“十三五”高等教育规划教材
高等院校电气信息类专业“互联网+”创新规划教材

数据结构与算法应用实践教程

（第2版）

主　编　李文书

北京大学出版社
PEKING UNIVERSITY PRESS

“十三五”高等教育规划教材
高等院校电气信息类专业“互联网+”创新规划教材

数据结构与算法应用实践教程

（第2版）

主　编　李文书

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书和传统同类图书的区别是除了介绍基本的数据结构知识，如线性表、栈、队列、链表、树、二叉树、AVL 树、红黑树、排序和查找之外，还介绍了一些 C 语言中的内存分配、结构数组和结构指针的有关概念及常见问题分析；另外，还介绍了相应知识点的应用实践。总体来说，本书选取的内容侧重于在实际中有广泛应用的数据结构及算法，有很好的实用价值。本书介绍的所有数据结构及算法都以不同复杂程度给出其实现编码。为了便于读者自学，每章末附有小结及习题与思考。

本书可以作为高等院校计算机学科和信息类学科本、专科的教材，也可以作为其他理工专业的选修教材，还可以作为从事计算机工程与应用工作的科技人员的参考用书。

图书在版编目(CIP)数据

数据结构与算法应用实践教程/李文书主编. —2 版. —北京：北京大学出版社，2017.2

(高等院校电气信息类专业“互联网+”创新规划教材)

ISBN 978-7-301-27833-8

Ⅰ. ①数… Ⅱ. ①李… Ⅲ. ①数据结构—高等学校—教材②算法分析—高等学校—教材 Ⅳ. ①TP311.12 ②TP301.6

中国版本图书馆 CIP 数据核字(2016)第 298027 号

书　　名	数据结构与算法应用实践教程(第 2 版) Shuju Jiegou yu Suanfa Yingyong Shijian Jiaocheng
著作责任者	李文书　主编
策 划 编 辑	郑　双
责 任 编 辑	黄红珍
数 字 编 辑	刘志秀
标 准 书 号	ISBN 978-7-301-27833-8
出 版 发 行	北京大学出版社
地　　址	北京市海淀区成府路 205 号　100871
网　　址	http://www.pup.cn　新浪微博:@北京大学出版社
电 子 信 箱	pup_6@163.com
电　　话	邮购部 62752015　发行部 62750672　编辑部 62750667
印 刷 者	北京鑫海金澳胶印有限公司
经 销 者	新华书店
	787 毫米×1092 毫米　16 开本　18.75 印张　441 千字
	2012 年 2 月第 1 版
	2017 年 2 月第 2 版　2017 年 2 月第 1 次印刷
定　　价	42.00 元

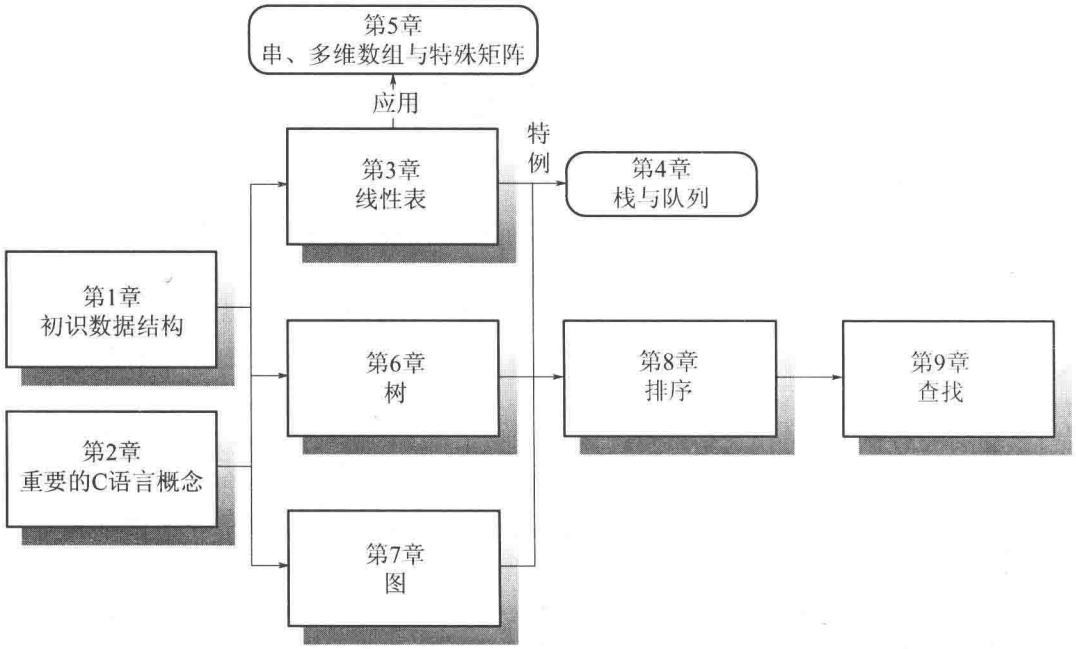
第 2 版前言

数据结构与算法是计算机科学和相关专业的核心课程，是软件设计的理论基础，是主要研究在非数值计算的程序设计问题中计算机的操作对象(数据元素)及它们之间的关系和运算，并对算法运行时间进行分析的学科。本课程的学习将为后续的操作系统、编译原理、软件工程、数据库概论等专业基础课和专业课程的学习，以及软件设计水平的提高打下良好的基础。

数据结构与算法是一门理论与实践并重的课程，学生在学习时不仅要掌握数据结构的基础知识，还要掌握编程的基本技巧。随着海量数据的增加，人们对能够处理这些数据程序的需求变得日益迫切。但是，在数据输入量很大时，程序的低效率现象变得非常明显。因此，这又要求对效率问题给予更大的关注。例如，一些特定问题通过精心的设计可以把对大量数据处理的时间限制从十几年减至不到一秒。因此，在某些情况下，对于影响算法实现运行时间的一些微小细节都需要进行认真的探究。如今，数据结构与算法已经成为软件开发工程师必备的基础知识之一。社会上大多数公司在招聘软件开发人员时都会考查应聘人员对数据结构与算法的熟练程度，并以此作为衡量应聘者水平的重要依据。

本书旨在使读者了解数据结构与算法这门课程，掌握其所含内容的内在规律，最终灵活运用，甚至有所发展。正如学写字先描红，不先模仿怎么创新呢？如果对现有的结论和成果都不了解来龙去脉，何谈再有新的突破呢？哲学上讲，只有遵从人类的认知规律，人们才能更容易地认识新事物，教科书为了达到其传授知识的目的，必须遵从人的认知规律。从数据结构的发展来看，它是应问题的需要而出现的，并为解决问题而服务。换而言之，对于数据结构的讲解，应当把重点放在算法上，在各种典型问题上提出新的数据结构；最终得出的认识是，为了特定的问题和算法而选用特定的数据结构，为了改进算法而改进数据结构，为了新的问题和算法而创造出新的数据结构。

本书系统介绍了数据结构的基础理论知识及算法设计方法，在内容选取上符合计算机学科和信息类学科人才培养目标的要求及教学规律和认知规律，在组织编排上体现了“先理论、后应用、理论与应用相结合”的原则，并兼顾学科的广度和深度，力求适用面广。全书共分为 9 章：第 1 章介绍数据结构的讨论范畴、基本概念、数据的逻辑结构、数据的物理结构及算法的描述与分析；第 2 章介绍一些重要的 C 语言概念，同时对 C 语言常见问题进行了分析；第 3 章介绍线性表的各种存储结构及相关应用；第 4 章介绍栈与队列的基本概念、各种存储结构及相关应用；第 5 章介绍串、多维数组和特殊矩阵的存储结构及相关应用；第 6 章介绍树、二叉树和森林的基本概念、各种存储结构及遍历、线索二叉树、二叉排序树及相关应用；第 7 章介绍图的概念、各种存储结构及遍历、最小生成树、关键路径、拓扑排序、最短路径及相关应用；第 8 章介绍 5 种基本的排序算法(插入排序、交换排序、选择排序、归并排序和基数排序)及相关应用；第 9 章介绍查找的基本概念、静态查找表及动态查找表的实现算法及相关应用。为了使读者对相关知识有一个更清晰的脉络，我们给出了各章之间依赖关系的结构图，如下图所示。



本书每一章都精心设计了经典的应用实践问题，并且附有一定数量、难度适宜的课后习题与思考，旨在引导读者不断深入地学习，学以致用，灵活处理一些实际问题，提高程序设计的能力。本书中的所有算法，均在 Visual C++下调试通过，不需任何修改就可直接上机运行、验证。可与本书配套使用的《数据结构重点难点问题剖析》(C 语言版)，已由浙江大学出版社出版，书中提供配套的习题和实习题，并可作为学习指导手册。

本书由李文书担任主编，胡杰、骆淑云、黄建、高玉娟、王哲、高海、尤苡名和张琛担任副主编，在写作过程中编者参考了国内外数据结构的最新教材和研究成果，得到了许多老教授的帮助和支持，在此对资料原作者和老教授们表示衷心的感谢！

由于编者水平有限，书中难免存在不妥之处，敬请读者指正。欢迎读者与编者联系，E-mail：wshlee@163.com。

编　者

2016 年 9 月

目　录

第 1 章　初识数据结构 …… 1
1.1　数据结构讨论范畴 …… 2
1.2　基本概念 …… 2
1.3　数据的逻辑结构 …… 4
1.4　数据的物理结构 …… 5
1.5　算法的描述与分析 …… 6
本章小结 …… 9
习题与思考 …… 10

第 2 章　重要的 C 语言概念 …… 13
2.1　内存分配 …… 14
2.2　结构数组、结构指针和位结构 …… 17
2.3　C 语言常见问题分析 …… 20
本章小结 …… 25
习题与思考 …… 26

第 3 章　线性表 …… 28
3.1　线性表的概念 …… 29
3.2　顺序表 …… 30
3.3　单向链表 …… 33
3.4　循环链表 …… 39
3.5　双向链表 …… 40
3.6　应用实践 …… 42
本章小结 …… 47
习题与思考 …… 48

第 4 章　栈与队列 …… 50
4.1　栈 …… 51
4.2　队列 …… 57
4.3　应用实践 …… 67
本章小结 …… 80
习题与思考 …… 81

第 5 章　串、多维数组与特殊矩阵 …… 83
5.1　串 …… 84
5.2　串的模式匹配 …… 96
5.3　多维数组 …… 103
5.4　特殊矩阵的压缩存储 …… 106
5.5　稀疏矩阵 …… 109
5.6　应用实践 …… 116
本章小结 …… 121
习题与思考 …… 121

第 6 章　树 …… 125
6.1　树的基本概念 …… 126
6.2　二叉树 …… 127
6.3　树和森林 …… 138
6.4　线索二叉树 …… 145
6.5　二叉排序树 …… 151
6.6　应用实践 …… 154
本章小结 …… 167
习题与思考 …… 168

第 7 章　图 …… 170
7.1　图的基本概念 …… 171
7.2　图的存储方式 …… 174
7.3　图的遍历 …… 180
7.4　最小生成树 …… 185
7.5　最短路径 …… 192
7.6　拓扑排序 …… 200
7.7　关键路径 …… 204
7.8　应用实践 …… 206
本章小结 …… 209
习题与思考 …… 209

第 8 章　排序 …… 212
8.1　基本概念 …… 213
8.2　插入排序 …… 214
8.3　交换排序 …… 219
8.4　选择排序 …… 225
8.5　归并排序 …… 229
8.6　基数排序 …… 232

8.7 排序方法比较 …… 236
8.8 应用实践 …… 237
本章小结 …… 239
习题与思考 …… 240
第9章 查找 …… 242
9.1 基本概念 …… 243
9.2 静态查找 …… 244
9.3 动态查找 …… 248
9.4 哈希查找 …… 275
9.5 应用实践 …… 283
本章小结 …… 286
习题与思考 …… 286
附录 关键词索引 …… 289
参考文献 …… 292

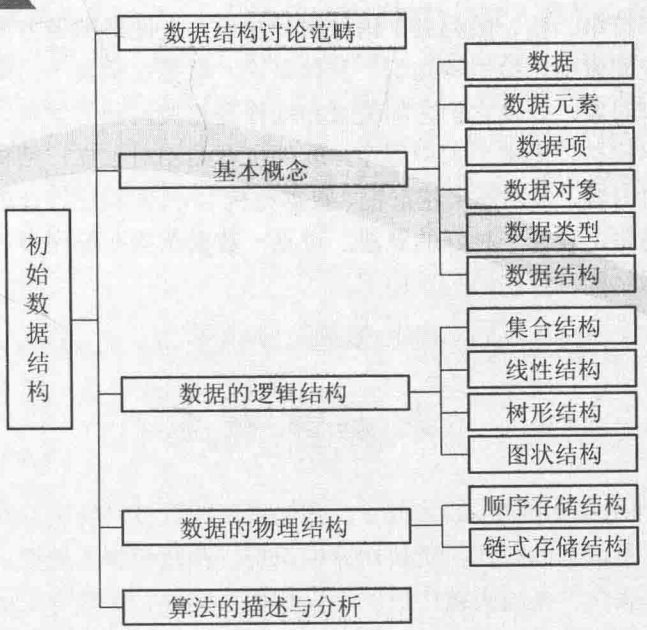
第1章

初识数据结构

学习目标

(1) 掌握数据结构的基本概念。
(2) 掌握抽象数据类型相关概念和软件构造方法。
(3) 掌握算法的含义及算法时间复杂度的计算方法。

知识结构图



重点和难点

本章讨论的都是一些基本概念，因此没有难点，重点在于了解有关数据结构的各个名词和术语的含义，以及语句频度计算和时间复杂度、空间复杂度的估算方法。

学习指南

(1) 掌握各名词、术语的含义，以及数据的逻辑结构和物理结构之间的关系。

(2) 了解抽象数据类型的定义、表示和实现方法。
(3) 理解算法5个要素的确切含义。
(4) 掌握计算语句频度和估算算法时间复杂度的方法。

1.1 数据结构讨论范畴

人们利用计算机的目的是解决实际的应用问题。在明确所要解决问题的基础上，经过对问题的深入分析和抽象，为其建立一个逻辑模型并分析基本的运算，然后确定恰当的数据结构表示该模型，在此基础上设计合适的数据存储和相关算法，最后完成具体的程序来模拟和解决实际问题。计算机求解问题的核心是算法设计，而算法设计又高度依赖于数据结构，数据结构的选择则取决于问题本身的需求。

在现实生活中，我们更多的不是解决数值计算的问题，而是借助一些更科学、有效的手段(如表、树和图等数据结构)，更好地处理问题。例如，在Web信息处理方面，我们需要图、字符、散列表、排序、索引、检索等知识；在人工智能方面，我们需要广义表、集合、有向图、搜索树等知识；在数据库方面，我们需要线性表、链表、排序、B+索引树等知识；在操作系统方面，我们需要队列、存储管理表、排序、目录树等知识；在编译原理方面，我们需要字符串、栈、散列表、语法树等知识；在图形图像方面，我们需要队列、栈、图、矩阵、空间索引、检索等知识。总体来说，数据结构是一门研究非数值计算的程序设计问题中操作对象，以及它们之间关系和操作等相关问题的学科。

20世纪70年代初，出现了大型程序，软件也开始相对独立，结构程序设计成为程序设计方法学的主要内容，人们越来越重视“数据结构”，认为程序设计的实质是对确定的问题选择一种好的结构，设计一种好的算法。可见，数据结构在程序设计当中占据了重要的地位。程序与数据结构、算法的关系如下：

程序=数据结构+算法

1.2 基 本 概 念

数据(data)是利用文字符号、数字符号及其他规定的符号对现实世界的事物及其活动所做的抽象描述。它是信息的载体，能被计算机识别、存储和加工处理。随着计算机技术的发展，数据这一概念的含义越来越广泛。不仅整数、实数、复数等是数据，字符、表格、声音、图形、图像等也都能够由计算机接收和处理，也都是数据。


【参考图文】

表示一个事物的一组数据称为一个数据元素(data element)，它是数据的基本单位，在程序中作为一个整体加以考虑和处理。在数据结构中，根据不同的需求，数据元素又被称为元素、顶点或记录。

数据项(data item)是具有独立含义的最小标识单位。在有些场合下，数据项又称为字段或域。例如，将一个学生的自然情况信息作为一个数据元素，而学生信息中的每一项(如学号、姓名、出生年月等)为一个数据项。

数据对象(data object)是性质相同的数据元素的集合，是数据的一个子集。既然数据对象是数据的子集，在实际应用中，处理的数据元素通常具有相同性质，在不产生混淆的情况下，我们都将数据对象简称为数据。

数据类型(data type)是和数据结构密切相关的一个概念，它最早出现在高级程序语言中，用以描述(程序)操作对象的特性专属。在用高级程序语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。数据类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些值上允许进行的操作。因此，数据类型是一个值的集合和定义在这个值集上的一组操作的总称。例如，C语言中的整型变量，其值集为某个区间上的整数(区间大小依赖于不同的机器)，定义在其上的操作为加、减、乘、除和取模等算术运算。

在C语言中，按照取值的不同，数据类型可以分为以下两类：

(1) 原子类型：不可以再分解的基本类型，包括整型、实型、字符型等。

(2) 结构类型：由若干个类型组合而成，是可以再分解的。例如，整型数组是由若干整型数据组成的。

抽象数据类型(abstract data type)是指一个数学模型及该模型上定义的一组操作的集合。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

事实上，抽象数据类型体现了程序设计中问题分解、抽象和信息隐藏的特性。抽象数据类型把实际生活中的问题分解为多个规模小且容易处理的问题，然后建立一个计算机能处理的数据类型，并把每个功能模块的实现细节作为一个独立的单元，从而使其具体实现过程隐藏起来。

为了便于在之后的讲解中对抽象数据类型进行规范的描述，我们给出了描述抽象数据类型的标准格式：

```
ADT 抽象数据类型名
    Data
        数据元素之间逻辑关系的定义
    Operation
        1
            初始条件
            结果描述
        2
            ...
        n
            ...
endADT
```

数据结构(data structure)是指相互之间存在一种或多种特定关系的数据元素集合，其主要研究数据的逻辑结构、物理结构及数据的运算。在计算机中，数据元素并不是孤立、杂乱无章的，而是具有内在联系的数据集合。数据元素之间存在的一种或多种特定关系，也就是数据的组织形式。为编写出一个“好”的程序，必须分析待处理对象的特性及各处理对象之间存在的关系。这也是研究数据结构的意义所在。


【参考图文】

一般来说，我们把数据结构分为逻辑结构和物理结构。

1.3 数据的逻辑结构

逻辑结构是指数据对象中数据元素之间的相互关系。逻辑结构是我们今后学习数据结构最需关注的内容。逻辑结构可分为以下 4 种：集合结构、线性结构、树形结构和图状结构。

1. 集合结构

集合结构中的数据元素除了同属于一个集合外，它们之间没有其他关系。各个数据元素是“平等”的，它们的共同属性是“同属于一个集合”。数据结构中的集合关系就类似于数学中的集合，如图 1.1 所示。

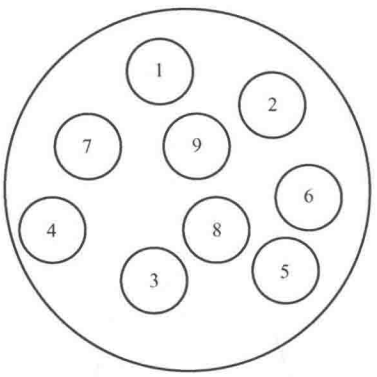


图 1.1 集合结构示意图

2. 线性结构

线性结构中的数据元素之间是一对一的关系，如图 1.2 所示。

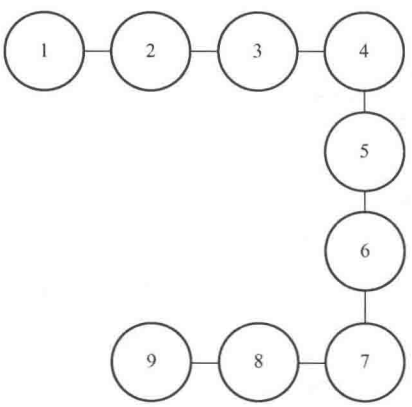


图 1.2 线性结构示意图

3. 树形结构

树形结构中的数据元素之间存在一对多的层次关系，如图 1.3 所示。

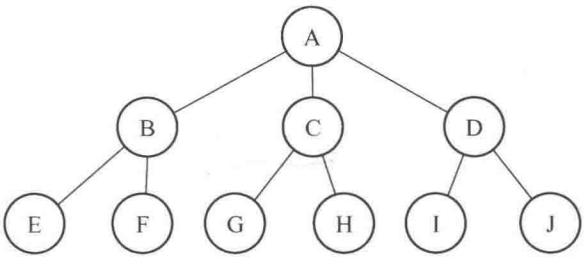


图 1.3　树形结构示意图

4. 图状结构

图状结构的数据元素是多对多的关系，如图 1.4 所示。

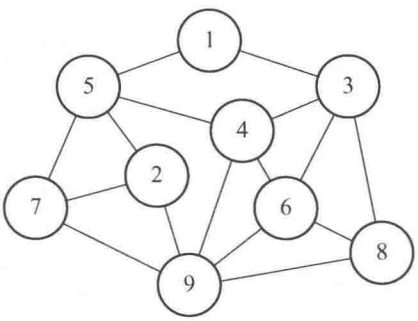


图 1.4　图状结构示意图

注意

在上述图例中，我们将每一个数据元素看作一个结点，用圆圈表示。元素之间的逻辑关系用结点之间的连线表示。如果这个关系是有方向的，那么用带箭头的连线表示。逻辑结构是针对具体问题的，是为了解决某个问题，在对问题理解的基础上，选择一个合适的数据结构表示数据元素之间的关系。

1.4　数据的物理结构

数据结构在计算机中的表示(又称为映像)称为数据的物理结构，也称为存储结构，是指数据的逻辑结构在计算机中的存储形式。数据的存储结构应正确反映数据元素之间的逻辑关系。如何存储数据元素之间的逻辑关系，是实现物理结构的重点和难点。

元素之间的关系在计算机中有两种不同的表示方法：顺序映像和非顺序映像，并由此得到数据元素两种不同的存储结构，即顺序存储和链式存储。

1. 顺序存储结构

把数据元素存放在地址连续的存储单元里，其数据间的逻辑关系和物理关系是一致的，借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系，如图 1.5 所示。

2. 链式存储结构

把数据元素存放在任意的存储单元里，这组存储单元可以是连续的，也可以是不连续

的。数据元素的存储关系并不能反映其逻辑关系，因此需要借助指示元素存储地址的指针(pointer)表示数据元素之间的逻辑关系，如图 1.6 所示。

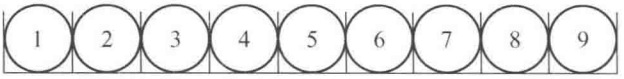


图 1.5　顺序存储结构示意图

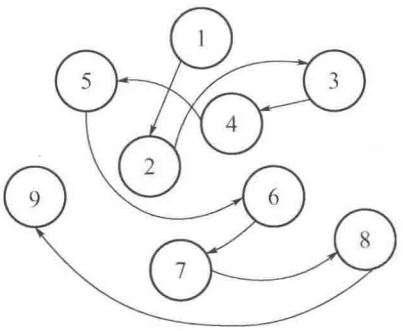


图 1.6　链式存储结构示意图

数据的逻辑结构和物理结构是密切相关的两个方面，同一逻辑结构可以对应不同的物理结构。以后读者会看到，任何一个算法的设计取决于选定的数据(逻辑)结构，而算法的实现依赖于采用的存储结构。逻辑结构是面向问题的，而物理结构是面向计算机的，其基本的目标是将数据及其逻辑关系存储到计算机的内存中。

1.5　算法的描述与分析

1.5.1　算法的描述

【参考图文】

算法是描述求解问题方法的操作步骤的集合。它主要有 4 种形式：框图形式、文字形式、伪码形式和程序设计语言形式。

框图形式简单、直观、易懂，但在描述比较复杂的算法时，显得不够方便，甚至难于把算法清晰、简洁地描述出来；文字形式同样简单、易懂，但在描述复杂算法时，显得不够简洁、清晰；伪代码形式与高级程序设计语言相仿，包括了高级语言的基本语言成分，并且较为简单，虽不能在计算机上直接运行，但容易编写和阅读，需要时改写为对应的高级语言程序也很容易；程序设计语言形式必须严格按照所使用的高级语言的语法规则来描述算法，可直接在计算机上运行获得结果。

算法有以下 5 个要素：

(1) 有穷性。有穷性是指一个算法必须总是在执行有穷步之后结束，且每一步都可在有穷时间内完成。

(2) 确定性。确定性是指算法中每一条指令必须有确切的含义，读者理解时不会产生二义性。并且，在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

(3) 可行性。可行性是指一个算法是能行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

(4) 输入。输入是指一个算法有零个或多个的输入，这些输入取自于某个特定的对象的集合。

(5) 输出。输出是指一个算法有一个或多个的输出，这些输出是同输入有着某些特定关系的量。

1.5.2　算法的分析

评价一个算法一般从正确性、可读性、健壮性、时间复杂度与空间复杂度五个主要方面进行。对于实际问题，我们主要分析其时间复杂度和空间复杂度。

一般从算法中选取一种基本操作，以其重复执行次数作为时间复杂度的依据，它取决于问题的规模 n 和待处理数据的初态。

算法需要占用的存储空间分为 3 部分：输入数据所占用的空间、程序代码所占用的空间和辅助变量所占用的空间。一般，输入数据所占用的空间与算法无关，取决于问题本身；程序代码所占用的空间对不同算法不会有数量级的差别。因此，空间复杂度主要考虑算法执行过程中辅助变量所占用的空间，一般以最坏情况下的空间复杂度作为算法的空间复杂度。

一般程序运行的时间与下列因素有关。

(1) 程序的输入。

(2) 编译的目标代码的质量。

(3) 执行程序机器指令的性质和速度。

(4) 构成程序的算法的时间复杂度。

正因为有如此多的因素，为了能比较客观地评价和比较算法，有必要分析一下各种因素对算法时间的影响及如何正确处理这些因素。

运行时间是输入规模的函数 f(n)，但是要记住 f(n)不等同于要求的时间复杂度 T(n)。由于在实际情况中，程序的输入不是一个确定的值 n，而是一个不确定的输入量。n 值表示输入数据的规模。这就涉及两个重要的概念：最坏时间复杂度和平均时间复杂度。

最坏时间复杂度：规模为 n 的所有输入量程序运行时间的最大值。

平均时间复杂度：规模为 n 的所有输入量程序运行时间的平均值。

由于平均时间复杂度比最坏时间复杂度要复杂，所以常常通过求最坏时间复杂度来表示某个算法的时间复杂度。但是必须要记住这两者是有区别的，也就是说，最坏时间复杂度最小的算法不一定是平均时间复杂度最小的算法。

由于算法的执行时间和运行程序的计算机有着密切的联系，所以 T(n)不能直接表达成 n 的函数，而要用“阶”来表示。

定义 1　$O(g(n))=\{f(n)\mid$若存在正常数 C 和 n_0，使得对所有的 $n\geq n_0$，有 $|f(n)|\leq C|g(n)|\}$。

定义 2　$\Omega(g(n))=\{f(n)\mid$若存在正常数 C 和 n_0，使得对所有的 $n\geq n_0$，有 $|f(n)|\geq C|g(n)|\}$。

定义 3　$\Theta(g(n))=\{f(n)\mid$ 若存在正常数 C_1、C_2 和 n_0，使得对所有的 $n\geq n_0$，有 $C_1|g(n)|\leq |f(n)|\leq C_2|g(n)|\}$。

由于存在重要结论 $T_1(n)+T_2(n)=O(\max(f(n), g(n)))$，所以一个程序的时间复杂度由程序中最复杂的部分组成，这一点是非常有用的。

另外，当算法的时间复杂度 T (n)与数据个数 n 无关系时，T (n) ≤ c * 1，所以此时算法的时间复杂度 T (n)=O (1)；当算法的时间复杂度 T (n)与数据个数 n 为线性关系时，算法的时间复杂度 T (n) = O (n)；以此类推分析一个算法中基本语句执行次数与数据个数的函数关系，就可求出该算法的时间复杂度。

在 C 语言表示的算法中，算法的时间复杂度一般与程序执行的步骤数有关系，一般是所有步骤的时间复杂度总和，所以需要对一些语句的运行时间做出估计。例如：

(1) 算数运算时间为 O(1)。

(2) 逻辑预算时间为 O(1)。

(3) 赋值运算时间为 O(1)。

(4) if 语句的运行时间为测试语句运行时间与后续执行语句运行时间的和。

(5) while 语句的运行时间为每次执行循环体的时间与循环次数之积。

(6) for 语句的运行实际与 while 相似。

(7) return 语句的运行时间为 O(1)。

我们希望随着问题规模 n 的增大其时间复杂度趋于稳定地上升，但上升幅度不能太大。常见 T(n)随 n 变化的增长率如图 1.7 所示。

注意

1) 一般常用的时间复杂度的关系

$O(1) \leqslant O(\log_2 n) \leqslant O(n) \leqslant O(n\log_2 n) \leqslant O(n^2) \leqslant O(n^3) \leqslant \cdots \leqslant O(n^k) \leqslant O(2^n)$

其中，2 项和 4 项中的 2 是对数的底，$k \geqslant 3$。

2) 使用 logn 或者 lgn，没有指明底数

原因在于，对于对数，公式 $\log_x y=(\log_m y)/(\log_m x)$成立。其中，x 和 m 是对数的底，而 m 是任何大于 0 且不等于 1 的数。当底数为 10 时，在数学上，习惯简写成 lg。从而有 $O(\log_m n)=O((\log_2 n)/(\log_2 m))=O((\log_3 n)/(\log_3 m))=\cdots$，显然 $\log_2 m$、$\log_3 m$ 是一个常量，可以从括号中提出来，于是有 $O(\log_m n)= O(\log_2 n)= O(\log_3 n)=\cdots$。我们看到，底数不管为 2 还是 3 还是其他任何大于 0 且不等于 1 的数，它们的上界都一样。于是为了统一和简便，都写成 O(logn)。

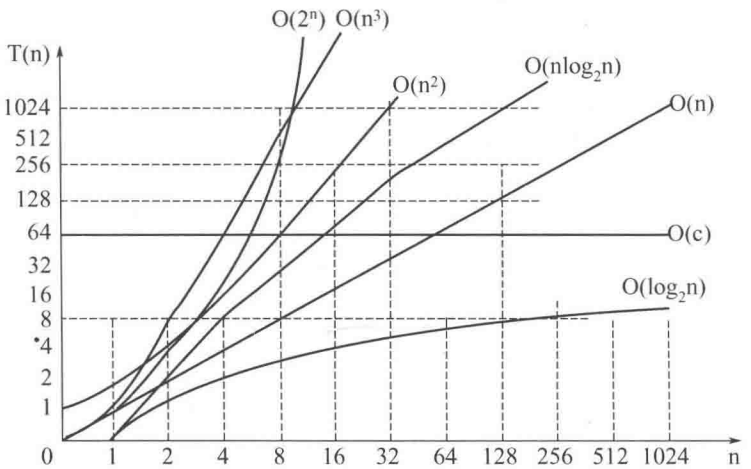


图 1.7　常见的 T(n)随 n 变化的增长率

接下来通过分析一些算法的实例来了解如何运用这些理论。

例 1.1　设数组 a 和 b 在前边部分已赋值，求两个 n 阶矩阵相乘运算算法的时间复杂度。

```
for (int i=0;i<n;i++)
{
   for (int j=0;j<n;j++)
   {
       c[i][j]=0;                                    //基本语句 1
       for (int k=0;k<n;k++)
          c[i][j]=c[i][j]+a[i][k]*b[k][j];           //基本语句 2
   }
}
```

解：设基本语句的执行次数为 f(n)，有 $f(n)=c_1n^2+c_2n^3$。因 $T(n)=f(n)=c_1n^2+c_2n^3=cn^3$，其中 c_1、c_2、c 可为任意常数，所以该算法的时间复杂度为 $O(n^3)$。

例 1.2　求下面程序段的时间复杂度。

```
int i = 1;
while (i <= n)
{
    i = i * 3;
}
```

解：设第一次循环 k=1，此时 i=1*3；第二次循环 k=2，此时 $i=1*3*3=3^2$；以此类推，第 T 次循环 k=T，此时 $i=3^T$。又 $i \leqslant n$，即 $3^T \leqslant n$，对此方程两边取对数，则 $T \leqslant \log_3 n$(关键是求 while 循环的次数)。从而该程序的时间复杂度为 $O(\log_3 n)$。

本 章 小 结

本章是为以后各章讨论的内容作基本知识的准备，介绍了数据结构和算法等基本概念。数据结构是由若干特性相同的数据元素构成的集合，并且在集合上存在一种或多种关系。根据关系的不同，可将数据结构分为 4 类：集合结构、线性结构、树形结构和图状结构。数据的存储结构是数据逻辑结构在计算机中的映像。由于数据是计算机操作对象的总称，它是计算机处理的符号的集合，集合中的个体为一个数据元素。数据元素可以是不可分割的原子，也可以由若干数据项组成，因此在数据结构中讨论的基本单位是数据元素，而最小单位是数据项。

我们可以用图 1.8 来说明数据对象、数据元素和数据项之间的关系。

由两种映像方法我们可以得到两类存储结构：一类是顺序存储结构，它以数据元素相对的存储位置表示关系，则存储结构中只包含数据元素本身的信息；另一类是链式存储结构，它以附加的指针信息(后继元素的存储地址)表示关系。

数据结构的操作是和数据结构本身密不可分的，两者作为一个整体可用抽象数据类型进行描述。抽象数据类型是一个数学模型以及定义在该模型上的一组操作，因此它和高级程序设计语言中的数据类型具有相同含义，而抽象数据类型的范畴更广，它不局限于现有

程序设计语言中已经实现的数据类型(它们通常被称为固有数据类型)，但抽象数据类型需要借用固有数据类型表示并实现。抽象数据类型的三大要素为数据对象、数据关系和基本操作，同时数据抽象和数据封装是抽象数据类型的两个重要特性。

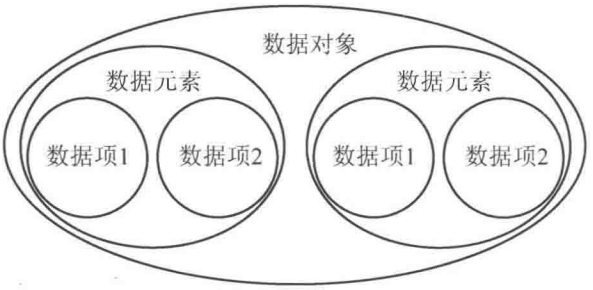


图 1.8　数据结构关系示意图

算法是进行程序设计的另一不可缺少的要素。算法是对问题求解的一种描述，是为解决一个或一类问题给出的一种确定规则的描述。一个完整的算法应该具有下列 5 个要素：有穷性、确定性、可行性、输入和输出。一个正确的算法应对苛刻且带有刁难性的输入数据也能得出正确的结果，并且对不正确的输入也能做出正确的反映。

算法的时间复杂度是比较不同算法效率的一种准则，算法时间复杂度的估算基于算法中基本操作的重复执行次数，或处于最深层循环内的语句频度。算法空间复杂度可作为算法所需存储量的一种量度，它主要取决于算法的输入量和辅助变量所占空间，若算法的输入仅取决于问题本身而和算法无关，则算法空间复杂度的估算只需考察算法中所用辅助变量所占空间。

习题与思考

1.1　单选题

1．算法的计算量的大小称为计算的(　　)。

A．效率　　B．复杂性　　C．现实性　　D．难度

2．计算机算法指的是(1　　)，它必须具备(2　　)这 3 个特性。

(1) A．计算方法　　B．排序方法

C．解决问题的步骤序列　　D．调度方法

(2) A．可执行性、可移植性、可扩充性　　B．可执行性、确定性、有穷性

C．确定性、有穷性、稳定性　　D．易读性、稳定性、安全性

3．从逻辑上可以把存储结构分为(　　)两大类。

A．动态结构、静态结构　　B．顺序结构、链式结构

C．线性结构、非线性结构　　D．初等结构、构造型结构

4．连续存储设计时，存储单元的地址(　　)。

A．一定连续　　B．一定不连续

C．不一定连续　　D．部分连续，部分不连续

5．在下面的程序段中，对 x 的赋值语句的频度为(　　)。