普通高等教育“十一五”国家级规划教材

21世纪大学本科
计算机专业系列教材

在线教学版 教学资源 练习与测试 互动教学 智能学习

算法设计与分析（第4版）

王晓东 编著

http://www.tup.com.cn

- 国家精品课程主讲教材
- 根据教育部“高等学校计算机科学与技术专业规范”组织编写
- 与美国 ACM 和 IEEE CS *Computing Curricula* 最新进展同步

清华大学出版社

普通高等教育“十一五”国家级规划教材

国家精品课程主讲教材

21世纪大学本科计算机专业系列教材

算法设计与分析
（第4版）

王晓东　编著

清华大学出版社

北京

内 容 简 介

为了适应我国21世纪计算机人才培养的需要，结合我国高等学校教育工作的现状，立足培养学生能跟上国际计算机科学技术的发展水平，更新教学内容和教学方法，提高教学质量，本书以算法设计策略为知识单元，系统地介绍计算机算法的设计方法与分析技巧，以期为计算机科学与技术学科的学生提供广泛而坚实的计算机算法基础知识。

另有配套的《算法设计与分析习题解答(第4版)》，对本书的全部习题做了详尽的解答。

本书内容丰富，观点新颖，理论联系实际，不仅可用作高等学校计算机类专业本科生和研究生学习计算机算法设计与分析的教材，而且也适合广大工程技术人员和自学读者学习参考。

图书在版编目(CIP)数据

算法设计与分析/王晓东编著. —4版. —北京：清华大学出版社，2018
(21世纪大学本科计算机专业系列教材)
ISBN 978-7-302-51010-9

Ⅰ. ①算… Ⅱ. ①王… 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(2018)第190896号

责任编辑：张瑞庆
封面设计：何凤霞
责任校对：梁　毅
责任印制：沈　露

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 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：22.5　　**字　　数**：550千字
版　　次：2003年1月第1版　2018年10月第4版　　**印　　次**：2018年10月第1次印刷
定　　价：49.90元

产品编号：079854-01

21世纪大学本科计算机专业系列教材编委会

前言

FOREWORD

以最低的成本、最快的速度、最好的质量开发出适合各种应用需求的软件，必须遵循软件工程的原则，设计出高效率的程序。一个高效的程序不仅需要编程技巧，更需要合理的数据组织和清晰高效的算法。这正是计算机科学领域里数据结构与算法设计所研究的主要内容。一些著名的计算机科学家在有关计算机科学教育的论述中提出，计算机科学是一种创造性思维活动，其教育必须面向设计。计算机算法设计与分析正是一门面向设计，且处于计算机科学与技术学科核心地位的教育课程。通过对计算机算法系统的学习与研究，理解和掌握算法设计的主要方法，培养对算法的计算复杂性进行正确分析的能力，为独立地设计算法和对给定算法进行复杂性分析奠定坚实的理论基础，对从事计算机系统结构、系统软件和应用软件研究与开发的科技工作者是非常重要和必不可少的。为了适应我国21世纪计算机人才培养的需要，结合我国高等学校教育工作的现状，立足培养学生能跟上国际计算机科学技术的发展水平，更新教学内容和教学方法，本书以算法设计策略为知识单元，系统地介绍计算机算法的设计方法与分析技巧，以期为计算机科学与技术学科的学生提供一个广泛而坚实的计算机算法基础知识。

全书共分11章。在第1章中首先介绍算法的基本概念，接着简要阐述算法的计算复杂性和算法的描述，然后围绕设计算法常用的基本设计策略组织第2章至第10章的内容。第2章介绍递归与分治策略，这是设计有效算法最常用的策略，是必须掌握的方法。第3章是动态规划算法，以具体实例详述动态规划算法的设计思想、适用性以及算法的设计要点。第4章介绍贪心算法，这也是一种重要的算法设计策略，它与动态规划算法的设计思想有一定的联系，但其效率更高。按贪心算法设计出的许多算法能导致最优解，其中有许多典型问题和典型算法可供学习和使用。第5章和第6章分别介绍回溯法和分支限界法，这两章所介绍的算法适合处理难解问题，其解题的思想各具特色，值得学习和掌握。第7章介绍概率算法，对许多难解问题提供高效的解决途径，是有很高实用价值的算法设计策略。第8章介绍NP完全性理论和解NP难问题的近似算法。首先介绍计算模型、确定性和非确定性图灵机，然后进一步深入介绍NP完全性理论，最后介绍解NP难问题的近似算法，这是当前计算机算法领域的热门研究课题，具有很高的实用价值。第9章介绍有关串和序列的高效算法。第10章通过实例介绍算法设计中常用的算法优化策略。最后，在第11章介绍算法设计中较新的研究领域——在线算法设计。

在本书各章的论述中，首先介绍一种算法设计策略的基本思想，然后从解决计算机科学与应用中出现的实际问题入手，由简到繁地描述几个经典的精巧算法，同时对每个算法所需

要的时间和空间进行分析。这样使读者既能学到一些常用的精巧算法,又能通过对算法设计策略的反复应用,牢固掌握这些算法设计的基本策略,以期收到融会贯通之效。在为各种算法设计策略选择用于展示其设计思想与技巧的具体应用问题时,本书有意重复选择某些经典问题,使读者能深刻地体会到一个问题可以用多种设计策略求解。同时,通过对解同一问题的不同算法的比较,更容易体会到每一个具体算法的设计要点。随着本书内容的逐步展开,读者也将进一步感受到综合应用多种设计策略可以更有效地解决问题。

本书采用面向对象的Java语言作为表述手段,在保持Java优点的同时,尽量使算法的描述简明、清晰。

为了便于读者加深对知识的理解,各章配有难易适当的习题,以适应不同程度读者练习的需要。

在本书的编写过程中,得到教育部高等学校计算机类专业教学指导委员会的关心和支持。福州大学"211工程"计算机与信息工程重点学科实验室和福建工程学院为本书的写作提供了优良的设备与工作环境。清华大学出版社负责本书编辑出版工作的全体人员为本书的出版付出了大量辛勤劳动,他们认真细致、一丝不苟的工作精神保证了本书的出版质量。南京大学宋方敏教授和福州大学傅清祥教授在百忙中认真审阅了全书,提出了许多宝贵的改进意见。在此,谨向每一位曾经关心和支持本书编写工作的各方面人士表示衷心的谢意!

由于作者的知识和写作水平有限,书稿虽几经修改,仍难免存在缺点。热忱欢迎同行专家和读者惠予批评指正,使本书在使用过程中不断改进,日臻完善。

作　者

2018年6月

目 录

CONTENTS

第 1 章 算法引论

1.1 算法与程序

对于计算机科学来说，算法(algorithm)的概念至关重要。通俗地讲，算法是指解决问题的方法或过程。严格地讲，算法是满足下述性质的指令序列。

(1) 输入：有零个或多个外部量作为算法的输入。

(2) 输出：算法产生至少一个量作为输出。

(3) 确定性：组成算法的每条指令是清晰的、无歧义的。

(4) 有限性：算法中每条指令的执行次数有限，执行每条指令的时间也有限。

程序(program)与算法不同。程序是算法用某种程序设计语言的具体实现。程序可以不满足算法的性质(4)即有限性。例如操作系统，它是在无限循环中执行的程序，因而不是算法。然而可把操作系统的各种任务看成一些单独的问题，每一个问题由操作系统中的一个子程序通过特定的算法实现，该子程序得到输出结果后便终止。

1.2 表达算法的抽象机制

算法层出不穷，变化万千，其对象数据和结果数据名目繁多，不胜枚举。最基本的有布尔值数据、字符数据、整数和实数等；稍复杂的有向量、矩阵、记录等；更复杂的有集合、树和图，还有声音、图形、图像等数据。

算法的运算种类五花八门，多姿多彩。最基本的有赋值运算、算术运算、逻辑运算和关系运算等；稍复杂的有算术表达式和逻辑表达式等；更复杂的有函数值计算、向量运算、矩阵运算、集合运算，以及表、栈、队列、树和图的运算等；此外，还可能有以上列举的运算的复合和嵌套。

高级程序设计语言在数据、运算和控制三方面的表达中引入许多使之十分接近算法语言的概念和工具，具有抽象表达算法的能力。高级程序设计语言的主要好处如下：

(1) 高级语言更接近算法语言，易学、易掌握，一般工程技术人员只需几周时间的培训就可以胜任程序员的工作。

(2) 高级语言为程序员提供了结构化程序设计的环境和工具，使得设计出来的程序可读性好、可维护性强、可靠性高。

(3) 高级语言不依赖于机器语言，与具体的计算机硬件关系不大，因而所写出来的程序可植性好、重用率高。

(4) 把繁杂琐碎的事务交给编译程序,所以自动化程度高,开发周期短,程序员可以集中时间和精力从事更重要的创造性劳动,提高程序质量。

算法从非形式的自然语言表达形式转换为形式化的高级语言是一个复杂的过程,仍然要做很多繁杂琐碎的事情,因而需要进一步抽象。

对于一个明确的数学问题,设计它的算法,总是先选用该问题的一个数据模型。接着弄清楚该问题数据模型在已知条件下的初始状态和要求的结果状态,以及这两个状态之间的隐含关系。然后探索从数据模型的已知初始状态到达要求的结果状态所需的运算步骤。这些运算步骤就是求解该问题的算法。

按照自顶向下、逐步求精的原则,在探索运算步骤时,首先应该考虑算法顶层的运算步骤,然后再考虑底层的运算步骤。所谓顶层运算步骤是指定义在数据模型级上的运算步骤,或称宏观步骤。它们组成算法的主干部分,这部分算法通常用非形式的自然语言表达。其中,涉及的数据是数据模型中的变量,暂时不关心它的数据结构;涉及的运算以数据模型中的数据变量作为运算对象,或作为运算结果,或二者兼而为之,简称为定义在数据模型上的运算。由于暂时不关心变量的数据结构,这些运算都带有抽象性质,不含运算细节。所谓底层运算步骤,是指顶层抽象运算的具体实现。它们依赖于数据模型的结构,依赖于数据模型结构的具体表示。因此,底层运算步骤包括两部分:一是数据模型的具体表示;二是定义在该数据模型上的运算的具体实现。底层运算可以理解为微观运算。底层运算是顶层运算的细化;底层运算为顶层运算服务。为了将顶层算法与底层算法隔开,使二者在设计时不互相牵制、互相影响,必须对二者的接口进行抽象。让底层只通过接口为顶层服务,顶层也只通过接口调用底层运算。这个接口就是抽象数据类型(abstract data types,ADT)。

抽象数据类型是算法设计的重要概念。严格地说,它是算法的一个数据模型连同定义在该模型上并作为算法构件的一组运算。这个概念明确地把数据模型与该模型上的运算紧密地联系起来。事实正是如此,一方面,数据模型上的运算依赖于数据模型的具体表示,数据模型上的运算以数据模型中的数据变量为运算对象,或作为运算结果,或二者兼而为之;另一方面,有了数据模型的具体表示,有了数据模型上运算的具体实现,运算的效率随之确定。如何选择数据模型的具体表示使该模型上各种运算的效率都尽可能高?很明显,对于不同的运算组,为使该运算组中所有运算的效率都尽可能高,其相应的数据模型的具体表示将不同。在这个意义下,数据模型的具体表示又反过来依赖于数据模型上定义的运算。特别是当不同运算的效率互相制约时,还必须事先将所有的运算相应的使用频度排序,让所选择的数据模型的具体表示优先保证使用频度较高的运算有较高的效率。数据模型与定义在该模型上的运算之间存在的这种密不可分的联系,是抽象数据类型概念产生的背景和依据。

使用抽象数据类型带给算法设计的好处主要有:

(1) 算法顶层设计与底层实现分离,使得在进行顶层设计时不考虑它所用到的数据、运算表示和实现;反过来,在表示数据和实现底层运算时,只要定义清楚抽象数据类型而不必考虑在什么场合引用它。这样做使得算法设计的复杂性降低了,条理性增强了。既有助于迅速开发出程序原型,又使开发过程少出差错,程序可靠性高。

(2) 算法设计与数据结构设计隔开,允许数据结构自由选择,从中比较,优化算法效率。

(3) 数据模型和该模型上的运算统一在抽象数据类型中,反映它们之间内在的互相依赖和互相制约的关系,便于空间和时间耗费的折中,可以灵活地满足用户要求。

(4) 由于顶层设计和底层实现局部化,在设计中出现的差错也是局部的,因而容易查找和纠正差错。在设计中常常要做的增、删、改也都是局部的,因而也都容易进行。因此,用抽象数据类型表述的算法具有很好的可维护性。

(5) 算法自然呈现模块化,抽象数据类型的表示和实现可以封装,便于移植和重用。

(6) 为自顶向下、逐步求精和模块化提供有效途径和工具。

(7) 算法结构清晰,层次分明,便于算法正确性的证明和复杂性的分析。

1.3 描述算法

描述算法可以有多种方式,如自然语言方式、表格方式等。在本书中,采用 Java 语言描述算法。Java 语言的优点是类型丰富,语句精练,具有面向过程和面向对象的双重特点,可以充分利用抽象数据类型这一有力工具表述算法。用 Java 描述算法可使整个算法结构紧凑,可读性强。在本书中,有时为了更好地阐明算法的思路,还采用 Java 与自然语言相结合的方式描述算法,本节简要概述 Java 语言的若干重要特性。

1. Java 程序结构

1) 应用程序和 applet

Java 程序有两种类型:应用程序(stand-alone program)和 applet。Java 应用程序一定有一个主方法 main,而 applet 的主方法名为 init。

Java 应用程序可在命令行中用命令语句

```
java programName
```

来执行,其中 programName 是应用程序名。在执行 Java 应用程序时,系统自动调用应用程序的主方法 main。

Java 的 applet 必须嵌入 HTML 文件,由 Web 浏览器或 applet 阅读器来执行。在执行 applet 时,系统自动调用 applet 的主方法 init。

Java 程序必须先编译后执行。系统在编译时,将 Java 源程序转化为 Java 字节码(bytecode)。Java 源程序文件的扩展名为 java,编译后字节码文件的扩展名为 class。

Java 字节码可以看作在一台虚拟计算机即 Java 虚拟机(JVM)上运行的语言。本地计算机通过 Java 虚拟机解释运行 Java 程序。

2) 包

Java 程序和类可以包(packages)的形式组织管理。Java 自带的包有 java.awt,java.io,java.lang,java.util 等。Java 用户可根据需要将自己的程序组织成适合各种应用的包。

3) import 语句

在 Java 程序中可以用 import 语句加载所需的包。例如,import java.io.*;语句加载 java.io 包。语句 import java.io.PrintStream;则加载 java.io 包中的 PrintStream 类。

2. Java 数据类型

Java 基本数据类型如表 1-1 所示。

除了基本数据类型,Java 还提供一些经过包装的非基本数据类型,如 Byte, Integer, Boolean, String 等。

表 1-1 Java 基本数据类型

类型	默认值	分配空间/位	取值范围
boolean	false	1	true,false
byte	0	8	−128～+127
char	\u0000	16	\u0000～\uFFFF
double	0.0	64	$\pm 4.9\times10^{-324}$～ $\pm 1.8\times10^{308}$
float	0.0	32	$\pm 1.4\times10^{-45}$～ $\pm 3.4\times10^{38}$
int	0	32	−2 147 483 648～2 147 483 647
long	0	64	-9.2×10^{17}～$+9.2\times10^{17}$
short	0	16	−32 768～32 767

Java 处理基本数据类型和非基本数据类型的方式大不相同。在声明一个具有基本数据类型的变量时,自动建立该数据类型的对象(或称实例)。例如,语句 int k;建立一个数据类型为 int 的对象 k,其默认值为 0。对非基本数据类型,情况则不一样。语句 String s;并不建立具有数据类型 String 的对象,而是建立一个数据类型为 String 的引用对象(内存地址)。该引用对象的名字是 s,其初始值为 null。数据类型为 String 的对象可用下面的 new 语句建立。

```
s=new String("Welcome");
String s=new String("Welcome");
```

其中,第一个语句假设 s 已经声明过;第二个语句声明变量 s,并用 new 语句建立对象。

其他非基本数据类型对象的声明和建立方式与此类似。

3. 方法

在 Java 语言中,执行特定任务的函数或过程统称为方法(methods)。例如,Java 的 Math 类给出的常见的数学计算的方法如表 1-2 所示。

表 1-2 Java 的 Math 类常见的数学计算方法

方 法	功 能	方 法	功 能
abs(x)	x 的绝对值	max(x,y)	x 和 y 中较大者
ceil(x)	不小于 x 的最小整数	min(x,y)	x 和 y 中较小者
cos(x)	x 的余弦	pow(x,y)	x^y
exp(x)	e^x	sin(x)	x 的正弦
floor(x)	不大于 x 的最大整数	sqrt(x)	x 的平方根
log(x)	x 的自然对数	tan(x)	x 的正切

对计算表达式$\dfrac{a+b+|a-b|}{2}$值的自定义方法 ab 描述如下:

```
public static int ab(int a, int b)
```

```
{
    return (a+b+Math.abs(a-b))/2;
}
```

1）方法的参数

上述方法 ab 中，a 和 b 是形式参数，在调用方法时通过实际参数赋值。Java 中所有方法的参数均为值参数。在调用方法时先将实际参数的值复制到形式参数中，然后再执行调用。因此，在执行调用后，实际参数的值不变。

2）方法重载

方法的参数个数以及各参数的类型定义了该方法的签名。例如，上述方法 ab 的签名为(int,int)。Java 允许方法重载，即允许定义有不同签名的同名方法。上面的方法 ab 可以重载如下：

```
public static double ab(double a, double b)
{
    return (a+b+Math.abs(a-b))/2.0;
}
```

Java 解释器根据方法调用时实际参数的签名选用确定的方法。

4. 异常

Java 的异常(exception)提供了一种处理错误的简洁的方法。当程序发现一个错误时，就引发一个异常，以便在程序最合适的地方捕获异常并进行处理。例如，方法 ab 要求输入参数均为正整数时，可将方法 ab 修改如下：

```
public static int ab(int a, int b)
{
   if (a<=0||b<=0)
      throw new IllegalArgumentException ("All parameters must be>0");
   else return (a+b+Math.abs(a-b))/2;
}
```

在执行运算前，先检测参数 a 和 b，一旦发现非正参数，就由 throw 语句引发一个异常。throw 语句类似于 return 语句，但它描述方法的异常终止。通常用 try 块来定义异常处理，在引发异常之前，执行 try 块体。在 try 块体之后，有一个或多个异常处理。每一个异常处理由一个 catch 语句组成。这个语句指明欲捕获的异常以及出现该异常时要执行的代码块。当 try 引发了一个已定义的异常时，控制就转移到相应的异常处理中。

```
public static void main(String [] args)
{
    try { f();}
    catch (exception1)
    { 异常处理;}
    catch (exception2)
    { 异常处理;}
    …
    finally
```