目录

第一部分

基础知识

第一章 简介

当你设计和分析算法时，你需要能够描述算法的运行方式以及如何设计算法。你还需要一些数学工具来证明你的算法是正确且高效的。这部分内容将帮助你入门。本书的后续部分将在此基础上展开。

第 1 章提供了算法及其在现代计算系统中的位置的概述。本章定义了算法的概念并列举了一些例子。它还提出了将算法视为一种技术的观点，与高速的硬件、图形用户界面（GUI）、面向对象系统和网络等技术并列。

在第 2 章中，我们首次见到了解决排序 n 个数字序列问题的算法。它们以伪代码形式编写，虽然不能直接转换为任何传统的编程语言，但足够清晰地传达了算法的结构，以便你能够在自己选择的编程语言中实现它。我们研究的排序算法包括插入排序（使用增量方法）和归并排序（使用递归技术，称为“分而治之”）。虽然每个算法所需的时间随 n 的值增加而增加，但增长的速率在这两个算法之间有所不同。我们在第 2 章中确定了这些运行时间，并开发了一个有用的“渐近”符号来表示它们。

第 3 章对渐近符号进行了精确定义。我们将使用渐近符号来界定函数的增长范围，最常见的情况是描述算法运行时间的函数，上下界都包括在内。该章节首先非正式地定义了最常用的渐近符号，并给出了如何应用它们的示例。然后，它正式地定义了五种渐近符号，并介绍了将它们组合使用的约定。第 3 章的其余部分主要是数学符号的展示，更多地是为了确保你使用的符号与本书一致，而不是教授你新的数学概念。

第 4 章进一步探讨了第 2 章中介绍的分治策略。它提供了两个额外的示例，展示了用于相乘方阵的分治算法，其中包括令人惊讶的 Strassen 算法。第 4 章介绍了解决递归关系的方法，这对描述递归算法的运行时间很有用。在代入法中，你猜测一个答案并证明它的正确性。递归树提供了一种生成猜测的方法。第 4 章还介绍了强大的“主方法”技术，你通常可以使用它来解决由分而治之算法引起的递归关系。虽然该章节提供了主定理所依赖的基础定理的证明，但你可以自由地使用主方法，而不必深入研究证明。第 4 章以一些高级主题结束。

第 5 章介绍了概率分析和随机算法。通常，你使用概率分析来确定算法的运行时间，在这种情况下，由于固有的概率分布的存在，相同大小的不同输入可能具有不同的运行时间。在某些情况下，你可能会假设输入符合已知的概率分布，以便对所有可能的输入求平均运行时间。在其他情况下，概率分布不是来自输入，而是来自算法执行过程中所做的随机选择。一个算法的行为不仅由其输入决定，还由随机数生成器产生的值决定，这就是随机算法。你可以使用随机算法来对输入强制施加概率分布，从而确保没有特定的输入会导致性能下降，甚至用于限制允许产生错误结果的算法的错误率。

附录 A-D 包含其他数学材料，在阅读本书时对你会有帮助。你可能在阅读本书之前已经看过附录章节中的大部分内容（尽管我们使用的特定定义和符号约定在某些情况下可能与你之前看到的有所不同），所以你应该将附录视为参考资料。另一方面，你可能尚未看到第一部分的大部分材料。第一部分的所有章节和附录都以教程风格编写。

第二章 算法在计算中的作用

什么是算法？为什么研究算法是值得的？相对于计算机中使用的其他技术，算法的作用是什么？本章将回答这些问题。

2.1 算法

非正式地说，算法是一种明确定义的计算过程，它以某些值或一组值作为输入，并在有限时间内产生某些值或一组值作为输出。因此，算法是一系列计算步骤，将输入转化为输出。

你也可以将算法视为解决明确定义的计算问题的工具。问题的陈述以一般性的方式指定了问题实例的所需输入/输出关系，通常是任意大的问题实例。算法描述了一种特定的计算过程，以实现所有问题实例的输入/输出关系。

举个例子，假设你需要将一组数字按照单调递增的顺序进行排序。这个问题在实践中经常出现，并为引入许多标准的设计技术和分析工具提供了丰富的素材。以下是我们如何正式定义排序问题：

输入：n 个数的一个序列 $\langle a_1, a_2, ..., a_n \rangle$。

输出：输入序列的一个排列（排序之后的）$\langle a'_1, a'_2, ..., a'_n \rangle$，满足 $\langle a'_1 \leq a'_2 \leq \cdots \leq a'_n \rangle$。

因此，给定输入序列 $\langle 31, 41, 59, 26, 41, 58 \rangle$，一个正确的排序算法将输出序列 $\langle 26, 31, 41, 41, 58, 59 \rangle$。这样的输入序列被称为排序问题的一个实例。一般来说，问题的一个实例包括计算问题解所需的输入（满足问题陈述中规定的约束条件）。

由于许多程序将其作为中间步骤使用，排序是计算机科学中的一项基本操作。因此，你可以选择使用许多优秀的排序算法。哪种算法对于给定的应用程序最佳取决于多个因素，包括待排序项的数量，项的部分排序程度，对项值可能存在的限制，计算机的体系结构以及要使用的存储设备类型：主内存、磁盘，甚至是过时的磁带。

对于一个计算问题的算法，如果对于每个作为输入提供的问题实例，它能在有限的时间内停止计算，并输出问题实例的正确解，那么这个算法是正确的。一个正确的算法解决了给定的计算问题。而一个不正确的算法可能在某些输入实例上根本无法停止计算，或者在停止时给出错误的答案。与你可能期望的相反，如果能够控制错误率，不正确的算法有时也可能是有用的。当我们学习用于查找大素数的算法时，我们将在第 31 章中看到一个具有可控错误率的算法的例子。然而，通常情况下，我们只关注正确的算法。

算法可以用英语、计算机程序甚至硬件设计来进行规定。唯一的要求是规定必须提供对应计算过程的精确描述。

算法解决哪种问题

排序远非唯一一个已经开发出算法的计算问题。（当你看到本书的厚度时，你可能已经怀疑到这一点。）算法的实际应用无处不在，包括以下示例：

- 人类基因组计划在实现以下目标方面取得了巨大进展：鉴定人类 DNA 中大约 3 万个基因，确定构成人类 DNA 的大约 30 亿个化学碱基对的序列，将这些信息存储在数据库中，并开发用于数据分析的工具。每个步骤都需要复杂的算法。虽然这些问题的解决方案超出了本书的范围，但许多解决这些生物学问题的方法使用了本书中介绍的思想，使科学家能够在有效利用资源的同时完成任务。动态规划（如第 14 章所述）是解决其中几个生物学问题的重要技术，特别是涉及确定 DNA 序列之间相似性的问题。这样做可以节省时间（包括人力和机器时间）和金钱，因为实验技术可以提取更多的信息。
- 互联网使全球人民能够快速访问和检索大量信息。借助巧妙的算法，互联网上的网站能够管理和操作这些大量的数据。一些必须使用算法的问题的示例包括找到数据传输的良好路径（解决此类问题的技术出现在第 22 章），以及使用搜索引擎快速找到包含特定信息的网页（相关技术在第 11 章和第 32 章）。
- 电子商务使得商品和服务能够在电子环境下进行协商和交换，并且它依赖于个人信息的隐私，例如信用卡号码、密码和银行对账单。电子商务中使用的核心技术包括公钥加密和数字签名（在第 31 章中介绍），这些技术基于数值算法和数论。

- 制造业和其他商业企业经常需要以最有利的方式分配稀缺资源。石油公司可能希望知道在哪里安置油井以最大化预期利润。政治候选人可能希望确定在哪里花钱购买竞选广告，以最大化赢得选举的机会。航空公司可能希望以最低成本的方式为航班分配机组，确保每个航班得到覆盖，并满足政府对机组排班的规定。互联网服务提供商可能希望确定在哪里投放额外资源，以更有效地为其客户提供服务。所有这些都是可以通过将它们建模为线性规划问题来解决的例子，这是第 29 章讨论的内容。

尽管这些例子的一些细节超出了本书的范围，但我们确实介绍了适用于这些问题和问题领域的基本技术。我们还展示了如何解决许多具体问题，包括以下问题：

- 你手上有一张道路地图，上面标记了相邻交叉口之间的距离，你希望确定从一个交叉口到另一个交叉口的最短路径。即使不允许路径相交，可能的路径数量也可能非常大。你如何选择所有可能路径中最短的一条呢？你可以首先将道路地图（它本身就是实际道路的模型）建模为一个图（我们将在第六部分和附录 B 中介绍）。在这个图中，你希望找到从一个顶点到另一个顶点的最短路径。第 22 章展示了如何高效解决这个问题。
- 给定一个以零件库形式表示的机械设计，其中每个零件可能包含其他零件的实例，按顺序列出零件，使得每个零件都出现在使用它的任何零件之前。如果设计包含 n 个零件，则存在 $n!$ 种可能的顺序，其中 $n!$ 表示阶乘函数。由于阶乘函数的增长速度甚至超过指数函数，你不可能可行地生成每个可能的顺序，然后验证在该顺序中，每个零件都出现在使用它的零件之前（除非只有几个零件）。这个问题是拓扑排序的一个实例，第 20 章展示了如何高效解决这个问题。
- 医生需要确定一张图像是否代表了一个恶性肿瘤或良性肿瘤。医生有很多其他肿瘤的图像可用，其中一些已知是恶性的，一些已知是良性的。恶性肿瘤很可能与其他恶性肿瘤更相似，而良性肿瘤更可能与其他良性肿瘤相似。通过使用聚类算法，如第 33 章所示，医生可以确定哪种结果更有可能。
- 你需要对一个包含文本的大文件进行压缩，以便占用更少的空间。有许多已知的方法可以实现这一目的，包括 “LZW 压缩算法”，它寻找重复的字符序列。第 15 章研究了一种不同的方法，即 “Huffman 编码”，它通过不同长度的位序列对字符进行编码，其中出现频率更高的字符使用较短的位序列进行编码。

这些列表远非详尽无遗（你可能从本书的厚度中推测出来了），但它们展示了许多有趣的算法问题所共有的两个特点：

1. 它们有许多潜在的解决方案，其中绝大多数并不能解决手头的问题。在不显式地检查每个可能的解决方案的情况下，找到一个能够解决问题或是一个 “最佳” 解决方案，可能会带来相当大的挑战。
2. 它们具有实际应用。在上述问题列表中，寻找最短路径提供了最简单的例子。运输公司，如货车或铁路公司，有着在道路或铁路网络中找到最短路径的经济利益，因为选择更短的路径可以降低劳动力和燃料成本。或者，互联网上的路由节点可能需要找到网络中的最短路径，以便快速路由一条消息。或者，一个想要从纽约开车到波士顿的人可能希望使用导航应用程序找到驾驶方向。

并非每个由算法解决的问题都有一个容易确定的候选解集。例如，给定一组表示定期时间间隔取样的信号样本的数值，离散傅里叶变换将时间域转换为频率域。也就是说，它将信号近似为正弦波的加权和，产生不同频率的强度，这些频率的加和近似于取样信号。离散傅里叶变换除了是信号处理的核心之外，还在数据压缩和大多项式和整数乘法中具有应用。第 30 章介绍了这个问题的高效算法，即快速傅里叶变换（通常称为 “FFT”）。该章还概述了一个硬件 FFT 电路的设计。

数据结构

这本书还介绍了几种数据结构。数据结构是一种存储和组织数据的方式，以便于访问和修改。选择适当的数据结构是算法设计的重要组成部分。没有一种单一的数据结构适用于所有目的，因此你应该了解其中几种数据结构的优势和限制。

技术

虽然您可以将本书作为算法的 “菜谱” 使用，但你可能会遇到一些问题，对于这些问题，你无法很容易地找到已发布的算法（例如，本书中的许多练习和问题）。本书将教你算法设计和分析的技巧，以便你能够独立开发算法，验证其正确性，并分析其效率。不同的章节涉及算法问题解决的不同方面。一些章节解决特定的问题，例如在第 9 章中找到中位数和顺序统计量，在第 21 章中计算最小生成树，在第 24 章中确定网络中的最大流。其

他章节介绍了一些技术，例如在第 2 章和第 4 章中的分治法、第 14 章中的动态规划以及第 16 章中的摊还分析。

难题

本书的大部分内容都是关于高效算法的。我们通常衡量算法的效率是通过速度：一个算法产生结果需要多长时间？然而，有一些问题我们并没有找到在合理时间内运行的算法。第 34 章研究了这些问题中的一个有趣子集，被称为 NP 完全问题。

为什么 NP 完全问题很有趣？首先，尽管我们从未找到过 NP 完全问题的高效算法，但也没有人能够证明不存在高效算法。换句话说，没有人知道是否存在适用于 NP 完全问题的高效算法。其次，NP 完全问题集合具有一个显著的特性，即如果其中任何一个问题存在高效算法，那么所有问题都存在高效算法。这种关系使得缺乏高效解决方案更加引人入胜。第三，几个 NP 完全问题与我们已知存在高效算法的问题相似但并不相同。计算机科学家对于问题陈述的微小变化如何导致已知最佳算法的效率发生巨大变化感到着迷。

你应该了解 NP 完全问题，因为它们在实际应用中出现得相当频繁。如果你需要为一个 NP 完全问题设计一个高效算法，你可能会花费很多时间进行无果的搜索。相反，如果你能证明该问题是 NP 完全问题，你可以把时间花在开发高效的近似算法上，也就是一种能够给出较好但不一定是最优解的算法。

举个具体的例子，考虑一个带有中央配送中心的送货公司。每天，公司会在中心配送中心装货，并将货物送到多个地址。在一天结束时，每辆卡车必须回到中心配送中心，以便准备下一天的装货。为了降低成本，公司希望选择一种送货顺序，使得每辆卡车行驶的总距离最小。这个问题就是著名的 “旅行推销员问题”，它是一个 NP 完全问题。目前没有已知的高效算法。然而，在一定的假设条件下，我们知道有一些高效算法可以计算出接近最小总距离的解。第 35 章讨论了这种” 近似算法”。

替代计算模型

多年来，我们可以指望处理器的时钟速度以稳定的速度增加。然而，物理限制对不断增长的时钟速度构成了根本性障碍：因为功率密度与时钟速度超线性增加，一旦时钟速度足够高，芯片就有熔化的风险。因此，为了每秒执行更多计算，芯片被设计成不仅包含一个处理” 核心”，而是几个处理核心。我们可以将这些多核计算机类比为在单个芯片上的几个顺序计算机。换句话说，它们是一种” 并行计算机”。为了从多核计算机中获得最佳性能，我们需要设计考虑并行性的算法。第 26 章介绍了一种” 任务并行” 算法模型，它利用了多个处理核心。这个模型在理论和实践的角度都有优势，许多现代并行编程平台也采用了类似于这种并行模型的方法。

本书中的大部分示例假设在算法开始运行时所有输入数据都是可用的。算法设计的大部分工作也是基于这个假设进行的。然而，在许多重要的实际示例中，输入数据实际上是随时间到达的，而算法必须在不知道未来将到达的数据的情况下决定如何进行。在数据中心，作业不断到达和离开，调度算法必须决定何时何地运行作业，而不知道未来将会到达哪些作业。在互联网中，必须根据当前状态路由流量，而不知道未来流量将到达的位置。医院急诊室必须根据患者的病情进行分诊决策，而不知道未来其他患者何时到达以及他们需要哪些治疗。接收输入数据的算法并非一开始就具有所有输入，而是随着时间推移，这些算法被称为在线算法，第 27 章对其进行了研究。

2.2 作为一种技术的算法

第三章 算法基础

3.1 插入排序

3.2 分析算法

3.3 设计算法

第四章 如何刻画算法的运行时间？

4.1 O记号，Ω记号和Θ记号

4.2 渐进记号：形式化定义

4.3 标准记号和常用函数

第五章 分治策略

第六章 概率分析与随机算法

6.1 雇佣问题

6.2 指示器随机变量

6.3 随机算法

6.4 概率分析和指示器随机变量的进一步使用

第二部分

排序和顺序统计量

第七章 简介

第八章 堆排序

第九章 快速排序

第十章 线性时间排序

第十一章 中位数和顺序统计量

第三部分

数据结构

第十二章 简介

第十三章 基本数据结构

第十四章 哈希表

第十五章 二叉搜索树

第十六章 红黑树

第四部分

高级设计与分析技术

第十七章 简介

第五部分

高级数据结构

第十八章 简介

第六部分

图算法

第十九章 简介

第七部分

算法问题选编

第二十章 简介

第八部分

附录：数学基础知识