普通高等教育计算机规划教材

数据结构与算法

第 3 版

张小莉　王　苗　罗文劼　编著

提供电子教案和习题解答

下载网址 http://www.cmpedu.com

机械工业出版社
CHINA MACHINE PRESS

普通高等教育计算机规划教材

数据结构与算法

第3版

张小莉　王　苗　罗文劼　编著

机械工业出版社

本书共包括 8 章内容，详细讲述了线性结构、树结构和图结构中的数据表示及数据处理的方法，并对查找和排序两种重要的数据处理技术做了细致的探讨。书中对每一类数据结构的分析均按照“逻辑结构—存储结构—基本运算的实现—时空性分析—典型实例—知识小结—扩展知识导读—练习题—实验题”的顺序来进行，算法全部采用 C 语言描述，很容易转换成程序。本书语言叙述通俗易懂，由浅入深，算法可读性好，应用性强。书中配有大量算法设计的例子，以便于读者理解和掌握数据结构中数据表示和数据处理的方法。

本书可作为高等院校计算机和信息类相关专业“数据结构”课程的教材也可作为高职高专同类专业的教学用书及各类工程技术人员的参考用书。

本书配有电子教案和习题解答，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：2399929378，电话：010-88379753）。

图书在版编目（CIP）数据

数据结构与算法 / 张小莉，王苗，罗文劼编著. —3 版. —北京：机械工业出版社，2014.2
普通高等教育计算机规划教材
ISBN 978-7-111-45795-4

Ⅰ. ①数… Ⅱ. ①张… ②王… ③罗… 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

中国版本图书馆 CIP 数据核字（2014）第 026065 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：和庆娣 师沫迪
责任印制：李 洋
北京华正印刷有限公司印刷
2014 年 4 月第 3 版 · 第 1 次印刷
184mm × 260mm · 19.25 印张 · 476 千字
0001—3000 册
标准书号：ISBN 978-7-111-45795-4
定价：42.00 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	网络服务
社 服 务 中 心：（010）88361066	教 材 网：http://www.cmpedu.com
销 售 一 部：（010）68326294	机工官网：http://www.cmpbook.com
销 售 二 部：（010）88379649	机工官博：http://weibo.com/cmp1952
读者购书热线：（010）88379203	**封面无防伪标均为盗版**

出 版 说 明

信息技术是当今世界发展最快、渗透性最强、应用最广的关键技术，是推动经济增长和知识传播的重要引擎。在我国，随着国家信息化发展战略的贯彻实施，信息化建设已进入了全方位、多层次推进应用的新阶段。现在，掌握计算机技术已成为21世纪人才应具备的基础素质之一。

为了进一步推动计算机技术的发展，满足计算机学科教育的需求，机械工业出版社聘请了全国多所高等院校的一线教师，进行了充分的调研和讨论，针对计算机相关课程的特点，总结教学中的实践经验，组织出版了这套“普通高等教育计算机规划教材”。

本套教材具有以下特点：

1）反映计算机技术领域的新发展和新应用。

2）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、多媒体光盘、课程设计和毕业设计指导等内容。

3）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。

4）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，强调知识、能力与素质的综合训练。

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和自学用书。

希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

“数据结构”是计算机及相关专业的一门重要的专业基础课，是介于“数学”“计算机硬件”和“计算机软件”之间的一门计算机科学与技术领域的核心课程，同时数据结构技术也被广泛应用于信息科学、系统工程、应用数学以及各种工程技术领域。该课程主要介绍如何合理地组织和表示数据、如何有效地存储和处理数据、如何正确地设计算法以及对算法的优劣进行分析和评价。

本书结合编者多年教学经验，在《数据结构与算法》（第2版）的基础上进行了修订，仍然定位于应用型本科层次，坚持以“面向应用，易教易学”为目标，数据结构和算法设计简单明了，语言叙述通俗易懂，讲解由浅入深，并在以下几方面有所改进。

1）章节结构的调整。为突出应用型本科的学习特点，使内容结构更合理，更有利于知识的前后贯通，加强了应用数据结构的能力的培养，对部分章节内容进行了改变。如在第1章中加入对递归内容的介绍和相关的练习及实验；将堆栈和队列并入基本线性结构中介绍；删掉了一般讲不到、难度较大或应用性差的一些问题，强调更常用的或应用性强的知识；增加了第8章扩展应用部分。

2）采用问题引入的方式，引导学生思考，使学生更快、更自然地进入到教学内容的学习中。主要章节的开始采用问题引入，从常识性或典型问题入手，启发学生思考，引起学生的学习兴趣，使学生对如何学习这一章，一开始就有明确的认识。

3）提供较多有针对性的示例，帮助学生灵活掌握知识要点。每一章讲解基本知识之后，都列举一些对应的应用问题，给出典型例题的分析与解决，帮助学生理解和掌握本章节的知识点。

4）丰富的配套练习。每章除了理论教学内容外，在每章的结尾有对本章知识点的总结，帮助学生回顾本章的内容，掌握学习重点。此外还包括练习题和实验题，帮助学生通过练习和实验全面掌握所学习的知识点。本书最后还给出实验要求、模拟试题和参考文献，为学生提供实验课程的指导和辅助学习的资料。

本书由罗文劼教授组织并统稿。其中第1、2、3章由张小莉和王苗共同编写，第4、5、8章由罗文劼编写，第6、7章由王苗编写。在本书的编写过程中，石强、苗秀芬、王硕等多位从事“数据结构”教学的老师对此书的编写提出了宝贵的意见和建议，在此一并表示感谢。

由于编者水平有限，书中难免存在疏漏之处，恳请各位读者批评指正。

编　者

目　录

第1章 绪 论

计算机科学是一门研究数据表示和数据处理的科学。数据就是对客观事务采用计算机能够识别、存储和处理的符号的表示。简言之，数据是计算机化的信息，是计算机可以直接处理的最基本和最重要的对象。无论是进行科学计算或数据处理、过程控制，还是对文件的存储和检索等计算机应用，都是对数据进行加工处理的过程。计算机对数据的处理并不是简单地将数据堆积在一起，而是使其具有某种内在的联系。因此，为了更有效地处理数据，设计出好的算法，编写结构清晰而且效率高的程序，必须研究数据的特性、数据间的相互关系及其对应的存储表示，并利用这些特性和关系设计出相应的算法和程序。

1.1 引言

“数据结构”是计算机科学与技术专业的专业基础课，是重要的核心课程。数据结构的知识为后续专业课程的学习提供必要的知识和技能准备，打好“数据结构”这门课程的扎实基础，对于学习计算机专业的其他课程，如“操作系统”“编译原理”“数据库管理系统”“软件工程”“人工智能”等都是十分有益的。而且，所有的计算机系统软件和应用软件都要用到各种类型的数据结构。因此，要想有效地使用计算机、充分发挥计算机的性能，必须学习和掌握好数据结构的有关知识。

1.1.1 学习数据结构的目的

在计算机发展的初期，人们使用计算机的目的主要是处理数值计算问题。使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先要从该具体问题抽象出一个适当的数学模型，然后设计或选择一个解此数学模型的算法，最后编出程序进行调试、测试，直至得到最终的解答。由于当时所涉及的运算对象是简单的整型、实型或布尔类型，数据量小且结构简单，所以程序设计者的主要精力是集中于程序设计的技巧上，而无须关心如何组织数据。

随着计算机应用领域的日益广泛和软、硬件技术的发展，非数值计算问题显得越来越重要。据资料统计，当今处理非数值计算性问题占用了 90%以上的机器时间。这类问题所涉及的数据结构更为复杂，数据元素之间的相互关系一般无法用数学方程式加以描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是要设计出合适的数据结构，才能有效地解决问题。下面所列举的例子就是属于这一类问题。

【例 1-1】 成绩检索系统。要求成绩检索系统提供自动查询的功能，如查询某个学生的单科成绩或平均成绩，查询某门课程的最高分等。

实现这个系统，首先需要考虑如何组织数据，然后再按照相应的算法编写程序，就可以实现计算机自动检索。例如，可以将每个学生的各项信息（学号、姓名、各项成绩等）用某种构造的数据类型表示，全部学生的信息按学号的次序排列，可以组织成一个线性表格，如表 1-1 所示，根据查询的需要可以设计出各种查询的算法。

表 1-1　学生成绩表

学号	姓名	考试成绩			平均成绩
		高等数学	C 语言	英语	
20131801	吴承志	90	95	85	90
20131802	李淑芳	88	76	91	85
20131803	刘　丽	92	78	82	84
20131804	张会友	81	78	72	77
20131805	石宝国	76	82	79	79
20131806	何文颖	86	90	91	89
20131807	赵胜利	76	78	80	78
20131808	崔文靖	82	93	86	87
20131809	刘　丽	80	85	81	82
…	…	…	…	…	…

类似的还有电话自动查号系统、图书信息检索系统、仓库库存管理系统等。在这类文档管理的数学模型中，计算机处理的对象之间通常存在一种简单的线性关系，这类数学模型可称为线性的数据结构。

【例 1-2】 棋盘布局问题。要求将 4 个棋子布在 4 行 4 列的棋盘上，使得任两个棋子既不在同一行或同一列，也不在同一对角线上。

此问题的处理过程不是根据某种确定的计算法则，而是利用试探和回溯的探索技术求解。为了求得合理布局，在计算机中要存储布局的当前状态。从最初的布局状态开始，一步步地进行试探，每试探一步形成一个新的状态，整个试探过程形成了一棵隐含的状态树，如图 1-1 所示。

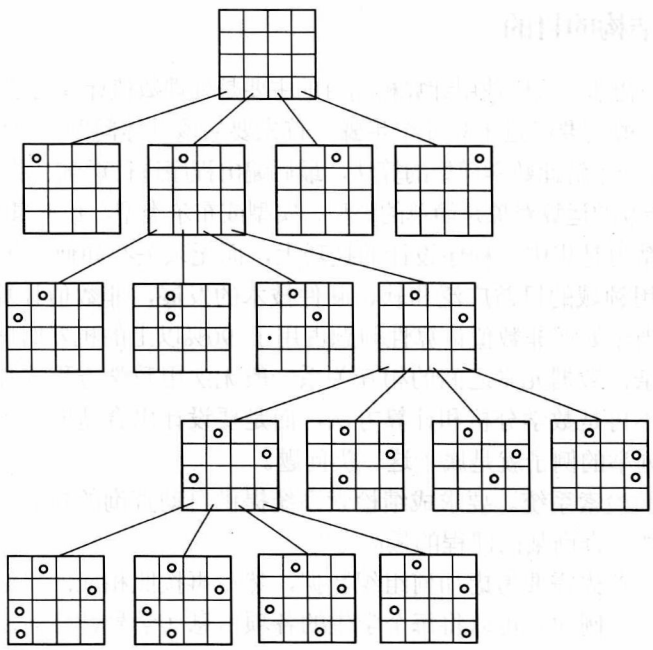

图 1-1　棋盘布局问题中隐含的状态树

回溯法求解过程实质上就是一个遍历状态树的过程。在这个问题中所出现的树也是一种数据结构，它可以应用在许多非数值计算的问题中。

【例 1-3】 教学计划编排问题。一个教学计划包含许多课程，在教学计划包含的许多课程之间，有些必须按规定的先后次序进行，有些则没有次序要求。即有些课程之间有先修和后续的关系，有些课程可以任意安排次序，如表 1-2 所示。

表 1-2 计算机专业的课程设置

课 程 号	课 程 名 称	先 修 课 程
c_1	计算机导论	无
c_2	数据结构	c_1，c_4
c_3	汇编语言	c_1
c_4	C 程序设计语言	c_1
c_5	计算机图形学	c_2，c_3，c_4
c_6	接口技术	c_3
c_7	数据库原理	c_2，c_9
c_8	编译原理	c_4
c_9	操作系统	c_2

各个课程之间的次序关系可用一个称作图的数据结构来表示，如图 1-2 所示。有向图中的每个顶点表示一门课程，如果从顶点 c_i 到 c_j 之间存在有向边<c_i，c_j>，则表示课程 i 必须先于课程 j 进行。

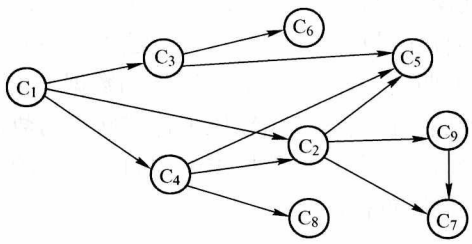

图 1-2 教学计划编排问题中表示课程间优先关系的有向图

由以上 3 个例子可见，描述这类非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。因此，可以说数据结构课程主要是研究非数值计算的程序设计问题中所出现的计算机操作对象以及它们之间的关系和操作的学科。

学习数据结构的目的是为了了解计算机处理对象的特性，将实际问题中所涉及的处理对象在计算机中表示出来并对它们进行处理。与此同时，通过算法训练来提高学生的逻辑思维能力，通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

1.1.2 “数据结构”课程的内容

数据结构与数学、计算机硬件和软件有十分密切的关系。“数据结构”是介于“数学”“计算机硬件”和“计算机软件”之间的一门计算机科学与技术专业的核心课程，是“高级程序设计语言”“编译原理”“操作系统”“数据库”“人工智能”等课程的基础。同时，数据结构技术也广泛应用于信息科学、系统工程、应用数学以及各种工程技术领域。

“数据结构”课程集中讨论软件开发过程中的设计阶段，同时涉及编码和分析阶段的若干基本问题。此外，为了构造出好的数据结构及其实现，还需考虑数据结构及其实现的评价与选择。因此，数据结构的内容包括三个层次的五个“要素”，如表 1-3 所示。

表 1-3 “数据结构”课程内容体系

层次＼方面	数据表示	数据处理
抽象	逻辑结构	基本运算
实现	存储结构	算法
评价	不同数据结构的比较及算法分析	

数据结构的核心技术是分解与抽象。通过对问题的抽象，舍弃数据元素的具体内容，就得到逻辑结构。类似地，通过分解将处理要求划分成各种功能，再通过抽象舍弃实现细节，就得到运算的定义。上述两个方面的结合使我们将问题变换为数据结构。这是一个从具体（即具体问题）到抽象（即数据结构）的过程。然后，通过增加对实现细节的考虑进一步得到存储结构和实现运算，从而完成设计任务。这是一个从抽象（即数据结构）到具体（即具体实现）的过程。熟练地掌握这两个过程是数据结构课程在专业技能培养方面的基本目标。

“数据结构”作为一门独立的课程在国外是从 1968 年才开始的，但在此之前其有关内容已散见于编译原理及操作系统之中。20 世纪 60 年代中期，美国的一些大学开始设立有关课程，但当时的课程名称并不叫数据结构。1968 年美国唐·欧·克努特教授开创了数据结构的最初体系，他所著的《计算机程序设计技巧》第一卷《基本算法》是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作。从 20 世纪 60 年代末到 70 年代初，出现了大型程序，软件也相对独立，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。从 20 世纪 70 年代中期到 80 年代，各种版本的数据结构著作相继出现。目前，数据结构的发展并未终结，一方面，面向各专门领域中特殊问题的数据结构得到研究和发展，如多维图形数据结构等；另一方面，从抽象数据类型和面向对象的观点来讨论数据结构已成为一种新的趋势，越来越被人们所重视。

1.2 数据结构的概念

在系统地学习数据结构知识之前，先对一些基本概念和术语赋予确切的含义。

1.2.1 基本概念和术语

1. 数据

数据（Data）是信息的载体，是所有能够被计算机识别、存储和加工处理的符号的总称。它是计算机程序加工的原料，应用程序可以处理各种各样的数据。在计算机科学中，数据是计算机加工处理的对象，它可以是数值数据，也可以是非数值数据。数值数据是一些整数、实数和复数，主要用于工程计算、科学计算和商务处理等；非数值数据包括字符、文字、图形、图像、语音等。

2. 数据项

数据项（Data Item）是具有独立含义的标识单位，是数据不可分割的最小单位。如学生

成绩表中“学号”“姓名”等。数据项有名和值之分，数据项名是一个数据项的标识，用变量定义，而数据项值是它的一个可能取值，如表 H 中“20131801”是数据项“学号”的一个取值。数据项具有一定的类型，依数据项的取值类型而定。

3. 数据元素

数据元素（Data Element）是数据的基本单位。在不同的条件下，数据元素又可称为元素、结点、顶点、记录等。例如，考试查分系统的学生成绩表中的一个记录、棋盘布局问题中状态树的一个状态、教学计划编排问题中的一个顶点等，都被称为一个数据元素。

有时，一个数据元素可由若干个数据项组成，例如，学籍管理系统中学生信息表的每一个数据元素就是一个学生记录。它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。这些数据项可以分为两种：一种叫做初等项，如学生的性别、籍贯等，这些数据项是在数据处理时不能再分割的最小单位；另一种叫做组合项，如学生的成绩，它可以再划分为数学、物理、化学等更小的项。通常，在解决实际应用问题时是把每个学生记录当做一个基本单位进行访问和处理的。

4. 数据对象

数据对象（Data Object）或数据元素类（Data Element Class）是具有相同性质的数据元素的集合。在某个具体问题中，数据元素都具有相同的性质（元素值不一定相等），属于同一数据对象（数据元素类），数据元素是数据元素类的一个实例。例如，在交通咨询系统的交通网中，所有的顶点是一个数据元素类，顶点 A 和顶点 B 各自代表一个城市，是该数据元素类中的两个实例，其数据元素的值分别为 A 和 B。

5. 数据结构

数据结构（Data Structure）是指互相之间存在着一种或多种关系的数据元素的集合。数据结构涉及数据元素之间的逻辑关系，数据在计算机中的存储方式和这些数据上定义的一组运算，一般称这 3 个方面为数据的逻辑结构、数据的存储结构和数据的运算。

（1）数据的逻辑结构

在任何问题中，数据元素之间都不会是孤立的，在它们之间都存在着这样或那样的逻辑关系，这种数据元素之间的关系称为逻辑结构。数据的逻辑结构包含两个要素：一个是数据元素的集合，另一个是关系的集合。

在形式上，数据的逻辑结构通常可以采用一个二元组来表示：

$$\text{Data\_Structure} = (D, R)$$

其中，D 是数据元素的有限集，R 是 D 上关系的有限集。

根据数据元素间关系的不同特性，数据的逻辑结构通常分为以下四类。

1）集合：在集合中，数据元素间的关系是“属于同一个集合”。集合是元素关系极为松散的一种结构。

2）线性结构：该结构的数据元素之间存在着一对一的关系。

3）树形结构：该结构的数据元素之间存在着一对多的关系。

4）图形结构：该结构的数据元素之间存在着多对多的关系，图形结构也称作网状结构。图 1-3 所示为四类逻辑结构的示意图。

由于集合是数据元素之间关系极为松散的一种结构，因此也可用其他结构来表示它。

数据的逻辑结构可以看做是从具体问题抽象出来的数学模型，它与数据的存储无关。我

们研究数据结构的目的是为了在计算机中实现对它的操作，为此还需要研究如何在计算机中表示一个数据结构。数据结构在计算机中的表示（又称映像）称为数据的物理结构，又称存储结构。它所研究的是数据的逻辑结构在计算机中的实现方法，包括数据结构中元素的表示及元素间关系的表示。

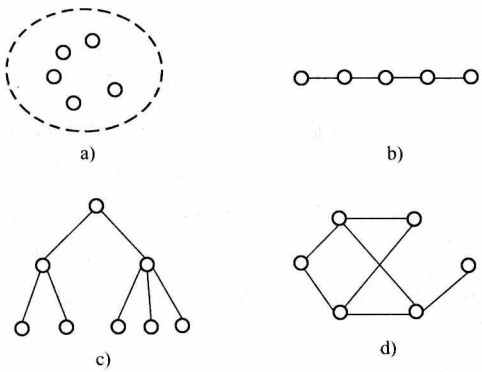

图 1-3　四类逻辑结构的示意图

a) 集合结构　b) 线性结构　c) 树形结构 d) 图形结构

（2）数据的存储结构

数据的存储结构最常用的是顺序存储和链式存储的方法。

1）顺序存储方法：是把逻辑上相邻的元素存储在物理位置相邻的存储单元中，结点间的逻辑关系由存储单元的邻接关系来体现，由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。

2）链式存储方法：对逻辑上相邻的元素不要求其物理位置相邻，元素间的逻辑关系通过附设的指针字段来表示，由此得到的存储表示称为链式存储结构，链式存储结构通常借助于程序设计语言中的指针类型或引用类型来实现。

除了顺序存储方法和链式存储方法外，有时为了查找的方便还采用索引存储方法和散列存储方法。

3）索引存储方法：是在存储结点信息的同时，还建立附加的索引表。索引表中的每一项包含关键字和地址，关键字是能够唯一标识一个数据元素的数据项，地址指示出数据元素所在的存储位置。索引存储主要是针对数据内容的存储，而不强调关系的存储，索引存储方法主要面向查找操作。

4）散列存储方法：是以数据元素的关键字的值为自变量，通过某个函数（散列函数）计算出该元素的存储位置。索引存储也是针对数据内容的存储方式。

以上四种存储方法中，顺序存储方法和链式存储方法是最基本最常用的，索引存储方法和散列存储方法在具体实现时需要用到前两种方法。在实际应用中，一种逻辑结构可以有不同的存储方法，选用何种存储结构来表示相应的逻辑结构要视具体情况而定，主要考虑运算的实现及算法的时空要求。

（3）数据的运算

运算是对数据的处理。运算与逻辑结构紧密相连，每种逻辑结构都有一个运算的集合。运算的种类很多，根据操作的结果，可将运算分为两种类型。

1）引用型运算。这类运算不改变数据结构中原有的数据元素的状态，只根据需要读取某些信息。

2）加工型运算。这类运算的结果会改变数据结构中原有数据的状态，如数据元素的内容、个数等。

数据的运算是定义在数据的逻辑结构上的，但运算的具体实现是在数据的存储结构上进行的。数据的运算是数据结构不可分割的一个方面，在数据的逻辑结构和存储结构给定之后，如果定义的运算集及运算的性质不同，也会导致完全不同的数据结构，例如后面章节中将要介绍的线性表、栈和队列等。

1.2.2 抽象数据类型

1．数据类型

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设计语言中，用以刻画程序中操作对象的特性。在用高级语言编写的程序中，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。类型显式地或隐含地规定了在程序执行期间变量或表达式所有可能的取值范围，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。

在高级程序设计语言中，数据类型可分为两类：一类是原子类型；另一类则是结构类型。原子类型的值是不可分解的。如 C 语言中整型、字符型、浮点型、双精度型等基本类型，分别用保留字 int、char、float、double 标识。而结构类型的值是由若干成分按某种结构组成的，因此是可分解的，并且它的成分可以是非结构的，也可以是结构的。例如，数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。在某种意义上，数据结构可以看成是“一组具有相同结构的值”，而数据类型则可被看成是由一种数据结构和定义在其上的一组操作所组成的。

2．抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义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。

抽象数据类型和数据类型实质上是一个概念。例如，各种计算机都拥有的整数类型就是一个抽象数据类型，尽管它们在不同处理器上的实现方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

但在另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型，还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的重用性，在近代程序设计方法学中，要求在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作，并在模块的内部给出这些数据的表示及其操作的细节，而在模块的外部使用的只是抽象的数据及抽象的操作。这也就是面向对象的程序设计方法。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作组成，而数据结构又包括数据元素及元素间的关系，因此抽象数据类型一般可以由元素、关系及操作三种要素来定义。

抽象数据类型的特征是使用与实现相分离，实行封装和信息隐蔽。就是说，在抽象数据类型设计时，把类型的定义与其实现分离开来。

1.3 算法

数据的运算是通过算法来描述的。算法与数据结构的关系紧密，在算法设计时先要确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。下面就从算法的特性、算法的描述、算法的性能分析与度量三个方面对算法进行介绍。

1.3.1 算法的特性

算法（Algorithm）是对特定问题求解步骤的一种描述，是指令的有限序列。其中每一条指令表示一个或多个操作。

一个算法应该具有下列特性。

1）有穷性。一个算法必须在有穷步之后结束，即必须在有限时间内完成。

2）确定性。算法的每一步必须有确切的定义，无二义性。算法的执行对应着相同的输入仅有唯一的一条路经，即相同的输入必然有相同的输出。

3）可行性。算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。

4）输入。一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。

5）输出。一个算法具有一个或多个输出，这些输出同输入之间存在某种特定的关系。

算法的含义与程序十分相似，但又有区别。一个程序不一定满足有穷性。例如，操作系统，只要整个系统不遭破坏，它将永远不会停止，即使没有作业需要处理，它仍处于动态等待中。因此，操作系统不是一个算法。另一方面，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的解，而程序则是算法在计算机上的特定的实现。一个算法若用程序设计语言来描述，则它就是一个程序。

算法与数据结构是相辅相成的。解决某一特定类型问题的算法可以选定不同的数据结构，而且选择恰当与否直接影响算法的效率；反之，一种数据结构的优劣由各种算法的执行来体现。

要设计一个好的算法通常要考虑以下的要求。

1）正确：算法的执行结果应当满足预先规定的功能和性能要求。

2）可读：一个算法应当思路清晰、层次分明、简单明了、易读易懂。算法不仅仅是让机器来执行的，更重要的是便于人的阅读和交流。

3）健壮：当输入不合法数据时，算法能够作适当处理，而不会产生莫名其妙的输出或引起其他严重的后果。

4）高效：指算法应具有较好的时空性能。

这些指标一般很难做到十全十美，因为它们常常相互矛盾，在实际的算法评价中应根据需要有所侧重。

1.3.2 算法的描述

算法可以使用各种不同的方法来描述。

最简单的方法是使用自然语言，优点是简单且便于人们对算法的阅读，缺点是不够严谨。

可以使用程序流程图、N-S 图等算法描述工具，其优点是描述过程简洁、明了，缺点是

不能够直接在计算机上执行，若要将它转换成可执行的程序，则还有一个编程的问题。

也可以直接使用某种程序设计语言来描述算法，不过直接使用程序设计语言并不容易，而且不太直观，常常需要借助于注释才能使人看明白。

为了解决理解与执行这两者之间的矛盾，人们常常使用一种称为伪码语言的描述方法来进行算法描述。伪码语言介于高级程序设计语言和自然语言之间，它忽略高级程序设计语言中一些严格的语法规则与描述细节，因此它比程序设计语言更容易描述和被人理解，而比自然语言更接近程序设计语言。它虽然不能直接执行但很容易被转换成高级语言。

本书采用类 C 语言作为算法的描述工具，这使得算法的描述简洁、清晰，不必拘泥于 C 语言的细节，又容易转换成 C 语言或 C++语言的程序。

1.3.3 算法的性能分析与度量

在程序设计中，对算法进行分析与度量是非常重要的。解决一个具体的应用实例，常常有若干个算法可以选用，因此程序设计者要判断哪一个算法在现实的计算机环境中对于解决问题是最优的。在计算机科学中，一般从算法的计算时间与所需存储空间来评价一个算法的优劣。

1．时间复杂度

将一个算法转换成程序并在计算机上执行时，其运行所需要的时间取决于下列因素：

1）硬件的速度。主要指机器的指令性能和速度，例如 64 位机一般要比 32 位机快；主频 2GHz 的机器一般比 1GHz 的机器快。

2）书写程序的语言。实现语言的级别越低，其执行效率就越高，如汇编语言的执行效率一般高于高级语言，C/C++语言的执行效率一般高于 Visual Basic 语言等。

3）编译程序所生成目标代码的质量。对于代码优化较好的编译程序其所生成的目标代码的质量较高。

4）问题的规模。例如，求 100 以内的素数与求 1000 以内的素数其执行时间必然是不同的。

显然，在各种因素都不能确定的情况下，很难比较出算法的执行时间。也就是说，使用执行算法的绝对时间来衡量算法的效率是不合适的。因此，为了突出算法本身的性能，可以将上述各种与计算机相关的软、硬件因素都确定下来，这样一个特定算法的运行工作量的大小就只依赖于问题的规模（通常用正整数 n 表示），或者说它是问题规模的函数。

一个算法的时间复杂度（Time Complexity）就是指算法的时间耗费，这里用 T(n) 表示。

一个算法执行所耗费的时间，是算法中所有语句执行时间之和，而每条语句的执行时间是该语句执行一次所用时间与该语句重复执行次数的乘积。一个语句重复执行的次数称为语句的频度（Frequency Count）。

算法的时间复杂度 T(n)表示为：$T(n)=\sum_{\text{语句}i}(t_i \times c_i)$

其中 t_i 表示语句 i 执行一次的时间，c_i 表示语句 i 的频度。

假设每条语句执行一次的时间均为一个单位时间，那么算法的时间耗费可简单表示为各语句的频度之和：$T(n)=\sum_{\text{语句}i} c_i$