HZ BOOKS
华章教育

高等学校计算机专业规划教材

算法设计与分析

骆吉洲　编著

Design and Analysis of Algorithms

机械工业出版社
China Machine Press

高等学校计算机专业规划教材

算法设计与分析

骆吉洲　编著

Design and Analysis of Algorithms

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

算法设计与分析 / 骆吉洲编著 . —北京：机械工业出版社，2014.10
（高等学校计算机专业规划教材）

ISBN 978-7-111-48316-8

I. 算… II. 骆… III. ① 电子计算机 – 算法设计 – 高等学校 – 教材 ② 电子计算机 – 算法分析 – 高等学校 – 教材 IV. TP301.6

中国版本图书馆 CIP 数据核字（2014）第 241863 号

本书以算法设计与分析的理论、方法和技术为主线，系统地介绍分治算法、动态规划算法、贪心算法、最小值最大值方法、搜索策略、随机算法、近似算法和在线算法等算法设计技术，以及循环不变量方法、反例方法、平摊分析方法、概率分析方法、近似度分析方法和竞争度分析方法等算法分析技术。在介绍这些理论、方法和技术的同时，还介绍计算几何、图论、元素选取、最大流、顶点覆盖和匹配等领域的一些基本算法。

本书可以作为计算机相关专业本科生、研究生教材，也可供从事算法设计与分析工作的工程技术人员参考。

出版发行：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：朱秀英　　责任校对：殷　虹
印　　刷：北京瑞德印刷有限公司　　版　　次：2014 年 11 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：16.25
书　　号：ISBN 978-7-111-48316-8　　定　　价：49.00 元

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

前　言

算法设计与分析是计算机科学永恒不变的主题，是计算机专业的核心课程之一。这门课程旨在培养学生以算法设计和分析为手段，时间高效、空间高效地解决问题的意识和能力，以及简洁、清晰、准确表达算法及其分析过程的技能，提高学生的算法修养，培养创新意识，培育研究能力。通过这门课程的学习，学生可以理解算法和算法复杂度的概念，掌握评判算法优劣的标准，较系统地掌握分治算法、动态规划算法、贪心算法、搜索策略、随机算法、近似算法和在线算法等算法设计技术，掌握循环不变量方法、反例方法、平摊分析方法、概率分析方法、近似度分析方法和竞争度分析方法等算法分析技术，为进一步深造打下良好的算法基础。

为实现上述培养目标，作者结合自己多年从事算法设计与分析课程教学和研究的经验，精心地选取和组织了这门课程的教学内容。全书以算法设计与分析方法为主线，分为11章，着重强调对算法设计与分析基本技术的掌握，舍弃了在数据结构、集合论与图论等先修课程中学生已经掌握的基本算法，包括绝大部分排序算法、字符串匹配算法和基本图论算法。第1章是绪论，建立算法的概念并阐述算法设计和分析的目的和意义。第2章是数学基础，总结全书算法分析过程中使用的主要数学工具和数学结论，便于在后续章节中着重于算法设计和分析的过程，尽量避免复杂的数学推导过程。第3～5章介绍算法设计与分析的基本技术，包括分治算法、动态规划算法和贪心算法。第6章介绍平摊分析技术及其应用。第7章介绍最大值最小值方法，讨论网络流和匹配等图论问题的算法。第8～10章介绍难解问题的三种处理方法，包括树的搜索策略、随机算法、近似算法。第11章介绍在线算法和竞争度分析。

对每一种算法设计与分析技术，均先通过简单的例子介绍技术的原理和要点，再通过一系列的例子阐述使用该技术时需要注意的各个侧面。例如，第3章的一系列算法设计实例着重强调利用分治算法求解问题时划分阶段、递归求解阶段和合并阶段需要注意的问题。第4章的一系列算法设计实例除阐明动态规划的要点之外，还展示了各种典型的优化子结构，等等。

本书还特别强调算法设计过程依赖于对问题的分析和对问题特征的把握，即算法设计通常从求解问题的蛮力算法开始，逐步观察和分析问题的特征并结合恰当的算法设计技术，逐渐设计出高效的算法。每章均包含了这样的算法设计实例，以引导读者建立从蛮力法入手，逐步分析问题特征，再利用问题特征设计算法的思维过程，让读者理解问题特征分析、基本算法和算法设计与分析之间的关系，体会从无到有、精益求精的算法设计过程，培养创新意识和研究能力。

本书还配备了大量的习题，以期学生在实践过程中加深对算法设计与分析方法的理解并提高算法设计与分析的速度。部分习题来自参考文献，部分习题来自科研实践，还有部分习题来自多年教学实践的总结。作者希望学生在完成习题的过程中，首先能够确切地理解计算问题。部分习题专门为这一目的设置，只要理解了计算问题，适当修改已有的算法即可解决。此外，作者希望学生能够仔细、完备地阐明求解习题得到的算法以及对算法的分析。写出算法及其分析过程会让被忽略的难点或问题彻底暴露，这个过程还可以排除那些不相干的想法。算法设计和分析过程往往需要多次重复书写才能得到一个清晰、简洁的结果。比较好的做法是：获得算法及其分析之后将它放在一旁，提交之前再看看它是否仍然是完备的、令人信服的和易于理解的。书写算法并完成对算法的分析可以培养学生简洁、清晰、准确地表达自己的能力，而这恰恰是一项非常有用的技能。

在过去的几十年中，专家和学者们编著了许多与算法设计与分析相关的教材。这些著作给本书的编写提供了参考和借鉴，在这里向相关作者表示感谢。同时，也感谢哈尔滨工业大学多年来提供的教学和研究机会，并感谢学生、朋友和家人的鞭策及支持。

由于作者水平有限，书中不当之处在所难免，敬请读者批评指正。如果读者有任何建议或意见，可以发送电子邮件至·luojizhou@hit. edu. cn。

作者

2014 年 6 月

教学建议

教学章节	教学内容与学时安排		
	教学内容	40学时课程	60学时课程
第1章 绪论	算法在计算机科学体系中的地位	2	2
	算法的概念		
	算法分析		
	算法设计方法		
第2章 数学基础	复杂度函数的阶	2	2
	标准符号和通用函数		
	递归方程		
第3章 分治算法	分治算法原理	2	2
	大整数乘法		
	Strassen矩阵乘法		
	快速傅里叶变换	2	2
	最邻近点问题		
	平面点集的凸包	2	2
	剪枝搜索方法		
	线性时间选择算法		
	二元线性规划的线性时间算法	2	2
	1-圆心问题的线性时间算法		
第4章 动态规划算法	动态规划原理	2	2
	最长公共子序列		
	矩阵链乘法	2	2
	0-1背包问题		
	最优二叉搜索树		1
第5章 贪心算法	贪心算法的基本原理	2	2
	活动选择问题		
	哈夫曼编码问题	2	2
	最小生成树问题		
	贪心算法的理论基础	2	2
	单位时间任务调度问题		
第6章 平摊分析	平摊分析方法	2	2
	动态表性能的平摊分析	2	2
	斐波那契堆及其操作代价的平摊分析		
	并查集及其操作代价的平摊分析		1

（续）

教学章节	教学内容与学时安排		
	教学内容	40学时课程	60学时课程
第7章 最大值最小值方法	最大流问题和最小割问题	2	2
	Ford-Fulkerson算法		
	Edmonds-Karp算法		
	推送复标算法	2	2
	复标前置算法		
	匹配与覆盖	2	2
	最大二分匹配		
	一般图上的最大匹配		1
	最大权值二分匹配	2	2
	稳定二分匹配		
第8章 树的搜索策略	问题解空间的树表示	2	2
	典型搜索策略		
	分支限界法的应用	2	2
	A^*算法及其应用	2	2
	博弈树和α-β剪枝		
第9章 随机算法	随机算法概述		2
	数值随机算法		
	随机选择和拉斯维加斯算法		
	快速排序和舍伍德算法		2
	素数测试和蒙特卡罗算法		
	最小割随机算法		
第10章 近似算法	近似算法的性能分析		2
	基于组合优化的近似算法		
	基于贪心思想的近似算法		2
	基于局部搜索的近似算法		
	基于动态规划的近似算法		2
	基于线性规划的近似算法		
	不可近似性		2
第11章 在线算法	在线算法与竞争度分析		2
	欧几里得最小生成树问题的在线算法		
	凸包在线算法		2
	线性链表在线更新算法		
	最短并行调度在线算法		1

目　录

CHAPTER 1

第1章 绪 论

1.1 算法在计算机科学体系中的地位

1.1.1 计算机理论模型和计算问题的分类

什么叫计算？什么叫有效的计算？什么样的问题存在有效的计算过程？围绕这些问题，在20世纪上半叶(特别是20世纪30年代)，许多数学家和逻辑学家展开了大量的研究和探索，并在“计算”这个概念上达成了共识。

定义 1-1 一个**计算**是能够在某种计算装置上机械地执行的一个操作序列。

例如，在计算机上，机器指令构成的一个序列是一个计算；在棋盘上，游戏者依据游戏规则交替地移动棋子也是一个计算……

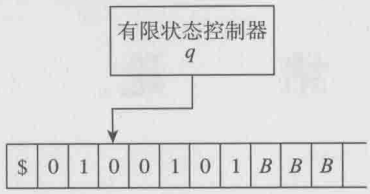
计算的概念表明，计算必须在特定的计算装置上完成。因此，人们提出了通用计算装置的许多数学模型，包括库尔特·哥德尔(Kurt Gödel)的递归函数模型、艾隆佐·契尔奇(Alonzo Church)的λ演算模型、厄米尔·勒昂·泊斯特(Emil Leon Post)的Post机模型和阿兰·图灵(Alan Turing)的图灵机模型等。人们还证明了Turing-Church命题：所有这些计算模型的计算能力是相同的，即能够在其中一个模型上执行的计算，也能够在其他计算模型上计算。

在这些计算模型中，图灵机由于其简洁性和直观性，被广泛地用来介绍计算模型。确定型图灵机(简称图灵机)由一个有限状态控制器、一个读写头、一个单向无限的存储带和有限条计算规则构成。有限状态控制器能够记住有限种状态，这类似于人的大脑能够记住若干种计算状态。读写头能够读写存储带上的符号，这类似于人的手和笔能够读写纸上的符号。存储带由存储单元构成，每个单元能够记录一个符号，每个单元内的符号可以被擦除和写入，这类似于演草纸。每一条形如 $\delta(q,a)=(q',b,R/L)$ 的计算规则规定了在当前状态 q 下从存储带上读到当前符号 a 时应该转入状态 q'，同时在存储带的当前位置写下符号 b，读写头向右(或向左)移动一个单元。

图1-1给出了图灵机的一个示例。该图灵机实现了二进制计数器加1操作，即从计数器低位向高位扫描找到首个等于0的二进制位并将它赋值为1，并将扫描过程遇到的等于1的全部二进制位赋值为0。在图1-1a给出的图灵机中有三种状态，q_0 表示从当前位置找到计数器的最低位；q_1 表示从低位向高位扫描，将遇到的1置0；q_t 表示找到了第1个等于0的位，并将它置1。这三种状态下的操作规则如图1-1b所示。给定计数器的初值和读写头的初始位置，根据操作规则进行一系列的操作，该图灵机可以实现二进制计数器的加1操作。例如，在图1-1a所示的计数器格局下，依次执行规则2、规则2、规则3、规则2、规则3、规则4、规则6和规则7构成的操作序列即可实现计数器加1。

如果允许图灵机在状态 q 下读到符号 a 时，图灵机的状态同时转移到状态子集 Q，即

操作规则形如 $\delta(q,a)=(Q,b,R/L)$，则称之为非确定型图灵机。除非特别说明，图灵机均指确定型图灵机。



a) 图灵机模型示例

1. $\delta(q_0,\$)=(q_0,\$,R)$　2. $\delta(q_0,0)=(q_0,0,R)$
3. $\delta(q_0,1)=(q_0,1,R)$　4. $\delta(q_0,B)=(q_1,B,L)$
5. $\delta(q_1,B)=(q_1,B,L)$　6. $\delta(q_1,1)=(q_1,0,L)$
7. $\delta(q_1,0)=(q_t,1,L)$　8. $\delta(q_1,\$)=(q_t,\$,R)$

b)图灵机操作规则示例

图 1-1　图灵机示例

计算问题的输入数据表示在图灵机的存储带上，所占用存储单元的数量 n 称为问题的**输入规模**。图灵机求解计算问题时，计算过程执行的计算规则的个数表示成输入规模 n 的函数 $f(n)$，称为该图灵机的**时间复杂度**；计算过程使用的不同存储单元的个数表示成输入规模 n 的函数 $g(n)$，称为该图灵机的**空间复杂度**。

存在多项式时间复杂度的确定型图灵机的计算问题构成集合 P，存在多项式时间复杂度的非确定型图灵机的计算问题构成集合 NP。由于确定型图灵机是非确定型图灵机的特例，因此 $P\subseteq NP$。关于 $P=?NP$，现在还未解决。然而，NP 问题类中有一个特殊子类被称为 NP- 完全问题类，其中的每个问题称为 NP- 完全问题(亦称为 NPC- 问题)。如果一个 NPC- 问题存在多项式时间复杂度的确定型图灵机，则 $P=NP$。因此，NPC- 问题通常被认为是难解问题。

在 $NP\neq P$ 的假设下，目前人们公认 P 问题类、NP 问题类和 NPC 问题类之间的关系如图 1-2所示。

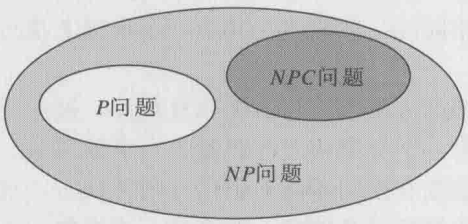


图 1-2　P 问题类、NP 问题类和 NPC 问题类之间的关系

本书第 3～11 章讨论一系列高效求解各类计算问题的方法，其中既包括高效处理 P- 问题的 4 类典型方法，也包括处理 NPC- 问题的 3 类典型方法，以及分析这些处理方法的若干种技术。但后续章节不再更多地讨论计算模型，也不讨论如何证明一个问题是 NPC- 问题。当遇到 NPC- 问题时，将加以说明，读者仅需将它们理解为难以处理的计算问题即可。

1.1.2　利用计算机求解问题

图灵机模型理论中一个重要的理论结果是**通用图灵机**的存在性命题。即存在一个通用图灵机 M，它能够模拟任何图灵机执行计算。如果将图灵机看做程序，则通用图灵机的存在意味着存在模拟所有程序运行的理论机器。计算机是通用图灵机的一种物理实现。

于是，用计算机求解问题 P，就是在图灵机模型下求解问题 P，即为问题 P 设计相应

的图灵机。将所得图灵机用高级程序设计语言表达出来就是图灵机的具体表示；高级程序设计语言通过编译得到机器代码，即为能够被计算机模拟运行的图灵机编码。

在设计图灵机的过程中，通常需要进行如下三个步骤。

1)确定计算过程需要的符号集，这决定了求解该问题的图灵机的存储带上每个存储单元表示的信息，这些信息需要经过编码并利用恰当的数据结构进行表示之后，才能实现在计算机上的存储。例如，在前面的二进制计数器中，图灵机的每个存储单元存储计数器的1位，在计算机上用1位即可实现；在排序整数时，图灵机的每个存储单元存储一个整数，在计算机上需要用1个或多个字节才能实现存储；在讨论图论问题时，图灵机每个存储单元存储一个顶点或一条边，在计算机上同样需要用1个或多个字节才能实现存储。

2)确定图灵机的状态集，这要求给出求解问题的过程中可能出现的各种状态，通常本步骤同第3步一同实现。

3)确定图灵机的有穷计算规则，这要求给出各种状态下遇到各种数据时应该采取的操作步骤，以及操作步骤完成后的状态转移。显然，这里的一个操作步骤可能需要用多条计算机机器指令才能实现。因此，计算的时间复杂度并不直接反映该计算在具体计算机上的执行时间。

多个图灵机可以完成相同的计算任务，它们选用的符号集、状态集、操作规则可以不同，进而其复杂度也可能各不相同。图灵机的时间(和空间)复杂度可以用来比较图灵机的计算效率。给定完成同一计算任务的两个图灵机 T_1 和 T_2，其时间复杂度函数分别记为 $f(n)$和$g(n)$，T_1 和 T_2 的每个计算规则被计算机模拟的时间开销分别为 a 和 b，于是 T_1 和 T_2 在计算机上被模拟执行的开销分别为 $a \cdot f(n)$和 $b \cdot g(n)$。由于 a 和 b 是常数，直接比较 $f(n)$和 $g(n)$即可比较 T_1 和 T_2 在计算机上被模拟执行的时间效率。对于空间复杂度，也有类似的结论。

因此，利用计算机求解计算问题的过程即设计高效率的图灵机、编码图灵机、模拟执行图灵机的过程。

1.1.3　计算机科学的知识体系

围绕计算机求解问题的过程，计算机科学知识体系如图1-3所示。

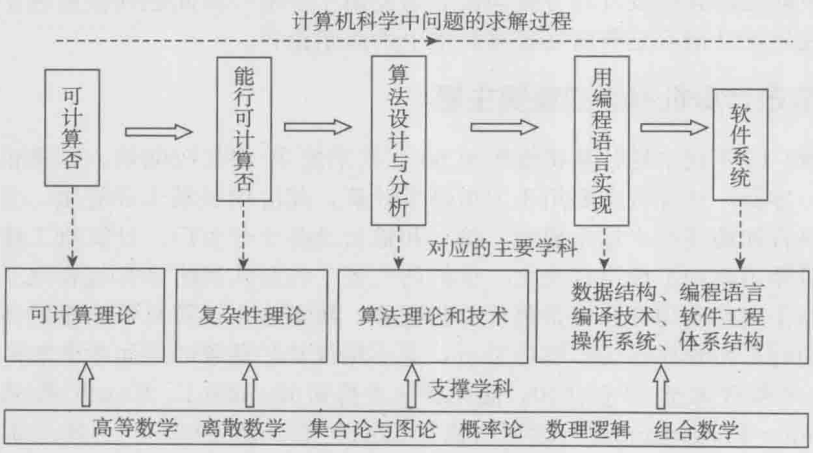


图1-3　计算机科学知识体系

利用计算机求解计算问题 A，首先要判定问题 A 能否被计算机求解，即是否存在求解问题 A 的总能停机的图灵机，将这样一个图灵机称为求解问题 A 的一个**算法**。在不能停机的图灵机上，计算是无效的，因为无法判定该图灵机永远不会停止还是在输入数据上继续执行更多的操作后才能停止。存在算法的计算问题称为可计算问题。遗憾的是，几乎所有的问题均是不可计算的，因为每个图灵机均可以被编码成二进制形式(机器代码)，进而可以看成一个自然数。故所有可计算问题构成的集合可以看成自然数集，它是可数的。不可计算问题的个数是不可数的。例如，下面的问题是不可计算的：1)判断无理数 π 的无限不循环小数表示形式中是否存在连续 7 个 1；2)判定 x_1，…，x_n 的任意多项式是否存在整数根；3)判定是否存在一个计算机程序识别所有计算机病毒，等等。研究计算模型、计算模型的等价性、问题的可判定性和可解性的科学称为可计算理论。

如果计算问题 A 是可计算的，则仍需判定它是否是能行可计算的，即判定是否存在时间复杂度 $f(n)$ 是 n 的多项式的算法。求解问题 A 的复杂度最低的算法的时间复杂度，称为问题 A 的复杂度。如果问题 A 的复杂度是问题输入规模 n 的多项式，则称 A 是多项式时间可计算的(或可有效计算的或能行可计算的)。研究问题复杂度的学科称为计算复杂性理论。计算复杂度理论的研究内容包括特定问题的固有复杂度、最好复杂度、最坏复杂度，计算问题的复杂度分类及 $P=?NP$。

如果计算问题 A 是能行可计算的，则接下来需要针对该问题进行算法设计和分析，即设计求解问题 A 的复杂度尽量低的算法。这需要恰当地表示计算问题的输入和求解过程的各种数据和状态，给出计算过程应当采取的计算规则，并分析计算过程中所用资源的多少。研究算法设计和分析的理论、方法和技术的学科即算法设计与分析，其研究内容包括算法设计的理论、方法、技术和算法分析的理论、方法和技术。

给定求解问题 A 的算法，用高级程序设计语言，结合计算机的硬件体系结构和操作系统，将算法中的数据用恰当的数据结构进行表示并用计算机程序实现算法的各个操作规则，再利用恰当的编译器将计算机程序编译成计算机机器代码就得到了相应的计算机软件。

在求解计算问题的各个环节及相应的各个学科中，高等数学、离散数学、集合论与图论、概率论、数理逻辑、组合数学等知识，以及计算思维等，起着支撑作用。

可见，算法设计与分析在计算机科学知识体系中处于核心地位：问题的可计算性、问题的复杂度及问题的算法设计与分析均需要研究用于求解计算问题的信息表示和有限计算规则；高级程序设计语言是算法在较高层次上的表示语言。

1.1.4 算法是计算机科学的重要主题

自 20 世纪 40 年代中期计算机诞生至 60 年代末的 20 多年时间里，计算机科学的主题并未明确。一方面，计算机主要用于大型科学计算，其应用领域十分有限。另一方面，高级程序设计语言和编译技术远未成型、输入和输出设备十分落后。计算机工作者的工作语言还主要以机器语言和汇编语言为主，他们的主要工作是从事将计算机程序手工或半自动地翻译成机器语言并利用穿孔纸带将计算机程序、数据输入计算机等繁重的体力劳动。少有人对编写的计算机程序进行严格的分析，更不用说从较抽象的层面考虑如何求解问题。

20 世纪 60 年代末至 70 年代初，唐纳德·克鲁斯(Donald E. Knuth)教授出版了《The Art of Computer Programming》(以下简称 TAOCP)第一卷(1967 年)、第二卷(1968 年)和第三卷(1971 年)。该著作明确了算法的概念，归纳和总结了之前计算机程序设计的基本技术，对算法或者计算机程序进行严格的数学分析，并建立了相应的数学方法。TAOCP

为严谨地分析算法奠定了基础，由此确立了算法设计与分析在计算机科学体系的主题地位。由于 TAOCP 对计算机科学的重要影响，唐纳德·克鲁斯于 1974 年获得图灵奖，成为图灵奖(ACM Turing Award)最年轻的获奖者。

70 年代末，随着计算机硬件性能的快速提升、标准输入和输出设备的出现、操作系统和数据库系统的出现，以及高级程序设计语言的完善和现代编译技术的日益成熟，人们可以从更高、更抽象的层面考虑如何用计算机求解问题，即设计和分析求解问题的计算机算法；算法设计与分析的技术也日趋成熟，由此推动了计算机科学的飞速发展。

1.1.5　算法设计与分析的意义

首先，算法是计算机的灵魂。如前所述，计算机本身就是通用算法(即通用图灵机)的物理实现，它能够模拟所有算法的执行过程，因此计算机不能独立于算法而存在。计算问题的求解过程即算法设计、分析和实现的过程。它是计算机软件工程的基础。

其次，算法是计算机各领域的核心。随着计算机的飞速发展，各领域的计算机工作者与计算机硬件的距离越来越远。他们通常不再直接与计算机硬件打交道，而是从更加抽象、更富逻辑的角度出发解决计算问题。目前，计算机各个领域的研究问题均主要是围绕本领域内的计算问题开展算法设计与分析。

最后，在网络时代，由于计算机计算能力的增长和信息量的增长速度不匹配，越来越多的挑战需要依靠卓越的算法来解决。虽然计算机在摩尔定律的作用下，其计算能力每年都在飞速增长且价格不断下降，但其计算能力仍无法满足人们的计算需求。事实上，日益先进的记录和存储手段使每个人的信息量都在爆炸式的增长，现在每人每天都会创造出大量数据(照片、视频、语音、文本等)。互联网的信息流量和日志容量也在飞快增长。在科学研究方面，随着研究手段的进步，数据量更是达到了前所未有的程度，很多实验每秒均产生 TB 级的数据量。处理这些海量数据，单靠计算机硬件性能的提升和高超的编程技艺是远远不够的，必须借助算法设计与分析作为手段，为求解计算问题选用高效的算法。

算法还可以改变人类的生活。例如，基因研究可能因新算法而产生新的医疗方式。在安全领域，有效的算法可能避免下一个 9·11 的发生。在气象方面，算法可以改善天气预报以拯救生命。

1.2　算法的概念

简单地说，算法是有效的计算。有效性有三层含义。其一是算法必须是有目的的计算，即算法能够根据输入给出相应的输出，无目的的计算是无效计算。其二是算法必须在有限计算步骤内停止，不停止的计算是无效的，因为无法判断计算不停止的原因是由于它无法求解给定的问题还是由于截至当前的计算时间不足以求解给定的问题。其三是算法执行的每个计算步骤必须是在输入数据和之前的计算步骤得到的计算结果的基础上采取的明确的可机械完成的计算步骤。

定义 1-2　**算法**是一个满足下列条件的计算：

1)有穷性/终止性：有限步内必须停止；

2)确定性：每一步都是严格定义和确定的动作；

3)能行性：每一个动作都能够被精确地机械执行；

4)输入：有一个满足给定约束条件的输入；

5)输出：给出满足给定约束条件的计算结果。

算法用来求解问题。在计算机科学中，问题应该如何刻画呢？一个计算问题要求对给定的输入计算得到满足一定约束条件的输出结果。因此，计算问题的本质是刻画输入和输出之间的关系。

定义 1-3 **问题** A 是定义在输入集 Input 和输出集 Output 上的二元关系，$A \subseteq \text{Input} \times \text{Output}$。任意二元组$(x, y) \in A$ 称为问题 A 的一个**实例**，$x \in \text{Input}$ 称为该实例的**输入**，$y \in \text{Output}$ 称为该实例的**输出**。

例 1-1 整数排序(SORT)问题要求对任意输入整数序列$<a_1, a_2, \cdots, a_n>$进行排序，并按照从小到大的次序依次输出该序列中每个元素得到序列$<b_1, b_2, \cdots, b_n>$。SORT 问题可以形式化定义如下，其中 A_{Mset} 表示集合 A 是多重集(即元素在集合 A 中可以多次出现)。(<1,3,2>, <1,2,3>)是 SORT 问题的一个实例，<1,3,2>是这个实例的输入，而<1,2,3>是该实例的输出。 ◀

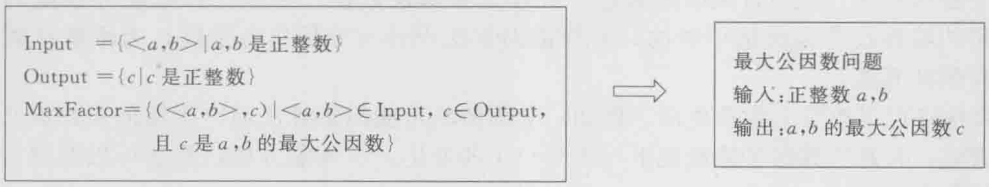
排序问题

$$\text{Input} = \{<a_1, a_2, \cdots, a_n> \mid a_i \text{ 是整数}, 1 \leqslant i \leqslant n\}$$

$$\text{Output} = \{<b_1, b_2, \cdots, b_n> \mid b_i \text{ 是整数}, 1 \leqslant i \leqslant n, \text{且 } b_1 \leqslant b_2 \leqslant \cdots \leqslant b_n\}$$

$$\text{SORT} = \{(<a_1, a_2, \cdots, a_n>, <b_1, b_2, \cdots, b_n>) \mid <a_1, a_2, \cdots, a_n> \in \text{Input},$$
$$<b_1, b_2, \cdots, b_n> \in \text{Output 且} \{a_1, a_2, \cdots, a_n\}_{\text{Mset}} = \{b_1, b_2, \cdots, b_n\}_{\text{Mset}}$$

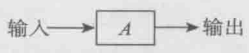
问题的形式化定义精确地描述了问题的输入和输出，以及二者之间的约束关系。然而，形式化定义繁琐且不直观。在算法设计和分析中，约定用接近自然语言的方式来描述计算问题，而不引起任何歧义。例如，最大公因数问题 MaxFactor 要求对任意输入的两个正整数 a 和 b，求出 a 和 b 的最大公因数 c；其形式化定义如图 1-4a 所示，其直观描述如图 1-4b 所示。本书约定对问题的描述采用如图 1-4b 所示的直观描述。



a) 形式化定义　　b)直观定义

图 1-4　最大公因数问题的定义

可见，问题定义了输入和输出之间的约束关系。求解问题 P 的算法 A 要求对问题 P 的任意输入 x，产生相应的输出结果 y，使得$<x, y> \in P$。算法用于求解问题，而不仅仅求解计算问题的一个或几个实例，即算法可以看成如右侧的黑箱结构。



描述算法需要指明算法求解的问题，明确地给出算法的各个操作步骤。虽然算法最终运行于计算机上，但算法的计算步骤不必表示成计算机的机器指令。事实上，算法中的各个步骤只需能够用计算机高级程序设计语言精确地实现即可。高级程序设计语言及其编译器能够将实现操作步骤的程序代码翻译成可以在计算机上精确地、机械地执行的机器代码。算法的各个计算步骤的精细化粒度取决于以下两个因素。其一是计算步骤的精确含义能否被算法实现者(或其他读者)准确地理解和实现；其二是计算步骤完成的基本操作的含义是否清晰。

下面给出了一个算法示例。这是求解最大公因数问题的欧几里得算法，它是目前已

知的人类最早设计的算法。算法以两个正整数 a、b 为输入，并输出 a 和 b 的最大公因数。该算法的理论基础是代数基本定理，即用一个非零整数除另一个整数必然产生商和余数。据此，算法第 1 步将 a、b 中较大者 $\max(a,b)$ 赋值给变量 m，将较小者 $\min(a,b)$ 赋值给变量 n。然后，算法进入 While 循环(第 2～5 步)。每次 While 循环，首先求得 m 除以 n 的余数 r；由代数基本定理易知，m、n 的最大公因数即 n、r 的最大公因数。故算法在第 4 步和第 5 步分别将 n 和 r 赋值给变量 m、n；然后，算法进入下一轮循环，直到 n 为 0。

欧几里得算法 EuclidComFactor

输入：正整数 a,b

输出：a 和 b 的最大公因数 c

1. $m\leftarrow\max(a,b)$，$n\leftarrow\min(a,b)$；　　/* a,b 中较大者赋值给 m，较小者赋值给 n */
2. While　$n>0$　Do
3. 　　$r\leftarrow m \bmod n$；　　/* $m \bmod n$ 表示 m 除以 n 的余数 */
4. 　　$m\leftarrow n$；
5. 　　$n\leftarrow r$；
6. 输出 m；

本书约定书写算法时，首先给出算法的名称，然后描述算法求解的计算问题，再依次列出算法的操作步骤，每个步骤前用阿拉伯数字编号。书写操作步骤时，用“←”表示赋值操作，“=”表示逻辑相等关系。首字母大写的关键字 While、For、Repeat、If、Else、Then、Do 等用于控制算法流程，其含义同高级程序设计语言中相应关键字的含义。“/*”和“*/”之间的文字是对操作步骤的注释，在必要的地方用于对操作步骤进行进一步的解释。操作步骤前的空格用于对齐位于同一层次的语句。例如，在欧几里得算法中，第 3、4、5 步对齐，表明它们均是 While 循环中的语句；而第 6 步与第 2 步对齐，则表明 While 循环结束后执行第 6 步的操作。

1.3　算法分析

算法设计完成之后，需要对算法进行分析，包括算法正确性分析和算法复杂度分析。

1.3.1　算法正确性分析

算法的正确性分析旨在证明算法在问题的所有输入上能够产生正确的输出。不正确的算法是无意义的。

定义 1-4（算法正确性）　求解问题 $P\subseteq\text{Input}\times\text{Output}$ 的算法 A 是**正确的**，如果算法 A 在 $\forall x\in\text{Input}$ 上的计算都最终停止并给出输出 $A(x)$ 使得 $(x,A(x))\in P$。

可见，正确算法在每一个输入均最终停止，并产生正确的输出。反之，不正确算法可能在某些输入实例上不停止或者在某些输入实例上产生不正确的输出。严格地讲，算法在被实现之前必须对其正确性给出严格的证明；**程序调试不能代替算法的正确性证明**，因为程序调试只能证明算法有错误，而不能证明算法没有错误。

证明算法的终止性的一般方法是给出算法的终止条件，并证明算法经过有限步骤必然能达到这一条件。