算法图解 (图灵程序设计丛书)

作者：巴尔加瓦(Aditya Bhargava)

版权信息

书名：算法图解

作者：[美] Aditya Bhargava

译者：袁国忠

ISBN：978-7-115-44763-0

版权声明

谨以此书献给我的父母、Sangeeta和Yogesh

前言

我因为爱好而踏入了编程殿堂。*Visual Basic 6 for Dummies* 教会了我基础知识，接着我不断阅读，学到的知识也越来越多，但对算法却始终没搞明白。至今我还记得购买第一本算法书后的情景：我琢磨着目录，心想终于要把这些主题搞明白了。但那本书深奥难懂，看了几周后我就放弃了。直到遇到一位优秀的算法教授后，我才认识到这些概念是多么地简单而优雅。

几年前，我撰写了第一篇图解式博文。我是视觉型学习者，对图解式写作风格钟爱有加。从那时候起，我撰写了多篇介绍函数式编程、Git、机器学习和并发的图解式博文。顺便说一句，刚开始我的写作水平很一般。诠释技术概念很难，设计出好的示例需要时间，阐释难以理解的概念也需要时间，因此很容易对难讲的内容一带而过。我本以为自己已经做得相当好了，直到有一篇博文大受欢迎，有位同事却跑过来跟我说：“我读了你的博文，但还是没搞懂。”看来在写作方面我要学习的还有很多。

在撰写这些博文期间，Manning出版社找到我，问我想不想编写一本图解式图书。事实证明，Manning出版社的编辑对如何诠释技术概念很在行，他们教会了我如何做。我编写本书的目的就是要把难懂的技术主题说清楚，让这本算法书易于理解。与撰写第一篇博文时相比，我的写作水平有了长足进步，但愿你也认为本书内容丰富、易于理解。

致谢

感谢Manning出版社给我编写本书的机会，并给予我极大的创作空间。感谢出版人Marjan Bace，感谢Mike Stephens领我入门，感谢Bert Bates教我如何写作，感谢Jennifer Stout的快速回复以及大有帮助的编辑工作。感谢Manning出版社的制作人员，他们是Kevin Sullivan、Mary Piergies、Tiffany Taylor、Leslie Haimes以及其他幕后人员。另外，还要感谢阅读手稿并提出建议的众人，他们是Karen Bensdon、Rob Green、Michael Hamrah、Ozren Harlovic、Colin Hastie、Christopher Haupt、Chuck Henderson、Pawel Kozlowski、Amit Lamba、Jean-Francois Morin、Robert Morrison、Sankar Ramanathan、Sander Rossel、Doug Sparling和Damien White。

感谢一路上向我伸出援手的人：Flaskhit游戏专区的各位教会了我如何编写代码；很多朋友帮助审阅手稿、提出建议并让我尝试不同的诠释方式，其中包括Ben Vinegar、Karl Puzon、Alex Manning、Esther Chan、Anish Bhatt、Michael Glass、Nikrad Mahdi、Charles Lee、Jared Friedman、Hema Manickavasagam、Hari Raja、Murali Gudipati、Srinivas Varadan等；Gerry Brady教会了我算法。还要深深地感谢算法方面的学者，如CLRS1 、高德纳和Strang。我真的是站在了巨人的肩上。

1 《算法导论》四位作者的姓氏（Thomas H. Cormen、Charles E. Leiserson、Ronald L. Rivest和Clifford Stein）首字母缩写。——译者注

感谢爸爸、妈妈、Priyanka和其他家庭成员，感谢你们一贯的支持。深深感谢妻子Maggie，我们的面前还有很多艰难险阻，有些可不像周五晚上待在家里修改手稿那么简单。

最后，感谢所有试读本书的读者，还有在论坛上提供反馈的读者，你们让本书的质量更上了一层楼。

关于本书

本书易于理解，没有大跨度的思维跳跃，每次引入新概念时，都立即进行诠释，或者指出将在什么地方进行诠释。核心概念都通过练习和反复诠释进行强化，以便你检验假设，跟上步伐。

书中使用示例来帮助理解。我的目标是让你轻松地理解这些概念，而不是让正文充斥各种符号。我还认为，如果能够回忆起熟悉的情形，学习效果将达到最佳，而示例有助于唤醒记忆。因此，如果你要记住数组和链表（第2章）之间的差别，只要想想在电影院找座位就坐的情形。另外，不怕你说我啰嗦，我是视觉型学习者，因此本书包含大量的图示。

本书内容是精挑细选的。没必要在一本书中介绍所有的排序算法，不然还要维基百科和可汗学院做什么。书中介绍的所有算法都非常实用，对我从事的软件工程师的工作大有帮助，还可为阅读更复杂的主题打下坚实的基础。祝你阅读愉快！

路线图

本书前三章将帮助你打好基础。

- 第1章：你将学习第一种实用算法——二分查找；还将学习使用大O表示法分析算法的速度。本书从始至终都将使用大O表示法来分析算法的速度。
- 第2章：你将学习两种基本的数据结构——数组和链表。这两种数据结构贯穿本书，它们还被用来创建更高级的数据结构，如第5章介绍的散列表。
- 第3章：你将学习递归，一种被众多算法（如第4章介绍的快速排序）采用的实用技巧。

根据我的经验，大O表示法和递归对初学者来说颇具挑战性，因此介绍这些内容时我放慢了脚步，花费的篇幅也较长。

余下的篇幅将介绍应用广泛的算法。

- 问题解决技巧：将在第4、8和9章介绍。遇到问题时，如果不确定该如何高效地解决，可尝试分而治之（第4章）或动态规划（第9章）；如果认识到根本就没有高效的解决方案，可转而使用贪婪算法（第8章）来得到近似答案。
- 散列表：将在第5章介绍。散列表是一种很有用的数据结构，由键值对组成，如人名和电子邮件地址或者用户名和密码。散列表的用途之大，再怎么强调都不过分。每当我需要解决问题时，首先想到的两种方法是：可以使用散列表吗？可以使用图来建立模型吗？
- 图算法：将在第6、7章介绍。图是一种模拟网络的方法，这种网络包括人际关系网、公路网、神经元网络或者任何一组连接。广度优先搜索（第6章）和狄克斯特拉算法（第7章）计算网络中两点之间的最短距离，可用来计算两人之间的分隔度或前往目的地的最短路径。
- K最近邻算法（KNN）：将在第10章介绍。这是一种简单的机器学习算法，可用于创建推荐系统、OCR引擎、预测股价或其他值（如“我们认为Adit会给这部电影打4星”）的系统，以及对物件进行分类（如“这个字母是Q”）。
- 接下来如何做：第11章概述了适合你进一步学习的10种算法。

如何阅读本书

本书的内容和排列顺序都经过了细心编排。如果你对某个主题感兴趣，直接跳到那里阅读即可；否则就按顺序逐章阅读吧，因为它们都以之前介绍的内容为基础。

强烈建议你动手执行示例代码，这部分的重要性再怎么强调都不过分。可以原封不动地输入代码，也可从www.manning.com/books/grokking-algorithms 或https://github.com/egonschiele/grokking_algorithms 下载，再执行它们。这样，你记住的内容将多得多。

另外，建议你完成书中的练习。这些练习都很短，通常只需一两分钟就能完成，但有些可能需要5~10分钟。这些练习有助于检查你的思路，以免偏离正道太远。

读者对象

本书适合任何具备编程基础并想理解算法的人阅读。你可能面临一个编程问题，需要找一种算法来实现解决方案，抑或你想知道哪些算法比较有用。下面列出了可能从本书获得很多帮助的部分读者。

- 业余程序员
- 编程培训班学员
- 需要重温算法的计算机专业毕业生
- 对编程感兴趣的物理或数学等专业毕业生

代码约定和下载

本书所有的示例代码都是使用Python 2.7编写的。书中在列出代码时使用了等宽字体。有些代码还进行了标注，旨在突出重要的概念。

本书的示例代码可从出版社网站下载，也可从https://github.com/egonschiele/grokking_algorithms 下载。

我认为，如果能享受学习过程，就能获得最好的学习效果。请尽情地享受学习过程，动手运行示例代码吧！

作者在线

购买英文版的读者可免费访问Manning出版社管理的专用网络论坛，并可以评论本书、提出技术性问题以及获得作者和其他读者的帮助。若要访问并订阅该论坛，可在浏览器的地址栏中输入www.manning.com/books/grokking-algorithms 。这个网页会告诉你注册后如何进入论坛、可获得哪些帮助以及讨论时应遵守的规则。

Manning出版社致力于为读者和作者提供能够深入交流的场所。然而，作者参与论坛讨论纯属自愿，没有任何报酬，因此Manning出版社对其参与讨论的程度不做任何承诺。建议你向作者提些有挑战性的问题，以免他失去参与讨论的兴趣！只要本书还在销售，你就能通过出版社的网站访问作者在线论坛以及存档的讨论内容。

第 1 章　算法简介

本章内容

- 为阅读后续内容打下基础。
- 编写第一种查找算法——二分查找。
- 学习如何谈论算法的运行时间——大O表示法。
- 了解一种常用的算法设计方法——递归。

1.1　引言

算法 是一组完成任务的指令。任何代码片段都可视为算法，但本书只介绍比较有趣的部分。本书介绍的算法要么速度快，要么能解决有趣的问题，要么兼而有之。下面是书中一些重要内容。

- 第1章讨论二分查找，并演示算法如何能够提高代码的速度。在一个示例中，算法将需要执行的步骤从40亿个减少到了32个！
- GPS设备使用图算法来计算前往目的地的最短路径，这将在第6、7和8章介绍。
- 你可使用动态规划来编写下国际跳棋的AI算法，这将在第9章讨论。

对于每种算法，本书都将首先进行描述并提供示例，再使用大O表示法讨论其运行时间，最后探索它可以解决的其他问题。

1.1.1　性能方面

好消息是，本书介绍的每种算法都很可能有使用你喜欢的语言编写的实现，因此你无需自己动手编写每种算法的代码！但如果你不明白其优缺点，这些实现将毫无用处。在本书中，你将学习比较不同算法的优缺点：该使用合并排序算法还是快速排序算法，或者该使用数组还是链表。仅仅改用不同的数据结构就可能让结果大不相同。

1.1.2　问题解决技巧

你将学习至今都没有掌握的问题解决技巧，例如：

- 如果你喜欢开发电子游戏，可使用图算法编写跟踪用户的AI系统；
- 你将学习使用K最近邻算法编写推荐系统；
- 有些问题在有限的时间内是不可解的！书中讨论NP完全问题的部分将告诉你，如何识别这样的问题以及如何设计找到近似答案的算法。

总而言之，读完本书后，你将熟悉一些使用最为广泛的算法。利用这些新学到的知识，你可学习更具体的AI算法、数据库算法等，还可在工作中迎接更严峻的挑战。

需要具备的知识

要阅读本书，需要具备基本的代数知识。具体地说，给定函数$f(x) = x \times 2$，$f(5)$的值是多少呢？如果你的答案为10，那就够了。

另外，如果你熟悉一门编程语言，本章（以及本书）将更容易理解。本书的示例都是使用Python编写的。如果你不懂任何编程语言但想学习一门，请选择Python，它非常适合初学者；如果你熟悉其他语言，如Ruby，对阅读本书也大有帮助。

1.2 二分查找

假设要在电话簿中找一个名字以K打头的人，（现在谁还用电话簿！）可以从头开始翻页，直到进入以K打头的部分。但你很可能不这样做，而是从中间开始，因为你知道以K打头的名字在电话簿中间。

又假设要在字典中找一个以O打头的单词，你也将从中间附近开始。

现在假设你登录Facebook。当你这样做时，Facebook必须核实你是否有其网站的账户，因此必须在其数据库中查找你的用户名。如果你的用户名为karlmageddon，Facebook可从以A打头的部分开始查找，但更合乎逻辑的做法是从中间开始查找。

这是一个查找问题，在前述所有情况下，都可使用同一种算法来解决问题，这种算法就是**二分查找**。

二分查找是一种算法，其输入是一个有序的元素列表（必须有序的原因稍后解释）。如果要查找的元素包含在列表中，二分查找返回其位置；否则返回`null`。

下图是一个例子。

下面的示例说明了二分查找的工作原理。我随便想一个1～100的数字。

你的目标是以最少的次数猜到这个数字。你每次猜测后，我会说小了、大了或对了。

假设你从1开始依次往上猜，猜测过程会是这样。

这是**简单查找**，更准确的说法是**傻找**。每次猜测都只能排除一个数字。如果我想的数字是99，你得猜99次才能猜到！

1.2.1 更佳的查找方式

下面是一种更佳的猜法。从50开始。

小了，但排除了**一半**的数字！至此，你知道1～50都小了。接下来，你猜75。

大了，那余下的数字又排除了一半！使用二分查找时，你猜测的是中间的数字，从而每次都将余下的数字排除一半。接下来，你猜63（50和75中间的数字）。

这就是二分查找，你学习了第一种算法！每次猜测排除的数字个数如下。

不管我心里想的是哪个数字，你在7次之内都能猜到，因为每次猜测都将排除很多数字！

假设你要在字典中查找一个单词，而该字典包含240 000个单词，你认为每种查找最多需要多少步？

如果要查找的单词位于字典末尾，使用简单查找将需要240 000步。使用二分查找时，每次排除一半单词，直到最后只剩下一个单词。

因此，使用二分查找只需18步——少多了！一般而言，对于包含n个元素的列表，用二分查找最多需要$\log_2 n$步，而简单查找最多需要n步。

对数

你可能不记得什么是对数了，但很可能记得什么是幂。$\log_{10} 100$相当于问“将多少个10相乘的结果为100”。答案是两个：$10 \times 10 = 100$。因此，$\log_{10} 100 = 2$。对数运算是幂运算的逆运算。

对数是幂运算的逆运算

本书使用大O表示法（稍后介绍）讨论运行时间时，log指的都是$\log_2$。使用简单查找法查找元素时，在最糟情况下需要查看每个元素。因此，如果列表包含8个数字，你最多需要检查8个数字。而使用二分查找时，最多需要检查$\log n$个元素。如果列表包含8个元素，你最多需要检查3个元素，因为$\log 8 = 3$（$2^3 = 8$）。如果列表包含1024个元素，你最多需要检查10个元素，因为$\log 1024 = 10$（$2^{10} = 1024$）。

说明

本书经常会谈到log时间，因此你必须明白对数的概念。如果你不明白，可汗学院（khanacademy.org）有一个不错的视频，把这个概念讲得很清楚。

说明

仅当列表是有序的时候，二分查找才管用。例如，电话簿中的名字是按字母顺序排列的，因此可以使用二分查找来查找名字。如果名字不是按顺序排列的，结果将如何呢？

下面来看看如何编写执行二分查找的Python代码。这里的代码示例使用了数组。如果你不熟悉数组，也不用担心，下一章就会介绍。你只需知道，可将一系列元素存储在一系列相邻的桶（bucket），即数组中。这些桶从0开始编号：第一个桶的位置为#0，第二个桶为#1，第三个桶为#2，以此类推。

函数`binary_search`接受一个有序数组和一个元素。如果指定的元素包含在数组中，这个函数将返回其位置。你将跟踪要在其中查找的数组部分——开始时为整个数组。

```
low = 0
high = len(list) - 1
```

你每次都检查中间的元素。

```
mid = (low + high) / 2  ←---如果(low + high)不是偶数，Python自动将mid向下取整。
```