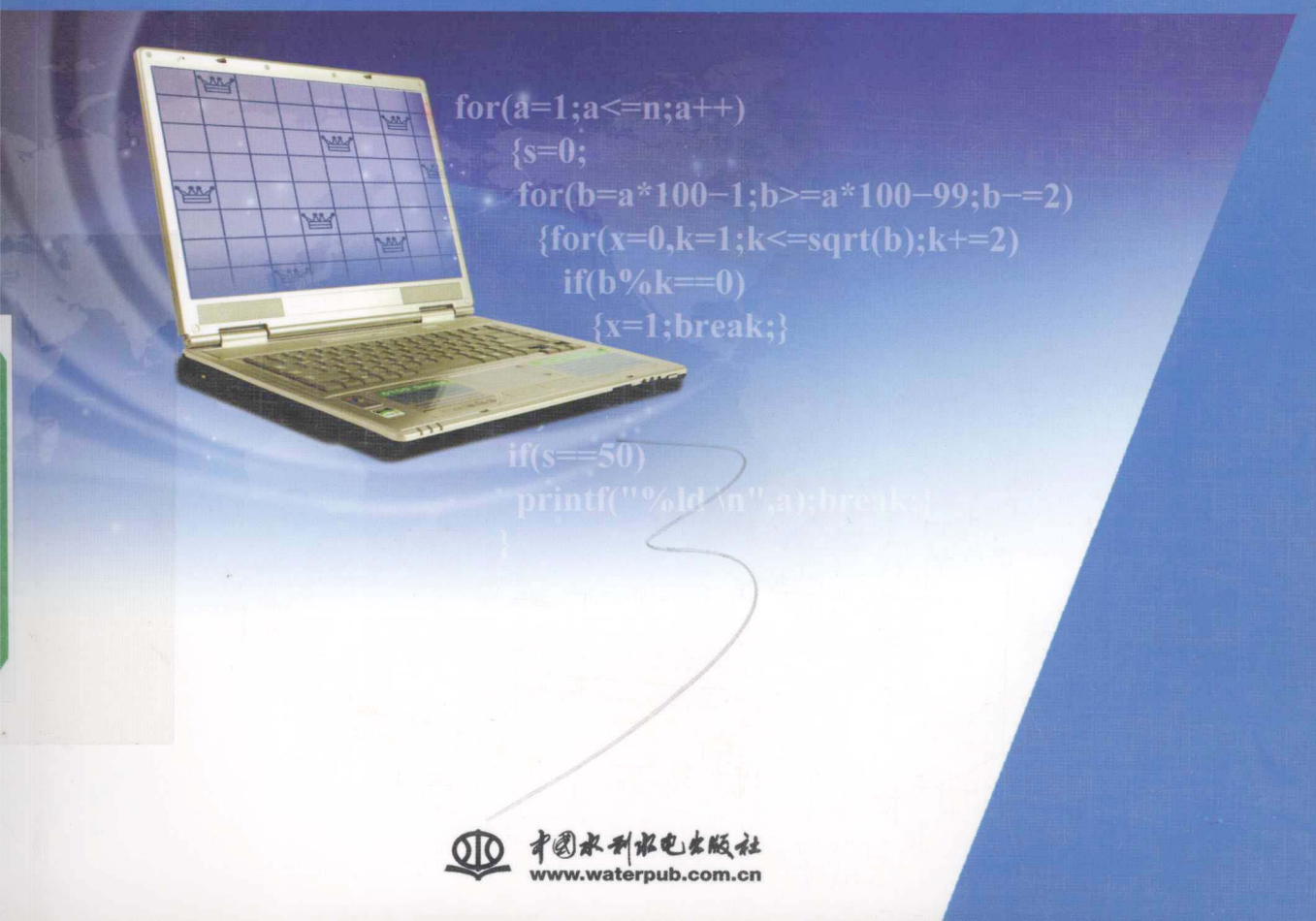
21世纪高等学校精品规划教材

算法设计与分析实用教程

杨克昌　严权峰　编著



中国水利水电出版社
www.waterpub.com.cn

21世纪高等学校精品规划教材

算法设计与分析实用教程

杨克昌　严权峰　编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书遵循“精选算法，面向设计，突出案例应用，注重能力培养”的编写宗旨，精选枚举、递推、递归、回溯、动态规划、贪心算法与模拟等常用算法，精心组织各算法应用的典型案例，注重算法设计与分析及算法改进与优化，力求理论与实际相结合，算法设计与案例应用相统一。每一个案例的应用求解，从问题提出、算法设计与描述，到算法测试与分析、算法改进与优化，环环相扣，融为一体。

书中所有应用案例的算法设计均给出设计要点与描述，可在 VC++ 6.0 编译通过。

本书可作为各高等院校计算机及相关专业“算法设计与分析”课程教材，供各级程序设计竞赛培训选用，也可作为广大程序设计爱好者与软件开发人员的参考书。

本书配套的教学课件、源代码与习题参考解答，可以从中国水利水电出版社网站以及万水书苑免费下载，网址为：http://www.waterpub.com.cn/softdown/ 或 http://www.wsbookshow.com。

图书在版编目（CIP）数据

算法设计与分析实用教程 / 杨克昌，严权峰编著
. -- 北京 : 中国水利水电出版社，2013.6
21世纪高等学校精品规划教材
ISBN 978-7-5170-0978-8

Ⅰ. ①算… Ⅱ. ①杨… ②严… 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. ①TP301.6

中国版本图书馆CIP数据核字(2013)第136360号

策划编辑：杨庆川　责任编辑：张玉玲　加工编辑：孙　丹　封面设计：李　佳

书　名	21世纪高等学校精品规划教材 **算法设计与分析实用教程**
作　者	杨克昌　严权峰　编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路1号D座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（发行部）、82562819（万水）
经　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市铭浩彩色印装有限公司
规　格	184mm×260mm　16开本　18.5印张　467千字
版　次	2013年6月第1版　2013年6月第1次印刷
印　数	0001—3000册
定　价	35.00元

前　言

进入21世纪，随着计算机的广泛普及与信息技术的深入发展，培养应用型软件开发人才成为提高国家科技水平的重要方面。国家“973”信息技术与高性能软件基础规划项目首席科学家顾钧教授与中国工程院院士李国杰教授指出：“我国的软件开发要算法先行，这样才能推动软件技术的研究与开发，提高我国企业软件产品的技术竞争力和市场竞争力。”

“算法设计与分析”是计算机科学技术学科体系的一个核心问题，是大学计算机专业一门重要的专业基础课。计算机算法设计是应用计算机解决实际问题的核心环节，是一种创造性思维活动，其教育必须面向应用。针对目前相关教材中，算法内容贪多求全、贪广求深，算法理论与实际应用脱节的现状，探索与改革适合计算机本科层次教学的“算法设计与分析”教材，直接关系到学生应用算法设计解决实际问题能力的培养与提高，是实现算法课程教学目标的当务之急。

为此，我们在《计算机常用算法与程序设计教程》（“十一五”国家级规划教材，人民邮电出版社，2008.11）与《计算机常用算法与程序设计案例教程》（清华大学出版社，2011.7）两本书的基础上进行整合优化，推出适合计算机本科教学实际的《算法设计与分析实用教程》，对算法的组织与案例的选取、算法理论阐述与案例实际应用结合进行精心设计，力求适合高校本科教学目标与知识结构的要求。

本书遵循**“精选算法，面向设计，突出案例应用，注重能力培养”**的编写宗旨，具有以下“四注重”特色。

（1）注重常用算法的选取与组织

本书在常用算法的选取方面，克服以往贪多求全、贪广求深的弊端，结合本科教学实际，精心选取了枚举、递推、递归、动态规划、回溯、贪心算法与模拟等常用算法，避免出现本科阶段与研究生阶段的教学内容混杂不分的局面。其中，模拟算法中的“竖式运算模拟”是我们总结推广应用于数论高精计算的创新成果。

对选取的常用算法，在介绍算法的基本理论与设计规范的基础上，推介该算法设计应用的常用模式，联系典型案例讲述该算法中要求本科生掌握的基本内容与应用设计，删除一些难度大、理论深、应用少的带研究性质的算法内容。

（2）注重典型案例的精选与提炼

算法是程序设计的核心，算法需要典型案例的展示与支撑。培养学生学习算法设计的兴趣，激发学生应用算法设计解决实际问题的学习热情，不是一两句空洞说教所能奏效的，必须通过一些有趣的、有启发性的典型案例来引导。教程在算法材料的组织上，克服以往罗列算法多、应用算法设计解决实际问题少、算法理论与实际应用脱节的弊端，对选取的每一种算法，设计了基础型、应用型与综合型三种难度梯度的典型案例，包括基础的数值求解、常见的数据处理、有趣的智力测试、巧妙的模拟探索，既有启发引导的基础案例，也有难度较大的综合案例，既有构思巧妙的新创趣题，也有历史悠久的经典名题，难度适宜，深入浅出。

这些典型案例的精选与提炼有利于提高学生学习算法设计的兴趣，有利于学生在计算机

实际案例求解上开阔视野，使之在算法思路的开拓与设计技巧的应用上有一个深层次的锻炼与提高。通过典型案例来展开算法设计与分析，突出算法设计在解决实际案例中的核心地位与指导作用。其中一些难度较大的综合型案例可作为课后的专题研究与课程设计选用。

（3）注重算法设计与实现的紧密结合

算法是程序设计的核心与灵魂，程序是算法的一种描述与实现手段。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。教程采用功能丰富、应用面广、高校学生使用率最高的C语言描述算法，并直接在VC++环境下测试通过，大大缩减了算法设计与程序实现的距离。对每一个实际案例，从案例提出、算法设计与描述，到算法测试与分析、算法改进与优化，环环相扣，融为一体。学生看得懂、学得会、用得上，可以收到立竿见影、举一反三的效果，力求理论与实际相结合、算法设计与应用实际相统一，突出算法在解决实际案例中的核心地位与引导作用，切实提高学生对算法思想的理解和算法设计的把握，有效提高学生应用算法设计解决实际问题的水平和能力。

（4）注重算法设计的改进与优化

算法设计不可能是一成不变的，教材对某些典型案例提供了多种算法设计，编写出不同表现形式与不同设计风格的算法，并对一些常规设计实施多层次多方位的变通、改进与优化，集中体现了算法设计的灵活性和多样性。对不同算法的复杂性分析，确立不同算法的特点与适应环境，比较出不同算法的优劣。

变通出成果，变通长能力。算法改进与优化的过程既是提高案例求解效率的过程，也是对学生算法设计能力培养与提高的过程，更是优化意识与创新能力增强的过程。

为方便师生教学，本书配套的教学课件、源代码与习题的参考解答均可从指定网站下载。

本书可作为各高等院校计算机及相关专业“算法设计与分析”、“计算机程序设计应用基础”等课程教材，供各级程序设计选拔赛与ACM复习参考；也可供IOI、NOI及各省程序设计竞赛培训选用；还可作为广大程序设计爱好者与软件开发人员的参考书。

本书由杨克昌、严权峰进行策划、编著与统稿。在本书的编写过程中，湖南理工学院的王岳斌教授、周持中教授以及甘靖、方世林等老师给予了多方面的指导与帮助，唐雁、陈凯、姜镇林等同学阅读了书稿并测试了相关算法，在此深表感谢。

尽管每一算法设计都经反复核实检查与检测调试，但因涉及内容较广，难免存在差错，恳请专家与读者批评指正。

编著者

2013年6月

目 录

第 1 章　算法及其复杂性分析

算法理论研究的是算法的设计技术与分析技术。算法设计是指应用计算机求解一个问题时，如何设计一个有效的算法；算法分析则是对已设计的算法，如何评价与判断其优劣。

算法设计与算法分析是相互依存的，设计出的算法需要测试、检验和评价，对算法的分析反过来将改进与优化算法设计。

1.1　算法及其描述

算法（algorithm）的概念在计算机科学技术领域几乎无处不在，在各种计算机软件系统的设计与实现中，算法往往处于核心地位。

算法在中国古代文献中称为“术”，最早出现在《周髀算经》和《九章算术》。而英文名称“Algorithm”来自于 9 世纪波斯数学家花拉子米（比阿勒·霍瓦里松，拉丁转写：al-Khwarizmi），因为比阿勒·霍瓦里松在数学上提出了算法这个概念。“算法”原为“algorism”，即“al-Khwarizmi”的音转，意思是“花拉子米的运算法则”，在 18 世纪演变为“algorithm”。

本节论述算法的定义、特征及算法的描述。

1.1.1　算法定义与特性

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课题，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。我们学习算法的重点就是把人类找到的求解问题的方法、步骤，以过程化、形式化、机械化的形式表示出来，以便让计算机执行，从而解决更多实际问题。

1. 算法定义

什么是算法？我们首先给出算法的定义。

算法是解决问题的方法或过程，是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。在数学和计算机科学之中，算法是一个计算的具体步骤，常用于计算、数据处理和自动推理。

当面临某一问题时，需要找到用计算机解决这个问题的方法与步骤，算法就是解决这个问题的方法与步骤的描述。

2. 算法三要素

算法由操作、控制结构与数据结构三要素组成。

（1）操作

加、减、乘、除等算术运算；

大于、小于、等于、大于等于、小于等于、不等于等关系运算；

与、或、非等逻辑运算；

输入、输出、赋值等操作。

（2）控制结构

顺序结构：各操作依次执行；

选择结构：由条件是否成立来选择操作执行；

循环结构：重复执行某些操作，直到满足某一条件才结束；

模块调用：一个模块调用另一个模块（包括自身直接或间接调用的递归结构）。

（3）数据结构

算法的处理对象是数据，数据之间的逻辑关系、数据的存储方式与处理方式就是数据的数据结构。

常见的数据结构有：数组（Array）、堆栈（Stack）、队列（Queue）、链表（Linked List）、树（Tree）、图（Graph）、堆（Heap）、散列（Hash）等。

3. 算法的基本特征

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。

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）算法有零个或多个输入

有些输入数据需要在算法执行过程中输入，有些算法看起来没有输入，实际上输入已被嵌入算法之中。

（5）算法有一个或多个输出

输出一个或多个与输入数据有确定关系的量，是算法对数据进行运算处理的结果。

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。

4. 算法的重要意义

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？”计算机的计算能力每年都在飞速增长，价格在不断下降。可我们不要忘记，日益先进的记录和存储手段使我们需要处理的信息量也在快速增长，互联网的信息流量更在爆炸式地增长。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。例如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。三维图形、海量数据处理、机器学习、语音识别都涉及极大数据量处理。

算法并不局限于计算机和网络。在网络时代，越来越多的挑战需要靠卓越的算法来解决。如果你把计算机的发展放到数据飞速增长的大环境下考虑，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设计水平的高低。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，将提高算法设计水平作为提升国家科技竞争力的重要方面。

对同一个计算问题，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。

本书具体介绍枚举、递推、递归、回溯、动态规划、贪心算法与模拟等常用算法及其在实际案例求解中的应用。最后介绍几个算法综合应用与优化的案例。

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。

可以设计不同的算法来求解一个问题，可以采用不同的形式来表述同一个算法。

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 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，下面把 C 语言的语法要点作简要概括。

1. 标识符

可由字母、数字和下划线组成，但是标识符必须以字母或下划线开头。一个字母的大小写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 0 开头的数字序列）、十六进制常数（以 0x 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。注意，赋值号左边必须是变量，而不能是表达式。

（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

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

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〉(形参列表)

{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。

以上三种循环中，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，用以简化算法的框架描述。

例 1-1　求两个整数 a 和 b(a>b)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≠0，以 b 为 a，r 为 b，继续第 (1) 步。

注意到任两个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，应用C语言具体描述如下：

```
// 辗转相除求两个整数a、b最大公约数
main()
  { long  a,b,c,r;
    printf(" 请输入整数a,b: ");
    scanf("%ld,%ld",&a,&b);
    if(a<b)
       { c=a;a=b;b=c;}                    // 必要时交换a和b，确保a>b
    r=a%b;
    while(r!=0)
       { a=b;b=r;                         // 通过循环实施“辗转相除”
         r=a%b;
       }
    printf(" 最大公约数为: %ld",b);
  }
```

该算法中有输入，即输入整数a和b；有“辗转相除”处理，通过条件循环实施；最后有输出，即输出整数a和b的最大公约数。

例1-2 某学院有m个学生参加南湖春游，休息时喝汽水。南湖商家公告如下：

（1）买1瓶汽水定价1.40元，喝1瓶汽水（瓶不带走）1元。

（2）为节约资源，规定3个空瓶可换回1瓶汽水，或20个空瓶可换回7瓶汽水。

（3）为方面顾客，可先借后还。例如借1瓶汽水，还3个空瓶；或借7瓶汽水，还20个空瓶。

问m个学生每人喝1瓶汽水（瓶不带走），至少需多少元？

输入正整数m(2<m<10000)，输出至少需多少元（精确到小数点后第2位）。

解：注意到春游喝汽水无须带走空瓶，根据商家的规定作以下分析。

（1）如果人数为20人，买13瓶汽水，借7瓶汽水，饮完20瓶汽水后还20个空瓶（即相当于换回7瓶汽水还给商家），两清。此时每人花费为

$$13/20*1.40=0.91 \text{ 元}$$

（2）如果人数为3人，买2瓶汽水，借1瓶汽水，饮完3瓶汽水后还3个空瓶（即相当于换回1瓶汽水还给商家），两清。此时每人花费为

$$2/3*1.40=0.93 \text{ 元}$$

（3）如果只有2人或1人，每人喝1瓶汽水（瓶不带走），此时每人花费1元。

（4）注意到0.91<0.93<1，因而有以下的最省钱算法：

1）把m人分为x=m/20个大组，每组20人。每组买13瓶汽水（借7瓶汽水），饮完后还20个空瓶（即相当于换回7瓶汽水还给商家），两清。

2）剩下t=m-x*20人，分为y=t/3个小组，每组3人。每组买2瓶汽水（借1瓶汽水），饮完后还3个空瓶（即相当于换回1瓶汽水还给商家），两清。

3）剩下t=m-x*20-y*3人，每人花1元喝1瓶。

该算法得所花费用最低为：(13*x+2*y)*1.40+t（元）。

（5）费用最低的算法描述

```
// 喝汽水
main()
{ long m,t,x,y;
  printf("  请输入 m: "); scanf("%ld",&m);
  x=m/20;                    // 分 x 个大组，每组买 13 瓶汽水，借 7 瓶
  t=m-20*x;                  // 剩下大组外的 t 人
  y=t/3;                     // 剩下 t 人分 y 个小组，每组买 2 瓶汽水，借 1 瓶
  t=m-20*x-3*y;              // 剩下大小组外的 t 人，每人花 1 元喝 1 瓶
  printf("  喝%ld 瓶汽水，需%.2f 元。\n",m,(13*x+2*y)*1.40+t);
}
```

该算法有输入，即输入人数 m；有处理，即依次计算大组数 x、小组数 y 与剩下的零散人数 t；有输出，即输出最省费用。

由以上两例可知，应用 C 语言描述算法缩减了从算法写成完整 C 程序的距离，比应用其他方法描述更为简便。

1.2　算法复杂性分析

算法复杂性的高低体现在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。

计算机资源中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　算法的时间复杂度

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。时间复杂度是指在计算机科学与工程领域完成一个算法所需要的时间，是衡量一个算法优劣的重要参数。时间复杂度越小，说明该算法效率越高，则该算法越有价值。

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。

1. 算法分析的方法

要想充分理解算法并有效地应用算法求解实际案例，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，哪个算法的速度更快，我们一般就会认为这个算法性能更优。

数学方法能更为细致地分析算法，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，当计算参数 n 比较大

时，计算机对 n^2 和 n^2+2n 的响应速度几乎没有什么区别，我们便可以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方法为大写字母“O”记法，它可以帮助我们简化算法复杂度计算的许多细节，提取有决定意义的主要成分。

2. 算法的执行语句频数

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。

例 1-3 试计算下面三个程序段的执行频数。

```
（1）x=x+1;s=s+x;
（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;
     }
```

解：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 个算法的执行频数。

在（1）中，2 个语句各执行 1 次，共执行 2 次。

在（2）中，“k=1”执行 1 次；“k<=n”与“k++”各执行 n 次；3 个赋值语句，每个赋值语句各执行 n 次；共执行 5n+1 次；

在（3）中，“t=1”与“k=1”各执行 1 次；“k<=n”与“k++”各执行 n 次；“t=t*2”执行 n 次；“j=1”执行 n 次；“j<=t”、“j++”与内循环的赋值语句“s=s+j”各执行频数为

$$2+2^2+\cdots+2^n=2(2^n-1)$$

因而(3)中的执行频数为：$6\cdot 2^n+4n-4$。

3. 算法时间复杂度定义

算法的执行频数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 $f(n)=O(g(n))$，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，例 1-3 中的（1）与（2），其计算时间即时间复杂度分别为 O(1) 和 O(n)。

据以上定义，（3）的执行频数为 $6\cdot 2^n+4n-4$，取 c=8，对任意正整数 n，有

$$6\cdot 2^n+4n-4\leqslant 8\cdot 2^n$$

即得（3）的计算时间为 $O(2^n)$，即（3）所代表的算法时间复杂度为 $O(2^n)$。

算法(1)、(2)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(约定 logn 表示以 2 为底的对数，下同)：

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

算法（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

	logn	n	nlogn	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	4096	65536
n = 32	5	32	160	1024	32768	429967296

一般地，当 n 取值充分大时，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颇为困难。

4. 符号 O 的运算规则

根据时间复杂度符号 0 的定义，有以下定理。

定理 1-1　关于时间复杂度符号 0 有以下加与乘运算规则

$$0(f)+0(g)=0(\max(f,g)) \tag{1.1}$$

$$0(f)*0(g)=0(f*g) \tag{1.2}$$

证明　设 F(n)=0(f)，报据 0 定义，存在常数 c1 和正整数 n1，对所有的 n≥n1，有 F(n)≤c1*f(n)。同样，设 G(n)=0(g)，报据 0 定义，存在常数 c2 和正整数 n2，对所有的 n≥n2，有 G(n)≤c2*g(n)。

令 c3=max(c1,c2),n3=max(n1,n2),h(n)=max(f,g)。对所有的 n≥n3，存在 c3，有

F(n)≤c1*f(n)≤c3*f(n)≤c3*h(n)

G(n)≤c2*g(n)≤c3*g(n)≤c3*h(n)

则　F(n)+G(n)≤2*c3*h(n)

即　0(f)+0(g)≤2*c3*h(n)=0(h(n))=0(max(f,g))

令 t(n)=f(n)*g(n)，对所有的 n≥n3，有

F(n)*G(n)≤c1*c2*t(n)

即　0(f)0(g)≤c1*c2*t(n)=0(f*g)。

式（1.1）和(1.2)成立。

定理 1-2　如果 $f(n)=a_m n^m+a_{m-1}n^{m-1}+\cdots+a_1 n+a_0$ 是 n 的 m 次多项式，$a_m>0$，则

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

证明　当 n≥1 时，据符号 0 定义有