现代信息科学技术基础

算法设计与分析

耿国华　主编

高等教育出版社
HIGHER EDUCATION PRESS

现代信息科学技术基础

算法设计与分析

SUANFA SHEJI YU FENXI

耿国华　主编

高等教育出版社·北京
HIGHER EDUCATION PRESS BEIJING

图书在版编目(CIP)数据

算法设计与分析/耿国华主编. -- 北京：高等教育出版社，2012.1
ISBN 978-7-04-033445-6

Ⅰ.①算… Ⅱ.①耿… Ⅲ.①电子计算机-算法设计 ②电子计算机-算法分析 Ⅳ.①TP301.6

中国版本图书馆 CIP 数据核字(2011)第 229309 号

策划编辑	陈红英	责任编辑	陈红英	封面设计	李卫青	版式设计	余 杨
插图绘制	尹 莉	责任校对	姜国萍	责任印制	田 甜		

出版发行	高等教育出版社	咨询电话	400-810-0598
社 址	北京市西城区德外大街 4 号	网 址	http://www.hep.edu.cn
邮政编码	100120		http://www.hep.com.cn
印 刷	北京东君印刷有限公司	网上订购	http://www.landraco.com
开 本	787mm × 1092mm 1/16		http://www.landraco.com.cn
印 张	16	版 次	2012 年 1 月第 1 版
字 数	330 千字	印 次	2012 年 1 月第 1 次印刷
购书热线	010-58581118	定 价	33.80 元

物 料 号 33445-00

前　言

计算机科学是一门创造性很强的学科，其教学必须面向设计。算法理论是计算机科学的核心之一，“算法设计与分析”正是一门面向设计且处于计算机学科核心地位的研究型课程。学习者可通过对计算机算法的系统学习与研究，掌握算法设计的主要方法和经典策略，培养和提高算法复杂性分析能力，为进一步解决具体应用问题设计优秀算法、评价分析算法性能奠定坚实的基础。

本书内容分四部分，涉及主要的经典算法设计与分析技术，给出了算法解决应用问题的大量范例。

第一部分（第1章）为算法概述，介绍了算法的基本概念及算法分析的相关基础知识，包括算法分析准则、算法分析数学基础、算法复杂性分析方法。

第二部分（第2～7章）为经典算法设计与分析技术，介绍了递归与分治、动态规划、贪心算法、回溯法、分支限界法、随机算法六类算法的思想、实现方法及应用。以典型算法设计策略为知识单元，采用算法基本思想、算法描述、算法分析的模式展开，从算法设计和算法分析的理论入手，根据各类算法的基本技术原理，给出算法的分析与证明方法，并将经典算法与应用问题相结合，提供多类别应用的范例。

第三部分（第8章）为NP完全性理论，从计算本质角度讨论计算模型的意义与作用，介绍图灵机模型、随机存取模型、随机存取存储程序机模型等计算模型及其计算，给出NP完全性理论基础及NP完全问题求解的基本技术及分析方法。

第四部分（第9章）为神经网络智能算法，这些算法反映了模拟自然过程的神经网络、反向传播模型等智能算法的共同特点，引导学习者了解近年来智能算法的基本结构和思想。

本书采用目前流行的C++语言作为算法描述手段，算法描述中加上必要的注释以便于阅读与交流，书中所列算法均已上机调试通过。本书融入作者多年开发“算法设计与分析”课程的理论与实践经验，具有如下特色：

（1）强调方法，注重理解。本书取材注重贯穿算法思想与算法策略的理解，重点进行算法思路与算法性能的分析，范例突出算法步骤的实现技术过程。

（2）结构简明，内容丰富。本书参考了算法设计与分析的经典之作，内容包括基础、技术、理论和发展四个方面，涉及六大经典算法设计与分析技术的基本内容，以简洁的方

式描述核心方法,提供了用算法解决问题的范例与途径。

(3) 提供样例,示范应用。书中每类算法均编排应用问题示例,示范算法设计、分析步骤与技术实现过程;每章均附有大量的习题,有利于引导读者加深对内容的理解和应用;附录中编排适量试题,并附有参考答案,以示范解题步骤与分析证明过程。

本书由耿国华主编,冯筠副主编。具体编写分工如下:第1、2、3、8章及附录由耿国华、周明全编写,第4~7章由冯筠、刘晓宁、张德同、王克刚共同编写,第9章由王小凤编写。贺洁琼、刘倩、赵露露、张婧等研究生参加校对书稿、算法实验、课件制作、试题演算等工作,在此一并向他们表示感谢!本书附有电子教案,可从高等教育出版社学术出版网站(academic. hep. com. cn/download. shtml)下载。

由于作者水平有限,疏漏之处在所难免,敬请广大读者批评指正,以期得以改进与提高。

耿国华

2011年9月

目　录

第1章　算法概述

算法是计算机学科中最具有方法性质的核心概念，是计算机科学领域的基石之一，被誉为计算学科的灵魂。算法设计的优劣决定软件系统的性能，对算法进行研究能使我们深刻理解问题的本质及可能的求解技术。解决某个问题存在多种方法，寻求最优算法使得问题的解决更为方便和高效。所以我们不仅要为所解决的问题设计有效算法，还需对算法进行分析，以追求算法性能的最优化。本课程涉及算法设计和算法分析两个阶段，算法设计的任务是为解决某一给定问题设计有效方法，算法分析的任务是在比较解决特定问题多种方法优劣的基础上寻求最优方法。算法设计和分析正是分析问题和解决问题的结合。

本章对算法的基本概念、特性及算法设计与分析的任务等进行了概述，给出算法设计的基本过程和分析算法的准则，作为算法复杂度分析的基础，归结了算法分析中常用的数学基础知识，详细介绍了渐进时间复杂度函数阶的定义以及低阶(O)、高阶(Ω)、同阶(θ)3种渐进状态关系定义和分析方法。

1.1　算法的概念

约公元前300年，欧几里得在其著作《几何原本》第七卷中阐述了著名的欧几里得算法：给定两个正整数 m 和 n，求解其最大公约数，即求解能同时整除 m 和 n 的最大正整数。

算法步骤如下：

(1) 以 n 除以 m，并令所得余数为 r(r 必小于 n)。

(2) 若 $r=0$，算法结束，输出结果 n；否则继续步骤(3)。

(3) 将 n 置换为 m，r 置换为 n，并返回步骤(1)继续进行。

其算法可表示为例1.1的形式。

例1.1　求两个正整数 m、n 的最大公约数。

```
int gcd (int m ,int n)
        /*计算两个整数 m、n 的最大公约数
        输入:非负整数 m、n,其中 m、n 不同时为零
        输出:m、n 的最大公约数*/
```

```
{ while(n!=0)
    r=m%n;
    m=n;
    n=r;
  return m;
}
```

1.1.1 算法的定义和特性

1. 算法的定义

算法的非形式化定义:算法是规则的有限集合,是为解决特定问题而规定的一系列操作。

算法的形式化定义[1]:算法是一个四元组,即(Q,I,Ω,F),其中:

(1) Q 是一个包含子集 I 和 Ω 的集合,表示计算状态。

(2) I 表示计算的输入集合。

(3) Ω 表示计算的输出集合。

(4) F 表示计算的规则,是一个由 Q 到它自身的函数,具有自反性,即对于任何一个元素 $q \in Q$,有 $F(q)=q$。

一个算法是对于所有的输入元素 x 都在有穷步骤内终止的一个计算方法。在算法的形式化定义中,对于任何一个元素 $x \in I$,x 均满足以下性质:

$$x_0 = x, \quad x_{k+1} = F(x_k), \quad k \geqslant 0$$

该性质表示任何一个输入元素 x 均为一个计算序列,即 $x_0,x_1,x_2,\cdots,x_k$。对任何输入元素 x,该序列表示算法在第 k 步结束。

2. 算法的特性

(1) 有限性。一个算法必须保证执行有限步之后结束。例如,在欧几里得算法中,由于 m 和 n 均为正整数,在步骤(1)之后,r 必小于 n,若 r 不等于0,下一次执行步骤(1)时,n 值已经减小,而正整数的递降序列最后必然要终止。因此,无论给定 m 和 n 的原始值有多大,步骤(1)的执行都是有穷次的。

(2) 确定性(无二义)。算法的每一步骤必须有确切的定义,不能有歧义。例如,在欧几里得算法中,步骤(1)中明确规定"以 n 除 m",而不能有类似"以 n 除 m 或以 m 除 n"这类有两种可能的做法的规定。

(3) 可行性。算法原则上能精确地运行,在现有条件下是可以实现的。

(4) 输入。一个算法有0个或多个输入,以刻画运算对象的初始状态。例如,在欧几里得算法中,有两个输入——m 和 n。

(5) 输出。一个算法有一个或多个输出,以反映对输入数据加工后的结果。由于算

法需要给出解决特定问题的结果,没有输出结果的算法是毫无意义的。注意,这里所指的是广义输出,包括提供处理结果的多种形式。例如,在欧几里得算法中只有一个输出,即步骤(2)中的 n。

其中前 3 个特性较为集中地表现处理步骤,后两个特性主要涉及输入/输出接口。

算法可用图 1.1 来描述。

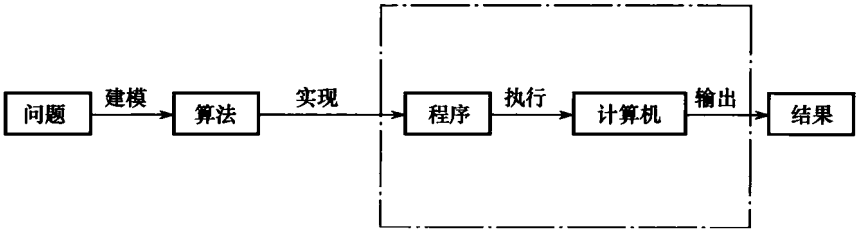

图 1.1 算法描述

3. 算法与程序的区别

算法描述了问题处理的方式或步骤,程序则是采用具体语言规则实现了算法的功能,算法要依靠程序来完成功能,算法是程序的灵魂。算法可用语言、文字、框图来描写,可用计算机、纸笔人工模拟执行。程序不一定满足有穷性,可直接在机器环境下运行。

4. 算法描述方式

算法的描述可以采用不同的方式,主要有以下 4 种:

1) 自然语言

日常生活中使用自然语言(如汉语、英语或数学语言)进行描述,通俗易懂,容易掌握,但是不严谨,容易有二义性,通常用于介绍算法的概要思路。前面关于欧几里得算法的描述使用的就是自然语言方式。

2) 框图

框图(也叫流程图)是描述算法的常用工具,是采用美国国家标准化协会规定的一组图形符号及文字说明来描述计算过程的图形。框图直观地表示算法的整个结构,着重处理流程的描述,便于检查修改,但不方便表达数据处理流程。

3) 高级语言

高级语言就是用计算机程序设计语言直接表达算法,是可以在计算机上直接运行的源程序。高级语言描述算法具有严格、准确的优点,但用于描述算法,也有语言细节过多的弱点。

4) 类语言

类语言接近于高级语言但又不是严格意义上的高级语言,是一种介于自然语言与计

算机语言之间的算法描述方式。类语言具有高级语言的一般语句结构，撇掉语言中的细节，以便把注意力主要集中在算法处理步骤本身的描述上。类语言结构性较强，比较容易书写和理解，不拘泥于具体语言的语法结构，以灵活的形式表现被描述对象，可将注意力集中在处理步骤中。例如类 PASCAL 语言、类 C 语言。

我们在学习和研究算法时往往使用类语言形式，不仅直观、方便、有利于交流，而且在设计算法时能较好地考虑算法执行时的动态性。

1.1.2 求解问题的基本过程

算法是问题的程序化解决方案。算法求解问题一般遵循以下步骤。

1. 明确问题性质并分析需求

在设计一个算法前，应该对给定的问题有完全的理解。常见的方法是仔细阅读问题的描述，提出疑问，试着手工处理一些小规模的例子，考虑一下特殊情况，等等。可以将问题简单地分为数值型问题和非数值型问题，不同类型的问题可以有针对性地采用不同的方法进行处理。

2. 建立问题的描述模型

对于数值型问题可以建立数学模型，通过数学模型来描述问题；对于非数值型问题，一般可以建立一个过程模型，通过过程模型来描述问题。

3. 选择解决方法

模型确定之后，可以针对不同的模型采用适当的处理方法。

4. 设计处理算法

对于数值型问题，可采用数值分析现成的经典算法，也可以根据问题的实际情况专门设计算法。

对于非数值型问题，可通过构建数据结构或算法分析设计进行处理，也可选择一些成熟的穷举法、分治法、回溯法等典型方法应用于处理过程。

5. 程序化

将设计好的算法用特定的程序设计语言实现，并在具体的计算机上运行。在编写程序时可能出现错误或者运行效率低下的情况，因此需要反复修改程序，以达到预期的目标。

6. 算法分析

根据评价标准研究各种算法特性的优劣，对算法的改进起到积极的作用。

1.1.3 算法设计示例——计算最大公约数

解决同一个问题有许多方法可以采用,具体方法的可行性、空间消耗、运行效率等都不相同,需要我们进行算法设计与分析。下面以计算最大公约数的 3 种算法为例说明算法设计与分析的意义。

1. 计算最大公约数的欧几里得算法

(1) 以 n 除以 m,并令所得余数为 r(r 必小于 n)。

(2) 若 $r=0$,算法结束,输出结果 n;否则继续步骤(3)。

(3) 将 n 置换为 m,r 置换为 n,并返回步骤(1)继续进行。

上述算法基于的方法是重复应用等式 $\gcd(m,n)=\gcd(n,m \bmod n)$,直到 $m \bmod n$ 等于 0。因为 $\gcd(m,0)=m$,m 的最后取值就是 m 和 n 的最大公约数。

2. 计算最大公约数的连续整数检测法

(1) 将 $\min\{m,n\}$ 赋值给 t。

(2) m 除以 t,如果余数为 0,进入步骤(3),否则进入步骤(4)。

(3) n 除以 t,如果余数为 0,则 t 为最大公约数,返回 t 的值,否则进入步骤(4)。

(4) 把 t 的值减 1,返回步骤(3)。

3. 计算最大公约数的质因数分解法

(1) 找出 m 的所有质因数。

(2) 找出 n 的所有质因数。

(3) 从步骤(1)求得的 m 的质因数分解式和步骤(2)求得 n 的质因数分解式中,找出所有相同的公因数。

(4) 将步骤(3)找到的公因数相乘,结果为所求的 $\gcd(m,n)$。

1.2 算法设计与分析任务

在算法设计阶段,主要是如何设计解决给定的问题的有效算法也就是构造问题的解,算法设计的任务是对各类具体的问题设计高质量的算法,以及研究设计算法的一般规律和方法。常用的算法设计方法主要有分治法、动态规划、贪心算法和回溯法等。算法设计是一个构造专用工具的过程,永远不会存在一种能解决所有问题的万能方法;算法设计是一个复杂的、创造性劳动,要求设计者能运用已有知识和抽象思维,逐步形成算法的基本思想,构造出算法的具体步骤,以正确解决问题。

在算法分析阶段,主要涉及分析判断某一算法质量的准则和技术,对算法进行有效性评价。对于设计出的每一个具体的算法,算法分析的主要任务是利用数学工具讨论它的

各种复杂度。复杂度分析的结果可以作为评价算法质量的标准之一，也可为改进算法提供参考。分析算法的复杂度需要较强的数学基础与技巧，针对不同的算法，需采用不同的分析方法。

算法的好坏对计算机解决问题的能力，如速度和规模，有十分重要的作用。如果我们把算法设计比喻为创作一部电视剧，算法分析则是观赏与对电视剧的评论。

算法设计与分析具有密切的联系，它们相互影响。算法需要进行检验和评价，反过来，算法评价的结果也可影响算法设计，以便改进算法的性能。

1.3 算法分析准则

算法分析是对一个算法所需计算时间和存储空间所做的定量分析。需要一定的准则和方法来分析算法的优劣，主要考虑算法的正确性、可读性、健壮性、高效率和低存储量这4个方面。

1. 正确性

算法的正确性最为重要。一个正确的算法应当对所有合法的输入数据都能得到应该得到的结果。对于那些简单的算法，可以通过调试验证其正确与否。要精心挑选那些具有“代表性的”，甚至有点“刁钻”的数据进行调试，以保证算法对“所有”的数据都是正确的。一般来说，调试并不能保证算法对所有的数据都正确，只能保证对部分数据正确，调试只能验证算法有错，不能验证算法无错。要保证算法的正确性，通常要用数学归纳法去证明。

算法的正确性是指假定给定有意义输入，算法经有限时间计算，可产生正确答案。先建立精确命题，证明给出某些输入后，算法将产生结果；然后证明这个命题。一个算法的正确性有两方面的含义：解决问题的方法选取是正确的，也就是数学上的正确性；实现这个方法的一系列指令是正确的。在算法分析中我们更看重的是前者。

正确性的4个层次[2]如下：程序不含语法错误；程序对几组输入数据能得出满足规格要求的结果；对典型的、苛刻的、带有刁难性的几组输入有正确的结果；对一切合法的输入数据都能产生满足规格要求的结果。

例1.2 求 n 个数的最大值问题。给出核心处理的示意算法如下：

```
max = 0;
for (i = 1 ;i <= n ;i ++ )
   { scanf("%f", &x);
     if (x > max) max = x;
   }
```

分析：如上求最大值的算法无语法错误。当输入的 n 个数全为正数时，结果是正确

的。如果输入的 n 个数全为负数时，求得的最大值为 0，显然这个结果不对，由这个简单的例子可以说明算法正确性的内涵。

思考：上面求最大值的算法应当为第几层次？这是否为正确的算法？

2. 可读性

算法的重要作用之一是便于阅读和交流，可读性有助于我们对算法的理解、调试和修改。

3. 健壮性

健壮性也称鲁棒性，它是指程序对于规范要求以外的输入情况的处理能力。所谓健壮的系统，是指对于规范要求以外的输入能够判断出该输入不符合规范要求，并具有合理的处理方式的系统。

4. 高效率和低存储量

评价算法效率的主要技术指标有算法运行的时间复杂度和空间复杂度两个方面。算法的效率通常是指算法的执行时间。对于一个具体问题的解决通常可以有多个算法，执行时间短的算法其效率就高。所谓的存储量需求，是指算法在执行过程中所需要的最大存储空间，这两者都与问题的规模有关。

在以上 4 种准则中，算法设计的最主要要求是算法的正确性和运行效率。

1.4 算法分析基础

要对算法进行性能分析，计算机基础知识与数学相关知识是必不可少的。本节将简要介绍与算法分析相关的数学符号、背景知识和基本方法。

1.4.1 常用数学术语

1. 计量单位

按照 IEEE 规定的表示法标准，位用“b”表示，字节用“B”表示，千字节用“KB”表示，兆字节用“MB”表示，毫秒用“ms”表示。它们之间的关系如下：1 MB 等于 2^{20} B，1 KB 等于 2^{10} KB，即 1 024 B；1 ms 等于 1/1 000 s。

2. 阶乘函数

任何大于 1 的自然数 n 阶乘的表示为 $n! = 1 \times 2 \times 3 \times \cdots \times n$，阶乘函数随着 n 的增大迅速增大。由于直接计算阶乘函数非常耗时，所以有时使用一个公式来作为近似计算式是非常有用的，即关于阶乘的斯特林公式为