国家精品在线开放课程配套教材

西安交通大学计算机基础系列教材

算法设计与问题求解

主编 乔亚男

副主编 崔舒宁 仇国巍 李波

高等教育出版社

国家精品在线开放课程配套教材
西安交通大学计算机基础系列教材

算法设计与问题求解

主编　乔亚男
副主编　崔舒宁　仇国巍　李波

高等教育出版社·北京

出版说明

今天，从硅谷到北京，大数据、人工智能、云计算、物联网等新技术在不断被谈论，展示着前所未有的生命力。这使得计算机科学与技术在整个产业中占据了越来越重要的地位，从而也对培养具备计算思维能力、能够运用计算机解决各类专业问题的创新型高素质人才提出了迫切的要求。

随着 21 世纪的到来，人类进入了信息社会，计算机和网络成为了人类生活与工作中不可缺少的伙伴和助手。作为信息社会中的技术人员，建立和掌握利用计算机求解各种专业问题的思路和方法，成为了一种必备的能力。计算机科学既是一门独立的学科，又是所有专业研究的基础。《国家中长期教育改革和发展规划纲要（2010—2020 年）》中指出，全面提高教育质量是高等教育发展的核心任务，是建设高等教育强国的基本要求。计算机基础教学承担着所有非计算机专业本科生的计算机能力培养，其教学质量和效果不仅直接影响到学生的计算机能力，在一定程度上也影响到本科教学质量。

教材是教学体系的反映，也是学习者获取知识的主要来源之一。为了进一步提升计算机基础教育质量，培养学生的计算思维能力，高等教育出版社联合西安交通大学计算机基础课程教学团队在深入理解教育部高等学校大学计算机课程教指委发布的《计算机基础课程教学基本要求》基础上，结合多年教学和研究经验，编写出版了这套面向理工类专业、体现信息技术与教育深度融合的新形态“计算机基础系列教材”。整套教材从入门到进阶，并通过最终的《软硬件综合设计指导》，力求形成相对独立的计算机基础教学体系，以培养学生对计算机的认知能力、编程能力和问题求解能力。这套教材总体上具有以下两个方面的显著特点。

（1）整套教材由浅入深、相互衔接、层次分明，反映了一个相对完整的计算机基础知识体系，可满足不同的学习需求。

整套教材分为 4 个层次。第一层次的《大学计算机》以数据的表示、存储、处理、分析为主线，介绍系统平台原理、C 程序设计，并从大数据的视角，概述了数据的组织与分析方法。作为入门级教材，该书将首先帮助读者在理解基础理论基础上，能具备基本的 C 语言编程能力、简单算法设计和数据结构基础。

第二个层次是程序设计进阶，包括《算法设计与问题求解》、《C#程序设计》和《C++程序设计》3 本教材，分别侧重于经典算法描述、可视化编程方法、面向对象程序设计等三个方面，以满足读者在不同方向上的学习需求。

第三个层次的《微机原理与接口技术》在计算机、通信、电子工程、工业自动化和机

械制造等专业学科中具有十分重要的地位。教材采用案例驱动方式，详细描述了微机系统的工作原理和I/O接口控制系统的设计方法，可帮助读者具备接口软硬件的初步设计能力。

第四个层次以“项目设计”能力培养为主题，通过《软硬件综合设计指导》等设计实践类教材，将前三个层次的知识融会贯通，帮助读者建立项目设计的基本思路和方法。

2）教材与在线课程资源双向关联，形成了书网一体化的新形态教材。

该系列教材与即将在“中国大学MOOC”平台发布的“计算机基础系列在线开放课程群”配套，全部教材都有对应的MOOC。通过在教材的关键知识点描述中嵌入二维码，可以从教材直接跳转到线上课堂，使读者在文字阅读的同时能方便地从视频中获得教师对知识点的讲解；通过在线上课堂中加入地址链接，可以实现在视频学习过程中同步阅读到详细的文字描述。同时，借助在线课堂便于动态调整和跟进新技术发展的特点，实现相对稳定的纸质教材与动态更新的数字课堂的深度融合。

计算机基础教学的培养目标是帮助所有非计算机专业的本科生具备利用计算机解决各种专业问题的能力。随着2017年新工科概念的提出，实践能力和创新能力成为关键词。因此，这套教材的编写中，除了关注理论知识的系统和衔接之外，还尽可能注重了实践应用性，让读者能在理解和重复书中运行结果的同时，能够自然地将书中的知识运用到实际的问题求解中。

信息技术是21世纪发展最为迅速的技术，未来这套教材的内容和容量也会随着技术的发展而不断改革和调整。但是，纸质教材的更新永远难以跟上信息技术的发展，这也是努力建设与在线开放课程实现双向关联的新形态教材的初衷。

我们也共同希望这一尝试能成为“互联网+”时代教材改革的风向标，为我国计算机基础教育的改革发展贡献力量。

西安交通大学
高等教育出版社
2018年7月

序

一流本科教育是建设世界一流大学的重要基础和支撑。《国家中长期教育改革和发展规划纲要（2010—2020 年）》中指出，全面提高教育质量是高等教育发展的核心任务，是建设高等教育强国的基本要求。计算机基础教育承担着培养高校所有本科生的计算机基本知识、基本能力的重任，其教学质量和效果不仅直接影响学生的计算机能力，而且也将影响整个本科教学质量。为了进一步推动计算机基础教育的发展，教育部高等学校大学计算机课程教学指导委员会发布了《高等学校计算机基础教学发展战略研究报告暨计算机基础课程教学基本要求》（以下简称《基本要求》），针对计算机基础教学的现状与发展，提出了教学改革的指导思想、知识结构和课程设置。

西安交通大学计算机基础课程教学团队长期坚持教学研究和改革探索，研究成果和教学成效始终处于国内领先地位。近年来，随着大规模在线开放课程在全球的兴起以及国家对在线课程建设的要求，先后建设了 3 门计算机基础 MOOC，其中两门获评首批国家级精品在线课程。在此基础上，课程团队启动建设了由 5 门理论课+1 门综合设计实训课构成的“计算机基础系列在线课程群”，为学习者提供了体系完整、内容先进、逻辑清晰的学习平台。

教材是实现教学要求的重要保证。配合系列 MOOC 建设，团队教师在分析研究国内外相关领域课程体系基础上，结合计算机技术最新发展，编写了这套面向非计算机理工类专业学生的“计算机基础系列教材”。这套教材具有以下三个特点。

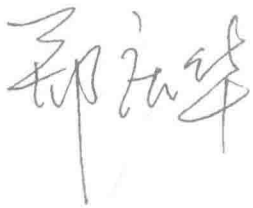
1. 体系完整，内容先进，逻辑清晰。构成了从基础理论到综合设计比较系统的计算机基础教学框架，符合高校理工类专业学生特点和学习需求。整套教材以教指委《基本要求》为指导，以“计算思维能力来自于设计实践”为理念，以“使学习者具备利用计算机求解一般专业问题的能力和对计算机新技术的感知能力”为目标，在《大学计算机》教材中，不仅融入了各种新技术的描述，还从大数据视角，介绍了数据的组织与分析方法，并用较大篇幅介绍了 C 程序设计方法，奠定基本编程能力；在此基础上，分别通过《算法设计与问题求解》、《C#程序设计》和《C++程序设计》等 3 本软件类教材和采用案例驱动式描述的《微机原理与接口技术》，分别介绍了高质量程序设计和 I/O 接口系统设计的思想和方法。最后，通过《软硬件综合设计指导》等实践类教材，介绍了多个典型案例，实现了从理论知识到综合设计的有机结合，帮助读者建立利用程序语言求解问题的能力或初步的接口控制系统设计能力。

2. 线下和线上深度结合。基于“互联网+”思维，实现教材与线上课堂的双向关联，

形成相对稳定的纸质教材与动态更新的数字课堂的深度融合。教材的每段关键知识点中都嵌入了二维码，扫描二维码，可以从教材直接跳转到线上课堂中相应知识点的视频讲解；通过线上课堂中的地址链接，可以在视频学习过程中同步阅读到教材的数字化文字描述。同时，借助 MOOC 教学内容易于动态调整和扩充的特点，实现教材内容的相对稳定和与教材链接的数字课堂资源随时更新，形成了线上线下融合的新形态教材。

3. 编写团队经验丰富，水平高，能力强。团队成员均有十年以上教学和研究经验，对计算机基础教育理论、教学方法和学习需求有非常深入的理解和体会。团队主要教师均获得过国家或省、校级教学成果奖，并先后主编出版过国家“十五”、“十一五”和“十二五”规划教材，具有丰富的教材编写经验。近年来，他们坚持将信息技术融入教育教学改革，结合大规模在线开放课程建设和在线学习行为分析研究，在多个理工类专业教学班中开展了基于 MOOC 的混合教学实践和在线学习过程的跟踪管理，已取得了一些研究成果，在《中国大学教学》等重要教学期刊和 ACM TURC 国际学术会议上发表了数篇研究论文。这套教材也反映了他们多年教学研究和实践的体会。

这套教材将会随着信息技术的发展和高校计算机基础教育的改革进程不断调整，希望各位专家、教师和读者不吝提出宝贵意见和建议，编写团队将认真吸取各方意见，不断改进和完善，为我国计算机基础教育的教材建设和人才培养做出新的更大的贡献。



西安交通大学教授 副校长
教育部大学计算机课程教学指导委员会副主任委员
2018 年 7 月

前言

随着信息技术在各行各业应用的不断深入，具备良好地使用计算机技术解决本专业应用问题的能力，已经成为一个合格大学生必须具备的条件之一。目前在大多数高校，“大学计算机基础”与“计算机程序设计”都是各专业的必修课程，但长期以来，一直有一种现象在困扰着我们：很多系统学习过计算机基础课程并且考试成绩优秀的学生，仍然很难写出一个正确的程序来解决本专业应用环境中的实际问题。究其原因，主要有三点。

首先，在大多数高校，当前的计算机基础课程设置仍显不够。“大学计算机基础”与“计算机程序设计”课程合起来大约 120 个课时，其中上机练习只有 40 个课时，区区 40 个小时，对于高中没有完全接触过编程的学生而言，只能做到刚刚入门。

其次，由于学生的计算机初始水平参差不齐，为了照顾大多数学生的进度，教师通常只能按照较为基础的水平来要求学生，学生只要达到完成课后习题的要求，能够通过考试就好；至于算法效率、编程风格和数据结构等进阶内容，只能浅尝辄止地讲解。

最后，绝大部分高校都将两门计算机基础课程安排在大一，而通常各个专业的学生接触本专业的专业课都在大三，中间一年多没有计算机相关课程，很多学生几乎没有再练习过，自然在专业课需要编程知识时捉襟见肘，难以应付了。

为了解决这个问题，亟需在传统的计算机基础课程与专业课程之间搭建一座桥梁，做好“做预设的程序习题”到“用程序解决未知问题”之间的衔接。《算法设计与问题求解》就基于如上的需求，旨在为计算机素质教育和计算机教学改革提出和建设一种新概念程序设计教材，围绕应用环境中实际问题的求解过程来阐述和讲解程序设计思想方法和相关技术知识，向学生展示如何设计和选择合适的数据结构来表示实际问题中的处理对象，如何把一个实际问题转化成一个程序可计算的逻辑模型以及如何考虑程序运行的效率来满足问题求解对时间的要求等。

通过学习本书，学生应该理解和掌握经典算法和数据结构，了解高级算法的原理；应该具备结合本专业实际应用，设计出高效算法和数据结构的能力；应该具备利用开源平台和工具软件，快速实现应用原型的能力。本书全部内容编排为 8 章，具体如下。

第 1 章，绪论部分，介绍本书的总体情况，了解算法的相关概念，对本书涉及的一些编程语言细节和函数库进行讲解。

第 2 章，熟悉数论相关算法，会用数值法求解基本数学问题，如多项式四则运算问题、多项式插值问题、非线性方程求解和线性方程组求解问题等。

第 3 章，掌握线性表的定义及其运算；顺序表和链表的定义、组织形式、结构特征和

类型说明以及在这两种表上实现的插入、删除和查找的算法。掌握栈和队列的定义、特征及在其上所定义的基本运算，在两种存储结构上对栈和队列所施加的基本运算的实现。

第 4 章，掌握树的定义、性质及其存储方法；二叉树的链表存储方式、结点结构和类型定义；二叉树的遍历算法；哈夫曼树的构造方法。掌握图的基本概念及术语；图的两种存储结构(邻接矩阵和邻接表)的表示方法；图的遍历(深度优先搜索遍历和广度优先搜索遍历)算法；最小生成树的构造。

第 5 章，掌握贪心算法的基本概念；了解贪心算法的性质和优缺点；了解最优解/次优解与计算性能之间的关系；图的最小生成树生成算法：Prim 算法和 Kruskal 算法。

第 6 章，掌握动态规划的基本概念，了解动态规划算法的基本思想、适用情况以及求解基本步骤；了解最优性原理；典型动态规划问题的解决：0-1 背包问题和最长公共子序列问题。

第 7 章，了解遗传算法的概念和设计方法；函数最值和旅行商问题的求解。

第 8 章，了解人工神经网络的概念，了解感知器算法和 BP 算法。

本书第 1、6 章由乔亚男编写，第 2、3 章由仇国巍编写，第 4、5 章由崔舒宁编写，第 7、8 章由李波编写。

本书已在西安交通大学相关专业试用 3 年，试用效果良好，达到了最初的设计目的。使用本书的学生均已学习过“大学计算机基础”课程，有了一定的编程基础，亟需趁热打铁地进一步提高软件开发水平。经过学习，学生的平均程序驾驭能力(可定义为不调试一次性编写多长的程序不会引起代码混淆)已经从 30～40 行提升到了 70～80 行，同时接触的程序设计目标已经从“习题”过渡到了“实际问题”，兴趣提升了，水平也提升了，也更能满足工科专业高年级专业课对学生计算机水平的要求。

受篇幅、时间以及作者水平等限制，书中不妥之处，恳望广大读者批评指正。

编者

2018 年 6 月

目录

第 1 章

绪论

在当今科技社会中，人们慢慢地接受了这样一个事实：计算机可以解决非常多的问题。但计算机究竟能解决什么样的问题，是用什么方法解决的？在搞清楚这个问题之前，应当首先知道这样一个事实：通常意义下的计算机是没有思维的机器，它并不能自主解决问题，而只能机械地执行指令，而这些指令是人设计出来的。机器能解决什么样的问题，怎样解决，需要靠人教给它。机器的优势是速度快、容量大、严格而精确。人应当充分发挥计算机的优势，避免劣势，让计算机用正确的方法做正确的事。哪些事情是“正确的事”？这取决于计算机的特点。怎样的方法是“正确的方法”？这正是本书要讨论的内容。人们使用计算机来解决问题的方法称为算法（algorithm）。

1.1 算法的概念

Mooc 微视频：算法的概念

1.1.1 从计算机的优势和劣势谈起

计算机的第一个优势：速度。

很多时候人不是不知道怎么解决问题，而是嫌方法太麻烦。有这样一道题目：用 1、2、3、4、5、6、7、8、9 这 9 个数字拼成一个 9 位数（每个数字恰好用一次），使得它的前 3 位、中间 3 位、最后 3 位的比值是 1 : 2 : 3。例如 192 384 576 就是一个合法的解，因为 192 : 384 : 576 = 1 : 2 : 3。

有一个办法是奏效的：列举出所有可能的 9 位数 123 456 789、123 456 798、123 456 879、…、987 654 321，一个一个检查是否符合条件。理论上来说这是可行的，可几乎没有人愿意这么做，因为这样的 9 位数有 362 880 个，即使人可以每秒检查一个数（这已经是很快的速度了！），也需要 100 个小时。计算机就不一样了，它的运算速度很快，同样是这个“笨方法”，人需要算 100 小时，计算机只需要不到 0.1 秒。学习过程序设计的学生可以轻松地在最多半个小时之内编写一个程序，然后花 0.1 秒运行一下，得到 4 个符合要求的结果：192 384 576、219 438 657、273 546 819、327 654 981。

计算机的第二个优势：记忆。

在计算机未损坏的情况下，它所存储的内容不会消失、不会改变，永远得到精确的保存，而且能够轻易地读写和复制。这一点人是做不到的。人为什么要在合上书之前夹一枚

书签？听课时为什么要做笔记？都是避免把曾经记住的东西忘掉。“记忆”分成两部分，用计算机术语来说，“记”是写（write），“忆”是读（read）。计算机内部有很多可供读写的存储单元，而读和写就是存储单元的基本操作。

计算机的第三个优势：精确。

神速计算加上永恒记忆，并不是计算机强大能力的全部。有一些任务，不需要神速计算，也不需要永恒记忆，可是如果不借助于工具，人就是做不好。“失之毫厘，谬之千里”说的就是这样的情况。在地上划一根长长的线段，然后闭上眼睛尽量沿着线走，你能笔直地走到线的尽头吗？也许你感觉你走得非常直，可是在走过相当长的距离后，几乎所有人都会走歪。看着你的手表，当秒针穿过 0 秒时闭上眼睛，默数 5 分钟。睁开眼睛后对照你的手表，真的是 5 分钟吗？人做事受感觉的影响，而感觉往往是不准确的。计算机可以严格按照指令工作，因此只要有了可操作的数学定义，这类事情用计算机完成很合适。

但是计算机作为一种只能机械接受人类指令的机器，当然也有它的限制与劣势。

计算机的速度快，并不是说可以给它无限大的工作量。同样的问题，解决方法不同，计算机的工作量也不同。为了计算 $1+2+3+\cdots+n$，可以让计算机做 $n-1$ 次加法，也可以让计算机直接计算 $n(n+1)/2$，即 1 次加法，1 次乘法和 1 次除法。刚才两种算法（是的，“连加 n–1 次”和“计算 $n(n+1)/2$”就是两个算法！）的差别告诉我们：同一个问题可以有多个算法，有的运算量大，有的运算量小。具体使用哪个算法由人说了算，计算机只会按部就班地执行。

计算机的容量大，也不是说可以要它保存无限多的东西。同样的问题，解决方法不同，需要保存的东西也不同。小学老师可以让我们背“九九乘法表”，然后用它算 12×34 或者 56 ×78 这样的两位数乘法，同样也可以让我们把所有两位数乘两位数的积全部背下来。这样的话，12×34 根本就不用算了，直接可以背出来。速度提高了，可是需要记更多的东西。小学教育的选择是，少记一点，宁愿每次多花点时间算。计算机也一样，不同的算法可能使用不同大小的内存，不过速度也许会有差异，这需要权衡。

计算机是精确的，缺少人类所具有的模糊能力，没有专门的认知系统，也不会产生灵感或者闹情绪。精确可以带来好处，也可以引起麻烦。计算机能执行的指令，简单来说就是四则运算、逻辑运算、存储器操作和输入、输出等。它很难解决如“这幅画漂亮吗？”这样模糊的问题。人可以回答，但却很难用精确的语言描述出他为什么这样回答。最后需要提醒读者注意的是，有很多算法对于人来讲是直观的，但是转化成计算机可以执行的运算指令却需要花一番功夫。例如，要求一个奇形怪状的容器能装多少水，计算机算起来比较麻烦（而且充满了数学味，不直观），而人却可以简单地做个实验就可以得到结果。要把一个东西塞到瓶子里，人可以往各个方向使劲塞，而计算机却很难使用这种“没头没脑”的“算法”。一句话：人和计算机是不同的，这个差别不只是速度、记忆容量和精确性的问题。

▶ 1.1.2 问题和算法

速度快、容量大和结果精确是计算机的硬件条件。为了解决问题，只有硬件条件远远

不够。人们需要针对各种问题设计不同的算法，并把算法转化为计算机可以直接运行的程序（program）。

算法的好处在于通用性。当针对一个问题（problem）设计出算法后，该问题的多个实例（instance）都能被解决。换句话说，一个加法程序不仅能计算 1 + 1，也能计算 100 + 100 和 1 000 + 1 000。另一方面，如果一个算法只能求解问题的一个实例，它的能力是十分有限的。一般情况下只需要把这个实例和对应的答案记录下来，算法本身就没有意义了。

一个问题通常有多个输入参数，当各个输入参数都确定时，该问题的实例也确定下来了。问题求解结束以后，答案放在输出参数里。可以给一个问题下一个类似这样的定义：

问题：正整数加法
任务：计算正整数 a 和 b 的和 c
输入：a、b（a 和 b 是不超过 10 000 的正整数）
输出：c（c 是 a 和 b 的和）

有了问题定义，算法设计者很明确地知道了任务，而用户也知道如何使用该算法（即如何提供输入参数，输出参数代表什么含义）。一个不完整的例子如下：

问题：找最大元素
任务：找出 n 个数中的最大元素

这个算法该如何设计、如何使用呢？不知道。如果算法的设计者认为输出应当是“第几个数最大”（输出序号），而使用者认为输出的是“最大数等于多少”（输出数值），那么就会出现问题。一个好的问题定义还应当包括一个或多个输入输出的例子，进一步澄清任务和输入、输出规定，例如：

问题：全等判定
任务：判断 n 个数是否全部相同
输入：n 个数
输出：如果全部相同，输出 Yes，否则输出 No
样例输入 1：5 3 3 6
样例输出 1：No
样例输入 2：7 7 7 7 7 7
样例输出 2：Yes

通过例子，相信读者一定会非常明确这个问题的任务和输入、输出格式了。

这个“全等判定”问题显然不是一个困难的问题。如果由人来做，只需要“扫一眼”就可以了。是不是可以写这样的算法呢？

算法：扫一眼算法

把所有数“扫一眼”。如果刚才曾发现两个数不一样，输出 No，否则输出 Yes。

这样的算法是无法被计算机执行的（但是可以被人轻松地“执行”），因为这个“算法”里的话十分模糊，“扫一眼”、“发现”等词语都没有被精确定义（例如，有可能眼花了没发现），计算机自然无法执行。

那么哪些算法是计算机可以执行的呢？这里给出一个看起来正确的算法：

算法：相邻枚举算法

从前到后考察每一个输入参数，如果它和紧跟其后的数不相等，输出 No；如果始终没有发现不相等的情况，输出 Yes。

该算法的思路是，如果每个数都和“后边”的数相等，那么所有数相等，否则不全相等。思路没有问题，但细节存在不少问题：首先，如果很多数和它后边的数都不相等，会输出很多“No”，不符合输出规定；其次，最后一个数的“紧跟在它后边的数”是什么？没定义。这些问题说明：把算法写成计算机可以准确执行的形式其实并不容易。

▶ 1.1.3 什么是算法

广义地说，算法是按部就班解决一个问题或完成某个目标的过程。算法设计是一个古老的研究领域。自古以来，人们总是对发现更好的目标求解方法充满兴趣，不论是取火、建造金字塔还是对邮件进行排序。而计算机算法的研究当然是一个新的领域。一些计算机算法采用的方法早在计算机发明之前就存在，但大多数计算机算法的设计需要新的方法和技术。首先，告诉计算机诸如“察看小山，如果发现敌情就拉响警报”是不够的。一台计算机必须了解“察看”的确切含义，知道如何识别敌情，懂得如何拉响警报。一台计算机可接受的指令应当是定义明确、长度有限的基本操作序列。将通常的命令转换为计算机可以理解的指令是一个困难的过程，而该过程就是编程。

计算机上的编程，所需要的不仅仅是将那些为人所理解的命令转换为计算机可以理解的语言。在大多数情况下，程序员必须设计出完全崭新的算法来求解问题。计算机能处理数十亿、数万亿比特单位的信息，能在一秒内完成数百万条基本指令。在这个数量级上进行算法设计是一种崭新的实践，有很多方面会与人们的直觉相反，因为人们通常只对自己能感知的事物进行思考。遗憾的是，一些能很好解决小问题的程序在处理大问题时就变得很糟。因此当进行大规模计算时不要忽视算法的复杂度和有效性。

问题的另一个方面是，人们在日常生活中所执行的算法一般不太复杂，执行的次数也不太频繁，因为回报非常低，通常不值得耗费太多精力来开发完美的算法。例如，在日常生活中，从超市采购回家，打开杂货袋放置食品的过程就可能不是最有效的，有效的过程应该考虑杂货袋中的食品以及橱柜的结构，但很少有人会去想这件事，更不会有人为它去

专门设计算法。反之，进行大规模商业包装和拆装就必须有一种好的方法。另一个例子是修剪草坪，可以考虑如何使来回次数最少、修剪时间最短，或者到垃圾堆的往返距离最短。除非你非常讨厌修剪草坪，否则就不会花一个小时的时间来思考如何使修剪时间缩短几分钟这样的问题。但另一方面，计算机可以处理非常复杂的任务，而且同一任务往往必须执行多次，因此值得花费时间来设计有效的方法，就算最终得到的方法更加复杂或难以理解也无妨，因为潜在的回报是巨大的。

对与直觉相悖的大型问题求解算法的需要以及了解这些算法的复杂度是学习算法这门学科的动力所在。首先，必须认识到直观的方法并不总是最好的，寻找更好的方法有时非常重要，为此，程序员必须学习新的方法和技术。本书给出了多种算法的说明。然而，就像要成为一个好棋手仅仅靠记住许多棋局是不够的一样，就算你学习了大量的算法也不足以应付各种情况，因此必须了解算法背后的原理，必须知道如何应用它们，当然更重要的是，何时应用它们。

算法的设计和实现类似于房屋的设计和建筑。先从最基本的概念出发考虑房子的修建。设计师的工作是给出满足要求的规划；工程师的工作是确认规划是正确且可行的（以保证房子不会在短时间内倒塌）；建设工人的工作是按照规划建筑房子。当然，在所有阶段都要考虑造价并进行分析。每个阶段的工作是不同的，但它们又有所联系甚至交织在一起。算法的设计也如此，先考虑基本的思想和方法，然后做出规划。必须证明规划是可行的并且代价是可接受的。最后的工作是在具体的计算机上实现。简单地说，可将过程分为 4 步：设计、正确性证明、分析和实现。再次，每步都不同但又相关，每步都不能不顾及其他而在真空中完成。当然，很少能按照线性次序走过历程。在构造算法的每个阶段都可能出现困难。由此可能需要修改设计，还可能需要再次进行可行性证明、调整代价以及改变算法实现。

Mooc 微视频：算法设计的要求

1.2 算法设计的要求

一个算法是一组有穷的规则，这些规则给出求解特定类型问题的运算序列；但除此之外，一个算法还有 5 个重要特征，只有具有以下所有性质才能称为是一种解决特定问题的算法。

（1）正确性（correct）。也就是说，它必须完成所期望的功能，把每一次输入转换为正确的输出。一个算法可能包含零个或多个输入，一个或多个输出。

（2）具体步骤（concrete steps）。一种算法应该由一系列具体步骤组成。“具体”意味着每步所描述的行为对于必须完成算法的人或机器是可读、可执行的。每一步必须在有限的时间内执行完毕。因此，算法好像给出了通过一系列步骤解决问题的“工序”，其中的每一步都是人们力所能及的，是否能够完成每一步依赖于谁或者什么来执行这个工序。例如，烹饪书中关于小甜饼的制作方法对于指导一位厨师来说很具体，但是对于一个自动小甜饼加工工厂却是不够的。

（3）确定性（no ambiguity）。一个算法的每个步骤都必须精确地定义；要执行的动作每一步都必须严格地和无歧义地描述清楚。但是如果直接使用自然语言来描述算法会带来很多问题，有可能读者未能准确地理解作者的意图。为了克服这个困难，作者设计了用于描述算法的形式地定义的程序设计语言或计算机语言，其中每一个语句都有非常确定的意义。本书的许多算法都将以 C 语言或受到一定限制的 C++语言给出，在一种计算机语言下一个计算方法的表达就叫做一个程序。

（4）有限性（finite）。一个算法在有限步骤之后必然要终止。如果一种算法的描述是由无限步组成的，就不可能将它写出来，也不可能将它作为计算机程序来实现。大多数算法描述语言均提供一些实现重复行为的方法，比如 C 语言中的 while 和 for 循环结构。循环结构具有简短的描述，但是实际执行的次数由输入来决定。

（5）终止性（terminable）。算法必须可以终止，即不能进入死循环。

下面把算法的概念同菜谱的概念做一下比较。一个菜谱大抵具有有限性（尽管有人说，心急水不沸），输入（鸡蛋、面粉等）以及输出（速食便餐等）等性质，但是众所周知它缺少确定性。经常有这样的情况，菜谱的指导是不确定的："加少许盐"。"少许"被定义为"少于 1/8 茶匙"，或许盐是明确地定义的；然而，把盐加在哪儿呢?是在顶上还是在边上?像"轻微地搅拌直到混合物变碎为止"或者"在小深平底锅中把法国白兰地酒加热"这样一些指导对于训练有素的厨师可能是十分适当的说明，但是一个算法必须被描述到这样一种程度，就是即使一部完全不了解任何人类常识的计算机也能遵循它。

应当说明，就实用而言，有限性的限制实际上是不够强的。一个有用的算法不只要求步骤有限，而且要求非常有限的、合理的步骤数。比如，严格来说，所有的密码都可以使用穷举算法来破译，但如果密码位数足够长，足够无规律的话，即使用世界上最快的计算机，也需要动辄成千上万年的时间来穷举，这种"有限"对于实践是没有意义的。在实践中，不仅要算法，而且还要在某种不明确定义的意义下的好算法。

1.3 算法效率的度量

算法设计出来以后，应该先进行分析。一般来说，需要估计算法的时空开销，即需要运行多长时间？需要多大内存？这是个很实际的问题，因为如果一个程序需要运行 100 年，或者需要很多内存，那么就算它能得出正确的答案也没有用，因为人们等不了这么久，或者买不起这么多内存。同一问题可用不同算法解决，而一个算法的质量优劣将影响到算法乃至程序的效率。算法分析的目的在于选择合适算法和改进算法。一个算法的评价主要从时间复杂度和空间复杂度来考虑。

1.3.1 时间复杂度

一个算法执行所耗费的时间，从理论上是不能算出来的，必须上机运行测试才能知道。但问题在于编程之前不可能精确地知道一个算法要运行多长时间，因为它还没有被转化成