高等教育规划教材

算法设计与应用

汪荣贵　杨　娟　薛丽霞　编著

提供电子教案

下载网址 http://www.cmpedu.com

机械工业出版社
CHINA MACHINE PRESS

高等教育规划教材

算法设计与应用

汪荣贵　杨　娟　薛丽霞　编著

机械工业出版社

本书深入浅出、全面系统地介绍了常用算法设计与应用技术，内容包括算法设计的策略、排序与查找算法、树模型算法、图模型算法、网络流模型算法、组合优化算法、深度学习算法、若干重要的专用算法等。本书将算法的经典内容、前沿内容以及相关的应用技术进行整合，形成一套完整、统一的体系结构，使得读者在学习算法理论知识的同时，还能系统地掌握算法在应用方面的知识，为后续学习打下扎实的算法设计与应用基础。全书各章自成体系，可分别作为独立单元进行选择学习，以满足读者的差异化需求。每章均配有一定数量的习题，供读者练习。

本书内容丰富、思路清晰、实例讲解详细、图例直观形象，适合作为计算机及相关专业的本科生教材，也可供工程技术人员和自学读者学习参考。

本书配有授课电子课件，需要的教师可登录 www.cmpedu.com 免费注册，审核通过后下载，或联系编辑索取（QQ：2850823885，电话：010-88379739）。

图书在版编目（CIP）数据

算法设计与应用 / 汪荣贵，杨娟，薛丽霞编著. —北京：机械工业出版社，2016.12

高等教育规划教材

ISBN 978-7-111-57805-5

Ⅰ. ①算… Ⅱ. ①汪… ②杨… ③薛… Ⅲ. ①算法设计－高等学校－教材 Ⅳ. ①TP301.6

中国版本图书馆 CIP 数据核字（2017）第 206630 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：郝建伟 陈瑞文 责任校对：张艳霞

责任印制：李 昂

中国农业出版社印刷厂印刷

2017 年 9 月第 1 版 • 第 1 次印刷

184mm×260mm • 22.75 印张 • 555 千字

0001－3000 册

标准书号：ISBN 978-7-111-57805-5

定价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

服务咨询热线：（010）88379833

读者购书热线：（010）88379649

网络服务

机 工 官 网：www.cmpbook.com

机 工 官 博：weibo.com/cmp1952

教育服务网：www.cmpedu.com

金 书 网：www.golden-book.com

出 版 说 明

当前，我国正处在加快转变经济发展方式、推动产业转型升级的关键时期。为经济转型升级提供高层次人才，是高等院校最重要的历史使命和战略任务之一。高等教育要培养基础性、学术型人才，但更重要的是加大力度培养多规格、多样化的应用型、复合型人才。

为顺应高等教育迅猛发展的趋势，配合高等院校的教学改革，满足高质量高校教材的迫切需求，机械工业出版社邀请了全国多所高等院校的专家、一线教师及教务部门，通过充分的调研和讨论，针对相关课程的特点，总结教学中的实践经验，组织出版了这套“高等教育规划教材”。

本套教材具有以下特点：

1）符合高等院校各专业人才的培养目标及课程体系的设置，注重培养学生的应用能力，加大案例篇幅或实训内容，强调知识、能力与素质的综合训练。

2）针对多数学生的学习特点，采用通俗易懂的方法讲解知识，逻辑性强、层次分明、叙述准确而精炼、图文并茂，使学生可以快速掌握，学以致用。

3）凝结一线骨干教师的课程改革和教学研究成果，融合先进的教学理念，在教学内容和方法上做出创新。

4）为了体现建设“立体化”精品教材的宗旨，本套教材为主干课程配备了电子教案、学习与上机指导、习题解答、源代码或源程序、教学大纲、课程设计和毕业设计指导等资源。

5）注重教材的实用性、通用性，适合各类高等院校、高等职业学校及相关院校的教学，也可作为各类培训班教材和自学用书。

欢迎教育界的专家和老师提出宝贵的意见和建议。衷心感谢广大教育工作者和读者的支持与帮助！

机械工业出版社

前　言

计算机已融入人们的生活，改变人们的生活方式，并把人们带进了如今的信息时代和互联网时代。很多年轻人通过掌握计算机核心技术而改善了自己的生活，甚至改变了自己的命运，实现了人生的辉煌。因此，学好计算机、成为高水平计算机专业人才是众多有志青年梦寐以求的目标。然而，计算机知识浩瀚无边，无数事实表明：对于初学者来说，如果在学习的开始阶段不能抓住一些关键核心知识进行重点学习，那么很快就会被计算机知识的海洋湮没而一事无成。

那么作为一个计算机初学者，首先需要重点学习的核心知识是什么？

事实上，高水平的计算机专业人员多是通过编写程序的方式进行研发工作的。因此，要想学好计算机，首先必须学好编程知识，能够写出高水平的程序。那么什么叫程序？初学者又该如何学习才能成为真正的编程高手？

著名的瑞士计算机科学家、图灵奖获得者尼克劳斯·沃尔斯（N. Wirth）教授给出如下著名的程序定义公式并被广大计算机业内人士认可：

程序=数据结构+算法

从本质上说，计算机是一种信息处理工具，处理对象是五花八门的各类数据，必须采用合理的方式或结构来存放这些数据，以便能够进行方便快速的存取。这就是数据结构要解决的问题，也是优秀程序员必须掌握的基本知识。

算法则是问题求解的计算思路或步骤。计算机毕竟只是一部机器，它不能直接面对和解决现实问题，而需要人类将现实问题抽象成一定的数学模型，再通过一定的算法对数学模型进行求解。因此，算法是分析问题、解决问题的核心内容，是程序的灵魂。算法设计技术是程序员安身立命的根本，也是计算机科学与技术学科的重要基石。

因此，要学好计算机，首先必须学好数据结构与算法设计知识。

算法设计是一种思维的艺术，需要一定的抽象思维能力和数学知识。这为广大初学者学习和掌握算法设计知识带来一定的困难，使得算法设计技术一直难以得到普及，在一定程度上阻隔了众多计算机学子的梦想。因此，多年来笔者一直很想编写一本适合的算法设计教材来解决这个问题，以满足广大读者学习和掌握算法设计技术的需要。

为使本书达到上述目的，笔者着重考虑如下两个要点：

第一、突出应用。一项技术有了具体的应用，大家自然会感兴趣，兴趣是最好的老师和学习动力。因此，本书采用比较独特的编写方式，不仅在每个章节穿插丰富的应用实例来验证算法的正确性、有效性和实用性，而且介绍和讨论了若干专用算法的设计技术，使得读者在学习算法设计理论知识的同时，还能够系统地掌握算法应用方面的知识。

第二、强调可读性。本书站在本科生低年级的思维角度进行编写，在保证表达准确的前提下，尽可能用最朴实的语言深入浅出地介绍算法设计技术，着重突出算法设计的思想和本质，而不仅仅是数学上的形式化描述。同时，本书辅以生动的图形解释，使读者能够对算法有更加清晰、全面的理解。需要说明的是，本书并没有为了增加可读性而降低知识的深度，只是通过

比较恰当的方式把相关知识表达得更加清楚明白，使得广大读者通过自己的努力就可以不太困难地掌握算法设计与应用的核心技术。

本书内容主要包括算法设计的策略、排序与查找算法、树模型算法、图模型算法、网络流模型算法、组合优化算法、若干专用算法的设计与应用，全面、系统地介绍了计算机算法设计的基本理论与应用知识。本书力图将算法的经典内容与前沿内容有机地结合在一起，形成一套完整、统一的体系结构，既有算法设计策略的总结，又有具体算法专题的论述，既有计算理论上的深度剖析，又有专门的算法实用性的讨论，使得广大读者在获取算法设计理论知识的同时，能比较系统地掌握算法应用方面的知识，为今后的工作和进一步学习打下扎实的算法设计与应用基础。

本书首先在第 1 章介绍算法设计的基础知识，包括算法的概念、表示和效率分析方法，然后从算法设计策略出发，在第 2 章详细介绍了算法设计的基本策略，包括蛮力与贪心策略、递归与分治策略、回溯与分支限界策略、动态规划策略等，并结合实例分析相关策略的应用场合，使得读者能够从宏观上了解和掌握算法设计的核心思想。随后，在此基础上用连续 5 章的篇幅详细分析和讨论了若干基本算法，即排序算法、树模型算法、图模型算法、网络流算法和查找算法，这 5 章是算法设计的核心知识，读者应熟练掌握。本书在随后的第 8 章介绍了组合优化及智能算法的设计技术，包括组合优化的基本算法和启发式算法、深度学习模型及算法。这部分内容已接近或到达算法学术研究前沿，可为读者进一步学习深造和研发高水平智能程序提供良好的基础。最后，本书在第 9 章详细介绍了若干重要专用算法的设计技术，包括数据压缩算法、数据加密算法以及字符串匹配算法，使读者能通过学习和使用这些专用算法较好地解决实际问题。

本书读者对象为计算机科学与技术专业、软件工程专业、信息安全专业、物联网专业、电子信息工程专业、自动化专业、信息与计算科学专业的本科生和低年级研究生，以及相关专业的广大科技工作者。考虑到算法内容浩瀚、种类繁多，掌握全部内容需要大量的时间，为方便广大读者进行差异化学习，笔者将全书各章设计为自成体系、可以独立学习的相对完整的单元，读者可以根据自身需要选择学习本书的部分章节或全部章节。

本书由汪荣贵、杨娟、薛丽霞编著。感谢在编写过程中研究生张清杨、丁凯、李永福、胡健根、张永刚、谢云飞、钟欣、孙伟、曹浩宇、刘雷雷、汪庆辉同学，以及本科生徐冲冲、王丽、李飞、江迪、杨茜、郑岩、张鸿翔、陈龙、李芸洁、曹瑞娟、王兴鹏、徐玲、刘睿德、姚旭晨等同学提供的帮助，感谢机械工业出版社、合肥工业大学计算机与信息学院的大力支持。

由于时间仓促，书中难免存在不妥之处，敬请读者不吝指正！

编　者

目　录

第 1 章　算法设计的基础知识

算法这个词并不陌生，因为大家从小学就开始接触算法。例如，做算术四则运算就是按照一定的算法步骤一步一步地计算，最终得到某种运算结果。简单地说，算法即为算术之法，就是通过运算的方式按照步骤逐步实现对问题的求解。事实上，算法思想源于东方，正如我国最高科技奖获得者吴文俊大师所述："我们传统数学在从问题出发以解决问题为主旨的发展过程中，建立了以构造性和机械化为其特色的算法体系，这与西方数学以欧几里得《几何原本》为代表的所谓公理化演绎体系正好遥遥相对"。

算法的这种基于构造性和机械性的运算体系非常契合计算机的运算特点和需求。因此，随着计算机的诞生和兴起，算法设计理论与技术迅速得到广泛重视和深入研究。如今关于算法设计的知识成果丰富多彩、不胜枚举，已经形成一个庞大的知识宝库，支撑着整个计算机科学与技术体系。本章可以看作打开这个宝库的钥匙，从算法的基本概念、表示方法、性能分析等几个方面介绍算法设计最基本的知识，为学习后续各章提供最基本的概念和方法。

1.1　计算机与算法

计算机是处理数据并将数据转化为有用信息的电子设备。任何计算机都是由程序指令控制的，程序指令规定计算机的用途，并告诉计算机需要完成的工作。因此，要使计算机工作就需要编写计算机程序，告诉计算机如何按照步骤执行程序，以实现最终目标。值得注意的是，根据计算机的特点，在告诉计算机需要它完成什么任务的同时，还需要进一步确定或选择让它怎样去完成，这正是计算机算法的由来。因此，对算法的处理逻辑进行合理设计并正确分析相关的计算性能是利用计算机解决问题的关键。任何受过良好训练的计算机专业人士都必须知道怎样去构造算法、理解算法和分析算法。算法的重要性，可以这样理解：科学的殿堂里陈列着两颗熠熠生辉的宝石，一颗是微积分，另一颗就是算法，微积分成就了以物理学为基础的工业文明，而算法则成就了以计算机技术为基础的现代信息社会。

1.1.1　计算机问题求解

用计算机处理数据或解决问题时，虽然可以根据不同的具体情况采用多种方法，但基本步骤是相同的，可大致分为问题分析与建模、算法设计与实现、算法分析与性能评价这 3 个步骤，如图 1-1 所示。

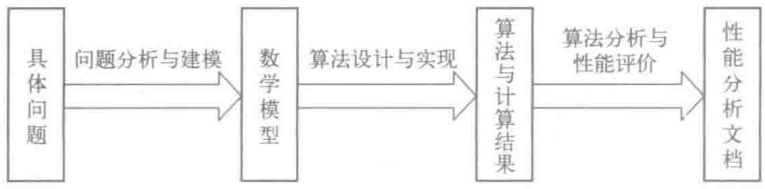

图 1-1　计算机问题求解

1. 问题分析与建模

现实问题纷繁复杂，而计算机能够处理的数据只能是二进制数据。因此，必须对现实问

题进行分析、抽象，建立相应的数学模型，把对现实问题的求解转化为对抽象数学模型的求解，以满足计算机处理问题的特点和基本要求。对现实问题进行分析的主要内容如下：首先要确定问题的逻辑结构和问题的基本功能，然后在结合数学、物理、计算机等与问题领域相关知识的基础上，建立问题求解的相关数学模型。问题分析与建模是计算机求解过程中一个非常重要的环节，如果在该环节没有对问题进行充分全面的考虑，则很有可能给随后的算法设计与实现环节带来很大麻烦，甚至设计出不符合实际要求或不可行的算法。

2. 算法设计与实现

建立好数学模型之后，就可以通过算法设计获得相应的算法来实现对所建数学模型的求解。所谓计算机算法设计，就是设计出解决某一特定问题或某一类问题的一系列处理步骤，并且要求这些步骤可以通过计算机的基本操作来实现。计算机算法设计一般需要结合数据结构特点，因为数据的结构特点有时会影响算法设计策略的选择。在设计算法时，一般先考虑是否可以借鉴现有知识或算法。如果现有算法不能解决此类问题，就要根据问题分析结果，逐步求精地设计出所需算法。在算法设计完成后，要将其表示成计算机语言，以便能够通过计算机来解决现实中的具体问题。算法的实现主要通过编程语言来完成，不同编程语言实现的算法有时会有一些小的差异。

3. 算法分析与性能评价

算法分析是对所设计算法的性能特征进行分析、评价和总结。我们总希望所设计的算法具有很多良好的特征，其中最主要的特征就是算法的正确性和算法效率。所有算法的正确性都需要经过严格的数学证明，但在很多情况下这是一个非常困难甚至无法完成的任务（关于算法正确性的数学分析与证明，已经超出了本书的范畴，本书主要通过一些实际算例的结果来验证算法的正确性）。算法的效率分析主要是对算法运行的时间效率和内存使用的空间效率进行分析和度量，这是本书重点学习的内容之一，后面将有详细介绍。

除了算法的正确性和效率，还需要分析和讨论算法的简单性和普适性。简单易懂的算法能使更多人接受并使用，应在保证算法正确性和效率的前提下，尽可能设计出简单的算法。普适性一般具有两方面的含义：一是指算法能够基本上无差别地适合解决某一大类问题；二是指算法所接受的输入具有一般性，而不是只在某个或某些特定输入情况下才能取得较好的效果。对于第一个方面，有时较难做到，如二次方程的标准求解公式不可能推广到任意次数的多项式方程；对于第二个方面，则要求所设计的算法能够处理此类现实问题所有可能的输入，并取得较好的效果。

1.1.2 算法的概念

算法一词出自《周髀算经》，即算术之法。目前，关于算法的概念有多种定义，至今没有一种得到公认，但可从这些不同的定义中找到一些基本的共识。为求同存异，下面从广义和狭义两个角度来诠释与理解算法的概念。

从广义上讲，算法是指通过运算的方式按照某种机械的步骤逐步实现对问题的求解，从这个角度看，现实生活中的很多工作流程都可以看作算法，如做菜的菜谱、理发的流程等，都可以叫算法。从狭义上讲，算法是一个由已知推求未知的过程，对于符合一定规范的输入，它能够在有限的时间内获得所需的输出。在计算机专业领域，主要是从狭义的角度来理解算法，通过图 1-2，可以更加形象地理解狭义的算法的含义。

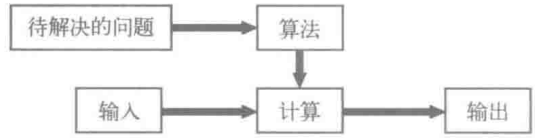

图 1-2　狭义算法的含义

本书的算法设计，主要是从计算机专业角度来理解算法。在计算机专业领域，对算法的理解不仅是狭义的，而且对算法概念的诠释更加明确具体，即算法是指用于计算机解决问题的清晰有穷指令序列，且满足以下 5 条基本性质：

1）有穷性。算法中每条指令的执行次数和时间均有限。执行次数或执行时间无穷的算法，对实际生产生活几乎没有意义，而且还会造成资源浪费。

2）确定性。对算法的描述必须无歧义，以保证算法执行结果是确定的，且符合要求和期望。如果算法对于同样的输入，在相同环境下产生不确定的结果，是不能接受的。

3）输入。一个算法有 0 个或多个输入，以确定运算对象的初始情况。所谓 0 个输入是指算法本身给定了初始条件。

4）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。没有输出的算法是毫无意义的。

5）可行性。算法中有待实现的运算都是基本运算，算法原则上要能够精确地运行，且人们用笔和纸做有限次运算后就可以完成。

从计算机专业角度来看，一个完整的算法应该具备以下三个基本要素。

（1）基本运算和操作

算法的实现形式有多种，但是这些实现形式都具有相同的基本运算和操作。这些基本运算和操作有以下 4 类。

- 算术运算：加减乘除等运算。
- 逻辑运算：或、与、非等运算。
- 关系运算：大于、小于、等于、不等于等运算。
- 数据传输：输入、输出、赋值等运算。

（2）控制结构

一个算法的功能结构不仅取决于所选用的操作，而且还与各操作之间的执行顺序紧密相关。算法的控制结构确定了算法的基本框架，决定了各个操作的执行顺序。算法控制结构有三种，即顺序结构、选择结构和循环结构。

（3）数据结构

在计算机领域，算法的操作对象是数据。一方面，为了方便快速地存取，数据一般是以一些特定的结构进行存储的，另一方面，具体实际问题及其数学模型的结构特点决定了数据之间总是存在着一些特定的逻辑关系或逻辑结构。这些存储结构和逻辑结构统称为数据结构。

1.1.3　算法的常用表示方法

一个算法可以采用多种形式进行表示，常用的表示方法有自然语言、流程图、伪代码、程序设计语言等。

1．自然语言

自然语言方式是指用普通语言描述算法的方法。自然语言方式的优点是简单、方便，适合描述简单的算法或算法的高层思想。例如：

解一元二次方程 $ax^2+bx+c=0$，$a\neq 0$ 的算法，用自然语言表示如下：

1）写出 a、b、c 的值。

2）求出 b^2-4ac 的值。

3）代入求根公式。

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a},(a\neq 0,b^2-4ac\geqslant 0) \tag{1-1}$$

4）输出 x_1，x_2 的值。

5）结束。

自然语言算法描述方式的主要问题是语义模糊，容易产生歧义，很难准确地描述复杂的、技术性强的算法。

2．流程图

流程图是描述算法的一种常用工具，它采用美国国家标准化协会（American National Standards Institute，ANSI）规定的一组图形符号来表示算法流程。算法流程图可以很方便地表示顺序、选择和循环结构。使用流程图表示算法可以避免自然语言的模糊缺陷，且独立于任何一种程序设计语言，有利于不同环境的算法设计。仅从算法描述的角度来看，流程图能够比其他方法更好地描述算法。

流程图的基本组件如图 1-3 所示。

图 1-3　流程图的基本组件

下面是流程图的 3 种基本结构。

1）顺序结构如图 1-4 所示。

2）循环结构如图 1-5 所示（EXP 表示指数运算，下同）。

3）选择结构如图 1-6 所示。

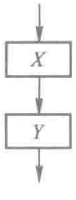

图 1-4　顺序结构

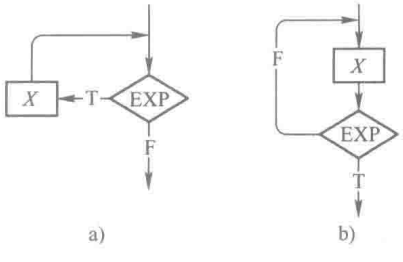

图 1-5　循环结构

a) do while 型循环结构　b) do until 型循环结构

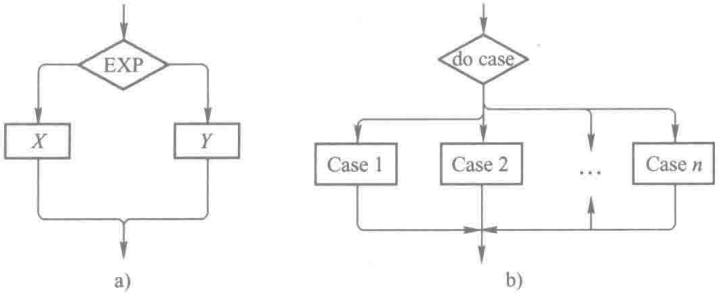

图 1-6　选择结构

a) 双分支选择结构　b) 多分支选择结构

例如，求一元二次方程 $ax^2+bx+c=0$，$a\neq 0$ 的解，采用流程图表示如图 1-7 所示。

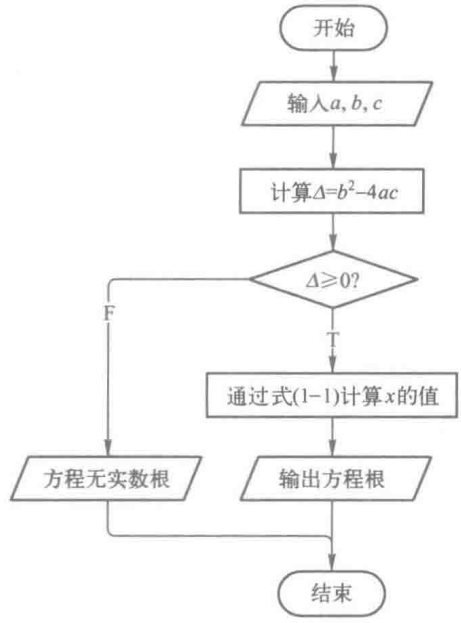

图 1-7　流程图

虽然流程图能够很好地描述算法，但是采用流程图表示的算法，难以阅读、修改，可靠性和可维护性难以保证。同时，算法流程图随意性太大，结构化、层次感不明显，不容易表示数据结构。

3．伪代码

伪代码不是真正的编程语言，而是一种算法描述语言，使用介于自然语言和程序设计语言之间的文字符号来描述算法。使用伪代码描述算法，能够使被描述的算法容易地以任何一种编程语言实现。伪代码可以综合使用多种编程语言中的语法、保留字，甚至会用到自然语言。因此，伪代码结构清晰、简单且易于修改，可读性较好，并且类似于自然语言。

例如，一元二次方程 $ax^2+bx+c=0$，$a\neq 0$ 的求解算法，可用如下伪代码表示：

```
input a , b , c;
    Δ=b²-4ac;
    x1= (-b + sqrt(Δ))/(2a);
    x2= (-b - sqrt(Δ))/(2a);
print x1, x2;
end
```

算法 1-1　一元二次方程的求解（伪代码描述）

4．程序设计语言

使用自然语言、流程图、伪代码描述的算法，计算机是不能直接执行的。为了使计算机能够按照要求工作，必须使用程序设计语言编写相关程序。目前比较流行的程序设计语言有 C++、Java、C、Python 等，当使用不同的程序设计语言描述同一算法时，描述形式可能会存在差异，这是由程序设计语言本身性质所决定的。

一般而言，用计算机程序设计语言描述的算法是最清晰、最严谨的。然而，使用特定程序设计语言描述算法增加了算法设计人员程序设计语言的学习时间，限制了与其他算法设计人员的交流，降低了算法设计的效率。同时，使用程序设计语言描述算法需要更多地考虑程序语言的细节，可能会干扰算法的设计思路。

例如，一元二次方程 $ax^2+bx+c=0$，$a\neq 0$ 的求解算法，用 C++程序描述如下：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
  double a,b,c,d,e,x1,x2;
  cout<<"请输入 a,b 和 c：  "<<endl;
  cout<<"a";
  cin>>b;
  cout<<"b: ";
  cin>>b;
  cout<<"c: ";
  cin>>c;
  cout<<"求解的方程为：
  "<<"*x*x"<<"+"<<b<<"*x"<<"+"<<c<<"=0"
  <<endl;
  a=1;
  d=b*b-4*a*c;
  if(d<0)
   {
  cout<<"There is no x."<<endl;
}
  if(d==0)
   {
     cout<<"There is only x."<<endl;
     x1=x2=(-b)/(2*a);
```

```
        cout<<"x1=x2="<<x1<<endl;
    }
    if(d>0)
    {
        e=sqrt(d);
        x1=(-b+e)/(2*a);
        x2=(-b-e)/(2*a);
        cout<<"x1="<<x1<<endl;
        cout<<"x2="<<x2<<endl;
    }
    return 0;
}
```

算法 1-2　一元二次方程求解（程序设计语言描述）

本书后面各章节的算法将主要采用伪代码进行描述，以利于读者将理论与实际相结合，通过自己动手描述和实现算法，加深对算法的理解。

1.2　算法的效率分析

从实际应用考虑，如果一个算法的运行时间过久，如需要长达数年的时间，那么该算法就不适合用来解决实际问题。同样，一个算法如果需要占用很大的内存开销，导致系统无法为其他程序提供必需的内存资源，也是不能够接受的。因此，对于设计出来的算法，需要对其使用时间和内存空间的效率进行分析，这就是算法的效率分析。分析算法效率的目的主要有两个，即对算法的改进提供思路和在解决某一问题时如何选择最优算法。与算法的普适性、简单性不同的是，可以对算法的效率分析做到精确的定量研究。下面介绍如何对算法效率进行分析。

1.2.1　算法效率的度量

算法效率主要考虑时间复杂度和空间复杂度两个因素，算法的时间复杂度是指算法运行的速度，算法的空间复杂度则是指算法需要的内存空间。对于一个算法，其时间复杂度和空间复杂度往往是相互影响的。当片面要求一个好的时间复杂度时，可能会使空间复杂度的性能变差，可能导致占用较多的存储空间；反之，当片面要求一个好的空间复杂度时，可能会使时间复杂度的性能变差，可能导致占用较长的运行时间。因此，在具体的算法设计中，经常会根据实际问题的需要，用空间换时间，以追求较高的运算速度；或者以时间换空间，以追求较小的内存占用量。

1．问题规模的度量

在很多情况下，对不同的问题规模使用相同的算法，其执行时间会有所不同。例如，有两个人，一人在深圳，另一人在上海，在乘坐相同交通工具的情况下，两个人到达北京所使用的时间是不同的。因此，算法的运行时间和内存占用量，一般都与问题规模有着非常密切的关系。例如，对一个数组进行排序，数组越大，排序的时间就越长。

从算法设计的角度看，需要一个指标对问题的规模进行度量，并将这个指标作为算法的规模输入量。此时，问题的规模主要反映在算法的规模输入量上，算法的效率就成为规模输入量的函数。规模输入量的选择对算法的执行效率往往会有很大的影响，例如，计算两个 n 阶矩阵的乘积，是选择矩阵阶数还是矩阵元素个数作为规模输入量，其算法效率是有很大差别的。因此，确定使用何种指标对问题的规模进行度量，是一个需要认真考虑的问题。

2．运行时间的度量

最简单的方法就是使用时间的标准度量单位来度量算法程序的运行时间，如秒（s），毫秒（ms）等。这种方法的最大缺陷就是它依赖于计算机的运行速度。例如，在一台装有 Intel 4004 的计算机上和一台装有 Intel 酷睿 i7 5960X 的计算机上运行相同的程序，所需要的时间会有很大差异。因此，算法在某台计算机上实现所需要的时间对于其他计算机而言是没有参考意义

的。当然，也可以统计算法的每一步操作的执行次数，用算法操作执行的总次数来度量时间效率。但是，这种方法也是十分困难的，且没有必要。因为算法的具体执行次数常常很难确定，而且我们并不关注那些无关紧要的操作步骤。因此，对于算法的时间效率分析，我们希望找到一个不依赖于上述因素的时间度量方法。

事实上，虽然一个算法中包含了很多种操作，但是决定算法时间效率的是那些很耗费时间的操作。因此，在算法分析中只需关心那些耗时的操作，就可以很好地评价算法的时间效率。这些耗时的操作称为基本操作，它们对算法执行时间占用最大。所以，可以用基本操作的执行次数来度量算法的时间效率。例如，矩阵乘法需要完成两种操作：乘法和加法，对多数计算机而言，乘法比加法更耗费时间，所以选取乘法作为基本操作。

因此，对规模为 n 的算法，可以通过统计其基本操作执行次数来度量算法的时间效率。设 c_{op} 为算法在特定计算机上一个基本操作的执行时间，其值为常量；$C(n)$为该算法需执行的基本操作的次数，则可用式（1-2）来估计算法在该计算机上的运行时间：

$$T_n \approx c_{op} C(n) \tag{1-2}$$

因此，可以采用规模输入量 n 的函数 T_n 来度量算法效率。如果 T_n 伴随 n 的增长而增长得很快，则说明算法效率很低；如果 T_n 伴随 n 的增长而增长得很慢，则说明算法效率很高。

3．算法的空间复杂度度量

算法的空间复杂度是对算法在运行过程中占用存储空间大小的度量，它也是问题规模 n 的函数。算法在计算机内存上所占用的存储空间，包括如下三个方面：存储算法本身所占用的存储空间、算法的输入/输出数据所占用的存储空间以及算法在运行过程中临时占用的存储空间，即指令空间、数据空间和环境栈空间。令 $S(n)$表示算法的空间复杂度，其通过计算算法所需的存储空间来实现，记作：

$$S(n)=O(f(n)) \tag{1-3}$$

式中，n 为问题的规模输入量；$f(n)$为语句关于 n 所占存储空间的函数，O 为求空间复杂度运算符。

根据程序编译运行的情况，可以把一个程序所需的空间分成两个部分：固定部分和可变部分，固定部分通常包括指令和基本数据等所需的空间，而可变部分通常指的是动态分配的空间和递归栈的空间。因此，对于空间复杂度的分析也分为固定部分和动态部分。对于一个特定程序而言，固定部分所需要的空间通常是一个常数值，而动态部分的大小不仅取决于算法本身，还与输入的规模和输入的数据有关。如果输入数据所占空间只取决于问题本身，与算法无关，那么只需分析该算法在实现时所需的辅助单元即可。若算法执行时，所需空间相对于输入数据量而言是个常数，则称此算法为原地工作，即算法所需辅助空间为一个不随问题规模变化的确定的值，此时空间复杂度为 $O(1)$。

通过以上分析，可以看出算法的时间复杂度和空间复杂度之间存在一定联系。此外，算法的所有性能之间都存在或多或少的相互影响、相互制约的关系。因此，设计算法时应当综合考虑算法的各项性能、算法的使用频率、算法处理的数据量的大小、算法描述语言的特性、算法运行的计算机系统环境等因素，权衡利弊、适当取舍，以获得最合适的算法。

1.2.2 函数增长的阶

从前面的算法效率度量中可以看出，无论是时间复杂度还是空间复杂度，算法的效率都是问题规模的某个函数，并且函数值随着问题规模的增长而增长。如果函数值随问题规模的增长而增长得很快，则说明算法效率很低，反之，如果函数值随问题规模的增长而增长得很慢，

则说明算法效率很高。因此，为了更好地分析算法的效率，需要一些合适的方法来描述或度量函数值随自变量增长而增长的速度。如果不能精确地算出函数增长的速度，那么至少也要针对函数增长的不同速度划分出若干性质不同的等级，这就是函数增长的阶。

当输入规模足够大时，精确表示的运行时间中的常系数和低阶项会被输入规模所掩盖。此时，只需要考虑高阶项即可。例如，若复杂度为多项式 x^2+x，x 的复杂度会被 x^2的复杂度“吸收”，则可只考虑最高阶。事实上，在分析和度量算法效率的时候，重点考虑的是当问题规模充分大时的函数值增长的情况。因此，当分析和度量算法效率时，一般都使用渐近效率的概念，即从极限的角度考虑运行时间如何随输入规模的增长而增长。渐近效率更高的算法，对大规模的输入是更适合的。

为了更加准确地阐释和度量算法复杂度随问题规模增长而增长的速度，计算机科学家在算法领域引入了一些函数增长符号，下面简要介绍这些符号。

（1）渐近上界符号 O

函数增长的渐近上界符号 O 的定义如下：

对于函数 $g(n)$，$O(g(n))=\{f(n)$：存在正常数 c 和 n_0，当 $n\geqslant n_0$ 时，使得 $0\leqslant f(n)\leqslant cg(n)\}$。

可以看出 $O(g(n))$表示一个函数集合，函数 $f(n)$的上界是由函数 $g(n)$的常数倍所确定的。事实上，符号 O 在一个常数因子内给出了函数 $f(n)$的一个上界，如图 1-8 所示。

为表示函数 $f(n)$是集合 $f(n)=O(g(n))$中的一个元素，记 $f(n)=O(g(n))$。根据 O 的定义，易知它具有如下运算性质：

1）
$$O(f)+O(g)=O(f+g) \tag{1-4}$$

2）
$$O(f)\cdot O(g)=O(f\cdot g) \tag{1-5}$$

3）若 $g(n)=O(f(n))$，则
$$O(f)+O(g)=O(f) \tag{1-6}$$

4）
$$f=O(f) \tag{1-7}$$

需要注意的是，函数 $f(n)$和 $g(n)$处于 n_0 之前的取值情况是无关紧要的。

例如，可以证明 $f(x)=x^2+2x+1$ 是 $O(x^2)$。

事实上，当 $x>1$ 时容易估计 $f(x)$ 的大小，因为当 $x>1$ 时，$x<x^2$ 且 $1<x^2$，所以只要 $x>1$ 就有 $0\leqslant x^2+2x+1\leqslant x^2+2x^2+x^2=4x^2$。因此，可以取 $C=4$ 和 $k=1$，$f(x)$ 是 $O(x^2)$，即只要 $x>1$ 就有 $f(x)<x^2$。注意，这里不需要使用绝对值，因为当 x 为正数时，等式中的所有函数值都为正数。

（2）渐近下界符号 Ω

类似于前面所讲，符号 O 给出函数 $f(n)$的一个渐近上界，函数增长的渐近下界符号 Ω 则给出函数 $f(n)$的一个渐近下界，如图 1-9 所示。符号 Ω 的定义如下：

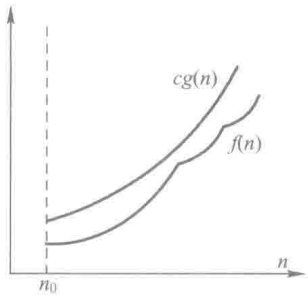

图 1-8　符号 O：$f(n)=O(g(n))$

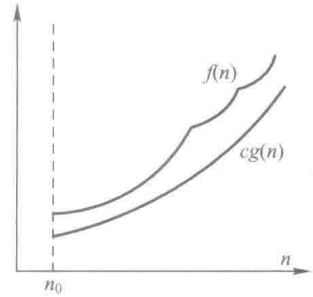

图 1-9　符号 Ω：$f(n)=\Omega(g(n))$