第 1 章　算法在计算中的作用

什么是算法？为什么要对算法进行研究？相对于计算机中使用的其他技术来说，算法的作用是什么？在本章中，我们就要来回答这些问题。

1.1　算法

简单来说，所谓**算法**(algorithm)就是定义良好的计算过程，它取一个或一组值作为**输入**，并产生出一个或一组值作为**输出**。亦即，算法就是一系列的计算步骤，用来将输入数据转换成输出结果。

我们还可以将算法看作是一种工具，用来解决一个具有良好规格说明的计算问题。有关该问题的表述可以用通用的语言，来规定所需的输入/输出关系。与之对应的算法则描述了一个特定的计算过程，用于实现这一输入/输出关系。

例如，假设需要将一列数按非降顺序进行排序。在实践中，这一问题经常出现。它为我们引入许多标准的算法设计技术和分析工具提供了丰富的问题场景。下面是有关该排序问题的形式化定义：

输入：由 n 个数构成的一个序列$\langle a_1, a_2, \cdots, a_n\rangle$。

输出：对输入序列的一个排列(重排)$\langle a'_1, a'_2, \cdots, a'_n\rangle$，使得 $a'_1 \leqslant a'_2 \leqslant \cdots \leqslant a'_n$。

例如，给定一个输入序列$\langle 31, 41, 59, 26, 41, 58\rangle$，一个排序算法返回的输出序列是$\langle 26, 31, 41, 41, 58, 59\rangle$。这样的一个输入序列称为该排序问题的一个实例(instance)。一般来说，某一个问题的实例包含了求解该问题所需的输入(它满足有关该问题的表述中所给出的任何限制)。

在计算机科学中，排序是一种基本的操作(很多程序都将它用作一种中间步骤)，因此，迄今为止，科研人员提出了多种非常好的排序算法。对于一项特定的应用来说，如何选择最佳的排序算法要考虑多方面的因素，其中最主要的是考虑待排序的数据项数、这些数据项已排好序的程度、对数据项取值的可能限制、打算采用的存储设备的类型(内存、磁盘、磁带)等。

如果一个算法对其每一个输入实例，都能输出正确的结果并停止，则称它是正确的。我们说一个正确的算法解决了给定的计算问题。不正确的算法对于某些输入来说，可能根本不会停止，或者停止时给出的不是预期的结果。然而，与人们对不正确算法的看法相反，如果这些算法的错误率可以得到控制的话，它们有时也是有用的。关于这一点，在第 31 章中研究用于寻找大质数的算法时介绍了一个例子。但是，一般而言，我们还是仅关注正确的算法。

算法可以用英语、以计算机程序或甚至是硬件设计等形式来表达。不论采用哪种形式，唯一的要求就是算法的规格说明必须提供关于待执行的计算过程的精确描述。

算法可以解决哪些类型的问题？

研究人员并不仅仅是针对排序这一计算问题设计了大量的算法(读者在看到本书的厚度时可能也会这么猜想的)。算法的实际应用面很广，例如：

- 人类基因项目的目标是找出人类 DNA 中的所有 100 000 种基因，确定构成人类 DNA 的 30 亿种化学基对的各种序列，将这些信息存储在数据库中，并开发出用于进行这方面数据分析的工具。这些步骤中的每一个都需要复杂的算法。该项目所涉及的各个问题的解

决方案已超出了本书的范围，但本书中有好几章中的思想在解决这些生物问题时都用到了，这样就使得科学家们可以有效地利用已有资源来完成任务，并且，当利用实验室技术可以提取出更多的信息时，就可以带来人、财、物、时间等方面的节约。

- 因特网使得全世界的人们都能够快速地访问和检索大量的信息。为了能实现这一目的，人们采用了巧妙的算法来管理和操纵大量的数据。这方面必须解决的问题包括寻找好的数据传输路径(第 24 章将介绍解决这些问题的技术)、利用搜索引擎来快速地找到包含特定信息的网页等(有关技术将在第 11 章和第 32 章中介绍)。
- 电子商务使得商品和服务可以以电子的形式进行谈判和交易。然而，电子商务要想得到广泛应用的话，非常重要的一点就是保持信用卡号、密码、银行结单等信息的私密性。公共密钥加密技术和数字签名技术(将在第 31 章中介绍)是这一领域内所使用的核心技术，它们的基础就是数值算法和数论理论。
- 在制造业和其他商业应用中，是否能有效地分配稀有资源常常是非常重要的。例如，石油公司可能希望确定该在何处打井，以求最大化预期效益。美国总统候选人可能希望确定该把竞选宣传的资金花在何处，以使赢得竞选胜利的可能性最大。航空公司可能希望以尽可能小的代价来将机组人员分配到不同的航班上，以便做到既到考虑到每一个航班，又不会违反政府有关航空人员调度的规定。因特网服务提供商可能希望确定该把额外的资源置于何处，以便能够更有效地服务其客户。所有这些都是可以利用线性规划求解问题的例子，这一技术将在第 29 章中介绍。

尽管这些例子中某些细节已经超出了本书的范围，我们仍给出了适用于这些问题和问题领域的底层支撑技术。此外，在本书中，我们还说明了如何解决许多具体的问题，例如：

- 给定一幅道路交通图，上面标注出了每一对相邻交叉路口之间的距离。我们的目标就是确定一个交叉路口到另一个交叉路口之间的最短路线。即使不允许每一条路线自我交叉，可能的路线数量也会是巨大的。在所有可能的路线中，该如何来选出最短的路线呢？这里，用一个图来对道路交通图进行建模(前者本身就是对实际道路的一种建模；有关图的内容将在第 10 章和附录 B 中介绍)，希望在图中找出一个顶点到另一个顶点之间的最短路径。在第 24 章中，将看到如何来有效地解决这一问题。
- 给定由 n 个矩阵所组成的一个序列$\langle A_1, A_2, \cdots, A_n\rangle$，希望确定其乘积 $A_1A_2\cdots A_n$。因为矩阵乘法是可以结合的，因而存在着若干合法的乘法顺序。例如，如果 $n=4$，可以按照以下几种顺序来执行矩阵乘法：$(A_1(A_2(A_3A_4)))$，$(A_1((A_2A_3)A_4))$，$((A_1A_2)(A_3A_4))$，$((A_1(A_2A_3))A_4)$，$(((A_1A_2)A_3)A_4)$。如果这些矩阵都是正方矩阵(因而其大小都是一样的)，乘法的顺序对矩阵乘法将花多少时间是没有影响的。然而，如果这些矩阵的大小不同的话(但其大小对矩阵乘法来说是相容的)，那么，乘法的顺序如何就会带来很大的差别了。可能的乘法结合顺序的数量是 n 的指数级的，因此，要尝试所有可能顺序的话，可能会花很长的时间。在第 15 章中，我们将会看到，如何用一种称为动态规划的技术来更为有效地解决这一问题。
- 给定一个方程 $ax\equiv b \pmod n$，其中 a、b 和 n 都是整数，希望找出所有(在模 n 时)满足该方程的整数 x。方程的解可能有零个、一个或多个。可以简单地尝试依次用 $x=0$，1，…，$n-1$ 来代入该方程，但第 31 章中给出了一种更为有效的方法。
- 给定平面上 n 个点，希望找出这些点的凸壳，即包含这些点的最小凸多边形。从直观上看，可以将每一个点看成是由一块板上突起的一个钉子表示的。因而，包围这些点的凸壳可以看成是一根包围了所有这些钉子的绷紧的橡皮绳。每一个令橡皮绳发生方向变化

的钉子都是该凸壳的一个顶点(33.3 节的图 33-6 给出了一个例子)。这些点的 2^n 个子集中的任何一个都可能是该凸壳的顶点。知道哪些点是该凸壳的顶点还不够，还需要知道它们出现的顺序。因此，该凸壳的顶点构成有多种选择。第 33 章给出了两种用于寻找凸壳的好方法。

上面这些例子远远没有穷尽所有可能的情况(单从本书的重量就能看出这一点了)，但也体现了许多有趣算法的两个共同特征：

1)有很多候选的解决方案，其中大部分都不是我们所需要的。找到真正需要的解决方案往往不是一件容易的事。

2)有着实际的应用。在上面列出的问题中，最短路径问题提供了最简单的例子。运输公司(如汽车货运或铁路公司)对于如何在公路或铁路网中找出最短路径，有着经济方面的利益，因为选取更短的路线可以降低劳动力和燃料成本。还有，在因特网上，一个路由结点也需要在网络中寻找最短路径，以便快速地路由消息。

数据结构

本书还包含了几种数据结构。数据结构是存储和组织数据的一种方式，以便于对数据进行访问和修改。没有哪一种数据结构可以适用于所有的用途和目的，因而，了解几种数据结构的长处和局限性是相当重要的。

技术

尽管可以将本书当作一本有关算法的“菜谱”(cookbook)来使用，但是，仍然可能在将来的某一天，碰到一个问题后，一时不太容易找到一个已有的算法来解决它(例如，本书的练习和思考题中有很多就是这样的情况)。本书将教给读者一些算法设计和分析的技术，以便读者自行设计算法、证明其正确性和理解其效率。

一些比较难的问题

本书的大部分都是关于一些比较高效的算法的。衡量算法效率的常用标准是速度，即一个算法得到最后结果所需要的时间。然而，有一些问题至今还没有已知的有效解法。第 34 章研究了这些问题的一个有趣的子集，即 NP 完全问题。

为什么说 NP 完全问题有趣呢？首先，尽管迄今为止都没有谁能找出 NP 完全问题的有效解法，但也没有人能够证明 NP 完全问题的有效解法是不存在的。换句话说，NP 完全问题是否存在有效算法是未知的。其次，NP 完全问题集有一个显著的特点，即如果该集合中的任何一个问题存在有效的算法，则该集合中的其他所有问题都存在有效算法。NP 完全问题之间的这种关系，使得缺乏有效的算法变得更为令人着急。第三，有几个 NP 完全问题类似于(但又不完全同于)一些有着已知有效算法的问题。对一个问题陈述的一点小小的改动，就会对其已知最佳算法的效率带来很大的变化。

对 NP 完全问题有所了解是很有价值的，因为有些 NP 完全问题会时不时地在实际应用中冒出来。例如，如果要求你找出有关某一 NP 完全问题的有效算法，你很可能会花费大量时间去探寻，结果却是徒劳无益的。如果你能证明该问题是 NP 完全的，就可以把时间花在设计一个有效的算法上，该算法可以给出比较好的、但不一定是最佳可能的结果。

我们来看一个具体的例子。假设有一个货车运输公司，它有一个中央仓库。每一天，它都要在仓库中将货物装满货车，并让它驶往若干个地方去送货。在一天结束时，这辆货车必须最后回到仓库，以便下一天再装货物。为了降低成本，该公司希望选择一条送货车行驶距离最短的送货顺序。这一问题即著名的“旅行商人问题”，并且是个 NP 完全问题。对于该问题，尚没有已知的

有效算法。但是，在特定的假设下，该问题存在着有效的算法，它们能够给出比最短可能的距离
9 长不了多少的总体距离。第 35 章讨论了这种“近似算法”。

练习

1.1-1 给出一个真实世界的例子，其中包含着下列的某种计算问题：排序，确定多矩阵相乘的最佳顺序，或者找出凸壳。

1.1-2 除了运行速度以外，在真实世界问题背景中，还可以使用哪些效率指标？

1.1-3 选择你原来见过的某种数据结构，讨论一下其长处和局限性。

1.1-4 上文中给出的最短路径问题和旅行商人问题有哪些相似之处？有哪些不同之处？

1.1-5 举出一个现实世界的问题例子，它只能用最佳解决方案来解决。再举出另一个例子，在其中“近似”最优解决也足以解决问题。

1.2 作为一种技术的算法

假设计算机无限快，并且计算机存储器是免费的，那么你还有任何理由来研究算法吗？如果你没有别的理由，就是仍然希望证明你的解决方案能够终止并给出正确的结果的话，那么答案就是“是的”。

如果计算机无限快，那么对于某一个问题来说，任何一个可以解决它的正确方法都是可以的。你很可能希望自己的实现能够符合良好的软件工程实践要求(亦即，具有良好的设计和文档说明)，但往往会采用最容易实现的方法。

当然，计算机可以做得很快，但还不能是无限快。存储器可以做到很便宜，但不会是免费的。因此，计算时间是一种有限的资源，存储空间也是一种有限的资源。这些有限的资源必须被
10 有效地使用，那些时间和空间上有效的算法就有助于做到这一点。

效率

解决同一问题的各种不同算法的效率常常相差很大。这种效率上差距的影响往往比硬件和软件方面的差距还要来得大。

例如，在第 2 章中，我们将介绍两个排序算法。第一个称为插入排序算法，对 n 个数据项进行排序的时间大约等于 $c_1 n^2$，其中 c_1 是一个不依赖于 n 的常量。亦即，该算法所需的时间大致正比于 n^2。第二个是合并排序算法，它排序 n 个数据项所需的时间大约是 $c_2 n \lg n$，其中 $\lg n$ 表示 $\log_2 n$，c_2 是另一个同样也不依赖于 n 的常量。插入排序算法与合并排序算法相比，通常有着更小的常量因子，即 $c_1 < c_2$。后面我们还将看到，与对输入规模 n 的依赖相比，常量因子对运行时间的影响要小得多。合并排序算法的运行时间中有个因子 $\lg n$，而插入排序算法的运行时间中有个因子 n，它要比前者大得多了。尽管对于较小的输入规模来说，插入排序要比合并排序更快些，但是，一旦输入规模 n 变得足够大了以后，合并排序的 $\lg n$ 与 n 相比的优势就远远不止是弥补两者之间常量因子上的差距了。无论 c_1 比 c_2 小多少，总会有一个转折点，过了这个点以后，合并排序就会比插入排序运行得更快了。

我们来看一个具体的例子。让一台更快的、运行插入排序的计算机(计算机 A)与一台较慢的、运行合并排序的计算机(计算机 B)进行比较。两者都要对一个大小为一百万个数的数组进行排序。假设计算机 A 每秒能执行 10 亿条指令，而计算机 B 每秒只能执行一千万条指令。因此，在计算能力方面，计算机 A 要比计算机 B 快 100 倍。为了使这一差距更为明显，假设让世界上最能干的程序员采用机器语言，来为计算机 A 编写插入排序算法的代码，所得到的代码需要 $2n^2$

条指令来排序 n 个数(此处，$c_1=2$)。另一方面，让一位平均熟练水平的程序员，采用某种具有低效编译器的高级语言来为计算机 B 编写合并排序算法的代码，所得到的代码有 $50n\lg n$ 条指令(这里 $c_2=50$)。为了排序一百万个数，计算机 A 花的时间为：

$$\frac{2\cdot(10^6)^2 \text{ 条指令}}{10^9 \text{ 条指令/秒}} = 2000 \text{ 秒}$$

计算机 B 花的时间为：

$$\frac{50\cdot 10^6 \lg 10^6 \text{ 条指令}}{10^7 \text{ 条指令/秒}} \approx 100 \text{ 秒}$$

计算机 B 由于采用了一个运行时间增长得更为缓慢的算法，尽管它用的是效率较低的编译器，运行速度也比计算机 A 快了 20 倍！当对一千万个数据进行排序时，合并排序的优势就更为明显了：这时，插入排序要花大约 2.3 天的时间，合并排序只需不到 20 分钟的时间。一般来说，随着问题规模的增长，合并排序的相对优势也会愈加明显。

11

算法和其他技术

上面给出的例子说明了算法就像计算机硬件一样，是一种技术。总体的系统性能不仅依赖于选择快速的硬件，还依赖于选择有效的算法。算法领域的研究和其他计算机技术领域一样，正不断地取得飞速的进展。

读者可能会想，与其他先进的计算机技术(如以下列出的)相比，算法对于当代的计算机是否真的那么重要？

- 具有高时钟主频、流水线技术、超级标量结构的硬件
- 易于使用的、直观的图形用户界面(GUI)
- 面向对象的系统
- 局域网和广域网技术

答案是“是的”。尽管对于有些应用来说，在应用这一层面上没有什么特别明显的算法方面的要求(例如，一些简单的 Web 应用就是这样的)，但大多数问题对算法还是有一定程度要求的。例如，假设有一种基于 Web 的服务，它用于确定如何从一个地方旅行至另一个地方。(在写作本书时，已经有了好几种这样的服务了。)其实现将依赖于快速的计算机硬件、图形用户界面、广域网技术，甚至还可能要依赖于面向对象技术。然而，除此之外，它还需要为某些操作设计算法，如寻找路由(很可能采用最短路径算法)、显示地图、插入地址等。

此外，即使是那些在应用层面上对算法性内容没什么要求的应用，其实也是相当依赖于算法的。该应用要依赖于快速的计算机硬件吧？硬件的设计就要用到算法。该应用要用到图形用户界面吧？任何 GUI 的设计也要依赖于算法。该应用要依赖于网络技术吧？网络路由对算法也有着很大的依赖。该应用是采用某种非机器代码的语言编写而成的吧？那么，它就要由编译器、解释器或汇编器来处理，所有这些软件都要大量用到各种算法。算法是当代计算机中用到的大部分技术的核心。

随着计算机性能的不断增长，可以利用计算机来解决比以往更大的问题。从上文有关插入排序与合并排序的比较中可以看出，正是对于更大的问题规模，不同算法在效率方面的差异才会变得特别显著。

是否拥有扎实的算法知识和技术基础，是区分真正熟练的程序员与新手的一项重要特征。利用当代的计算技术，无需了解很多算法方面的东西，也可以完成一些任务。但是，有了良好的算法基础和背景的话，可以做的事就要多得多了。

12

练习

1.2-1 给出一个实际应用的例子，它在应用这一层次上要求有算法性的内容。讨论其中所涉及算法的功能。

1.2-2 假设我们要比较在同一台计算机上插入排序和合并排序的实现。对于规模为 n 的输入，插入排序要运行 $8n^2$ 步，而合并排序要运行 $64n\lg n$ 步。当 n 取怎样的值时，插入排序的性能要优于合并排序？

1.2-3 对于一个运行时间为 $100n^2$ 的算法，要使其在同一台机器上，比一个运行时间为 2^n 的算法运行得更快，n 的最小取值是多少？

思考题

1-1 算法运行时间的比较

对于下表中的每一个函数 $f(n)$和时间 t，求出可以在时间 t 内被求解出来的问题的最大规模 n。假设解决该问题的算法解决该问题需要 $f(n)$微秒。

	1 秒	1 分钟	1 小时	1 天	1 个月	1 年	1 个世纪
$\lg n$							
$\sqrt{n}$							
n							
$n\lg n$							
n^2							
n^3							
2^n							
$n!$							

13

本章注记

有许多全面介绍算法这一主题的书都是非常不错的，如 Aho、Hopcroft 和 Ullman[5，6]，Baase 和 Van Gelder[26]，Brassard 和 Bratley[46，47]，Goodrich 和 Tamassia[128]，Horowitz，Sahni 和 Rajasekaran[158]，Kingston[179]，Knuth[182，183，185]，Kozen[193]，Manber[210]，Mehlhorn[217，218，219]，Purdom 和 Brown[252]，Reingold，Nievergelt 和 Deo[257]，Sedgewick[269]，Skiena[280]，以及 Wilf[315]等著作。Bentley[39，40]和 Gonnet[126]对算法设计中一些更具体实际的问题进行了讨论。对算法这一领域的综述可以参见《Handbook of Theoretical Computer Science，Volume A》[302]，以及《CRC Handbook on Algorithms and Theory of Computation》[24]。Gusfield[136]、Pevzner[240]、Setubal 和 Medanis[272]、Waterman[309]等教材则对计算生物学中用到的各种算法做了概述性的介绍。

14