题时，算法不必唯一。数据的组织方式、解题思路的不同，会导致不同的算法。

例如，将计数器 *count* 设置为全局变量，并初始化为 0。解决“计算逆序数”问题一个案例的算法还可以表示为如下的形式。

```
GET-THE-INVERSION(A, N)                ▷A[1..N]表示一个序列
1 if N<2
2     then return
3 for i←1 to N-1
4   do if A[i]>A[N]                    ▷检测到一个逆序
5     then count←count+1               ▷累加到计数器
6 GET-THE-INVERSION(A, N-1)
```

算法 0-2　解决“计算逆序数”问题一个案例的递归算法伪代码过程

这是一个“递归”算法，它在定义的内部（第 6 行）进行了一次自我调用。受上述算法 0-1 正确性命题证明的启发，这个算法的思想是基于先计算出 $A[1..N-1]$中关于 $A[N]$的逆序数 *count*，然后将问题归结为计算 $A[1..N-1]$的逆序数的子问题。用相同的方法解决子问题（递归调用自身，注意表示 A 的长度的第 2 个参数变成 $N-1$）把子问题的解与 *count* 合并就可得到原问题的解。其实，算法 0-2 与算法 0-1 仅仅是表达形式不同，本质上等价的：后者用末尾递归（第 6 行递归调用自身）隐式地替代算法 0-1 中第 3～6 行的外层 **for** 循环。所以，算法 0-2 也是正确的。

0.5 算法分析

解决同一问题的不同算法所消耗的计算机系统的时间（占用处理器的时间）和空间（占用内部存储器空间）资源量可能有所不同。算法运行所需要的资源量称为算法的复杂性。一般来说，解决同一问题的算法，需要的资源量越少，我们认为越优秀。计算算法运行所需资