WUTP

普通高等学校计算机科学与技术专业新编系列教材

Algorithm Design and Analysis

算法设计与分析

主编　刘任任

```
#include <stdio.h>
#include <stdio.h>
void main()
void main()
{
{
   void swap(int *ptr1,int *ptr2);
   void swap(int *ptr1,int *ptr2);
  int x,y, *ptr1, *ptr2;
  int x,y, *ptr1, *ptr2;
  printf("input x,y:");scanf("%d,%d",&x,
  printf("input x,y:");scanf("%d,%d",&x,
  printf("%d\t%d\n",x,y);ptr1=&x;ptr2=
  printf("%d\t%d\n",x,y);ptr1=&x;ptr2=
  if(x<y)
  if(x<y)
  swap(ptr1,ptr2);
  swap(ptr1,ptr2);
  printf("%d\t%d\n",x,y);
  printf("%d\t%d\n",x,y);
}
}
void swap(int *ptr1,int *ptr2)
void swap(int *ptr1,int *ptr2)
```

武汉理工大学出版社
Wuhan University of Technology Press

普通高等学校计算机科学与技术专业新编系列教材

Algorithm Design and Analysis

算法设计与分析

主　编　刘任任
副主编　汤　惟

武汉理工大学出版社
Wuhan University of Technology Press

内 容 提 要

本书主要介绍了:算法的基本概念及相关基本知识;常用的一些非数值算法的设计方法(分治法、贪心法、动态规划法、回溯法和分支限界法);字符串的匹配算法;NP 完全问题的近似算法;概率算法;目前常见的通用型数据压缩算法;公钥密码学基础。

本书可作为计算机科学与技术专业的本科生教材,也可供有关计算机工作者阅读。

图书在版编目(CIP)数据

算法设计与分析/刘任任主编.—武汉:武汉理工大学出版社,2003.12
普通高等学校计算机科学与技术专业(本科)新编系列教材
ISBN 7-5629-2034-6

Ⅰ.算… Ⅱ.刘… Ⅲ.①算法设计-高等学校-教材 ②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(2002)第 106861 号

出版发行:武汉理工大学出版社(武汉市武昌珞狮路 122 号 邮编:430070)
http://www.techbook.com.cn
E-mail:tiandq@mail.whut.edu.cn duanchao@mail.whut.edu.cn
经 销 者:各地新华书店
印 刷 者:安陆市鼎鑫印务有限责任公司
开 本:787×960 1/16
印 张:10.5
字 数:213 千字
版 次:2003 年 12 月第 1 版
印 次:2003 年 12 月第 1 次印刷
印 数:1—5000 册
定 价:14.00 元

普 通 高 等 学 校
计算机科学与技术专业新编系列教材
编 审 委 员 会

出版说明

当今世界已经跨入了信息时代，计算机科学与技术正在迅猛发展。尤其是以计算机为核心的信息技术正在改变整个社会的生产方式、生活方式和学习方式，推动整个人类社会进入信息化社会。为了顺应时代潮流，适应计算机专业调整及深化教学改革的要求，充分考虑到不同层次高校的教学现状，满足广大高校的教学需求，武汉理工大学出版社经过广泛调研，与国内近30所高等院校的计算机专家进行探讨，决定组织编写“普通高等学校计算机科学与技术专业新编系列教材”。

我们在组织编写新编本套系列教材时，**以培养现代化高级人才为重任，以提高学生综合素质、培养学生应用能力和创新能力为目的，以面向现代化、面向世界、面向未来为准绳，注重系列教材的特色和实用性，反映最新的教学与科研成果，体现本专业的时代特征。**同时，面对教育改革的需要、人才的需要和社会的需要，在编写本教材时，借鉴、学习国外一流大学的先进教学体系，结合国内的实际需要，吸取具有先进性、实用性和权威性的国外教材的精华，以更好地促进国内教材改革顺利进行。从时代和国际竞争要求的高度来思考，为打造一套高起点、高水平、高质量的系列教材而努力。

本套教材具有以下特色：

与时俱进，内容科学先进——充分体现计算机学科知识更新快的特点，及时更新知识，确保教材处于学科前沿，以拓宽学生知识面，培养学生的创新能力。

紧跟教学改革步伐，体现教学改革的阶段性成果——符合全国高校计算机专业教学指导委员会、中国计算机学会教育委员会制订的“计算机学科教学计划2000”的内容要求。

实现立体化出版，适应教育方式的变革——本套教材努力使用和推广现代化的教学手段，凡有条件的课程都准备组织编写、制作和出版配合教材使用的实验、习题、课件、电子教案及相应的程序设计素材库。

本套教材首批25种预定在2003年秋季全部出齐。我们的编审者、出版者决不敢稍有懈怠，一定高度重视，兢兢业业，按最高的质量标准工作。教材建设是我们共同的事业和追求，也是我们共同的责任和义务，我们诚恳地希望大家积极选用本套教材，并在使用过程中给我们多提意见和建议，以便我们不断修订、完善全套教材。

武汉理工大学出版社

2002年10月

前　言

早在20世纪70年代，计算机科学巨匠、图灵奖获得者D. E. Knuth曾指出，计算机科学就是研究算法的学问。因此，算法设计与分析是计算机科学的核心问题。计算机科学是一项创造性的思维活动，其教育必须面向设计，而计算机算法设计与分析正是面向设计的、处于核心地位的教育课程。它应当立足于基础课和专业基础课坚实的基础之上，其目的是通过对计算机领域的许多常见问题和有代表性算法的学习、研究、了解，掌握算法设计的一些主要方法，提高分析的基本技能和某些技巧，达到能独立设计算法和对给定算法进行复杂度分析的初级水平。无论从事计算机专业哪一方面的工作，这些知识都是必备的。特别是对计算机系统结构、系统软件和应用软件等专业，更是必不可少的专业知识。

从应用范围来看，算法可以分为数值算法和非数值算法两大类；从工作方式来看，算法又可以分为串行算法和并行算法两大类。因为数值算法在《数值分析》中介绍得较多，而本书作为《数据结构》的后继课程，主要讲述非数值算法。由于篇幅和时间的限制，本书暂只介绍串行算法。本书中的算法均用自然语言来表述其思路，再以类C语言来描述，力求简洁明了、通俗易懂。考虑到有关排序、图、集合的算法在《数据结构》课程中已有较详细的讲述，本书不再重复。

本书共分为11章。第1章介绍了算法的基本概念，并对分析算法的准则、描述算法的语言以及本书用到的基本数据结构作了简要的阐述。第2章至第6章分别介绍了常用的一些非数值算法的设计方法，它们分别是分治法、贪心法、动态规划法、回溯法和分支限界法。这些都是一些通用的算法，可应用于大部分问题之中，所以本书选取了一部分有代表意义的问题来进行讲解。第7章主要介绍了字符串的三种匹配算法，之所以将其单独列为一章，主要是考虑到目前计算机处理的数据类型中，字符串占有相当大的比重，它的处理比一般的数据类型更为复杂。而设计一个理想的匹配算法，需要坚实的理论基础和高超的设计技巧。第8章介绍了NP完全问题和近似算法。NP完全问题是20世纪70年代提出的理论计算机科学中的前沿课题，而近似算法则是目前针对NP完全问题行之有效的方法。第9章概率算法是一类比较特殊的算法，它相对于其他确定型的算法有其独特的优势和应用范围。第10章介绍了目前最常用的几类通用型数据压缩算法，数据压缩应用广泛，极具实用价值。第11章介绍了公钥密码学

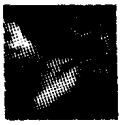
的基本理论和目前最常用的 RSA 公钥体制以及数字签名技术。密码学是一门古老而又年轻的学科，既具理论价值又具有实际意义，本章只介绍了它的研究成果，而对其理论基础不予深究。

本书的第 1、8、11 章由刘任任编写，第 2 章由韩海编写，第 3、4 章由汤惟编写，第 5、6、9、10 章由刘新编写，第 7 章由肖建华编写，全书由刘任任统稿。

由于时间仓促，作者水平有限，错误、缺点在所难免，恳请各位读者指教。

编 者

2003 年 10 月



目　录

1 引 论

1.1 什么是算法

算法和数学、计算等基本概念一样，要精确定义它并不是很容易的事。一般认为，算法是用计算装置（如电子计算机）能够理解的语言描述的解题过程。在计算机科学中，算法已逐渐成了用计算机解一类问题的精确、有效方法的代名词。非形式地说，算法就是一组有穷的规则，这些规则确定了解决某一类问题的一个运算序列，并且具有以下五个性质：

- 有穷性：一个算法必须在执行了有穷步运算之后终止。
- 确定性：一个算法中每一步运算的含义必须是确切的、无二义性的。
- 能行性：一个算法中要执行的运算都是相当基本的操作，能在有限时间内完成。
- 输入：一个算法有 0 个或多个输入，这些输入均取自某一特定的集合。
- 输出：一个算法将产生一个或多个输出，这些输出可理解为“对输入的计算结果”。

【例 1.1】 给定两个正整数 m 和 n，求 m 和 n 的最大公因数(m, n)。

这个问题通常可用所谓辗转相除法求解。此方法在 13 世纪我国的《数书九章》中就有记载，在西方此方法称为欧几里得算法。今描述如下：

步骤 1 【求余数】：用 n 除 m。令 r 是所得余数，即有 $m = qn + r$，$q \geqslant 0$，$0 \leqslant r < n$。转步骤 2。

步骤 2 【判余数】：若 $r = 0$，则输出 n，算法结束。否则转步骤 3。

步骤 3 【置换数】：执行 $m \leftarrow n$，$n \leftarrow r$，转步骤 1。

我们来看“辗转相除法”是否具备算法的五个性质。首先“辗转相除法”的三个步骤中的每个运算的含义是确切的，并且无二义性；其次，每个运算都是一些算术（除法）运算、非零判定操作和数的置换操作等相当基本的运算或操作；再次，注意到步骤 3 执行完后要返回到步骤 1，即出现了一个循环。但因为输入 m

和 n 都是给定的有限数，每次相除后所得余数 r 若不为零，则总有 $r < \min(m, n)$。因此，这就保证了经过有限次循环后，总有 $r = 0$。此时信号就会终止，并产生一个输出。总之，“辗转相除法”具备了算法所必须的五个性质。故它是一个算法。

要注意的是，一个算法并不等同于一个计算程序或者一个过程。“算法”同“过程”的区别在于：对任何合法输入，算法将在有限的时间内通过有穷步计算后终止，而过程则可以是有穷的，也可以是无穷的。

1.2 分析算法的准则

分析算法是一种有趣的智力活动，它可以充分发挥我们的聪明才智。更重要的是，从经济观点来看，分析算法可以使我们知道为完成一项任务所设计的算法的好坏，从而促使我们去设计一些更好的算法，以收到少花钱多办事、办好事的经济效益。

要分析一个算法，首先就要确定使用哪些运算以及执行这些运算所用的时间。一般这些运算可以分为两类：一类是基本运算，它包括四种基本的整数算术运算：加、减、乘、除；还可以包括浮点算术、比较、对变量赋值和过程调用等。这些运算所用时间虽然不同，但一般都只花费一个固定量的时间，因此，它们称为其时间是限界于常数的运算。另一类运算则不然，它们可能由一些更基本运算的任意长序列所组成。例如，两个字符串的比较运算可以是一系列字符比较指令，而字符比较指令又可使用移位和位比较指令。当比较一个字符的时间限界于常数时，比较两个字符串的时间总量就取决于它们的长度。

第二件要做的事是确定能反映出算法在各种情况下工作的数据集，即是要求我们编造出能产生最好、最坏和有代表性情况的数据配置，通过使用这些数据配置来运行算法，以了解算法的性能。这部分工作是算法分析最重要和最富有创造性的工作之一。有关这方面更多的叙述在以后讨论一些具体算法时再进行。

对一个算法要作出全面地分析可分成两个阶段来进行，即事前分析和事后测试。由事前分析，求出该算法的一个时间限界函数（它是一些有关参数的函数）。而由事后测试收集此算法的执行时间和实际占用空间的统计资料。

假设在程序中的某个地方有语句：$x = i$，如果要确定执行这条语句的时间总量，需要两项基本信息：该语句的频率计数（frequency count，即该语句执行的次数）和每执行一次这一语句所需要的时间。这两个数的乘积就是时间总量。

由于该语句的执行一次的时间依赖于特定的机器、编译程序和运行环境，因此事前分析只限于确定每条语句的执行次数，该次数与环境无关，而且独立于写

算法的程序设计语言，从而可以由算法直接确定。

在实际的算法分析中，往往可能分析出一个算法的计算时间或频率总数可以用某种函数来表示，比如说能用一个多项式来表示。但是由于算法本身可能相当复杂以及其它许多因素，使得我们在事前分析阶段根本就写不出这多项式的完整形式，甚至连最高次项的系数都不能写出，而只能写出该项的次数并判断出这个多项式与最高次项的关系。尽管这是件令人非常遗憾的事，但幸运的是这种关系仍反映了算法在计算时间上的基本特性，关于这一点，稍后即可看出，因此在算法的事前分析阶段，一般我们都满足于确定这种关系。下面给出这种关系的数学描述。

计算时间的渐近表示：假设某算法的计算时间是 $f(n)$，其中变量 n 可以是输入或输出量，可以是两者之和，也可以是它们之一的某种测度（例如，数组的维数，图的边数等等）。$g(n)$是在事前分析中确定的某个形式很简单的函数，例如 n^2，$\log n$[①] 等。它是独立于机器和语言的函数，而 $f(n)$则与机器和语言有关。

定义 1.1 如果存在两个正常数 c 和 n_0，对于所有的 $n \geqslant n_0$，有

$$|f(n)| \leqslant c|g(n)|$$

则记作 $f(n) = O(g(n))$。

因此，当说一个算法具有 $O(g(n))$的计算时间时，指的是如果此算法用 n 值不变的同一类数据在某台机器上运行时，所用的时间总是小于 $|g(n)|$一个常数倍。所以 $g(n)$是计算时间 $f(n)$的一个上界函数，$f(n)$的数量级就是 $g(n)$。当然，在确定 $f(n)$的数量级时总是试图求出最小的 $g(n)$，使得 $f(n) = O(g(n))$。我们也把 $f(n)$称为具有 $O(g(n))$的渐进时间复杂度。

我们已经证明了这样一个结论，若多项式：$f(n) = a_m n^m + \cdots + a_1 n + a_0$，则 $f(n) = O(n^m)$。

这一结论说明计算时间为 m 阶的多项式的算法，其时间都可以用 $O(n^m)$来表示，和其系数无关。这也就说明了数量级的改进对于算法的效率有着决定性的影响。

从计算时间上可以把算法分成两类，凡可用多项式来对其计算时间限界的，称为多项式时间的算法；而计算时间用指数函数限界的算法称为指数时间的算法。以下是六种常见的计算时间为多项式的关系：

$$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3)$$

指数时间算法一般有如下关系：

$$O(2^n) < O(n!) < O(n^n)$$

当 n 取得很大时，指数时间算法和多项式时间算法在所需的时间上非常悬

① 除非另有声明，本书所用的对数均以 2 为底。

殊。因此,只要有人能将现有指数时间的算法中任何一个简化为多项式时间的算法,都是一个伟大的成就。

符号 O 作为算法性能描述的工具,它表示计算时间的上界函数。为了进一步刻画算法的性能,有时也希望确定时间的下界函数,为此,引进另一个数学符号。

定义 1.2 如果存在两个正常数 c 和 n_0 对于所有的 $n > n_0$,有

$$|f(n)| \geqslant c|g(n)|$$

则记为 $f(n) = \Omega(g(n))$。

在某些情况下,某算法的计算时间既有 $f(n) = \Omega(g(n))$ 又有 $f(n) = O(g(n))$,即 $g(n)$ 既是 $f(n)$ 的上界又是它的下界。为简便起见,引进另一个数学符号来表示这种情况。

定义 1.3 如果存在正常数 c_1, c_2 和 n_0,对于所有的 $n > n_0$,有

$$c_1|g(n)| \leqslant |f(n)| \leqslant c_2|g(n)|$$

则记为 $f(n) = \Theta(g(n))$。

在某些情况下,算法中基本操作重复执行的次数还随问题的输入数据集不同而不同,例如,在长度为 n 的数组中顺序查找某个具体的数,有如下算法:

```
int search(int a[n], int x)
{ for (i = 0; i < n; i + + )
    if (a[i] == x ) return i;
  return  - 1;
}
```

若待查找的数 x 出现在数组的最前面,则此循环中的判断语句只需执行一次;若 x 不在此数组中,则判断语句要执行 n 次,这之间的差别是显而易见的。为了更准确地描述算法在不同情形下的效率,我们还需引入下面两个定义。

定义 1.4 算法的平均情形复杂度

设 D_n 是对于所考虑问题来说大小为 n 的输入集合,并设 I 是 D_n 的一个元素,$p(I)$ 是 I 出现的概率,$t(I)$ 是算法在输入 I 时所执行的基本运算的次数。那么算法的平均情形复杂度定义为:

$$A(n) = \sum p(I)t(I) \quad (I \in D_n)$$

定义 1.5 算法的最坏情形复杂度

$$W(n) = \mathrm{MAX}\{t(I)\} \quad (I \in D_n)$$

在上面的顺序查找中,最坏情形就是 x 不在数组中的情况,需要查找 n 次。而平均情形的计算要复杂得多。我们先要假定 x 不在数组中的概率是 1/2,而它出现在数组中任意一个位置的概率都是 $1/2n$,则套用定义 1.4 的公式,可以求

得 $A(n) = n * 3/4$。

算法的最坏情形复杂度描述的是算法在极端情况下的效率，而平均情形复杂度则更能描述算法的“一般表现”。我们已经知道，在实际应用中，平均情形复杂度的求得往往依赖于大规模的统计结果以计算概率，而有时这种统计结果难于得到；而在另外一些算法中，计算平均情形复杂度将遇到目前还无法解决的一些数学难题。所以在无法计算平均情形复杂度的情况下，我们就只能计算最坏情形复杂度，并以此作为算法效率度量的标准。

1.3 描述算法的语言和基本的数据结构

在本书后面各章节描述算法时，均先以自然语言表述算法的基本思想，然后再以类 C 语言对算法进行形式化地描述，这样既增强了算法的可读性，又不失其严谨性，避免歧义的产生。

表 1.1 是本书使用的类 C 语言的有关说明。

表 1.1 类 C 语言语法示例

<table>
<tr><td>1.预定义常量和类型</td><td colspan="2"># define TRUE 1
define FALSE 0
typedef int Status;
typedef struct Node * Tnode</td></tr>
<tr><td>2.数据类型</td><td colspan="2">int 整型；float 浮点型；char 字符型；string 字符串型；
struct 结构体型；*[简单类型] 指针类型</td></tr>
<tr><td>3.基本操作的算法</td><td colspan="2">函数类型 函数名(函数参数表){
//算法说明
语句序列；
}</td></tr>
<tr><td rowspan="5">4.赋值语句</td><td>简单赋值：</td><td>变量名 = 表达式；</td></tr>
<tr><td>串联赋值：</td><td>变量名 1 = 变量名 2 = ... = 变量名 k = 表达式；</td></tr>
<tr><td>成组赋值：</td><td>(变量名 1,...,变量名 k) = (表达式 1,...,表达式 k)；
结构体名 = 结构体名；
结构体名 = (值 1,...,值 k)；
数组名[起始下标..终止下标] = 数组名[起始下标..终止下标]</td></tr>
<tr><td>交换赋值：</td><td>变量名 < - - - > 变量名；</td></tr>
<tr><td>条件赋值：</td><td>变量名 = 条件表达式？表达式？表达式 T:表达式 F</td></tr>
</table>

续表 1.1

5.选择语句	(1)if(表达式) 语句;
	(2)if(表达式) 语句;
	else 语句;
	(3)switch(表达式){
	case 值 1:语句序列 1;break;
	……
	case 值 n:语句序列 n;break;
	default:语句序列 n + 1;break;
	}
	(4)switch{
	case 条件 1:语句序列 1;
	……
	case 条件 n:语句序列 n;break;
	default:语句序列 n + 1;break;
	}

6.循环语句	for(赋初值表达式;条件;修改表达式序列)语句;
	while(条件)语句;
	do{ 语句序列}while(条件);

7.结束语句	return [表达式]; 函数结束并带值返回 return; 函数结束语句 break; case 结束语句 exit(异常代码); 异常结束语句 error(出错信息); 异常结束并给出错误信息
8.输入和输出语句	scanf([格式串],变量 1,…,变量 n); printf([格式串],变量 1,…,变量 n); cout < < 变量 1 < < … < < 变量 n;
9.注释	/* 文字序列 */ //文字序列
10.基本函数	max(表达式 1,…,表达式 n)求 n 个参数中的最大值 min(表达式 1,…,表达式 n)求 n 个参数中的最小值 abs(表达式)求表达式的绝对值 floor(x) 或⌊x⌋ 不大于 x 的最大整数,称向下取整 ceil(x) 或⌈x⌉ 不小于 x 的最小整数,称向上取整
11.逻辑运算	&& 与运算; || 或运算; ! 取反运算
12.算术运算	+ 加; - 减; * 乘; / 除; % 模运算; + + 自加; - - 自减
13.关系运算	< 小于; > 大于; ≥大于等于; ≤小于等于; = = 等于; ≠不等于

本书还将用到几种常见的数据结构，它们分别是：

(1)线性表

一个线性表是 n 个数据元素的有限序列，在一个非空集合中，它存在惟一的一个被称为“第一个”的数据元素，存在惟一的一个被称为“最后一个”的数据元素，除第一个之外，每个数据元素均只有一个前驱，除最后一个之外，每个数据元素均只有一个后继。

(2)树

树是 n 个结点的有限集合。在一棵非空树中，有且仅有一个特定的称为根的结点；当 $n>1$ 时，其余结点可分为 $m(>0)$ 个互不相交的有限集 $T_1, T_2, \cdots, T_m$，其中每个集合本身又是一棵树，并且称为根的子树。

(3)图

一个图 G 是一个有序三元组，$G=(V, E, \Psi)$，其中：

① V 是非空顶点集合；

② E 是边集合，$E \cap V=\varnothing$；

③ Ψ 是 E 到 $\{uv \mid u, v \in V\}$ 的映射，称为关联函数，$u \in V, v \in V$。

读者如果熟练地掌握了这几种数据结构，将对学习本书有很大的帮助。有关这三种数据结构更多的知识，读者可以查阅《数据结构》一书。

本章小结

本章主要介绍了算法分析的一些基本准则。这其中最重要的概念是算法的复杂度，因为它是衡量算法效率的度量。一个算法复杂度的高低体现在所需的时间和空间两个方面。分析空间复杂度相对而言比较简单，我们更着重于分析算法的时间复杂度。这里的“分析”是指事前分析，这样一来，比较算法的效率就可以不依赖于特定的编译、运行环境，而只需考虑特定问题的输入数据、输入规模和算法本身采用的策略，这也是影响算法效率最主要的因素。

在分析算法的时间复杂度时，常用的一个概念是“渐进时间复杂度”，用符号 O 来表示，它描述了算法在问题规模 n 充分大的情况下所需时间的增长情况，而且它往往是我们比较算法优劣的一个基准。

在后续各章中我们将会逐步学到对于一些具体问题如何来设计合适的算法，以及如何分析这些算法的复杂度。

思考题与习题

1.1 比较下列函数阶的关系

(1) $f(n)=\log n$, $g(n)=\log_3 n$

(2) $f(n)=n$, $g(n)=\log n$

(3) $f(n)=n\log n$, $g(n)=\log n$

(4) $f(n)=2^n$, $g(n)=n^2$

(5) $f(n)=\log^2 n$, $g(n)=\log n^2$

1.2 如果 x 出现在线性表 L 中第 $i(i=2,3,4,\cdots,n)$ 个位置的概率恰好是其前一个位置