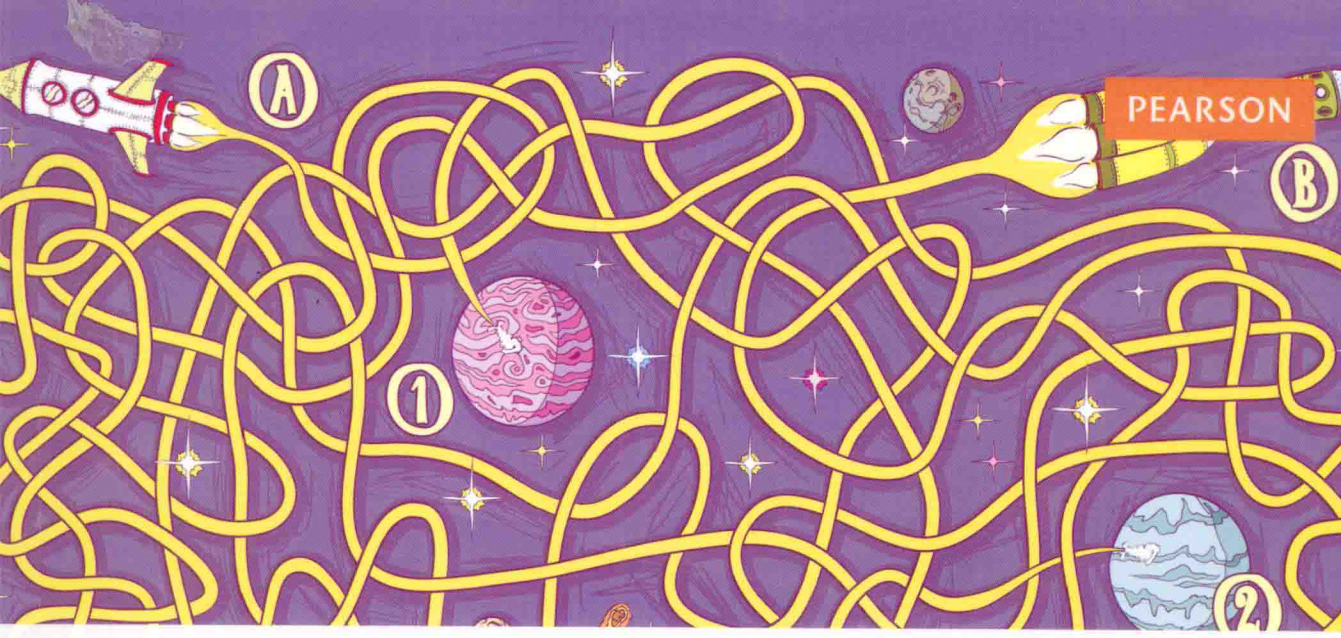
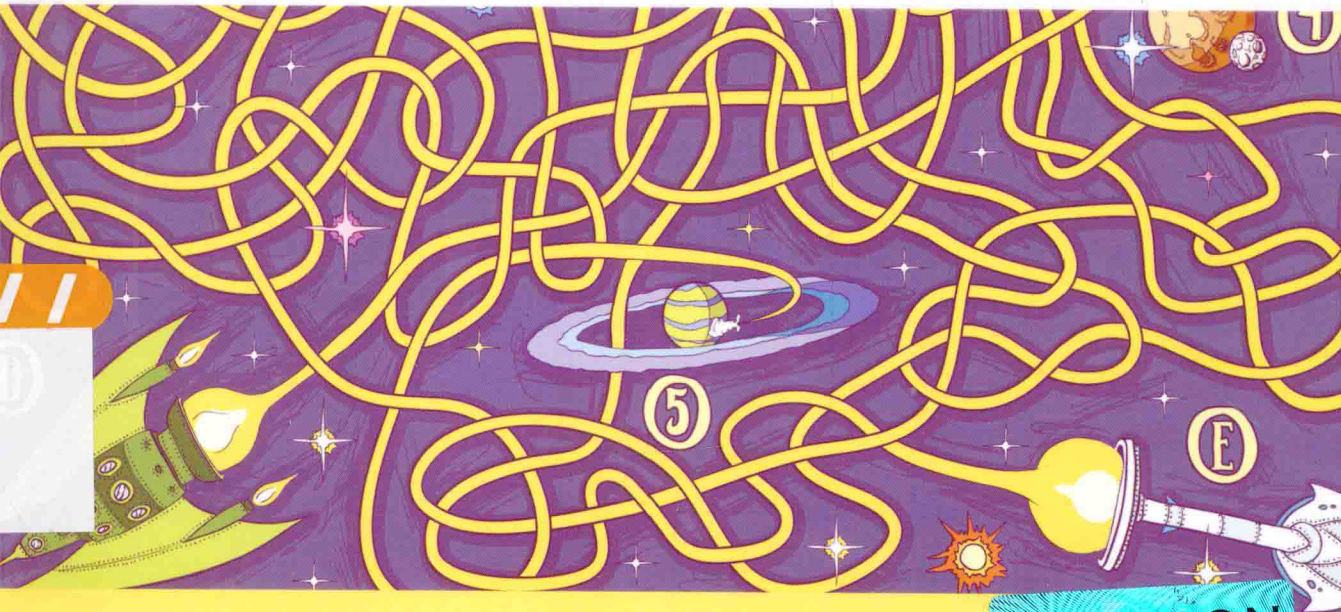


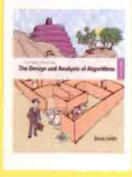
Introduction to The Design and Analysis of Algorithms Third Edition

算法设计与分析基础

（美）Anany Levitin 著
潘 彦 译

（第3版）





- 畅销十余年的国外经典教材，影响全球数十万名读者
- 条理清晰，逻辑严密，透过经典算法来揭示算法精髓
- 全面覆盖十大通用算法设计技术
- 600多道习题有难有易又有趣，涉及ACM竞赛题、算法谜题和面试题

清华大学出版社

Introduction to **The Design and Analysis of Algorithms** Third Edition

算法设计与分析基础

（第3版）

（美）Anany Levitin 著
潘 彦 译

清华大学出版社
北京

内 容 简 介

作者基于丰富的教学经验，开发了一套全新的算法分类方法。该分类法站在通用问题求解策略的高度，对现有大多数算法准确分类，从而引领读者沿着一条清晰、一致、连贯的思路来探索算法设计与分析这一迷人领域。本书作为第3版，相对前版调整了多个章节的内容和顺序，同时增加了一些算法，并扩展了算法的应用，使得具体算法和通用算法设计技术的对应更加清晰有序；各章累计增加了70道习题，其中包括一些有趣的谜题和面试问题。

本书十分适合用作算法设计和分析的基础教材，也适合任何有兴趣探究算法奥秘的读者使用，只要读者具备数据结构和离散数学的知识即可。

北京市版权局著作权合同登记号　图字：01-2014-2598

图书在版编目(CIP)数据

算法设计与分析基础/(美)莱维汀(Levitin, A.)著；潘彦译. —3版. --北京：清华大学出版社，2015
书名原文：Introduction to the Design and Analysis of Algorithms
ISBN 978-7-302-38634-6

Ⅰ. ①算… Ⅱ. ①莱… ②潘… Ⅲ. ①算法设计 ②算法分析 Ⅳ. TP301.6

中国版本图书馆 CIP 数据核字(2014)第279849号

责任编辑：文开琪　汤涌涛
封面设计：杨玉兰
责任校对：周剑云
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：三河市君旺印务有限公司
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：27.5　　**字　　数**：657千字
版　　次：2004年6月第1版　2015年2月第3版　　**印　　次**：2015年2月第1次印刷
印　　数：1～3500
定　　价：69.00元

产品编号：044086-01

译 者 序

十年前，本书第 1 版面世。

十年后，迎来了第 3 版。

十年不长。作者 Anany Levitin 仍然在维拉诺瓦大学从事算法基础教学，兢兢业业不断更新和完善着这本算法经典教材。清华大学出版社的诸位仍然辛勤耕耘在教材出版的第一线，在行业并不十分景气的情况下，恪守职业尊严，努力为大家奉献一部部优秀的教材和读物。正是由于这些作者、编者多年不变的持续付出，计算机教育事业才有了不断发展下去的动力。

十年也不短。十年前的读者想必已经从莘莘学子成为了企业骨干，很多已经成家立业，事业有成了吧？大家有没有在从事和算法有关的工作？算法学习给大家带来了什么有益的改变？多想听听大家的心声。作为译者本人来说，翻译第 1 版时刚刚三十岁，而现在已过不惑之年。当年接手本书的初衷仅仅是希望提供一本易懂的翻译教材，尽量减少读者阅读的障碍。但实际上，从这本书受益最大的可能还是译者本人。首先，翻译本书的过程提高了自身的综合能力。其次，有机会逐字逐句精读这样一本严谨的教材是一种很好的学术训练，为本人后来的博士生涯增益不少。最后，本人目前从事算法交易，尽管很少用到现成算法，但本书提供的算法专业训练还是使我获益良多。

茫茫历史长河中，一本书的好坏可能并不重要，但如果每个人都能专注做好自己的事情，对人对己就会产生非常有益的影响。捧起本书的读者们，我衷心希望大家认真做事，做正确的事。因为，下一个十年你不会后悔这样的付出。

我要感谢本书原著者，让我有机会和一本好书一起成长。我要感谢第 1、第 2 版的读者，他们通过互联网对本书做出了非常积极的评价，还有读者不吝指出书中的错误，和大家交流非常开心。我要感谢出版社的领导，继续给予我信任，并容忍我并不算快的进度。我还要感谢本书的编辑，她十年如一日，以一贯的严谨为本书提供了质量保证，尽管从未谋面，我想我们已经是老朋友了。我最后要感谢爱人李靓的支持，她理解翻译工作的意义，为我提供了很多实际的帮助。

从作者本版的修订风格来看，第 3 版不会是最后一版，希望我有幸再次为广大读者执起译笔。

祝大家学习顺利！

潘彦

phil_pan@hotmail.com

前　言

一个人在接受科技教育时能得到的最珍贵的收获是能够终身受用的通用智能工具。[①]

——乔治·福赛思

无论是计算科学还是计算实践，算法都在其中扮演着重要角色。因此，这门学科中出现了大量的教材。它们在介绍算法的时候，基本上都选择了以下两种方案中的一种。第一种方案是按照问题的类型对算法进行分类。这类教材安排了不同的章节分别讨论排序、查找、图等算法。这种做法的优点是，对于解决同一问题的不同算法，它能够立即比较这些算法的效率。其缺点在于，由于过于强调问题的类型，它忽略了对算法设计技术的讨论。

第二种方案围绕着算法设计技术来组织章节。在这种结构中，即使算法来自于不同的计算领域，如果它们采用了相同的设计技术，就会被编成一组。从各方(例如[BaY95])获得的信心使我相信，这种结构更适合于算法设计与分析的基础课程。强调算法设计技术有三个主要原因。第一，学生们在解决新问题时，可以运用这些技术设计出新的算法。从实用的角度看，这使得学习算法设计技术颇有价值。第二，学生们会试图按照算法的内在设计方法对已知的众多算法进行分类。计算机科学教育的一个主要目的，就是让学生们知道如何发掘不同应用领域的算法间的共性。毕竟，每门学科都会倾向于把它的重要主题归纳为几个甚至一个规则。第三，依我看来，算法设计技术作为问题求解的一般性策略，在解决计算机领域以外的问题时，也能发挥相当大的作用。

遗憾的是，无论是从理论还是从教学的角度，传统的算法设计技术分类法都存在一些严重的缺陷。其中最显著的缺陷就是无法对许多重要的算法进行分类。由于这种局限性，这些书的作者不得不在按照设计技术进行分类的同时，另外增加一些章节来讨论特殊的问题类型。但这种改变导致课程缺乏一致性，而且很可能会使学生感到迷惑。

算法设计技术的新分类法

传统算法设计技术分类法的缺陷令我感到失望，它激发我开发一套新的分类法([Lev99])，这套分类法就是本书的基础。以下是这套新分类法的几个主要优势。

- 新分类法比传统分类法更容易理解。它包含的某些设计策略，例如蛮力法、减治法、变治法、时空权衡和迭代改进，几乎从不曾被看作重要的设计范例。
- 新分类法很自然地覆盖了许多传统方法无法分类的经典算法(欧几里得算法、堆排序、查找树、散列法、拓扑排序、高斯消去法、霍纳法则等，不胜枚举)。所以，新分类法能够以一种连贯的、一致的方式表达这些经典算法的标准内容。
- 新分类法很自然地容纳了某些设计技术的重要变种(例如，它能涵盖减治法的 3 个

① 译注：出自 *What to do till the computer scientist comes*(1968)。乔治·福赛思(George Forsythe，1917—1972)是一名数学家，他认为最重要的三大工具依次是自然语言、数学和计算机科学。

变种和变治法的3个变种)。

- 在分析算法效率时，新分类法与分析方法结合得更好(参见附录B)。

设计技术作为问题求解的一般性策略

在本书中，主要将设计技术应用于计算机科学中的经典问题(这里唯一的创新是引入了一些数值算法的内容，我们也是用同样的通用框架来表述这些算法的)。但把这些设计技术看作问题求解的一般性工具时，它们的应用就不仅限于传统的计算问题和数学问题了。有两个因素令这一点变得尤其重要。第一，越来越多的计算类应用超越了它们的传统领域，并且有足够的理由使人相信，这种趋势会愈演愈烈。第二，人们渐渐认识到，提高学生们的问题求解能力是高等教育的一个主要目标。为了满足这个目标，在计算机科学课程体系中安排一门算法设计和分析课程是非常合适的，因为它会告诉学生如何应用一些特定的策略来解决问题。

虽然我并不建议将算法设计和分析课程变成一门教授一般性问题求解方法的课程，但我深信，我们不应错过算法设计和分析课程提供的这样一个独一无二的机会。为了这个目标，本书包含了一些和谜题相关的应用。虽然利用谜题来教授算法课程绝不是我的创新，但本书打算通过引进一些全新的谜题来系统地实现这个思路。

如何使用本书

我的目标是写一本既不泛泛而谈，又可供学生们独立阅读的教材。为了实现这个目标，本书做了如下努力。

- 根据乔治·福赛思的观点(参见前面的引文)，我试图着重强调隐藏在算法设计和分析背后的主要思想。在选择特定的算法来阐述这些思想的时候，我并不倾向于涉及大量的算法，而是选择那些最能揭示其内在设计技术或分析方法的算法。幸运的是，大多数经典算法满足这个要求。
- 第2章主要分析算法的效率，该章将分析非递归算法的方法和分析递归算法的典型方法区别开来。这一章还花了一些篇幅介绍算法经验分析和算法可视化。
- 书中系统地穿插着一些面向读者的提问。其中有些问题是经过精心设计的，而且答案紧随其后，目的是引起读者的注意或引发疑问。其余问题的用意是防止读者走马观花，不能充分理解本书的内容。
- 每一章结束时都会对本章最重要的概念和结论做一个总结。
- 本书包含600多道习题。有些习题是为了给大家练习，另外一些则是为了指出书中正文部分所涉及内容的重要意义，或是为了介绍一些书中没有涉及的算法。有一些习题利用了因特网上的资源。较难的习题数量不多，会在教师用书中用一种特殊的记号标注出来(因为有些学生可能没有勇气做那些有难度标注的习题，所以本书没有对习题标注难度)。谜题类的习题用一种特殊的图标做标注。
- 本书所有的习题都附有提示。除了编程练习，习题的详细解法都能够在教师资源中找到。请发送邮件到coo@netease.com，申请教师相关资源(也可联系培生公司的当地销

售代表，或者访问 *www.pearsonhighered.com/irc*)。本书的任何读者都可以在 CS 支持网站 *http://cssupport.pearsoncmg.com* 上找到 PowerPoint 格式的幻灯片文件。如果对算法有兴趣，欢迎加入 QQ 群“算法学习交流”，群号：425283001。

第 3 版的变化

第 3 版有若干变化。其中最重要的变化是介绍减治法和分治法的先后顺序。第 3 版会先介绍减治法，后介绍分治法，这样做有以下几个优点。

- 较之分治法，减治法更简单。
- 在求解问题方面，减治法应用更广。
- 这样的编排顺序便于先介绍插入排序，后介绍合并排序和快速排序。
- 数组划分的概念通过选择性问题引入，这次利用 Lomuto 算法的单向扫描来实现，而将 Hoare 划分方法的双向扫描留至后文与快速排序一并介绍。
- 折半查找归入介绍减常量算法的章节。

另一个重要变化是重新编排第 8 章关于动态规划的内容，具体如下所述。

- 导述部分的内容是全新的。在前两版中用计算二项式系数的例子来引入动态规划这一重要技术，但在第 3 版中会介绍 3 个基础性示例，这样介绍的效果更好。
- 8.1 节的习题是全新的，包括一些在前两版中没有涉及的流行的应用。
- 第 8 章其他小节的顺序也做了调整，以便达到由浅入深、循序渐进的效果。

此外，还有其他一些变化。增加了不少与本书所述算法相关的应用。遍历图算法不再随减治法介绍，而是纳入蛮力算法和穷举查找的范畴，我认为这样更合理。在介绍生成组合对象的算法时，新增了格雷码算法。对求解最近对问题的分治法有更深入的探讨。改进的内容包括算法可视化和求解旅行商问题的近似算法，当然参考文献也有相应的更新。

第 3 版一共新增约 70 道习题，其中涉及算法谜题和面试问题。

先修课程

本书假定读者已经学习了离散数学的标准课程和一门基础性的编程课程。有了这样的知识背景，读者应该能够掌握本书的内容而不会遇到太大的困难。尽管如此，1.4 节、附录 A 和附录 B 仍然对基本的数据结构以及必须用到的求和公式与递推关系分别进行复习和回顾。只有 3 个小节(2.2 节、11.4 节和 12.4 节)会用到一些简单的微积分知识，如果读者缺少必要的微积分知识，完全可以跳过这 3 个涉及微积分的小节，这并不妨碍对本书其余部分的理解。

课程进度安排

如果打算开设一门围绕算法设计技术来讲解算法设计和分析理论的基础课程，可以采用本书作为教材。但要想在一个学期内完成该课程，本书涵盖的内容可能过于丰富了。大体上来说，跳过第 3～12 章的部分内容不会影响读者对后面部分的理解。本书的任何一个部分都可以安排学生自学。尤其是 2.6 节和 2.7 节，它们分别介绍了经验分析和算法可视化，

这两小节的内容可以结合课后练习[①]布置给学生。

下面给出了针对一个学期课程的教学计划，这是按照40课时的集中教学来设计的。

课 次	主 题	小 节
1	课程简介	1.1～1.3
2，3	分析框架；常用符号 O、Θ 和 Ω	2.1，2.2
4	非递归算法的数学分析	2.3
5，6	递归算法的数学分析	2.4，2.5(+附录B)
7	蛮力算法	3.1，3.2(+3.3)
8	穷举查找	3.4
9	深度优先查找和广度优先查找	3.5
10～11	减一算法：插入排序、拓扑排序	4.1，4.2
12	折半查找和其他减常量算法	4.4
13	减变量算法	4.5
14～15	分治法：合并排序、快速排序	5.1～5.2
16	其他分治法示例	5.3、5.4或5.5
16	减变量算法	5.6
17～19	实例化简：预排序、高斯消去法、平衡查找树	6.1～6.3
20	改变表现：堆和堆排序或者霍纳法则和二进制幂	6.4或6.5
21	问题化简	6.6
22～24	时空权衡：串匹配、散列法、B树	7.2～7.4
25～27	动态规划算法	8.1～8.4(选3节)
28～30	贪婪算法：Prim算法、Kruskal算法、Dijkstra算法、哈夫曼算法	9.1～9.4
31～33	迭代改进算法	10.1～10.4(选3节)
34	下界的参数	11.1
35	决策树	11.2
36	*P*、*NP* 和 *NP* 完全问题	11.3
37	数值算法	11.4(+12.4)
38	回溯法	12.1
39	分支界限法	12.2
40	*NP* 困难问题的近似算法	12.3

① 译注：“练习”的原文为project，一般应该翻译成“项目”，但国外一般将布置在课后完成的、较大型的、要求实际演练的习题称为project，国内没有相应的称呼，所以姑且译为“练习”。

致谢

我要向本书的评审表达衷心的感谢，还要感谢本书前两版的许多读者，他们提供了许多宝贵的意见和建议，帮助本书得以改进和完善。本书第 3 版尤其得益于下列人士的评审，包括 Andrew Harrington(芝加哥洛约拉大学)、David Levine(圣文德大学)、Stefano Lombardi(加州大学河滨分校)、Daniel McKee(宾州曼斯菲尔德大学)、Susan Brilliant(弗吉尼亚州立联邦大学)、David Akers(菩及海湾大学)以及两名匿名评审。

我要感谢培生出版社所有为本书付出不懈努力的工作人员和相关人士。尤其要感谢本书编辑 Matt Goldstein、编务助理 Chelsea Bell、市场经理 Yez Alayan 和产品总监 Kayla Smith-Tarbox。我还要感谢 Richard Camp 为本书审稿，Windfall Software 的 Paul Anagnostopoulos 和 Jacqui Scarlott 为本书排版并提供项目管理支持，以及 MaryEllen Oliver 为本书进行校对。

最后，我要感谢两位家人。另一半整天都在写书比自己本人写书更让人崩溃，我的妻子 Maria 已容忍我多年并任劳任怨地帮助我，本书 400 多幅插图以及教师手册都是凭她一己之力完成的。女儿 Miriam 是我多年的英语老师，她不但阅读了本书大量篇幅，还帮我为每章找到了合适的名人名言。

Anany Levitin
anany.levitin@villanova.edu

目　录

第1章 绪 论

有两种思想，就像摆放在天鹅绒上的宝石那样熠熠生辉，一个是微积分，另一个就是算法。微积分以及在微积分基础上建立起来的数学分析体系造就了现代科学，而算法则造就了现代世界。[①]

——大卫·柏林斯基

为什么要学习算法？如果你想成为一名计算机专业人士，无论从理论还是从实践的角度，学习算法都是必需的。从实践的角度来看，我们必须了解计算领域中不同问题的一系列标准算法。此外，我们还要具备设计新算法和分析其效率的能力。从理论的角度来看，对算法的研究(有时称为“**算法学**”，英文为 algorithmics)已经被公认为是计算机科学的基石。大卫·哈雷尔(David Harel)写了一本非常好的书，他直截了当地将其命名为《算法学——计算精髓》(*Algorithmics: the Spirit of Computing*)。书中是这样阐述这个问题的：

算法不只是计算机科学的一个分支。它是计算机科学的核心。而且，可以毫不夸张地说，它与绝大多数科学、商业和技术都密切相关。([Har92]，p. 6)

即使不是计算机相关专业的学生，学习算法的理由也是非常充分的。坦率地说，没有算法，就没有计算机程序。而且，随着计算机日益渗透到我们日常工作和生活的方方面面，需要学习算法的人也越来越多。

学习算法的另一个理由是可以用它来培养人们的分析能力。毕竟，算法可以看作解决问题的一类特殊方法——它虽然不是问题的答案，但它是经过准确定义以获得答案的过程。因此，无论是否涉及计算机，特定的算法设计技术都能看作问题求解的有效策略。当然，算法思想天生固有的精确性限制了它能够解决的问题种类。例如，我们找不到一种使人幸福快乐的算法，也找不到一种使人功成名就的算法。但另一方面，从教育角度来看，这种必要的精确性却是很重要的。唐纳德·克努特(Donald Knuth)，算法学历史上最卓越的计算机科学家之一，是这样论述这个问题的：

受过良好训练的计算机科学家知道怎样处理算法：如何构造算法、操作算法、理解算法以及分析算法。这些知识远不止为了写出良好的计算机程序而准备的。算法是一种一般性的智能工具，它必定有助于我们对其他学科的理解，不管是化学、语言学或音乐，还是其他学科。为什么算法会有这种作用呢？我们可以这样理解：人们常说，一个人只有把知识教给别人，才能真正掌握它。实际上，一个人只有把知识教给“计算机”，才能“真正”掌握它，也就是说，将知识表述为一种算法……比起简单按照常规去理解事物，尝试用算法将其形式化能使我们的理解更加深刻。([Knu96]，p. 9)

---

① 译注：引自《算法的诞生》(*The Advent of the Algorithm*，2000)，作者是大卫·柏林斯基(David Berlinski)，他的另一部代表作是《微积分之旅》。

我们从 1.1 节开始涉及算法的概念。作为例子，我们将针对同一问题(求最大公约数)使用三种不同的算法。这样做有几个理由。首先，这是大家在中学时代就非常熟悉的问题。其次，它揭示了一种重要的观点：同样的问题往往能用多种算法来解决。这三种算法之所以具有典型意义，是因为它们的解题思路不同，复杂程度不同，解题效率也各不相同。再次，我希望将其中一种算法作为本书最先介绍的算法，这不仅因为它历史悠久(早在两千多年前它就出现在了欧几里得的著作中)，还因为它不朽的力量和重要性。最后，对这三种算法的研究，能使我们从总体上观察一种算法通常具有的若干重要特性。

1.2 节讲述算法解题的分析和计算问题。我们将讨论有关算法设计和分析的一些重要内容。内容涉及算法解题的不同方面，包括问题的分析、如何正确表述算法以及算法的效率分析。这一节不能传授秘方为任意问题设计算法。这是一个公认的事实，即这种秘方是不存在的。但当大家日后进行自己的算法设计和分析工作时，肯定会用到 1.2 节的内容。

1.3 节专门讨论几种重要的问题类型。经验证明，无论对于算法教学还是对于算法应用，这些问题类型都是极其重要的。实际上，有些教材(例如[Sed88])就是围绕这些问题类型来组织内容的。尽管包括我在内的许多人都认为围绕算法设计技术来组织内容更有优势，但无论如何，了解这些重要的问题类型都是十分必要的。一是因为这些类型是实际应用中最常见的，二是由于本书从头到尾都会用它们来演示一些特殊的算法设计技术。

1.4 节回顾基本的数据结构。安排这一节的目的与其说是为了讨论这方面的内容，还不如说是希望把它作为读者的参考材料。如果需要了解更详细的内容，可以参考很多关于该主题的优秀书籍，其中大多数都是和某一种编程语言相关的。

1.1 什么是算法

虽然对于这个概念没有一个大家公认的定义，但我们对它的含义还是有基本共识的：

算法(algorithm)是一系列解决问题的明确指令，也就是说，对于符合一定规范的输入，能够在有限时间内获得要求的输出。

这个定义可以用一幅简单的图(图 1.1)来说明。

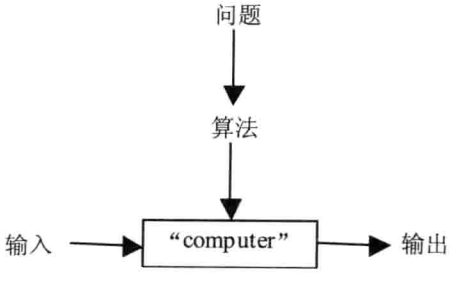


图 1.1 算法的概念

定义中使用了“指令”这个词，这意味着有人或物能够理解和执行所给出的命令，我

们将这种人或物称为 computer。请记住，在电子计算机发明以前，computer 是指那些从事数学计算的人。现在，computer 当然是特指那些做每件事情都越发不可或缺的、无所不在的电子设备。但要注意的是，虽然绝大多数算法最终要靠计算机来执行，但算法概念本身并不依赖于这样的假设。

为了阐明算法的概念，本节将以三种方法为例来解决同一个问题，即计算两个整数的最大公约数。这些例子会帮助我们阐明以下要点。

- 算法的每一个步骤都必须没有歧义，不能有半点儿含糊。
- 必须认真确定算法所处理的输入的值域。
- 同一算法可以用几种不同的形式来描述。
- 同一问题，可能存在几种不同的算法。
- 针对同一问题的算法可能基于完全不同的解题思路，而且解题速度也会有显著不同。

还记得最大公约数的定义吗？两个不全为 0 的非负整数 m 和 n 的最大公约数记为 gcd(m, n)，代表能够整除(即余数为 0)m 和 n 的最大正整数。古希腊数学家、亚历山大港的欧几里得(公元前 3 世纪)所著的《几何原本》，以系统论述几何学而著称，在其中的一卷里，他简要描述了一个最大公约数算法。用现代数学的术语来表述，**欧几里得算法**(Euclid's algorithm)采用的方法是重复应用下列等式，直到 m mod n 等于 0。

$$\gcd(m, n) = \gcd(n, m \bmod n) \qquad (m \bmod n \text{ 表示 } m \text{ 除以 } n \text{ 之后的余数})$$

因为 gcd(m, 0) = m(为什么？)，m 最后的取值也就是 m 和 n 的初值的最大公约数。

举例来说，gcd(60, 24)可以这样计算：

$$\gcd(60, 24) = \gcd(24, 12) = \gcd(12, 0) = 12$$

如果你对这个算法还没有足够的认识，可以做本节习题第 6 题，试着求一些较大数的最大公约数。

下面是该算法的一个更加结构化的描述：

用于计算 gcd(m, n)的欧几里得算法

第一步：如果 $n = 0$，返回 m 的值作为结果，同时过程结束；否则，进入第二步。

第二步：m 除以 n，将余数赋给 r。

第三步：将 n 的值赋给 m，将 r 的值赋给 n，返回第一步。

我们也可以使用伪代码来描述这个算法：

```
算法　Euclid(m, n)
      //使用欧几里得算法计算 gcd(m, n)
      //输入：两个不全为 0 的非负整数 m，n
      //输出：m，n 的最大公约数
      while n ≠ 0 do
        r ← m mod n
        m ← n
        n ← r
      return m
```

我们怎么知道欧几里得算法最终一定会结束呢？通过观察，我们发现，每经过一次循环，参加运算的两个算子中的后一个都会变得更小，而且绝对不会变成负数。确实，下一次循环时，n 的新值是 $m \bmod n$，这个值总是比 n 小。所以，第二个算子的值最终会变成 0，此时，这个算法也就结束了。

就像其他许多问题一样，最大公约数问题也有多种算法。让我们看看解这个问题的另外两种方法。第一个方法只基于最大公约数的定义：m 和 n 的最大公约数就是能够同时整除它们的最大正整数。显然，这样一个公约数不会大于两数中的较小者，因此，我们先有：$t = \min\{m, n\}$。我们现在可以开始检查 t 是否能够整除 m 和 n：如果能，t 就是最大公约数；如果不能，我们就将 t 减 1，然后继续尝试(我们如何确定该算法最终一定会结束呢？)。例如，对于 60 和 24 这两个数来说，该算法会先尝试 24，然后是 23，这样一直尝试到 12，算法就结束了。

用于计算 gcd(m, n)的连续整数检测算法

第一步：将 $\min\{m, n\}$的值赋给 t。

第二步：m 除以 t。如果余数为 0，进入第三步；否则，进入第四步。

第三步：n 除以 t。如果余数为 0，返回 t 的值作为结果；否则，进入第四步。

第四步：把 t 的值减 1。返回第二步。

注意，和欧几里得算法不同，按照这个算法的当前形式，当它的一个输入为 0 时，计算出来的结果是错误的。这个例子说明了为什么必须认真、清晰地规定算法输入的值域。

求最大公约数的第三种过程，我们应该在中学时代就很熟悉了。

中学时计算 gcd(m, n)的过程

第一步：找到 m 的所有质因数。

第二步：找到 n 的所有质因数。

第三步：从第一步和第二步求得的质因数分解式中找出所有的公因数(如果 p 是一个公因数，而且在 m 和 n 的质因数分解式分别出现过 p_m 和 p_n 次，那么应该将 p 重复 $\min\{p_m, p_n\}$次)。

第四步：将第三步中找到的质因数相乘，其结果作为给定数字的最大公约数。

这样，对于 60 和 24 这两个数，我们得到：

$$60 = 2\times2\times3\times5$$
$$24 = 2\times2\times2\times3$$
$$\gcd(60, 24) = 2\times2\times3 = 12$$

虽然学习这个方法的那段中学时光是令人怀念的，但我们仍然注意到，第三个过程比欧几里得算法要复杂得多，也慢得多(下一章中，我们将会讨论对算法运行时间进行求解和比较的方法)。撇开低劣的性能不谈，以这种形式表述的中学求解过程还不能称为一个真正意义上的算法。为什么？因为其中求质因数的步骤并没有明确定义：该步骤要求得到一个质因数的列表，但我们十分怀疑中学里是否曾教过如何求这样一个列表。必须承认，这并不是鸡蛋里挑骨头。除非解决了这个问题，否则我们不能下结论说，能够写一个实现这个