ALGORITHMS In a Nutshell

算法

技术手册

George T. Heineman,
Gary Pollice & Stanley Selkow 著
杨晨 李明 译

O'REILLY®
机械工业出版社
China Machine Press
HZ BOOKS

算法技术手册

George T. Heineman, Gary Pollice &
Stanley Selkow 著

杨晨 李明 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权机械工业出版社出版

机械工业出版社

图书在版编目（CIP）数据

算法技术手册/（美）海涅曼（Heineman, G. T.），波利切（Pollice, G.），塞克欧（Selkow, S.）著；杨晨等译. －北京：机械工业出版社，2009.10

书名原文：Algorithms in a Nutshell

ISBN 978-7-111-28674-5

I. 算… II. ①海… ②波… ③塞… ④杨… III. 电子计算机－算法理论－技术手册 IV. TP301.6-62

中国版本图书馆CIP数据核字（2009）第189407号

北京市版权局著作权合同登记

图字：01-2009-2368号

书　　名/ 算法技术手册
书　　号/ ISBN 978-7-111-28674-5
责任编辑/ 陈佳媛
封面设计/ Karen Montgomery，张健
出版发行/ 机械工业出版社
地　　址/ 北京市西城区百万庄大街22号（邮政编码100037）
印　　刷/ 北京京师印务有限公司
开　　本/ 178毫米×233毫米　16开本　21.75印张
版　　次/ 2010年3月第1版　2010年3月第1次印刷
定　　价/ 55.00元（册）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
客服热线：(010) 88378991; 88361066
购书热线：(010) 68326294; 88379649; 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

算法技术手册

O'Reilly Media, Inc.介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构O'Reilly Media, Inc.授权机械工业出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media, Inc.是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时也是联机出版的先锋。

从最畅销的*The Whole Internet User' Guide & Catalog*（被纽约公共图书馆评为20世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商——每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

译者序

算法，神秘而晦涩的词汇。算法，是计算机科学中最重要同时也是最基础的一环。从开始学习计算机，我们就深知，算法是整个计算机科学的核心。然而直至我们工作数年后，能够真正学好算法的人，却依旧是凤毛麟角。这并不是计算机教育的错，也不是计算机从业人员的错，更不是算法的错。长久以来，算法就像古老的咒语，算法背后高深的数学知识更让人望而生畏。其实，我们始终没有找到一条从理论走向实践的路。

在这里，我们很高兴能向大家介绍本书。它正是能够带领你学好算法的一本不可多得的好书。

本书的三位作者是伍斯特理工学院的教授，其中George T. Heineman毕业于达特茅斯学院和哥伦比亚大学，曾经获得过GE、IBM和AT&T的研究奖金，在软件工程方面有独到的研究。而Gary Pollice曾经供职于Rational Software、Sun等多家巨头，有着丰富的工业界经验，知道如何将学术和工业结合起来。Stanley M. Selkow毕业于卡内基梅隆大学和宾夕法尼亚大学，擅长图论和算法设计。本书由这三位伍斯特理工学院计算机理论专家合著，向我们展示了工业界和学术界对算法的不同看法以及如何高效地将理论和实践相结合。本书搭建了一条真正属于开发者的路。

本书的读者主要面向本科生以及程序设计人员，同样也适用于产品和项目管理人员。由于译者的知识和经验有限，翻译中难免有疏漏或错误，敬请广大读者谅解并批评指正。

杨晨 李明

2009年7月

作者简介

George T. Heineman是伍斯特理工学院计算机科学系的一名副教授,专注于软件工程方面的研究。他还参与编写了一本论述基于组件的软件工程方面的书:《Putting the Pieces Together》,由 Addison-Wesley于2001年出版。George是2005年国际组件软件工程研讨会的议程主席。

Gary Pollice自称是一个乖戾的人(其实就是一个顽固的、坏脾气的老人),在工业界花费了35年来探索他究竟希望成为什么。但是在2003年,他毅然决定以不成熟之身从工业界转移到学术的殿堂。在这里他可以用激进的话语影响下一代软件工程师,“为你的客户开发软件”,“学会如何成为团队的一员”,“要思考软件设计、代码质量、优雅程度和正确度”和“只要你成为牛人,那么即使成为书呆子也无所谓啦”。

Gary是伍斯特理工学院的一位实践教授(这是一个头衔,也就是说他在成为一名教授之前有一份真实的工作)。由于他对一起工作数年的WPI的毕业生们印象非常好,所以决定来WPI做一名教授。他和妻子Vikki,以及两条狗Aloysius和Ignatius一起住在麻省中部。他一直在做和极客相关的事情。你可以通过他的WPI网页*http://web.cs.wpi.edu/~gpollice/*来了解他,还可以随意地给他留言,抱怨或者赞美本书。

Stanley Selkow是伍斯特理工学院计算机系的一名教授,于1965年在卡内基理工学院(现卡内基梅隆大学)获电子电气工程学士学位,1970年在宾夕法尼亚大学获电子电气工程博士学位。在1968～1970年间,他在马里兰州贝塞斯达的国家卫生研究所从事公众健康服务相关的工作。自1970年,他先后在田纳西的诺克斯维尔和麻省的伍斯特从事教员工作,他也在蒙特利尔、重庆、洛桑和巴黎做过访问学者。他的主要研究领域是图论和算法设计。

封面介绍

本书封面的动物是一只寄居蟹(Pagurus bernhardus)。在世界各地有超过500种寄居蟹。它们大多为水生,生活在珊瑚礁和潮池的盐水中。一些寄居蟹,尤其是在热带地区的,是陆生的。例如一种名为盗蟹的寄居蟹,它能够长到椰子那么大。陆生寄居蟹在它们的壳里面存放着少量的水,用来帮助呼吸以及保持腹部湿润。

寄居蟹和其他螃蟹不一样,它们不需要一个坚硬的属于自己的外壳,它们寻找食肉动物不能食用的腹足动物(例如蜗牛)的外壳作为庇护所。它们特别偏好玉黍螺和海螺的

壳。随着身体的长大，它们需要寻找更大的壳寄居。如果他们将身体的任何部分暴露出来，那么会非常容易受到食肉动物的攻击；此外，如果没有一个合适的外壳，就会阻碍它们的成长，因为昆虫腹足动物的壳是有限的，所以竞争也是一个问题。

寄居蟹是十足（也就是说十只脚）甲壳动物。在它们的五对足中，第一对是钳子，或者是螯，较大的那个是它们用来防御和撕碎食物用的，而较小的是用来辅助进食的。第二对和第三对足帮助它们走路，最后两对足用于将它们固定在壳中。

寄居蟹具有明显的甲壳动物的特征：它们没有内骨架，而是有一个钙质的外骨骼。它们也有两只复眼，两对触角（它们用来感知味道和振动）并拥有三对口器。在触角的底部是一对触角腺，用来排出废物。

可以经常看见海葵附在寄居蟹的壳上。海葵以这种方式移动并食用寄居蟹的食物残渣，不过作为交换，海葵能够伪装寄居蟹以逃过海洋食肉动物的捕食，例如鱼和章鱼。其他的食肉动物也包括：鸟和其他的螃蟹，以及一些哺乳动物（例如人）。

寄居蟹号称“海洋的垃圾收集器”，它能够吃掉几乎所有的东西，例如海岸上腐烂的物质，因此它们在海岸清洁中扮演着一个非常重要的角色。作为杂食动物，它们的食物是极其多样化的，从虫子到有机废品（例如草和叶子）无所不有。

封面的图片来自于Johnson的《Library of Natural History》卷2。

目录

前言

就像《黑客帝国》里面的Trinity所说的：

> “Neo，是这个问题驱使着我们，是这个问题带你来到这儿。”
> 你知道这个问题，我也是。

作为本书的作者，我们将回答引领你到此的问题：

> 我能够使用某个算法解决我的问题吗？如果可以，那么怎么实现呢？

你也许并不需要理解一个算法为什么是正确的。如果你需要，那么请看看其他的资料，例如1180页的算法圣经 ——《算法导论》，作者是Thomas H. Cormen等（2001）。在那本书中你会了解到推论、定理以及证明；你也会从一些练习题和逐步递进的样例中看到算法是如何执行的。也许你会惊奇地发现，在算法导论中你找不到任何的实际代码，仅仅是一些伪代码的片段，伪代码是无数的算法教科书用来阐述算法的高级描述手段。在课堂上，这些教科书是非常重要的，但是在实际软件开发中，它们却起不到应有的作用，因为这些书假定伪代码都能够直接变成实际代码。

我们希望经验丰富的程序员在寻找问题的解决方案时，能够频繁参考本书。作为一名程序员，你每天要解决的问题都能在这里找到解决方案。在软件中，算法是决定成败的关键因素，在这里你能够了解到哪些决定能够改善关键算法的性能，也能够找到适合你的需求和解决方案的实际代码。

所有的算法都有实现，并且都使用测试工具经过仔细测试，以确保其正确性。而且，它们有足够的代码文档，能在这本书的代码库附录中找到它们。我们严格地遵照一系列的原则来设计算法、实现算法，以及编写这本书。如果这些原则对你很有意义，那么这本书也会同样有用。

原则：使用实际代码，而不是伪代码

为了计算最大网络流，一个实践者应该做些什么才能将图P-1的Ford-Fulkerson算法描述转换成实际代码呢？

Ford-Fulkerson Algorithm:
Input Graph G with flow capacity c, a source node s, and a sink node t
Output A flow f from s to t which is a maximum

```
1.  f(u,v) ← 0 for all edges (u,v)
2.  while (there is a path p from s to t in G_f such that c_f(u,v)>0 for all edges (u,v)∈p) do
3.      Find c_f(p) = min { c_f(u,v) | (u,v)∈p }
4.      foreach edge c_f(u,v)∈p do
5.          f(u,v) ← f(u,v) + c_f(p)    // Send flow along the path
6.          f(v,u) ← f(v,u) – c_f(p)    // The flow might be "returned" later
end
```

图P-1：教科书中常见的伪代码

图中的算法描述来自于维基百科（*http://en.wikipedia.org/wiki/Ford_Fulkerson*），这个描述与《算法导论》上的伪代码极其相似。最好还是不要期望一个软件的开发者能够根据这个Ford-Fulkerson算法的描述开发出实际的代码。翻到第8章，对比一下我们的代码。我们只使用有注释的，并且是精心设计过的代码。在你自己写的代码或者软件系统中使用我们提供的现成代码，或者这些代码的逻辑吧。

一些算法教科书确实有完整的C或者Java代码。但是这些教科书的目的通常是教初学者编程语言，或者是解释如何实现抽象数据类型。而且代码都只是在页面的狭窄边栏，作者通常都会忽略注释和错误处理，或者使用在实际应用中不会用到的快捷方法。我们相信程序员能够从有注释的，并且是精心设计过的代码中学到更多的东西，这就是我们为什么做如此多的工作来开发算法的实际解决方案。

原则：将算法和将要解决的问题分开

如果不和具体的问题联系起来，我们很难“泛化地”实现一个算法。我们反对那些给出了算法完全实现的书，这些书上的实现将泛化问题和特定问题纠结在一起，从而很难看出算法的原始结构。更糟糕的是，很多可用的实现依赖于一些特定的数据存储方式，虽然这种方式能够容易被机器理解，但是很难被人理解。

我们将会为泛化的问题和特殊的问题各设计算法的实现。例如，在第7章，当我们描述A*搜索算法的时候，我们用了一个例子，叫做八数码问题（在一个3×3格子的棋盘中移

动编号为1～8的小方块）。A*算法的实现依赖于一系列良好定义的接口。因为有良好定义的接口，实现这些接口的类能够清晰地封装八数码这类特定问题的细节。

在本书中，我们使用了大量的编程语言，并且遵循一种严格的设计规范，使得代码可读性高并且生成高效的解决方案。由于我们具备软件工程背景，所以根据泛化的算法和特定领域的解决方案设计一个清晰的接口是一件很容易的事情。按照这样的流程编码，产生的软件是易于测试、维护并且能够根据面临的问题即时扩展。另外一个好处是读者能够更加容易地阅读和理解算法的描述和结果。当选择了一个算法，我们将会告诉读者，如何将我们编写的可读并且高效的代码转换成高度优化（虽然稍稍降低了可读性）并且性能优秀的代码。毕竟，优化是在问题已经解决之后才进行的，而且客户需要的是运行更快的代码。即便如此，我也认为我们需要听从C. A. R. Hoare的建议：“过早的优化是一切问题之源”。

原则：仅仅讲述足够的数学

很多算法注重专门关注于证明算法的正确性，并且抽象地解释其细节。我们关注的却是如何将算法应用到实践中去。最后我们才会介绍一些数学知识，这些数学知识都是为了读者能够更好地理解数据结构和解的控制流程。

例如，一个人需要理解在很多算法中使用的集合和二叉树的性质。但是，没有必要证明二叉树的高度如何得到，并且解释红黑二叉树是如何平衡的。如果你需要了解这些细节，请阅读《算法导论》的第13章。我们仅仅是需要才会解释结果，如果读者需要理解如何证明结论，我们将会告诉读者在哪儿能够找到数学证明。

在这本书你将会学习到使用一些关键术语和分析技术，并且基于数据结构的功能来区分算法行为。

原则：用经验来支持数学分析

在这本书中，我们从数学角度来分析算法的性能，以帮助程序员了解在哪种情况下算法能够得到最好的性能。我们将会提供现成的代码样例，在相关代码库中，有大量的JUnit（*http://sourceforge.net/projects/junit*）测试样例为每个算法的实现提供了文档。我们也会生成基准性能数据，供分析算法性能时参考。

我们将每个算法归到一个特定的性能族中，并且提供基准测试数据来得到算法的性能，支持我们的分析结论。有一些算法是那些具有数学背景的算法设计人员证明能够非常高效但是却不可能实现的，我们需要避免使用这样的算法。我们将在各种平台上执行我们的算法，以证明算法的高效并不依赖于特定平台，而是由于其优秀的设计。

附录包含了我们采用的基准测试方法的全部细节，并且这个基准测试能够独立地验证书中描述的所有性能结论。我们能够给你的建议在开源社区非常常见：“你得到的利益也许会不一样”。虽然你不可能准确地复制我们的结论，但是你能看出我们描述的趋势。我们鼓励你在决定使用哪个算法的时候使用我们的这种基于经验的方法。

目标读者

如果你被困在一个沙漠孤岛上，并且只能选择一本算法书，我们推荐Donald Knuth的《计算机程序设计艺术（卷1～3）》（1998）。Knuth在这本书中描述了大量的数据结构和算法，并且进行了精巧的处理和分析。这本书包含了大量的脚注和练习，并且能够帮助一名程序员在接下来的时间中保持活力和竞争力。这些练习非常有挑战性，并且你能够直接接触到Knuth的思想。

但是你没有被困在孤岛上，对吗？你接手了一些劣质的代码，而且这些代码必须在周五前进行改进，你需要知道如何去处理才行！

在你面对一个算法问题，需要解决一个特定的问题或者对现有解决方案进行改进的时候，我们希望本书能够成为你的第一选择。为解决各种问题，我们广泛地讨论了现存的算法，并且遵循以下原则：

- 我们使用模式化的统一格式来描述每一个算法，进行每一次讨论并解释算法的重点。正因如此，我们的书才如此易读。我们才能够看出相似的设计会对不同的算法产生什么样的影响。
- 在这本书中，我们使用了不同的编程语言（包括C、C++、Java还有Ruby）来描述算法。得益于此，我们能够使用你熟悉的编程语言对算法进行详细的讨论。
- 我们描述了算法的期望性能，并且根据经验提供了支持这些结论的证据。无论你相信数学还是相信实际的运行时间，你都会被我们说服。

对软件工程师、程序员以及设计师来说，我们希望这本书能够派上用场。为了达到目标，你需要参考大量的关于实际解决方案和算法的资料，才能够解决手头的实际问题。你已经知道了如何用多种语言编写一个程序。你也知道了计算机科学的关键数据结构，例如数组、链表、栈、队列、散列表、二叉树还有有向图或者无向图。你不需要实现这些数据结构，因为它们大都在库函数中可以找到。

我们希望你能从这本书中学到如何选择并且测试解决方案以快速高效地解决问题，也能够学到一些高级数据结构和一些使用标准数据结构的新方法，来改善程序的性能。在选择算法时，如果你可以知道什么样的决定可以改善解决方案的效率，那么就能够提高你解决问题的能力。

本书组织方式

本书分为三个部分。第一部分（第1章～第3章）介绍了算法的必要数学基础，以便于读者理解本书的描述。在每个算法的论述中，我们使用了一种模式化的格式。我们仔细地设计这个格式，确保前后一致。第二部分（第4章～第9章）介绍了一系列的算法。这些章节的每个独立部分都是一个完备的算法描述。

第三部分（第10章和第11章）为那些感兴趣的读者提供一些较为高深的资源。当没有一个高效的解决方案来解决问题，而且“最后的线索”为解决问题提供了有意义的线索的时候，这个部分就能够告诉我们如何利用这些线索来解决问题。最后我们以一个重要领域的讨论结束本书。这个讨论在第2章之所以被忽略，是因为它的内容太过高深，太过前沿，甚至还没有被证明。第四部分包括了一个附录，这个附录描述了本书每章中对算法进行评测的方法以及数学分析。在业界，这是一种标准的基准测试方法，但是几乎没有算法教科书介绍这个方法。

本书体例

在印刷上的一些例行惯例：

代码（Code）

这些代码都是直接取自代码库，是现实中使用的代码。

斜体（*Italic*）

表示这个术语用于描述算法和数据结构。同样在伪代码描述中表示变量。

等宽字体（Constant Width）

表示实现中的软件元素，例如Java类、C语言的数组名称以及常量（`true`或者`false`）。

小型大写字母（SMALL CAPS）

表示算法名称。

在本书中，我们引用了大量的书籍、文章和网站。这些引用都用括号标示出来，例如（Cormen等，2001），每一章最后部分列出本章所使用的参考文献。正文中的引用列出作者的名字和文献的年份。

本书中的所有URL都在2008年8月时验证有效，而且我们抛弃了那些看起来不久就会失效的URL。短URL，例如*http://www.oreilly.com*，直接在文本中使用；否则，这些URL会放入参考文献中。