破解程序设计的奥秘

从算法到程序（第2版）

From Algorithm To Program 2nd Edition

徐子珊 编著

清华大学出版社

从算法到程序（第2版）

From Algorithm To Program 2nd Edition

徐子珊 编著

清华大学出版社
北京

内 容 简 介

本书第 1 章讨论算法设计、分析的基本概念。第 2 章讨论算法设计中最常用的几个数据结构，包括链表、栈、队列、二叉搜索树、散列表等。第 3 章讨论了算法设计的两个基本策略：渐增策略与分支策略。第 1～3 章的内容，为读者阅读本书以后的内容奠定了基础。第 4 章讨论几个代数计算的基本问题及其算法，包括矩阵运算、解线性方程组、多项式运算等。第 5 章讨论几个关于计算几何的基本问题及其算法，包括线段的相交判断、平面点集的凸包计算、最邻近点对问题等。第 6 章讨论了关于整数运算的基本问题，包括大整数的表示与运算、最大公约数计算、模运算、素数判定及整数因数分解等。第 4～6 章的内容为读者深入学习解决各种复杂问题奠定了解决数学计算问题的基础。第 7～9 章分别用回溯策略、动态规划策略及贪婪策略研究、解决计算机应用面临的最普遍、最典型的组合优化问题。第 10 章讨论图的搜索算法及其应用，包括深度优先搜索、拓扑排序、有向图的强连通分支计算、关节点计算、广度优先搜索、网络最大流及二部图的最大匹配等问题。第 11 章讨论了几个文本搜索的有趣算法，包括著名的 KMP 模式匹配算法、线性时间计算字符串中最长回文子串的 Manacher 算法、用动态规划策略寻求字符串中指定模式的最佳近似匹配的算法。对所有的的经典算法及数据结构，书中给出 C 语言的实现函数，形成一个通用的函数库，并详尽地加以解析。伴随各种算法的设计、分析及程序实现，书中给出了丰富多彩的应用问题及其解决方案的讨论，并给出了完整的程序代码。所有程序代码都经过反复调试，第 12 章介绍这些代码的使用方法。所有代码都以网络资源的方式提供给读者，访问下载地址为 www. tup. com. cn。

本书无论是对初学算法及程序设计入门的大学生读者还是对已经在职场打拼多年的程序员并有提高自身理论修养及技术水平愿望的读者都有开卷有益的意义。

图书在版编目（CIP）数据

从算法到程序/徐子珊编著. —2 版. —北京：清华大学出版社，2015
ISBN 978-7-302-40076-9

Ⅰ. ①从… Ⅱ. ①徐… Ⅲ. ①算法理论 Ⅳ. ①O141. 3

中国版本图书馆 CIP 数据核字(2015)第 089616 号

责任编辑：白立军
封面设计：傅瑞学
责任校对：白　蕾
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www. tup. com. cn，http://www. wqbook. com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup. tsinghua. edu. cn
质量反馈：010-62772015，zhiliang@tup. tsinghua. edu. cn
课件下载：http://www. tup. com. cn，010-62795954

印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：37. 75　　**字　　数**：918 千字
版　　次：2013 年 3 月第 1 版　2015 年 6 月第 2 版　　**印　　次**：2015 年 6 月第1 次印刷
印　　数：1～2000
定　　价：69. 50 元

产品编号：061661-01

第 2 版前言

本书第 1 版已经面世近 2 年了。承蒙读者厚爱及清华大学出版社的大力支持，遂有了今天第 2 版的问世。

根据广大读者的意见反馈，在第 1 版的基础上，除对原有内容中所含明显错漏之处进行修改以外，第 2 版增加了关于文本搜索的一些有趣的算法，包括著名的 KMP 模式匹配算法、线性时间内计算给定字符串中最长回文子串的 Manacher 算法和文本串中模式最佳近似匹配的动态规划算法。所有这些算法都涵盖于第 11 章中。考虑到原来的第 11 章介绍了验证运行本书各章应用问题程序时需加载文件等细节，这对喜欢动手的读者来说是很有帮助的，所以保留了原来的内容并将第 11 章讨论的 3 个应用问题程序的运行加载信息也补充了进去，作为第 12 章。所有这些添加、改动都是为了对读者阅读本书有所帮助，并且能通过对本书的阅读能让更多的年轻朋友在信息时代具有良好的计算思维能力和操控计算机的能力。

网络已经成为人们获取信息、数据的最方便快捷的工具了。本书第 1 版中源代码是以传统的光盘形式提供给读者，本意是方便读者随手可用。第 2 版将以网络资源形式提供给读者，具体的访问地址是 www. tup. com. cn。

为使作者和读者之间更方便、直接地交流沟通，作者的 QQ 号及空间地址公布如下。

QQ 号：513410359。

空间地址：user. qzone. qq. com/513410359? ptlang=2052。

再次感谢清华大学出版社的白立军先生，没有他的支持和帮助，无论是本书的第 1 版还是今天的第 2 版都不会如此顺利地送到读者的面前。

徐子珊

2015 年 3 月

第 1 版前言

学科的基本问题和基本方法是学科方法论的基本内容。计算机能解决的仅仅是计算问题而已。什么是计算问题？有哪些典型的计算问题？如何描述计算问题是计算学科的基本问题之一，也是计算机应用的前提。将问题与数据加以形式化表示，并设计解决计算问题的算法，评价算法的运行效率是计算学科的基本方法。本书的每一章都围绕一类或一个计算问题的形式化描述和算法及其分析展开。在开卷之前，先粗线条地向读者描述一下本书。

计算问题来自现实世界，现实世界五彩缤纷，计算问题多种多样。本书按典型计算问题的分类来组织各章内容，包括计数问题、代数计算问题、计算几何问题、数论问题、组合优化问题和图的搜索问题。

计数问题是最古老的但也是人类生活须臾不能离开的计算问题，特别是在现代科技与工业领域存在大量的计数问题。用计算机快速解决计数问题是实至名归。第 1 章讨论解决计数问题的基础是加法原理和乘法原理的应用。

为了让读者清楚地看到数据组织方式对算法设计方法及算法运行效率的影响，在第 2 章中浓缩了关于线性表、二叉树、散列表等最基础的几个数据结构。

数学中的计算问题更是比比皆是。数学问题的算法，如解线性方程组、计算多项式的变换、线段之间的位置关系等也是很多信息处理应用问题中经常要用的基本操作。本书用第 4～6 章的篇幅讨论代数、几何及数论中的典型计算问题的算法，所用的方法是第 3 章中介绍的渐增性策略和分治策略。

在多个可能解中寻求最优解的组合优化问题是计算学科面对的最典型的问题，因为人类的活动几乎都涉及资源的竞争，而有资源竞争就会产生组合优化问题。本书用第 7～9 章的内容来讨论组合优化问题的解决方法。讨论是按从大到小收缩解空间的线索展开。从无约束的回溯策略开始，到加上最优子结构性质及子问题重叠性质后的动态规划策略，解空间越小，算法效率越高。笔者试图以这样的全方位的讨论组合优化问题解决方法的形式，引导读者深入理解启发式解题思想方法。

在第 10 章讨论一个描述应用问题的重要数学模型——图。重点讨论图中顶点的搜索算法。利用搜索算法，讨论了诸如拓扑排序、关节点计算、网络流等几个经典的关于图的应用问题。

算法研究历史悠久。然而今天，算法理论研究落脚点在于指导计算机程序设计实践。本书讨论的每一个经典算法，均用 C 语言写成了通用的功能函数，并用这些函数解决了一系列有趣的应用问题。第 11 章汇总了这些函数的原型声明及数据结构的定义。

本书写作上除了上述的内容组织形式上的特点外，还用心于以下几点。

1. 理论严谨，语言规范

对每一个问题的算法，从问题的分析开始，包括思路的发展，算法的描述，正确性证明，运行时间的计算都加以详尽讨论，让读者能体验到计算学科的科学严谨性。算法设计与分

析以理论繁复著称。笔者试图以朴实的文字和平和的阐述展示对问题的分析,算法步骤的思考和运行时间估算。在确保科学性、正确性的前提下尽量使用与生活语言相近的词语而避免使用过多生僻的专用术语,让读者在阅读中感受本书语言的自然亲切。

2. 理论与实践互动

笔者对每一个理论算法都给出现有技术的程序实现,用以验证理论算法的正确性。虽然此前已经在理论上证明了算法的正确性,但通过实现了的程序的正确运行进一步证明理论是可行的。并且,算法的程序实现及对测试数据的调试运行能使读者深入理解算法的思想及其中细节微妙之处。对书中的每一个经典算法,均精选了1～2个应用问题,或说明如何直接调用算法解决该问题,或说明如何运用算法设计的思想解决问题。问题均选自ACM/ICPC的赛题或北京大学的网站http://poj.org/problemlist。

3. 小步推进,深入浅出

要设计一个算法来解决计算问题往往是比较复杂的。对复杂问题分析以及设计解决问题的算法并对其进行分析,进而实现为程序难免行文比较冗长。为减轻读者阅读疲劳,在保证内容完整性的前提下,适当地将问题分析、算法设计分析以及程序实现复杂过程按一定的内部逻辑分解成若干部分,一步一个小标题。读者可依次一步一步连续阅读,也可分多次,每次阅读一个部分。阅读时可通过小标题明确自己在整个过程中的那一部分,又不失对全局的掌控。

4. 图文并茂,生动形象

算法的基础是数学,数学讲的是逻辑思维。然而,逻辑思维并不排斥形象思维,形象思维有时可为逻辑思维深入推进助力。为帮助读者快速且正确地理解抽象概念,或思想方法,或微妙的技术细节,书中在适宜的地方插入很多精心绘制的插图。通过这些插图读者可对书中相应的文字或符号表述的理论、方法或技术内容有生动、形象的认识。

5. 通用代码,便于引用

本书中对所有算法的程序实现并非简单的代码堆砌。笔者对所实现的每一个C函数参数与返回值,数据与变量的设置及关键代码都进行了详尽的解析。并且将大多数算法和数据结构写成通用的代码,以光盘的形式向读者提供类似于C++的STL或Java的Collection Framework的通用库,便于读者在工作中或生活中需要时引用。本书算法中的伪代码的编写规范参照《Introduction to Algorithm》一书中的体例。

为方便选用本书作为算法课程教材的教师朋友使用,随书光盘中提供了PPT格式的课件。

徐子珊　记于山城重庆

2012年10月

目　录

第1章　计算问题

1.1　计算问题及其算法

1.1.1　计算问题及其描述

众所周知，计算机并不能解决所有问题。事实上，计算机只能解决一类称为“计算问题”的问题。所谓**计算问题**，指的是问题中所涉及的事物或其属性可用数据加以表示——称为输入数据，问题的解也可以表示成数据——称为输出数据，并且可以在有限步基本计算（算术运算、逻辑运算以及数据的暂存等）后，将特定的输入数据转换成正确的输出数据的问题。解计算问题就是将输入数据转换成正确的输出数据的过程。

利用计算机解决现实问题，首先需要将问题抽象成计算问题，也就是提炼出问题的输入数据和研究问题的解的数据特性。将问题表示为输入与输出的描述。例如，在一系列数据中查找特定值元素的查找问题可以描述如下。

输入：n 个数构成的序列 $A=<a_1,a_2,\cdots,a_n>$，特定值 x。

输出：若序列 A 中存在元素 $A[i]$，其值等于 x，返回从左到右的第一个值为 x 的元素的下标 i。否则，返回 -1。

例如，在线性表 $A=<3,6,0,4,1,7,9,5,2,8>$ 中查找特定值 x 的元素。图 1-1(a)为查找值为 $x=1$ 的元素，从 $A[1]$起依次要做 5 次检测，第 1 次找到值为 1 的元素。图 1-1(b)为查找值为 $x=11$ 的元素，从 $A[1]$起依次检测完所有元素（做 10 次检测），没有找到值为 11 的元素——最坏情形。图 1-1(c)为查找值为 $x=3$ 的元素，从 $A[1]$起仅做一次检测就找到值为 3 的元素——最好情形。

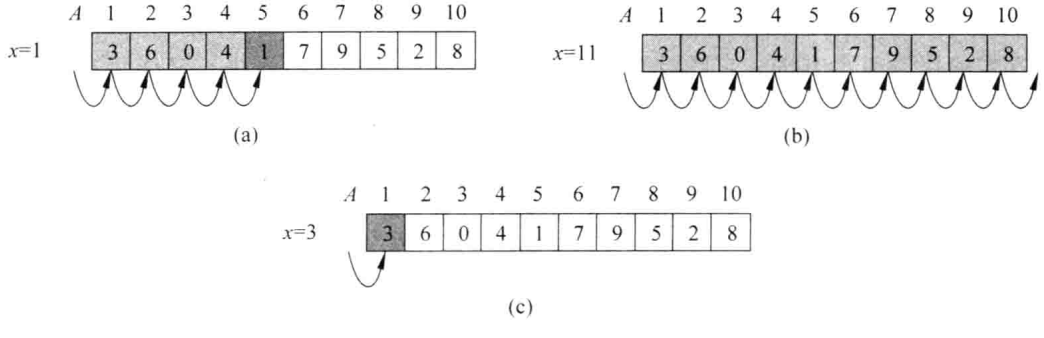

图 1-1　在一系列数据中查找特定元素

1.1.2 算法及其描述

将问题描述成其输入与输出后,就要考虑按什么样的顺序安排有限步的基本计算,将输入数据转换成输出数据,这个过程称为算法设计。安排好的计算步骤称为解决计算问题的**算法**。一个算法对问题输入的任何特定数据都能得到正确的输出数据,则称算法是正确的。对计算问题设计正确的算法,需要人们对问题有深入地理解,利用已有的数学和相关学科的科学知识以及生活常识,设计出从输入数据到输出数据的计算转换过程。例如,对上述在序列中查找特定值的问题,人们可以用如下**线性查找**算法。

解决计算问题的算法可以用各种方法加以描述。常用的有自然语言描述法、流程图描述法和伪代码描述法。用自然语言描述算法优点是表达能力很强,表述方便。例如,解决上述线性查找算法可用自然语言描述如下。

从序列的起点元素开始,依次检测元素的值是否等于 x。直至某个元素 a_i 的值等于 x,停止检测,返回 i。若检测完所有的元素(超过终点)未发现任何元素的值等于 x,则返回 -1。

用自然语言描述算法虽然方便,但有一个致命弱点:自然语言的字句可能存在歧义,即一个字句可以有不同的理解,这在描述算法时是不允许的。用**流程图**描述算法可以避免这一缺陷。仍然以线性查找算法为例,用流程图描述如图 1-2 所示。

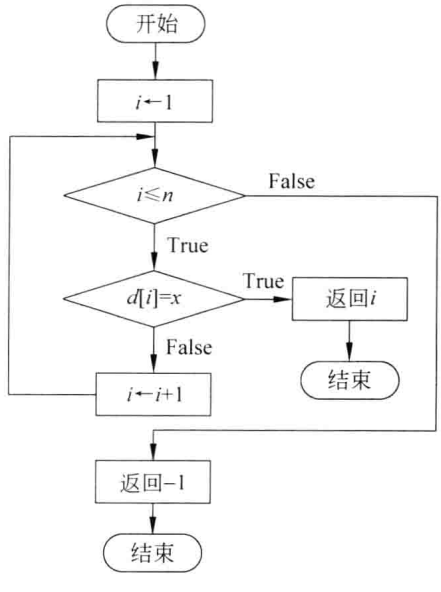

图 1-2 描述线性查找算法的流程图

用流程图描述算法克服了自然语言描述法的字句歧义性缺陷,且直观易读,但所需篇幅很大,当所要描述的算法流程比较复杂时,使用起来就不太方便。

比较实用的描述算法的工具是**伪代码**。这是一种有着类似于程序设计语言的严格外部语法(用 **if-then-else** 表示分支结构,用 **while-do** 或 **for-do** 表示循环结构),且有着内部宽松的数学语言表述方式的代码表示方法。它既没有歧义性的缺陷(严格的外部语法),又能用高度抽象的数学语言简练描述操作细节。以序列的线性查找算法为例,用伪代码描述如下。

```
LINEAR-SEARCH(A, x)
1 n←length[A]                ▷n 表示序列 A 中元素个数
2 i←1                        ▷从 A[1]开始
3 while i≤n                  ▷逐一检测 A[i]的值是否等于 x
4   do if A[i]=x
5        then return i       ▷若存在 A[i]=x,则返回 i
6      i←i+1
7 return −1
```

算法 1-1 在序列 A 中查找关键值为 x 的元素的线性查找过程 LINEAR-SEARCH

算法的伪代码表示除了上述的无歧义及抽象表达能力强的优势以外，有一个其他方法无可比拟的优点：其表达形式类似于高级程序设计语言代码表达形式，因此更便于将伪代码转换为程序代码。本书此后均用伪代码来描述算法。将算法描述为一个伪代码过程的其他好处在于过程的参数往往表示出了待解决的计算问题的输入，而过程的返回值表示出了该计算问题的输出。例如，解决在序列 A 中查找关键值为 x 的元素的算法 LINEAR-SEARCH(A,x)过程的参数 A 和 x 恰为该问题的输入，而在过程中第 6 行返回的 i 或在第 7 行返回的 -1 恰为问题的输出。

1.1.3　伪代码的使用约定

(1) 用分层缩进来指示块结构。例如，从第 2 行开始的 **while** 循环的循环体由 3 行组成，分层缩进风格也应用于 **if-then-else** 语句，如第 4～5 行的 **if-then** 语句。使用分层缩进而不是传统的诸如 **begin** 和 **end** 语句来表示块结构，大大降低了混乱，提高了清晰度。

(2) 循环结构 **while**、**for** 和 **repeat** 以及条件结构 **if**、**then** 和 **else** 具有与 Pascal 相仿的解释。然而，对 **for** 循环却有一点点不同：在 Pascal 中，循环计数变量的值在退出循环后是没有定义的，但在本书里，循环计数器在退出循环后仍然保留。所以，一个 **for** 循环刚结束时，循环计数器的值首次超过 **for** 循环上界。在插入排序正确性的讨论中就用到了这一特性。**for** 循环开头是 **for** $j \leftarrow 2$ **to** $length[A]$，所以当此循环结束时，$j = length[A]+1$(或等价地，$j = n+1$，因为 $n = length[A]$)。

(3) 符号 ▷ 表示本行其余部分是注释。

(4) 多重赋值形式 $i \leftarrow j \leftarrow e$ 的含义是变量 i 和 j 同赋予表达式 e 的值，它应当被理解为在赋值操作 $j \leftarrow e$ 之后紧接着赋值操作 $i \leftarrow j$。

(5) 变量(如 i、j 及 key)都局部于给定的过程。人们将不使用全局变量除非特别声明。

(6) 数组元素是通过数组名后跟括在方括号内的下标来访问。例如，$A[i]$表示数组 A 的第 i 个元素。记号".."用来表示数组中取值的范围。因此，$A[1..j]$表示 A 由$A[1]$、$A[2]$、…，$A[j]$j个元素构成的子序列。

(7) 组合数据通常组织在**对象**中，其中组合了若干个**属性**或**域**。用域名紧跟包括在方括号中的对象名来访问一个具体的域。例如，把一个数组当成一个对象，它具有说明其所包含的元素个数的属性 $length$。$length[A]$表示数组 A 的元素个数。虽然对数组元素和对象属性都使用方括号，通过上下文应当是能清楚辨别的。

表示数组或对象的变量被当成一个指向表示数组或对象的指针。对一个对象 x 的所有域 f，设 $y \leftarrow x$ 将导致 $f[y] = f[x]$。此外，若设 $f[x] \leftarrow 3$，则不仅有 $f[x] = 3$，且有 $f[y] = 3$。换句话说，赋值 $y \leftarrow x$ 后，x 和 y 指向同一个对象。

有时，一个指针不指向任何对象，此时，给它一个特殊的值 NIL。

(8) 过程的参数是**按值**传递的。被调用的过程以复制的方式接受参数，若对参数赋值，则主调过程不能看到这一变化。若传递的是一个对象，则指向数据的指针被复制，但对象的域没有复制。例如，若 x 是一个被调过程的参数，过程中的赋值 $x \leftarrow y$ 对主调过程是不可见的。但是，赋值 $f[x] \leftarrow 3$ 却是可见的。

(9) 布尔运算符 and 和 or 都是**短回路**的。也就是说，当人们计算表达式 x and y 时，先

计算 x。若 x 为 FALSE,则整个表达式不可能为 TRUE,所以人们不再计算 y。另一方面,若 x 为 TRUE,必须计算 y 以确定整个表达式的值。类似地,在表达式 x or y 中,计算表达式 y 当且仅当 x 为 FALSE。短回路操作符使得人们能够写出诸如"$x \neq$ NIL and $f[x]=y$"这样的布尔表达式而不必担心当 x 为 NIL 时去计算 $f[x]$。

1.1.4 算法分析

解决同一个问题,算法不必是唯一的。对表示问题的数据的不同的组织方式(数据结构),解决问题不同的策略(算法思想)将导致不同的算法。例如,若查找问题描述如下。

输入:n 个数构成的序列 $A=<a_1,a_2,\cdots,a_n>$,其中 $a_i \leqslant a_{i+1}$,$1 \leqslant i < n$;特定值 x。

输出:若序列 A 中存在元素,其值等于 x,返回一个值为 x 的元素在序列 A 中的位置——下标。否则,返回 -1。

可以用如下的算法来解决这一问题。

```
BINARY-SEARCH(A,x)
1 p←1,r←length[A]
2 while p≤r
3   do q←⌊(p+r)/2⌋
4     if A[q]=x
5       then return q
6     if A[q]<x
7       then p←q+1
8       else r←q-1
9 return -1
```

算法 1-2 在有序序列 A 中查找关键值为 x 的元素的二分查找过程 BINARY-SEARCH

该算法利用序列 A 的有序性,从 $p=1,r=n$ 开始,考察 A 的中点处元素 $A[\lfloor (p+r)/2 \rfloor]$ 的值是否等于 x,若是,则 q 就是所要求的下标值。否则,若该值小于 x,则丢弃 $A[p..q]$,在 $A[q+1..r]$中查找(调整 p 为 $q+1$),若该值大于 x,则丢弃 $A[q..r]$,在 $A[p..q-1]$中继续查找(调整 r 为 $q-1$)。周而复始,直至 $p>r$,即在 A 中无元素的值等于 x,返回 -1。

解决同一问题的不同的算法,消耗的时间和空间资源可能有所不同。算法运行所需要的计算机资源的量称为算法的复杂性。一般来说,解决同一问题的算法,需要的资源量越少,人们认为越优秀。计算算法运行所需资源量的过程称为算法复杂性分析,简称为**算法分析**。理论上,算法分析既要计算算法的时间复杂性,也要计算它的空间复杂性。然而,算法的运行时间都是消耗在已存储的数据处理上的,从这个意义上说,算法的空间复杂性不会超过时间复杂性。出于这个原因,人们多关注于算法的时间复杂性分析。本书中除非特别说明,所说的算法分析,局限于对算法的时间复杂性分析。

为客观、科学地评估算法的时间复杂性,人们设置一台抽象的计算机。它只用一个处理机,却有无限量的随机存储器。它的有限个基本操作——算术运算、逻辑运算和数据的移动(比如对变量的赋值)均在有限固定时间内完成,人们进一步假定所有这些基本操作都消耗一个时间单位,称此抽象计算机为随机访问计算机,简记为 RAM(Random Access