普通高等教育“十一五”国家级规划教材

高等学校计算机系列

计算机常用算法与程序设计教程

杨克昌　主编

人民邮电出版社
POSTS & TELECOM PRESS

普通高等教育“十一五”国家级规划教材

高等学校计算机系列

计算机常用算法与程序设计教程

杨克昌　主编

人民邮电出版社
北　京

图书在版编目（CIP）数据

计算机常用算法与程序设计教程 / 杨克昌主编. —北京：人民邮电出版社，2008.11

普通高等教育“十一五”国家级规划教材. 高等学校计算机系列

ISBN 978-7-115-17832-9

Ⅰ. 计… Ⅱ. 杨… Ⅲ. ①电子计算机—算法理论—高等学校—教材②程序设计—高等学校—教材 Ⅳ. TP301.6 TP311.1

中国版本图书馆 CIP 数据核字（2008）第 033281 号

内 容 提 要

本书遵循“内容实用，难易适当，面向设计，注重能力培养”的要求，讲述了穷举、回溯、分治、递归、递推、贪心算法与动态规划等计算机常用算法，同时简要介绍了模拟、智能优化与并行处理。本书注重常用算法的设计与应用，算法设计与程序实现的结合，以及算法的改进与程序优化，力求理论与实际相结合，算法与程序相统一。

书中所介绍的算法通常给出完整的 C 程序，并在 TC（VC++）环境下编译通过，为学习计算机常用算法与程序设计提供了范例。为便于读者练习，每章都附有习题，同时在附录中给出了习题求解的算法提示。

本书可作为高等院校计算机及相关专业“算法设计与分析”、“计算机常用算法与程序设计”课程的教材，也可供软件设计人员与计算机爱好者学习参考。

普通高等教育“十一五”国家级规划教材

高等学校计算机系列

计算机常用算法与程序设计教程

◆ 主　　编　杨克昌

责任编辑　张孟玮

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

北京华正印刷有限公司印刷

◆ 开本：787×1092　1/16

印张：17.25

字数：420 千字　　2008 年 11 月第 1 版

印数：1－3 000 册　　2008 年 11 月北京第 1 次印刷

ISBN 978-7-115-17832-9/TP

定价：28.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223

反盗版热线：(010)67171154

前　言

计算机程序设计是一种创造性思维活动，其教育必须面向应用。“计算机常用算法与程序设计”是计算机专业的核心课程，其教学目的是提高学生算法与程序设计水平，培养通过程序设计解决实际问题的能力。

通过对现有计算机专业本科生学习“算法设计与分析”课程效果的了解与调查，情况不容乐观。很多同学对所学过的算法描述与实施步骤不清楚，有些甚至对算法的基本概念与设计思想不甚了解，无法通过设计程序解决一些常见的应用问题。造成这一现象的原因是多方面的，缺少适合计算机本科层次的“算法与程序设计”教材是其中一个重要方面。作为普通高等教育“十一五”国家级规划教材，本书在内容选材与深度的把握上，在理论阐述与设计应用的结合上进行了精心设计，力图适合高校本科教学的目标与知识结构的要求。本书遵循“内容实用，难易适当，面向设计，注重能力培养”的宗旨编写，在以下几个方面对算法教材进行了改革探索。

（1）注重常用算法的设计与应用。教材在选材上避免贪多求全、贪广求深，以至出现本科阶段与研究生阶段的教学内容混杂不分的局面，只讲基本的常用算法，以及常用算法中要求本科学生掌握的基本内容，去除一些难度大、理论深、应用少的带研究性质的算法内容。

（2）注重算法设计与程序实现的紧密结合。算法与程序实际上是一个统一体，不应该也不能将它们对立或分割。教材在材料的组织上克服了以往罗列算法多、应用算法设计解决实际问题少、算法与程序设计脱节、算法理论与实际应用脱节的问题，在讲述每一种常用算法时，力求理论与实际相结合、算法与程序相统一，突出算法在解决实际问题中的应用，切实提高对常用算法的理解和掌握。针对每一种常用算法，精选典型的实际应用问题或课题，使用 C（C++）语言进行算法描述与程序设计，从问题提出、算法描述到程序实现连成一体，切实提高学生应用算法与程序设计解决实际问题的水平和举一反三的能力。

（3）注重算法改进与程序优化。教材对典型例题提供了多种不同的算法设

计，体现了算法的灵活性和适用性。算法与程序设计都不是一成不变的，算法改进与程序优化的过程是算法设计水平提高的过程。通过算法改进与程序优化，努力培养学生的优化意识与创新能力。

本书结构清晰简明，内容深入浅出。全书共分9章。

第1章算法与程序设计简介，对算法描述与算法的复杂度作了简要阐述。

第2章穷举与回溯，在简要介绍穷举这一基础算法的应用及其优化的同时，着重阐述了“聪明搜索”回溯法的设计要点与实际应用。

第3章递归与分治，这是最常用也是应用最广的有效算法。

第4章递推，在概括常用递推模式的基础上，着重列举了递推在数列与数阵求解，特别在应用递推求解计数问题方面的应用。

第5章贪心算法，举例说明了贪心算法在最优化求解方面高效率的应用。

第6章动态规划，介绍了在最优化问题求解中应用最广的动态规划的设计思想、设计要点与操作步骤，着重列举了动态规划在0-1背包、最长子序列与最优路径探索方面的应用。

第7章模拟，着重讲述了运算模拟、随机模拟与操作过程模拟等方面的具体应用。

第8章智能优化，简要介绍了模拟退火、遗传算法、粒子群算法、神经网络等智能优化算法的概念与基本内容。

第9章并行算法简介，介绍近年来发展非常迅速的并行计算的基本理论与简单应用。

本书由杨克昌任主编。第1、6、7章由杨克昌编写，第2、4章由吴小明编写，第3、8、9章由郭观七、李文彬、卢泽勇编写，第5章由谭用秋编写，最后由杨克昌统稿、定稿。本书的编写得到湖南理工学院领导和计算机与信息工程系的大力支持，编者在此表示衷心感谢。

由于编者学识水平所限，书中疏漏与错误之处在所难免，恳请广大读者批评指正。

编者

2008年2月

目　录

第 1 章 算法与程序设计简介

1.1 算法与算法描述

在计算机科学与技术中，算法（algorithm）是一个常用的基本概念。本节论述算法的定义与算法的描述。

1.1.1 算法

在计算机科学中，算法一词用于描述一个可用计算机实现的问题求解方法。算法是程序设计的基础，是计算机科学的核心。计算机科学家哈雷尔在《算法学——计算的灵魂》一书中指出："算法不仅是计算机学科的一个分支，它更是计算机科学的核心，而且可以毫不夸张地说，它和绝大多数科学、商业和技术都是相关的。"

在计算机应用的各个领域，技术人员都在使用计算机求解他们各自专业领域的问题，他们需要设计算法，编写程序，开发应用软件，所以学习算法对于越来越多的人来说变得十分必要。

什么是算法？我们首先给出算法的定义。

算法是指解决某一问题的运算序列。或者说算法是问题求解过程的运算描述，一个算法由有限条可完全机械地执行的、有确定结果的指令组成。指令正确地描述了要完成的任务和它们被执行的顺序。计算机按算法所描述的顺序执行算法的指令能在有限的步骤内终止，或终止于给出问题的解，或终止于指出问题对此输入数据无解。

算法是满足下列特性的指令序列。

1. 确定性

组成算法的每条指令是清晰的，无歧义的。

在算法中不允许有诸如"x/0"之类的运算，因为其结果不能确定；也不允许有"x 与 1 或 2 相加"之类的运算，因这两种可能的运算应执行哪一个，并不确定。

2. 可行性

算法中的运算是能够实现的基本运算，每一种运算可在有限的时间内完成。

在算法中两个实数相加是可行的；两个实数相除，例如求 2/3 的值，在没有指明位数时需由无穷个十进制位表示，并不可行。

3. 有穷性

算法中每一条指令的执行次数有限，执行每条指令的时间有限。

例如，如果算法中的循环步长为零，运算进入无限循环，这是不允许的。

4. 输入

一个算法有零个或多个输入。

5. 输出

一个算法至少产生一个量作为输出。

通常求解一个问题可能会有多种算法可供选择，选择的主要标准是算法的正确性和可靠性。其次是算法所需要的存储空间少和执行时间短等。

有人也许会认为：今天计算机运算速度这么快，算法还重要吗？诚然，计算机的计算能力每年都在飞速增长，价格也在不断下降。可我们不要忘记，日益先进的纪录和存储手段使我们需处理的信息量在爆炸式的增长，互联网的信息流量也在快速增长。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。无论是三维图形、海量数据处理、机器学习、语音识别，都需要极大的计算量。在网络时代，越来越多的挑战需要靠卓越的算法来解决。

算法并不局限于计算机和网络。在高能物理研究方面，很多实验每秒钟都能产生若干个 TB 的数据量，但因为处理能力和存储能力的不足，科学家不得不把绝大部分未经处理的数据舍弃。在气象方面，算法可以更好地预测未来天灾的发生，以拯救生命。所以，如果你把计算机的发展放到应用和数据飞速增长的大环境下，你一定会发现，算法的重要性不是在日益减小，而是在日益增强。

在实际工程中我们遇到许多的高难度计算问题，有的问题在巨型计算机上采用一个劣质的算法来求解可能要数个月的时间，而且很难找到精确解。但采用一个优秀的算法，即使在普通的个人计算机上，可能只需数秒钟就可以求得解答。计算机求解一个工程问题的计算速度不仅仅与计算机的设备水平有关，更取决于求解该问题的算法技术水平的高低。世界上许多国家，从大学到研究机关都高度重视对计算机算法的研究，已将提高算法设计水平看作是一个提升国家技术竞争力的战略问题。

对同一个计算问题来说不同的人会有不同的计算方法，而不同算法的计算效率、求解精度和对计算资源的需求有很大的差别。

本书具体介绍分治、递归与递推、贪心算法、回溯法、动态规划与模拟等常用算法。同时对智能优化与并行处理作简要介绍。

1.1.2 算法描述

要使计算机能完成人们预定的工作，首先必须为如何完成这些工作设计一个算法，然后

再根据算法编写程序。

一个问题可以设计不同的算法来求解，同一个算法可以采用不同的形式来表述。

算法是问题的程序化解决方案。描述算法可以有多种方式，如自然语言方式、流程图方式、伪代码方式、计算机语言表示方式与表格方式等。

当一个算法使用计算机程序设计语言描述时，就是程序。本书采用 C 语言与自然语言相结合来描述算法。之所以采用 C 语言来描述算法，因为 C/C++语言功能丰富、表达能力强、使用灵活方便、应用面广，既能描述算法所处理的数据结构，又能描述计算过程，是目前大学阶段学习计算机程序设计的首选语言。

为方便算法描述与程序设计，下面把 C 语言的基本要点作简要概括。

1. 标识符

可由字母、数字和下划线组成，标识符必须以字母或下划线开头， 大小写的字母分别认为是两个不同的字符。

2. 常量

整型常量：十进制常数、八进制常数（以 o 开头的数字序列）、十六进制常数（以 ox 开头的数字序列）。

长整型常数（在数字后加字符 L 或 l）。

实型常量（浮点型常量）：小数形式与指数形式。

字符常量：用单引号（撇号）括起来的一个字符，可以使用转义字符。

字符串常量：用双引号括起来的字符序列。

3. 表达式

（1）算术表达式

整型表达式：参加运算的运算量是整型量，结果也是整型数。

实型表达式：参加运算的运算量是实型量，运算过程中先转换成 double 型，结果为 double 型。

（2）逻辑表达式

用逻辑运算符连接的整型量，结果为一个整数（0 或 1），逻辑表达式可以认为是整型表达式的一种特殊形式。

（3）字位表达式

用位运算符连接的整型量，结果为整数。字位表达式也可以认为是整型表达式的一种特殊形式。

（4）强制类型转换表达式

用“(类型)”运算符使表达式的类型进行强制转换，如（float）a。

（5）逗号表达式（顺序表达式）

形式为：表达式 1，表达式 2，…，表达式 n

顺序求出表达式 1，表达式 2，…，表达式 n 的值，结果为表达式 n 的值。

（6）赋值表达式

将赋值号“=”右侧表达式的值赋给赋值号左边的变量，赋值表达式的值为执行赋值后

被赋值的变量的值。

（7）条件表达式

形式为：逻辑表达式？表达式 1：表达式 2

逻辑表达式的值若为非 0（真），则条件表达式的值等于表达式 1 的值；若逻辑表达式的值为 0（假），则条件表达式的值等于表达式 2 的值。

（8）指针表达式

对指针类型的数据进行运算。例如 p-2、p1-p2、&a 等（其中 p、p1、p2 均已定义为指针变量），结果为指针类型。

以上各种表达式可以包含有关的运算符，也可以是不包含任何运算符的初等量。例如，常数是算术表达式的最简单的形式。

表达式后加“；”，即为表达式语句。

4. 数据定义

对程序中用到的所有变量都需要进行定义，对数据要定义其数据类型，需要时要指定其存储类别。

（1）数据类型标识符有：

int（整型），short（短整型），long（长整型），unsigned（无符号型），char（字符型），float（单精度实型），double（双精度实型），struct（结构体名），union（共用体名）。

（2）存储类别有：

auto（自动的），static（静态的），register（寄存器的），extern（外部的）。

变量定义形式：存储类别 数据类型 变量表列

如：static float x,y

5. 函数定义

存储类别 数据类型 <函数名> (形参表列)

{函数体}

6. 分支结构

（1）单分支：

if(表达式) <语句 1> [else <语句 2>]

功能：如果表达式的值为非 0（真），则执行语句 1；否则（为 0，即假），执行语句 2。所列语句可以是单个语句，也可以是用{}界定的若干个语句。应用 if 嵌套可实现多分支。

（2）多分支：

```
switch(表达式)
   { case 常量表达式 1：<语句 1>
     case 常量表达式 2：<语句 2>
     …
     case 常量表达式 n：<语句 n>
     default：<语句 n+1>
```

}

功能：取表达式 1 时，执行语句 1；取表达式 2 时，执行语句 2；…，其他所有情形，执行语句 n+1。

case 常量表达式的值必须互不相同。

7. 循环结构

（1）while 循环：

while(表达式) 语句

功能：表达式的值为非 0（条件为真），执行指定语句（可以是复合语句）。 直至表达式的值为 0（假）时，脱离循环。

特点：先判断，后执行。

（2）Do-while 循环：

do　语句

while（表达式）

功能：执行指定语句，判断表达式的值非 0（真），再执行语句；直到表达式的值为 0（假）时，脱离循环。

特点：先执行，后判断。

（3）for 循环：

for（表达式 1；表达式 2；表达式 3）语句

功能：解表达式 1；求表达 2 的值：若非 0（真），则执行语句；求表达式 3；再求表达式 2 的值；……；直至表达式 2 的值为 0（假）时，脱离循环。

以上三种循环，若执行到 break 语句，提前终止循环。若执行到 continue，结束本次循环，跳转下一次循环判定。

顺便指出，在不致引起误解的前提下，有时对描述的 C 语句进行适当简写或配合汉字标注，用以简化算法框架描述。

例如，从键盘输入整数 n，按 C 语言的键盘输入函数应写为：

```
scanf("%d",&n);
```

可简写为：scanf(n);

或简写为：输入整数 n;

要输出整数量 $a(1),a(2),\cdots,a(n)$，按 C 语言的输出函数应写为：

```
for(k=1;k<=n;k++)
    printf("%d ",a[k]);
```

可简写为：

```
printf(a(1)-a(n));
```

或简写为：输出 a(1-n);

【例 1.1】 求两个整数 $a,b(a>b)$的最大公约数的欧几里德算法：

（1）a 除以 b 得余数 r；若 $r=0$，则 b 为所求的最大公约数。

（2）若 $r\neq 0$，以 b 为 a，r 为 b，继续（1）。

注意到任两整数总存在最大公约数，上述辗转相除过程中余数逐步变小，相除过程总会

结束。

欧几里德算法又称为“辗转相除”法，具体描述如下：

```
输入正整数 a，b;
if(a<b)
    {c=a;a=b;b=c;}                    /*  交换 a，b，确保 a>b */
r=a%b;
while(r!=0)
  { a=b;b=r;                          /*  实施“辗转相除” */
    r=a%b;
  }
printf(最大公约数 b);
```

1.2 算法复杂性分析

算法复杂性的高低体现运行该算法所需计算机资源的多少。算法的复杂性越高，所需的计算机资源越多；反之，算法的复杂性越低，所需的计算机资源越少。

计算机资源，最重要的是时间资源与空间资源。因此，算法的复杂性有时间复杂性与空间复杂性之分。需要计算机时间资源的量称为时间复杂度，需要计算机空间资源的量称为空间复杂度。时间复杂度与空间复杂度集中反映出算法的效率。

算法分析是指对算法的执行时间与所需空间的估算，定量给出运行算法所需的时间数量级与空间数量级。

1.2.1 时间复杂度

算法作为计算机程序设计的基础，在计算机应用领域发挥着举足轻重的作用。一个优秀的算法可以运行在计算速度比较慢的计算机上求解问题，而一个劣质的算法在一台性能很强的计算机上也不一定能满足应用的需求。因此，在计算机程序设计中，算法设计往往处于核心地位。如何去设计一个适合特定应用的算法是众多技术开发人员所关注的焦点。

要想充分理解算法并有效地应用于求解实际问题，关键是对算法的分析。通常我们可以利用实验对比方法、数学方法来分析算法。

实验对比分析很简单，两个算法相互比较，它们都能解决同一问题，在相同环境下，哪个算法的速度快我们一般就会认为这个算法性能好。

数学方法能更为细致的分析算法，能在严密的逻辑推理基础上判断算法的优劣。但在完成实际项目过程中，我们很多时候都不能去做这种严密的论证与推断。因此，在算法分析中，我们往往采用能近似表达性能的方法来展示某个算法的性能指标。例如，当 n 比较大的时，计算机对 n^2 和 n^2+2n 的响应速度几乎没有什么区别，我们便可直接认为这两者的复杂度均为 n^2。在分析算法时，隐藏细节的数学表示法成为大写 O 记法，它可以帮助我们简化算法复杂度的许多细节，提取主要成分，这和遥感图像处理中的主成分分析思想相近。

一个算法的时间复杂度是指算法运行所需的时间。一个算法的运行时间取决于算法所需执行的语句（运算）的多少。算法的时间复杂度通常用该算法执行的总语句（运算）的数量级决定。

就算法分析而言，一条语句的数量级即执行它的频数，一个算法的数量级是指它所有语句执行频数之和。

观察下面三个程序段：

（1）
```
x=x+1;
```

（2）
```
for(k=1;k<=n;k++)
    { x=x+y;
      y=x+y;
      s=x+y;
    }
```

（3）
```
for(t=1,k=1;k<=n;k++)
    {t=t*2;
     for(j=1;j<=t;j++)
       s=s+j;
    }
```

如果把以上 3 个程序段看成 3 个相应算法的主体，我们来看 3 个算法的数量级。

在（1）中，语句执行频数为 1，算法的数量级为 1；

在（2）中，每个赋值语句执行频数为 n，该算法的数量级为 $3n$；

在（3）中，内循环的赋值语句执行频数为 $2+2^2+\cdots+2^n=2(2^n-1)$。

算法的数量级直接决定算法的时间复杂度。

定义　对于一个数量级为 $f(n)$ 的算法，如果存在两个正常数 c 和 m，对所有的 $n \geqslant m$，有

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

则记作 $f(m)=O(g(n))$，称该算法具有 $O(g(n))$的运行时间，是指当 n 足够大时，该算法的实际运行时间不会超过 $g(n)$的某个常数倍时间。

显然，以上所列举的（1）与（2），其计算时间即时间复杂度分别为 $O(1)$，$O(n)$。

据以上定义，（3）的数量级为 $2(2^n-1)$，取 $c=2$，对任意正整数 n，有

$$2(2^n-1)<2\cdot 2^n$$

即得（3）的计算时间为 $O(2^n)$，即算法（3）的时间复杂度为 $O(2^n)$。

以上 3 个程序段前两个所代表的算法是多项式时间算法。最常见的多项式算法时间，其关系概括为

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

约定 $\log n$ 表示以 2 为底的对数。程序段（3）所代表的是指数时间算法。以下 3 种是最常见的指数时间算法，其关系为

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

随着 n 的增大，指数时间算法与多项式时间算法在所需的时间上相差非常大，表 1.1 具体列出了时间复杂度常用函数增长情况。

表 1.1　　时间复杂度常用函数增长情况

	$\log n$	n	$n\log n$	n^2	n^3	2^n
n=1	0	1	0	1	1	2
n=2	1	2	2	4	8	4
n=4	2	4	8	16	64	16
n=8	3	8	24	64	512	256
n=16	4	16	64	256	4 096	65 536
n=32	5	32	160	1 024	32 768	4 294 967 296

一般地，当 n 取值比较大时，在计算机上实现指数时间算法是不可能的，就是比 $O(n\log n)$ 时间复杂度高的多项式时间算法运行也很困难。

根据时间复杂度符号 O 的定义，有

定理 1.1　关于时间复杂度符号 O 有以下运算规则：

$$O(f)+O(g)=O(max(f,g)) \tag{1.1}$$

$$O(f)O(g)=O(fg) \tag{1.2}$$

证明　设 $F(n)=O(f)$，根据 O 定义，存在常数 c_1 和正整数 n_1，对所有的 $n\geqslant n_1$，有 $F(n)\leqslant c_1f(n)$。同样，设 $G(n)=O(g)$，根据 O 定义，存在常数 c_2 和正整数 n_2，对所有的 $n\geqslant n_2$，有 $G(n)\leqslant c_1g(n)$。

令 $c_3=max(c_1,c_2),n_3=max(n_1,n_2),h(n)=max(f,g)$。对所有的 $n\geqslant n_3$，存在 c_3，有

$F(n)\leqslant c_1f(n)\leqslant c_3f(n)\leqslant c_3h(n)$

$G(n)\leqslant c_2g(n)\leqslant c_3g(n)\leqslant c_3h(n)$

则　$F(n)+G(n)\leqslant 2c_3h(n)$

即　$O(f)+O(g)\leqslant 2c_3h(n)=O(h)=O(max(f,g))$

令 $t(n)=fg(n)$，对所有的 $n\geqslant n_3$，有

$F(n)G(n)\leqslant c_1c_2t(n)$

即　$O(f)O(g)\leqslant c_1c_2t(n)=O(fg)$。

式（1.1）、式(1.2)成立。

定理 1.2　如果 $f(n)=a_mn^m+a_{m-1}n^{m-1}+\cdots+a_1n+a_0$ 是 n 的 m 次多项式，$a_m>0$，则

$$f(n)=O(n^m) \tag{1.3}$$

证明　当 $n\geqslant 1$ 时，据符号 O 定义有

$$\begin{aligned} f(n)&=a_mn^m+a_{m-1}n^{m-1}+\cdots+a_1n+a_0\\ &\leqslant|a_m|n^m+|a_{m-1}|n^{m-1}+\cdots+|a_1|n+|a_0|\\ &\leqslant(|a_m|+|a_{m-1}|+\cdots+|a_1|+|a_0|)n^m \end{aligned}$$

取常数 $c=|a_m|+|a_{m-1}|+\cdots+|a_1|+|a_0|$，据定义，式（1.3）得证。

【例 1.2】　估算以下程序段所代表算法的时间复杂度。

```
for(k=1;k<=n;k++)
```