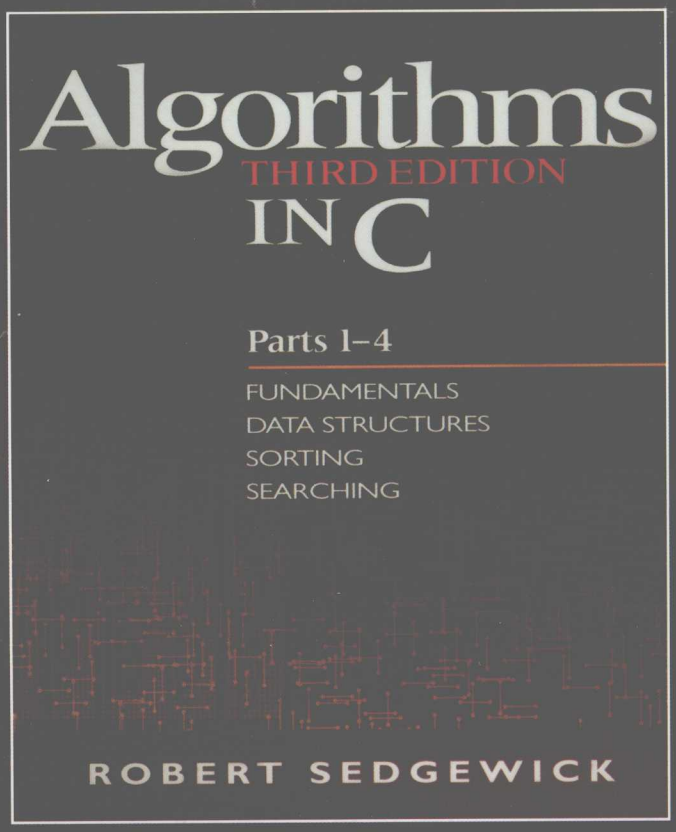
HZ BOOKS 华章教育
PEARSON Addison Wesley
计 算 机 科 学 丛 书
原书第3版

算法：C语言实现

（第1～4部分）

基础知识、数据结构、排序及搜索

（美） Robert Sedgewick 普林斯顿大学 著 霍红卫 西安电子科技大学 译



Algorithms in C

Parts 1-4: Fundamentals, Data Structures, Sorting, Searching

Third Edition

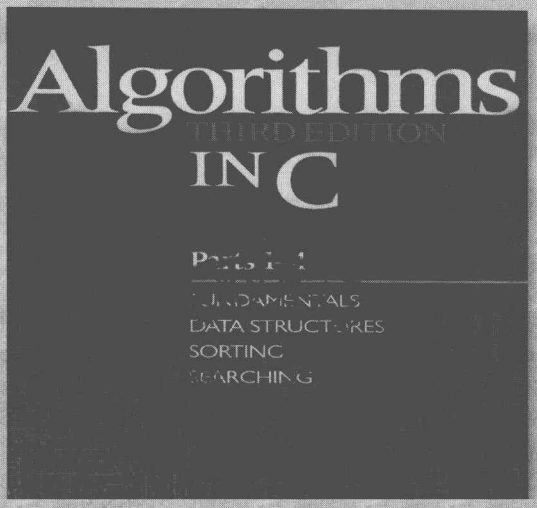
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第3版

算法：C语言实现
（第1～4部分）
基础知识、数据结构、排序及搜索

（美） Robert Sedgewick 普林斯顿大学 著 霍红卫 西安电子科技大学 译



Algorithms in C
Parts 1-4: Fundamentals, Data Structures, Sorting, Searching
Third Edition

机械工业出版社
China Machine Press

本书细腻讲解计算机算法的C语言实现。全书分为四部分，共16章。包括基本算法分析原理，基本数据结构、抽象数据结构、递归和树等数据结构知识，选择排序、插入排序、冒泡排序、希尔排序、快速排序方法、归并和归并排序方法、优先队列与堆排序方法、基数排序方法以及特殊用途的排序方法，并比较了各种排序方法的性能特征，在进一步讲解符号表、树等抽象数据类型的基础上，重点讨论散列方法、基数搜索以及外部搜索方法。书中提供了用C语言描述的完整算法源程序，并且配有丰富的插图和练习，还包含大量简洁的实现将理论和实践成功地相结合，这些实现均可用在真实应用上。

本书内容丰富，具有很强的实用价值，适合作为高等院校计算机及相关专业本科生算法课程的教材，也是广大研究人员的极佳参考读物。

本书版权登记号：图字：01-2006-3991

图书在版编目（CIP）数据

算法：C语言实现（第1～4部分），基础知识、数据结构、排序及搜索（原书第3版）/（美）塞奇威克（Sedgewick, R.）著；霍红卫译. —北京：机械工业出版社，2009.10

（计算机科学丛书）

书名原文：Algorithms in C, Parts 1~4: Fundamentals, Data Structures, Sorting, Searching, Third Edition

ISBN 978-7-111-27571-8

Ⅰ. 算…　Ⅱ. ① 塞…　② 霍…　Ⅲ. ① 电子计算机－算法理论　② C语言－程序设计　Ⅳ. ① TP301.6　② TP312

中国版本图书馆CIP数据核字（2009）第114468号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：李俊竹

北京京北印刷有限公司印刷

2009年10月第1版第1次印刷

184mm × 260mm · 29.5印张

标准书号：ISBN 978-7-111-27571-8

定价：79.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

译者序

本书是算法方面的优秀著作之一。它系统地阐述了算法的特征以及它们可能应用的场合，讨论了算法分析与理论计算机科学的关系，并通过实验数据和分析结果表明选择何种算法来解决实际问题。书中包含了基本概念、数据结构、排序算法和搜索算法。

这本书不仅适合于程序员和计算机科学专业的学生，而且也适合于那些想利用计算机并想使它运行更快或是想要解决更大问题的人们。书中的算法代表了过去50年来所研究的知识主体。对于大量应用问题，这些知识主体已经成为有效使用计算机不可缺少的部分。从物理学中的N-体模拟问题到分子生物学中的序列分析问题，在此所描述的基本方法在科学研究中已日显重要。另外，从数据库系统到Internet搜索引擎，这些方法已经成为现代软件系统的重要组成部分。随着计算机应用的覆盖面越来越广，基本算法的影响也日益显著。

本书主要内容及特点如下：

- 扩展介绍了数组、链表、串、树和其他基本数据结构。
- 为排序、选择、优先队列ADT实现和符号表ADT（查找）实现提供了多达100多个算法。
- 介绍了多路基数排序、随机BST、伸展树、跳跃表、多路trie等新的数据结构。
- 为算法提供了很多可视化的信息，还有大量实验研究和基本分析研究，从而为选择算法解决实际问题提供了依据。
- 增加了1000多个新练习，从而有助于深入了解算法的特征。
- 本书以大量图例说明算法的工作过程，使得算法更加易于理解和掌握。
- 适合作为高等院校算法设计课程的教材，同时可作为从事软件开发和工程设计的专业人员的参考书。

由于时间较紧及译者水平有限，译文难免有错误及不妥之处，恳请读者批评指正。

译者

于西安电子科技大学计算机学院

2009年4月

前　言

写本书的目的是为了对当今使用最为重要的计算机算法做一综述，并为需要学习这方面知识的越来越多的读者提供基础的技术。本书可以在学生掌握了所需的基本程序设计技巧，熟悉了计算机系统，但还未学过计算机科学或计算机应用高级领域的专业课程的时候，用作计算机科学的第二、第三或第四门课程的教科书。此外，由于本书包含了大量有用算法的实现，以及关于这些算法的性能特征的详细信息，因而它还可用于自学，或者作为从事计算机系统或应用程序开发人员的参考手册。宽广的视角使得本书成为计算机算法领域最合适的入门读物。

对于新的一版，我不仅完全重写了它的内容，而且还添加了一千多个练习、一百多幅图表和数十个新程序。我还给所有图表和程序添加了详细的注释。新的素材不仅涵盖了新的主题，而且还包含对经典算法的更完整解释。抽象数据类型是这本书的重点，这使得程序应用更广泛，并且与现代面向对象的程序设计环境更紧密。读过本书旧版本的人一定会发现，新版本包含了更为丰富的新信息，所有读者将发现大量的教学资料为掌握基本概念提供了有效途径。

由于新的素材数量过多，所以我们把新版本分为两卷（每一卷的容量都大约为旧版本的大小），本书是第一卷。这卷书中包含了基本概念、数据结构、排序算法和搜索算法；第二卷涵盖的高级算法及应用是以第一卷的基本抽象概念和方法为基础的。这个新版中的关于基本原理和数据结构的所有素材几乎都是新的。

这本书不仅适合于程序员和计算机科学专业的学生，而且也适合于想利用计算机并想使它运行更快或是想要解决更大问题的人们。这本书中的算法代表了过去50年来所研究的知识主体。对于大量应用问题，这些知识主体已经成为有效使用计算机的不可缺少的部分。从物理学中的N-体模拟问题到分子生物学中的序列分析问题，在此所描述的基本方法在科学研究中已日显重要。另外，对于从数据库系统到Internet搜索引擎，这些方法已经成为现代软件系统的重要组成部分。随着计算机应用的覆盖面越来越广，基本算法的影响也日益显著。本书的目标是要提供一种资源，使广大学生以及专业人士可以了解并有效利用这些算法解决计算机应用中出现的问题。

本书范围

本书共有16章，分为四大部分：基础、数据结构、排序和搜索。这里的说明是想使读者对尽可能多的基本算法有一个了解。本书描述的从二项队列到帕氏线索这个范围内的独创性的方法，都与计算机科学核心的基本范型相关。第二卷由另外四部分组成，涵盖了字符串算法、几何算法、图算法和高级主题。写这些书的主要意图是把各个领域中应用的基本方法集合在一起，从而为用计算机求解问题提供最好的方法。

如果你已经学过计算机科学的一两门课程，如C、Java或C++这样的高级程序设计语言课程，或者可能还有讲授程序设计系统的基本概念的课程，或者具有同等的程序设计经验，那么一定会非常欣赏本书提供的资料。因此，本书是为那些熟悉现代程序设计语言和现代计算

机系统的基本特性的人而编写的。书中给出的参考文献会有助于弥补背景知识的不足。

由于用来支持分析结果的大部分数学知识都包含在本书中（或者做出标记不在本书之中），因而尽管具有完备的数学知识肯定会有帮助，但专门对数学知识的准备不是必要的。

教学中的用法

在教学中如何使用本书内容具有很大的灵活性，这取决于教师的偏好以及学生所做的准备。这里所描述的算法多年以来已经得到广泛应用，而且无论对于实际的程序员还是计算机科学专业的学生，这些算法都代表了基本的知识主体。书中涵盖了足够的基本内容可用作数据结构课程的学习，也有足够详细的高级主题用于算法课程的学习。有些教师可能希望强调与实现和实践有关的内容，而另外一些教师则可能把重点放在分析和理论概念上。

教学中使用的电子文档、程序设计示例作业、为学生提供的交互式练习以及其他课程有关的资料都可在本书的主页上找到。

关于数据结构和算法的基础课程可以把重点放在第二部分的基本数据结构及其他们在第三、四部分实现中的应用。关于算法设计与分析的课程可以把重点放在第一部分和第五部分中的基础内容，然后在第三部分和第四部分研究算法达到良好渐近性能的方法。关于软件工程的课程可能会省略数学和高级算法的内容，并把重点放在如何把给出的算法实现集成到大的程序或系统中。关于算法的课程则可能进行综述并引入所有这些领域的概念。

本书的早期版本在近年来为世界各地的学院或大学用作计算机科学的第二或第三门课程的教材或其他课程的补充阅读材料。在普林斯顿大学，我们的经验表明这本书内容覆盖面广，为主修课程提供计算机科学的导引，并可在后续的算法分析、系统程序设计以及理论计算机科学的课程中对它进行扩充，同时为其他学科的学生提供一整套的技术，使他们能很快学以致用。

这一版中的大多数练习是新添加的，分为几种类型。一类练习的目的是为了测试对课文中内容的理解，要求读者能够完成某个例子或应用课文中描述的概念。另一类练习则涉及实现算法和把算法整理到一起，或者进行实验研究从而对各种算法进行比较以及了解其性质。还有一类练习则是一些重要信息的知识库，其详细程度本身不适合放在正文中。阅读并思考这些练习，会使每个读者受益匪浅。

算法的实用性

若希望更有效地使用计算机，可以把这本书用作参考书，或用于自学。具有程序设计经验的人可以从本书中找到有关某个特殊主题的信息。对于更大范围的读者，尽管某些情况下，某一章中的算法使用了前一章中的方法，但你仍可以独立于本书的其他章节阅读本书的某个章节。

本书的定位是研究有可能实用的算法。本书提供了算法的详尽信息，读者可以放心地实现和调试算法，并使算法能够用于求解某个问题，或者为某个应用提供相关功能。书中包括了所讨论方法的完整实现，并在一系列一致的示例程序中给出了这些操作的描述。由于我们使用了实际代码，而不是伪代码，因而在实际中可以很快地使用这些程序。通过访问本书的主页可以得到程序的代码清单。

实际上，书中算法的实际应用会产生数百幅图表。正是这些图表提供的立体视觉直观地发现了许多算法。

本书详细讨论了算法的特征以及它们可能应用的场合。尽管并不强调，但是书中论述了算法分析与理论计算机科学的联系。在适当的时候，书中都给出了经验性的数据和分析结果用以说明为什么选择使用某些算法。如果有趣，书中还会描述所讨论的实际算法与纯理论结果之间的关系。关于算法性能特征和实现的某种信息的综合、概括和讨论都会贯穿本书的始终。

编程语言

书中所有实现所用的程序设计语言均为C语言。任何特定语言都有优缺点。我们使用C语言是因为它是一种广泛使用的语言，并且能够为本书的实现提供所需的特征。由于没有多少结构是C语言所特有的，因而用C语言编写的程序可以很容易地变成用其他现代编程语言书写的程序。在适当的时候，我们会使用标准C语言中的术语，但本书并不打算成为C语言程序设计的参考手册。

在这一版中有很多新的程序，旧版本的很多程序也已更新，主要目的是使它们在用作抽象数据类型时更具有易读性。对程序所做的广泛的实验性比较研究贯穿在本书中。

本书以前的版本是用Pascal、C++和Modula-3来呈现基本程序的。在本书的主页上可得到这些代码。新程序的代码和用新语言，如Java书写的代码将在适当的时候添加进来。

本书的目标是以尽可能简单、直接的方式呈现算法。在尽可能的情况下利用一致的风格，使得相似的程序看起来相似。对于书中的许多算法，无论使用哪种语言，算法都具有相似性。例如，Quicksort是一种快速排序算法（取了一个著名的例子），无论它是用Algol60、Basic、Fortran、Smalltalk、Ada、Pascal、C、PostScript、Java表示的，还是其他无数程序设计语言和环境表示的，都证明它是一种有效的排序方法。

我们力争编写精致、简明和可移植的代码实现，但同时关注实现的效率，因而我们在开发的各个阶段就试图了解代码的性能特征。第一章包含这种方法的一个详细例子，用以说明如何用这种方法开发一个算法的高效C语言实现，并简略介绍了本书其余部分的内容。

致谢

对于本书早期的版本，很多人提供了有用的反馈信息。特别要提出的是普林斯顿大学和布朗大学的数百名学生们在过去数年的学习过程中一直承受着初稿的粗糙。特别要感谢Trina Avery和Tom Freeman对于第一版的出版所给予的帮助。还要感谢Janet Incerpi，是她的创造力和聪明才智使我们利用早期原始的数字排版硬件和软件制作出本书的第一版。感谢Marc Brown在算法可视化方面做的研究，他创建了本书中的诸多插图，而本书中的很多图表正来源于此。感谢Dava Hanson，对于我提出的关于C语言方面的所有问题，他总是乐于回答。我还要感谢众多读者，他们对各个版本提供了详尽的评论，这其中包括：Guy Almes、Jon Bentley、Marc Brown、Jay Gischer、Allan Heydon、Kennedy Lemke、Udi Manber、Dana Richards、Join Reif、M. Rosenfeld、Stephen Seidman、Michael Quinn和William Ward。

为了完成这一版本，我有幸与Addison-Wesley的Peter Gordon和Debbie Lafferty一起工作。他们耐心地指导这个项目的完成，使其经历了从一般性的修改到大幅度重写的过程。同样我还有幸与Addison-Wesley的许多专业人员一起工作。这个项目的性质使得本书带给他们非同寻常的挑战，对于他们的忍耐力我致以诚挚的感谢。

在写这本书的过程中，我得到了两位良师益友。在此我要特别向他们表示感谢。Steve Summit从技术的角度对各版本的初稿都做了仔细的检查，并向我提供了数千条详尽的意见，

尤其是关于程序方面的建议。Steve很清楚我的目标是要提供精致、有效且实用的实现代码，他的意见不仅帮助我给出实现一致性的度量标准，而且还帮助我对其中一部分作了重大的改进。Lyn Dupre也对我的手稿提供了数千条的意见，这些建议对我来说是无价之宝，不仅帮助我改正和避免了许多语法错误，而且更重要的是，使我找到一种一致性的编写风格，由此才使我把如此繁多的技术资料整理在一起。能够有机会向Steve和Lyn学习，我万分感激。他们的投入对于本书的完成至关重要。

我在这里所写的内容大多受益于Don Knuth的授课和著作，他是我在斯坦福大学的导师。尽管Don对本书没有直接的影响，但在本书中仍然能够感受到他的存在，因为正是他为算法研究奠定了科学基础，才使得像本书这样的工作得以完成。我的朋友兼同事Philippe Flajolet，是使算法分析发展成为一个成熟领域的主力，对这本书具有同样的影响力。

非常感谢普林斯顿大学、布朗大学以及法国国立计算机与自动化研究所（Institute National de Recherce en Informatique et Automatique，INRIA）给予的支持，在这些地方，我完成了本书大部分的工作。还要感谢美国国防部防御分析研究所（Institute for Defense Analysis）以及施乐的帕洛阿尔托研究中心（Xeror Palo Alto Research Center），我在他们那里的访问期间完成了本书的一些工作。本书的某些部分离不开国家自然科学基金（National Science Foundation）和海军研究中心（Office of Naval Research）的慷慨支持。最后，我要感谢Bill Bowen、Aaron Lemonick和Neil Rudenstine，他们为普林斯顿大学建立了一个良好的学术环境，使我能够在这样良好的环境中，在承担众多其他事务的同时完成本书的准备。

Robert Sedgewick

Marly-le-Roi，法国，1983年2月

普林斯顿，新泽西州，1990年1月

詹姆斯镇，罗得岛，1997年8月

有关练习的注释

给练习分类是一件充满风险的事情，因为本书的读者具备的知识背景和经验参差不齐。虽然如此，指导仍然是适宜的，所以许多练习都加了一个记号，以帮助你判断如何动手解决它们。

测试你对内容理解程度的练习标以空心三角符号，如下所示：

▷7.1 按前面例子的风格，给出快速排序算法应用于文件内容为EASYQUESTION每一步的排序结果。

通常，这样的练习是与正文中的例子直接相关。它们并不特别难，但是做这些练习可能教会你一个事实或一个概念，它们可能是你在阅读正文时感到困惑不解的问题。

给正文中添加新的和需要思考信息的练习标以空心圆符号，如下所示：

○13.23 比较你从练习13.22所得结果和从下面过程所得结果：利用程序13.2和程序13.3对一棵N个节点的随机树执行删除最大关键字，并重新插入该关键字的操作，其中$N = 10$，100和1000。对于每个N，要求达到N^2次的插入－删除对操作。

这样的练习鼓励你考虑与书中内容相关的重要概念，或者回答出现在你阅读正文时遇到的一个问题。即使你没有时间做这些练习，你也会发现阅读这些练习是非常有价值的。

具有挑战性的练习标以黑色圆点，如下所示：

•8.45 假设归并排序将文件按随机方式进行划分，而不是恰好平分。使用这样的方法对包含N个元素的文件进行排序，平均需要使用多少次比较？

这种练习可能需要花费大量时间才能完成，这取决于你的经验。一般而言，最有效的方法是分几个时期来解决它们。

少数难度极大的练习标以两个黑色圆点，如下所示：

••15.28 证明由N个随机位串所构建的线索的高度约为$2\lg N$。*提示*：考虑生日问题（见性质14.2）。

这种练习类似于研究文献上陈述的问题，但书中的内容可能为你试图（可能成功）解决它们做好了准备。

*对于考察你的程序设计能力和数学能力的练习，则没有明确记号。*这些要求程序设计能力或数学分析能力的练习是一种自我检查。我们鼓励所有的读者都通过实现算法以测试自己对算法的理解程度。对于程序员或者程序设计课程的学生来说，这样的练习很简单，而对于那些近来很少编程的人来说，则会有一定难度：

4.45 写一个客户程序从命令行的第一个参数中取一个整数N，然后打印出N个*扑克牌局*，方法是把N个项放到一个随机队列中（见练习4.4），然后打印出从队列中一次拣出五张牌的结果。

类似的情况，我们鼓励所有的读者努力探索有关算法性质的分析基础。对于一个科学工作者或者离散数学课程的学生来说，这样的练习很简单，而对于那些近来很少做数学分析的人来说，仍然会有一定的难度：

1.12 在一棵由加权快速合并算法在最坏情况下构造的2^n个节点的树中，试计算从一个节点到树根节点的平均距离。

还有更多的练习需要你去阅读和掌握。我希望这里有足够的练习能够激励你积极地加深对自己感兴趣主题的理解，而不仅仅满足于简单阅读正文所得到的收获。

目 录

第四部分 搜 索

第一部分 基础知识

第1章 引 言

本书的目的是研究各种重要且有用的*算法*（algorithm），即研究适合计算机实现的求解问题的方法。我们将会涉及许多不同领域中的应用，但把重点放在重要且有趣的基本算法上。我们将花费足够的时间理解每个算法的重要特征并考虑一些细节问题。我们的目标是学习大量现今计算机所用的最重要的算法，充分理解这些算法以达到学以致用的目的。

理解书中给出的程序所使用的策略是实现并测试这些程序，试验这些程序的各种变体，讨论它们在小规模例子上的操作，并试图在实际中可能遇到的更大规模的例子上试验它们。我们将利用C程序设计语言来描述算法，因而同时也就提供了有用的实现。我们的程序风格一致，很容易就能改写成其他现代程序设计语言。

我们还关注算法的性能特征，这有助于我们开发算法的改进版本，比较求解同一任务的不同算法，并能预测或保证求解更大问题的性能。理解算法如何执行可能需要试验或者数学分析或者两者都需要。我们考虑许多最重要算法的详细信息，在可行时直接研制分析结果，或者在必要时利用研究文献中的结果。

为了说明研制算法求解的一般方法，本章我们考虑包含求解特定问题的大量算法的一个详细例子。我们考虑的这个问题不是一个玩具问题；它是一个基本的计算任务，并且我们研制的解决方法也可用于大量应用中。我们从一种简单求解方法开始，然后探索这种解法的性能特征，这可以帮助我们理解如何改进算法。在重复几次这样的过程之后，我们就会得到求解问题的一个高效且有用的算法。这个原型例子为通篇使用这个一般方法奠定了基础。

最后对本书内容作一概略讨论以结束本章，其中包括简略描述书中的各个主要部分的组成，以及它们之间的相互关系。

1.1 算法

当我们写一个计算机程序时，一般而言我们是在实现事先设计的求解某个问题的方法。这个方法常常与使用的特定计算机无关；它很可能同样适合于许多计算机和计算机语言。我们必须学习的是如何解决问题的方法，而不是计算机程序本身。术语*算法*用在计算机科学中，用来描述适合于计算机程序实现的求解问题的方法。算法是计算机科学的基础：它们是许多领域研究的核心。

大多数算法关注的是计算中涉及的数据的组织方法。用这种方法建立的对象称为*数据结构*（data structure），它们也是计算机科学研究的核心。这样，算法与数据结构就结合在一起了。在本书中，我们把数据结构看作是算法的副产品或最终产物，因而我们必须研究这些数据结构以便理解算法。简单算法可以导致非常复杂的数据结构，反之，复杂算法可以利用简单的数据结构。我们将在这本书中研究许多数据结构的性质；事实上，将这本书称为《用C语

言表示的算法与数据结构》更合适。

当我们利用计算机帮助我们求解问题时，一般都会面对许多不同的方法。对于小规模的问题，利用哪一个方法是不重要的，只要能够有个方法正确解决问题就行。然而对于大规模问题（或需要求解大量小规模的问题的应用），我们的动机就是设计时间和空间都尽可能高效的方法。

我们学习算法设计的主要原因是这个学科可以使我们节省大量的时间和空间，甚至可能使原本不可能解决的问题得以解决。在我们处理数百万个对象的一个应用中，如果利用一个设计良好的算法会使程序快上数百万倍。我们将会在1.2节和书中许多地方看到这样的例子。与之相比，花额外的钱或时间购买并安装一台新的计算机可能使程序快10或100倍。无论应用领域是什么，精细的算法设计都是求解大规模问题的过程中极其有效的部分。

当开发大规模或复杂计算机程序时，就要做大量工作理解和定义要被求解的问题，控制它的复杂度，并把它分解成为能够容易实现的更小任务。通常分解之后，多数算法容易实现。然而，在大多数情况下，还有少数算法的选择非常关键，因为大多数系统资源将会消耗在运行这些算法上。本书所关注的算法就是这些类型的算法。我们将会研究各个应用领域中用于求解大规模问题的各种基本算法。

计算机系统中的程序共享变得更广泛，因而，尽管我们可能期望在本书中利用大部分的算法，但我们仍然希望实现其中的一小部分算法。然而，实现基本算法的一个简单版本可以帮助我们更好地理解算法，并能更有效地利用高级版本的算法。更重要的是，基本算法经常需要重新实现。这样做的主要原因是我们时常面对具有新特征的全新的计算环境（硬件和软件方面），原实现也许不能最好地利用这些特征。换句话说，我们常常实现适合于具体问题的基本算法，而不是调用系统例程，以使我们的解具有可移植性和持久性。重新实现基本算法的另一个原因是许多计算机系统中共享软件的机制并不足够强大，从而使标准程序适合于在特定任务上有效地执行（或者这样做还不够方便），因而有时实现新的程序更容易一些。

计算机程序常常被过度优化。费力确保一个特定算法的实现最有效也许并不值得，除非这个算法在大量任务中使用，或者多次使用。否则，一个精细的相对简单的实现就足够了。我们可以相信它会运行，与最好的可能版本相比，最坏情况下它的运行速度可能要慢5到10倍。这意味着它可能要多运行几秒钟。与此相比，首先选择恰当的算法可能加速100倍、1000倍，甚至更多。节省的运行时间可能达到数分钟、数小时，甚至更多。在本书中，我们侧重于这些最好算法的最简单的合理实现。

选择某个特定任务的最好算法可能是一个复杂的过程，也许涉及复杂的数学分析。计算机科学中研究这些问题的分支称为*算法分析*（analysis of algorithm）。分析表明我们研究的许多算法具有杰出的性能，还有一些算法经过实践表明可以很好地工作。我们主要的目的是学习求解重要任务的合理算法，然而还要仔细关注这些方法之间可比较的性能。不应该利用不清楚会消耗什么资源的算法，应努力了解我们到底期望我们的算法如何执行。

1.2　典型问题——连通性

假设给定整数对的一个序列，其中每个整数表示某种类型的一个对象，我们想要说明对p-q表示“p连接到q”。假设“连通”关系是可传递的：也就是说如果p和q之间连通，q和r之间连通，那么p和r也连通。我们的目标是写一个过滤集合中的无关对的程序。程序的输入为对p-q，如果已经看到的到那点的数对并不隐含着p连通到q，那么输出该对。如果前面的对确实隐含着p连通到q，那么程序应该忽略p-q，并应该继续输入下一对。图1-1给出了这个过程

的一个例子。

我们的问题是设计能够记录足够多它所看见的数对信息的程序，并能够判定一个新的对象对是否是连通的。非形式地，我们称设计这样一个算法的任务为连通性问题。这个问题出现在许多重要的应用中。我们这里简略地考虑三个例子用以表明问题的本质。

例如，整数可以表示大规模网络中的计算机，而对表示网络中的连接。这样，就可利用程序确定，是需要建立新的p和q能够通信的连接，还是利用已有连接建立通信路径。在这种应用中，可能需要处理数百万个和数十亿个连接，甚至更多。正如我们将要看到的那样，要解决那些没有高效算法的应用问题是不可能的。

3-4	3-4	
4-9	4-9	
8-0	8-0	
2-3	2-3	
5-6	5-6	
2-9		2-3-4-9
5-9	5-9	
7-3	7-3	
4-8	4-8	
5-6		5-6
0-2		0-8-4-3-2
6-1	6-1	

图1-1 连通问题示例

注：给定表示两对象之间连接的整数对序列（左），连通算法的任务是输出那些提供新的连通关系的对（中间）。例如，由于连通关系2-3-4-9隐含在前面的数对中（右边给出了这个证明），因而对2-9不是输出的一部分。

类似地，整数可以表示电网络中的连接点，而对表示连接这些点之间的连线。在这种情况下，如果可能，我们可以利用程序找出连接所有点且没有额外连接的一种方式。事实上，不能保证表中有足够多的边连接所有点，我们将会看到确定是否连通是程序的一个主要应用。

图1-2中说明了这两种类型应用的更大示例。考察这个图可以看出连通问题的难度，如何排列才能快速断定网络中的任何给定两点是连通的？

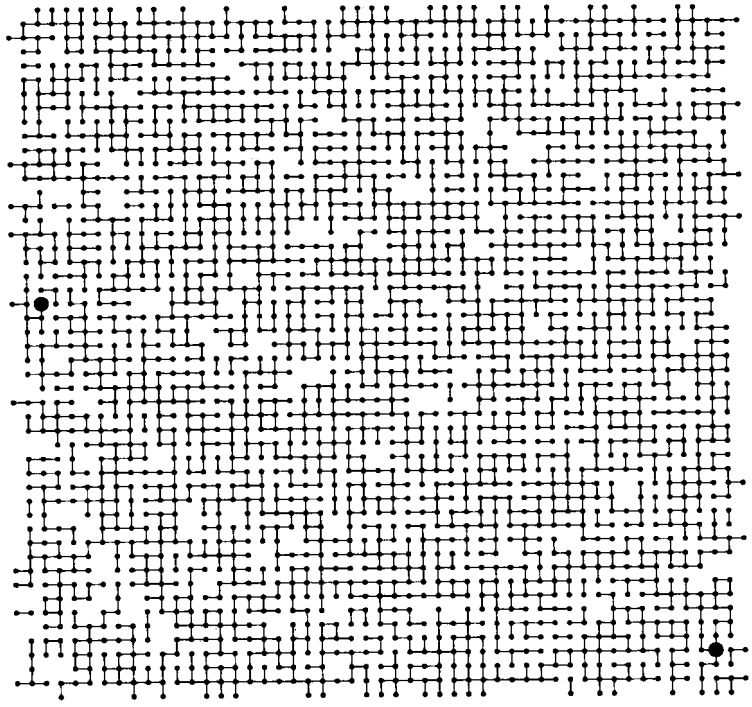


图1-2 大规模连通问题示例

注：连通问题中的对象可以表示连接点，对表示它们之间的连接。正如在这个理想化示例中表明的那样，它可以表示城市中连接建筑物的连线，或者表示计算机芯片上的连线。图形化的表示可以使人们看到不连通点，但算法必须在给定的整数对上才能工作。标示为大黑点的两个点是否是连通的？

还有一个例子出现在某种程序设计环境中，连通性可用来断言两个变量名是否等价。问题是在经过这样的断言序列之后，能够确定两个给定的名字是否等价。这个应用激发了我们打算考虑的几个算法的研制。它直接将我们的问题与一种简单抽象关联起来，为使算法具有广泛应用而提供了一种方法。我们即将看到这一点。

像上一段描述的变量名等价问题这样的应用程序要求我们把每个不同的变量名与一个整数关联起来。这种关联关系也隐含在前面描述的网络连接和电路连接的应用中。在第10章至第16章，我们将会以一种更高效的方法考虑提供这种连接关系的大量算法。因此，不失一般性，本章假设有N个对象，每个都与$0 \sim N-1$之间的一个整数名对应。

我们正在寻求完成特定和良定义任务的程序，可能还想要解决其他许多相关的问题。在研制算法时我们面对的首要任务之一是确信我们已经以合理的方式指定了问题。我们要求算法的越多，它完成任务所需要的时间和空间越多。不可能量化这个关系，并且我们在发现一个问题难以求解或是求解代价昂贵，或是在好的情况下，发现算法可以比原始说明提供更多有用的信息时，我们常常修改这个问题的说明。

例如，我们的连通问题的说明只要求我们的程序知道任意给定对p-q是否是连通的，并不能够表明连接那个对的任何方式。添加这样一个说明的要求会使问题更加困难，会涉及其他的算法，我们将在第5章简略讨论，并在第7章详细讨论。

前面这段提到的说明要比原始说明要求*更多*的信息，我们也可以要求*更少*的信息。例如，我们可能只想回答这样的问题："M个连接足以把N个对象都连接起来吗？"这个问题表明，要研制一个高效的算法，常常需要我们对正在处理的抽象对象进行高级推理。在这种情况下，由图论基本结果可以得出所有N个对象是连通的，当且仅当连通算法输出的对的个数恰好为$N-1$（见5.4节）。换句话说，连通算法永远不会输出多于$N-1$个对，这是因为一旦它输出$N-1$个对，则它从那个时刻遇见的任何对将会是连通的。因此，我们可以修改求解连通问题的程序，增加一个计数器就可以得到一个回答yes-no问题的程序，而不输出那些前面不连通的每个对，当计数器的值为$N-1$时，程序回答"yes"，否则回答"no"。这个问题只是我们希望回答关于连通性的许多问题中的一个例子。输入对的集合称为*图*（graph），输出对的集合称为图的*生成树*，它连接了所有对象。我们在第七部分考察图、生成树以及所有相关算法的性质。

努力确定算法执行的基本操作很重要，这使我们为连通问题设计的算法可以用于许多类似的问题。确切地说，每当我们得到一个新对时，我们必须首先确定它是否表示一个新的连接，然后把已经看到的连接信息合并到已得到的对象的连通关系中，使得它能够检查将要看到的连接。我们把这两个任务封装成为*抽象操作*，用整数输入值表示抽象集合中的元素，然后设计算法和数据结构，使其

- *查找*（find）包含给定数据项的集合。
- 用它们的*并集*（union）替换包含两个给定数据项的集合。

按照这些抽象操作组织我们的算法似乎并不妨碍求解连通性问题，并且这些操作可能用于求解其他的问题。在计算机科学中，特别是在算法设计中，开发更高层次的抽象是一项重要的过程，贯穿本书我们会看到大量这样的情况。在这一章里，我们利用非形式的抽象思维指导我们设计解决连通问题的程序。在第4章里，我们会看到如何用C代码封装抽象。

根据查找和合并抽象操作容易求解连通性问题。在从输入读取一个新的对p-q后，对于对中的每个数执行查找操作。如果对的成员在同一集合中，那么考虑下一对；如果它们不在同一集合中，则执行合并操作，并输出这个对。集合表示*连通分量*（connected component），即那些给定分量中的任何两个对象是连通的对象的集合。这种方法把开发连通问题算法解的过

程变为定义表示集合的数据结构以及开发高效利用这个数据结构的查找和合并算法。

有许多用于表示和处理抽象集合的方法，我们在第4章会更详细地考虑这些方法。在这一章里，我们集中在找到一种能够高效支持合并和查找操作的表示方法上，这些操作用于求解连通问题。

练习

1.1 给定输入0-2，1-4，2-5，3-6，0-4，6-0和1-3，给出连通算法所产生的输出。

1.2 列出图1-1示例中连接两个不同对象的所有不同方式。

1.3 描述一种统计正文中使用合并和查找操作求解连通问题之后的剩余集合的个数的简单方法。

1.3 合并－查找算法

开发求解给定问题高效算法的过程的第一步是*实现解这个问题的一个简单算法*。如果我们需要解决几个容易的特定问题的实例，那么简单实现就能完成这项工作。如果要用更复杂的算法，简单实现可以用于检查小规模例子的正确性，并成为评估算法性能的一个基准。我们总是关注算法的效率，但我们在开发解决问题的第一个程序时更关注的是确保程序的*正确性*。

首先考虑如何存储所有输入对，然后写一个遍历这些输入的函数，然后检查下一对对象是否是连通的。我们会用另一种方法。首先，实际应用中对的个数可能会很大，不能把它们全部放在内存中。其次，更重要的是，即使我们能够把它们都放在内存中，也没有一种简单方法能够由连接关系集合很快地确定两个对象是否是连通的！在第5章中会讨论使用这种方法的一个基本算法，但在这一章中我们考虑的方法更简单，因为它们可以求解难度更小的问题，且这些方法不要求存储所有对，因而是更高效的方法。这些方法利用整数数组，每个整数对应一个对象，用于保存实现合并和查找操作时所需要的必要信息。

数组是基本的数据结构，将在3.2节中详细讨论它。这里是使用它们的一个最简单的形式：我们声明1000个整数的数组，写为a[1000]。a[i]表示引用数组中的第i个整数，其中$0 \leqslant i < 1000$。

程序1.1是求解连通问题的*快速一查找算法*（quick-find algorithm）的一种简单实现。算法的基础是一个整型数组，当且仅当第p个元素和第q个元素相等时，p和q是连通的。初始时，数组中的第i个元素的值为i，$0 \leqslant i < N$。为实现p与q的*合并*操作，我们遍历数组，把所有名为p的元素值改为q。我们也可以选择另一种方式，把所有名为q的元素改为p。

程序1.1 连通问题的快速查找算法

这个程序从标准输入读取小于N的非负整数对序列（对p-q表示“把对象p连接到q”），并且输出还未连通的输入对。程序中使用数组id，每个元素表示一个对象，且具有以下性质，当且仅当p和q是连通的，id[p]和id[q]相等。为简化起见，定义N为编译时的常数。另一方面，也可以从输入得到它，并动态地为它分配id数组（见3.2节）。

```
#include <stdio.h>
#define N 10000
main()
  { int i, p, q, t, id[N];
    for (i = 0; i < N; i++) id[i] = i;
    while (scanf("%d %d\n", &p, &q) == 2)
      {
        if (id[p] == id[q]) continue;
        for (t = id[p], i = 0; i < N; i++)
```