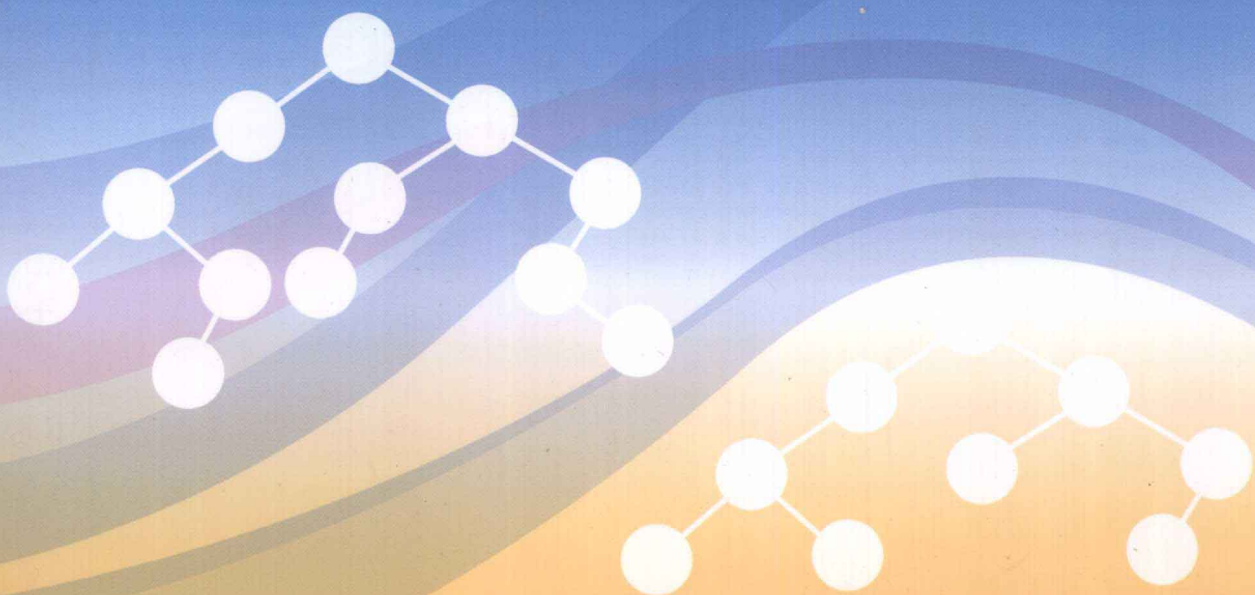


高等学校计算机专业规划教材

新编数据结构 及算法教程



林碧英 主编
石 敏 焦润海 编著



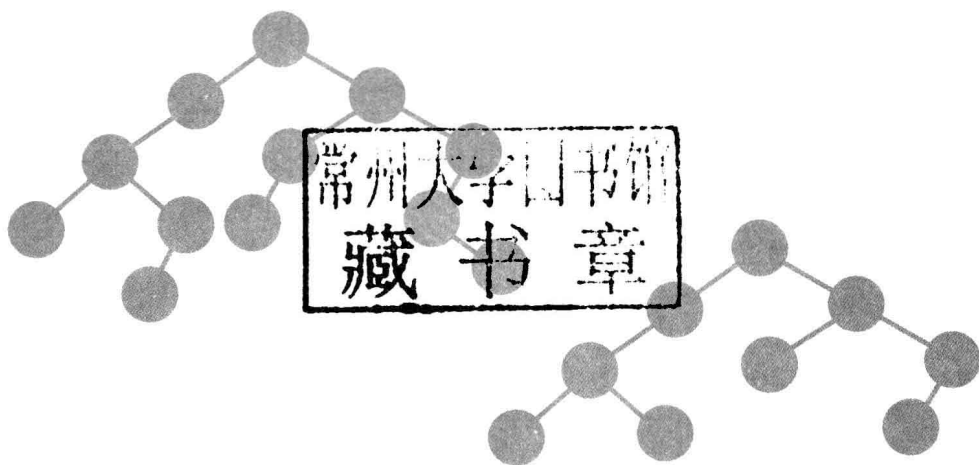
清华大学出版社

等学校计算机专业规划教材

新编数据结构 及算法教程

林碧英 主编

石 敏 焦润海 编著



清华大学出版社
北 京

内 容 简 介

本书介绍了数据结构的基本概念、基本知识以及数据结构的应用。全书按照三部分编写。第一部分是线性结构,包括线性表、栈与队列、数组和特殊矩阵;第二部分是非线性结构,包括树和二叉树、图;第三部分是数据处理技术,包括查找和排序,内容涵盖了全国硕士研究生计算机综合考试课程的数据结构知识。

本书适合作为各类高等院校、高等职业技术学校与计算机相关的各类专业的数据结构与算法的教学用书,也是从事软件设计人员一本难得的参考书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

新编数据结构及算法教程/林碧英主编. —北京:清华大学出版社,2012.9

(高等学校计算机专业规划教材)

ISBN 978-7-302-29370-5

I. ①新… II. ①林… III. ①数据结构—高等学校—教材 ②算法分析—高等学校—教学参考资料 IV. ①TP311.12

中国版本图书馆 CIP 数据核字(2012)第 158276 号

责任编辑:龙啟铭

封面设计:常雪影

责任校对:梁 毅

责任印制:李红英

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:北京市人民文学印刷厂

装 订 者:三河市深源装订厂

经 销:全国新华书店

开 本:185mm×260mm

印 张:25.75

字 数:614 千字

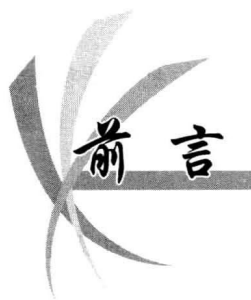
版 次:2012 年 9 月第 1 版

印 次:2012 年 9 月第 1 次印刷

印 数:1~3000

定 价:39.50 元

产品编号:048357-01



“数据结构”是计算机及相关专业的一门核心专业基础课,在计算机课程的教学计划中,它起着核心主导、承上启下的作用,是培养学生程序设计能力的一门重要课程,也是计算机及相关专业的大学生应聘、考研的一门必需课程。

长期以来,数据结构课程由于其抽象与动态性,使得在教学过程中始终存在“能听懂,编不了程,不会应用”的现象。目前大部分学生在学数据结构之前,只学了一门程序设计语言。受学时的限制,数据结构需要的链表基础,学生知之甚少,几乎没有编写过有关链表的程序,因此造成了学生学习“数据结构”的困难。另外,目前国内外的很多数据结构教科书都是在抽象层次上阐述,它们在介绍典型数据结构之前,虽然也使用了一些实际问题作为切入点,但最后并没有针对这些问题给出具体的解决方案,学生还是没有弄清楚为什么要学习这种结构,怎样用这种结构来解决实际问题。

由于以上问题,我们认为数据结构的教学改革势在必行。为此,我们对传统的教学方法进行改革尝试,以应用为主线,用案例驱动的教学方式,对每一种典型的数据结构教学采用以下方法:

- (1) 引入实际问题。
- (2) 分析实际问题中的数据和数据之间的关系,以及对数据需要做的常用操作。
- (3) 抽象出实际问题涉及数据的逻辑结构、存储结构和基本操作的设计与实现。
- (4) 用所学的数据结构设计和实现解决这些实际问题的算法。
- (5) 编码、调试并分析运行结果。

上述案例驱动的教学模式,我们已经实践了4年,取得了可喜的成绩。我们边教学边总结,编写了这本以案例驱动为主导,以解决实际问题为主线,以激发学生为学习热情的教材,目的是使得原本难教难学的课程变得生动形象,更贴近实际,从而切实提高学生的程序设计能力。

此外,本教材还具有以下特点:

- (1) 加强对基本操作实现的函数形参的分析。每个基本操作都配以示意图,显示存放在内存中的数据对象在操作完成前后的变化,分析操作的对象,需要的输入和输出,以及操作是否引起数据对象的变化,帮助读者熟练掌握基本操作的设计与实现。



(2) 数据结构中的很多算法涉及递归操作,初学者很难理解递归算法的执行过程,为此我们对较难理解的递归算法,配以图表,揭示每一步的变化,将抽象的想象变为可以看到的具體过程,提升读者对递归算法的设计能力。

(3) 对复杂的算法,我们用实际数据,采用图表结合的方式,将存储结构的变化和逻辑结构的变化同步展现,加强读者对算法的理解,使读者进一步掌握复杂算法的设计与实现。

(4) 各章内容均与授课对象进行过多次交流,广泛听取学生的意见,及时进行内容的更新和修改。

全书共8章,由林碧英老师统一编排、审核。第1章、第7章和第8章由石敏老师编写;第2章和第3章由焦润海老师编写;第4章、第5章和第6章由林碧英老师编写。各章的习题由苏辰隽、莫瑞芳整理。

我们只是在案例教学做了一些尝试,在教材的编写中可能还有很多不尽人意的地方,恳请广大读者多提宝贵意见。

编写组

2012年8月于北京



第 1 章 绪论 /1

1.1	数据结构的起源与发展	1
1.2	基本概念和术语	2
1.3	理解数据结构	3
1.4	数据的逻辑结构和存储结构	4
1.4.1	逻辑结构	5
1.4.2	存储结构	6
1.5	抽象数据类型	8
1.5.1	数据类型	8
1.5.2	抽象数据类型	8
1.6	算法分析与评价	11
1.6.1	数据结构与算法的关系	11
1.6.2	算法的定义	11
1.6.3	算法的 5 大特性	11
1.6.4	算法设计的要求	12
1.6.5	算法效率分析	13
1.6.6	算法的时间复杂度	14
1.6.7	算法存储空间需求	16
1.7	本章小结	17
1.8	习题	17

第 2 章 线性表 /20

2.1	问题的提出	20
2.1.1	问题中的数据分析	20
2.1.2	问题中的功能分析	21
2.1.3	问题中的数据结构	22
2.2	线性表	22
2.2.1	线性表的定义	22
2.2.2	线性表的存储结构和基本操作的实现	24
2.2.3	线性表的两种存储结构的区别	47

2.3	案例实现	48
2.3.1	基于顺序表的新生成绩管理系统	48
2.3.2	基于单向链表的新生成绩管理系统	52
2.4	其他形式的链表	54
2.4.1	单向循环链表	54
2.4.2	双向循环链表	57
2.5	线性表的应用	60
2.5.1	两个线性表的合并	60
2.5.2	一元多项式的应用	63
2.6	本章小结	69
2.7	习题与实验	70

第3章 栈与队列 /74

3.1	问题的提出	74
3.1.1	问题中的数据分析	74
3.1.2	问题中的功能分析	74
3.1.3	问题中的数据结构	75
3.2	栈	76
3.2.1	栈的定义	76
3.2.2	栈的存储结构和基本操作的实现	77
3.2.3	栈的两种存储结构的区别	87
3.2.4	案例实现：基于栈的括号匹配	87
3.3	栈的应用	89
3.3.1	表达式求值	89
3.3.2	栈与递归	94
3.4	队列	103
3.4.1	队列的定义	103
3.4.2	队列的存储结构和基本操作的实现	105
3.4.3	队列的两种存储结构的区别	116
3.4.4	案例实现：基于队列的医院挂号模拟系统	116
3.5	队列的应用	120
3.6	共用栈和双队列	124
3.6.1	共用栈	124
3.6.2	双端队列	126
3.7	本章小结	127
3.8	习题与实验	127



第4章 数组和特殊矩阵 /133

4.1	多维数组	133
4.1.1	数组的逻辑结构	133
4.1.2	数组的内存映像	133
4.2	特殊矩阵的压缩存储	136
4.2.1	对称矩阵	136
4.2.2	三角矩阵	138
4.2.3	带状矩阵	139
4.3	稀疏矩阵	140
4.3.1	稀疏矩阵的三元组表存储	140
4.3.2	稀疏矩阵的十字链表存储	146
4.4	本章小结	152
4.5	习题	152

第5章 树和二叉树 /155

5.1	问题的提出	155
5.1.1	问题中的数据分析	155
5.1.2	问题中的功能分析	156
5.1.3	问题中的数据结构	156
5.2	树的定义和基本术语	156
5.2.1	树的递归定义	156
5.2.2	树的基本术语	156
5.2.3	树的表示	158
5.2.4	树的抽象数据类型描述	159
5.3	二叉树	159
5.3.1	二叉树的定义	159
5.3.2	二叉树的性质	161
5.3.3	二叉树的抽象数据类型	162
5.3.4	二叉树的存储结构	163
5.3.5	二叉树的遍历及其应用	166
5.3.6	案例实现: 基于表达式二叉树的动态表达式计算	192
5.4	线索二叉树	192
5.4.1	线索二叉树的定义	193
5.4.2	线索二叉树的基本操作实现	194
5.4.3	基于中序线索二叉树的遍历算法	200
5.5	树、森林与二叉树的转换及其应用	203
5.5.1	树、森林与二叉树的转换	203



5.5.2	树的存储结构	204
5.5.3	树和森林的遍历	209
5.5.4	树的简单应用	210
5.5.5	案例实现:基于树结构的行政机构管理	217
5.6	哈夫曼树及其应用	220
5.6.1	最优二叉树——哈夫曼树	220
5.6.2	哈夫曼树及哈夫曼编码的构建算法	224
5.7	本章小结	229
5.8	习题与实验	229

第6章 图 /234

6.1	问题的提出	234
6.1.1	问题中的数据分析	235
6.1.2	问题中的功能分析	235
6.1.3	问题中的数据结构	235
6.2	图的定义和基本术语	235
6.2.1	图的定义	235
6.2.2	图的基本术语	235
6.2.3	图的分类与连通性	237
6.2.4	图的抽象数据类型定义	238
6.3	图的存储结构	239
6.3.1	图的邻接矩阵表示	240
6.3.2	图的邻接表表示	243
6.3.3	有向图的十字链表表示	246
6.3.4	无向图的邻接多重表表示	247
6.4	图的遍历	249
6.4.1	连通图的深度优先搜索(Depth-First Search)	249
6.4.2	连通图的广度优先搜索(Breadth-First Search)	253
6.4.3	非连通图的深度(广度)优先遍历	255
6.4.4	图的遍历算法应用	255
6.5	图的连通性	261
6.5.1	无向图的连通分量和生成树	261
6.5.2	最小生成树及应用	261
6.6	最短路径	272
6.6.1	求从某个源点到其余各点的最短路径	272
6.6.2	每一对顶点之间的最短路径	278
6.7	有向无环图及其应用	283
6.7.1	拓扑排序	284

6.7.2 关键路径	287
6.7.3 案例实现：教学计划编排系统	290
6.7.4 案例实现：基于有向无环图的表达式计算	295
6.8 本章小结	300
6.9 习题与实验	301

第7章 查找 /305

7.1 问题的提出	305
7.2 基本概念与描述	306
7.2.1 查找的基本概念	306
7.2.2 性能分析	307
7.2.3 内部查找和外部查找	308
7.2.4 C语言描述	308
7.3 线性表查找	308
7.3.1 顺序查找	309
7.3.2 二分查找	310
7.3.3 分块查找	314
7.3.4 案例实现：基于顺序查找的学生信息表查询	316
7.4 树表查找	320
7.4.1 二叉排序树	320
7.4.2 平衡二叉树	328
7.4.3 B-树和 B+树	344
7.4.4 案例实现：基于二叉排序树的学生信息管理	352
7.5 哈希表	357
7.5.1 哈希表概念	357
7.5.2 常用的哈希函数	358
7.5.3 解决冲突的方法	359
7.5.4 哈希表的查找及其性能分析	362
7.6 本章小结	364
7.7 习题与实验	365

第8章 排序 /369

8.1 问题的提出	369
8.2 基本概念	369
8.3 插入排序	371
8.3.1 直接插入排序	371
8.3.2 折半插入排序	373
8.3.3 希尔排序	374



8.4	交换排序	375
8.4.1	冒泡排序.....	375
8.4.2	快速排序.....	377
8.5	选择排序	379
8.5.1	简单选择排序.....	380
8.5.2	堆排序.....	381
8.6	归并排序	384
8.7	基数排序	386
8.7.1	多关键字排序.....	386
8.7.2	链式基数排序.....	387
8.8	案例实现：学生基本信息表的排序	390
8.9	各种内部排序方法的比较	397
8.10	本章小结.....	398
8.11	习题.....	399

参考文献 /402

1.1 数据结构的起源与发展

20 世纪 40 年代,第一台电子计算机问世。由于其产生的最初动力是人们想发明一种能进行科学计算的机器,因此在发展早期,其应用范围几乎只局限于科学和工程的计算,其处理的对象是纯数值型的信息,人们把这类问题称为数值计算问题。

近 30 年来,电子计算机的发展异常迅猛,运算速度不断提高,信息存储量日益扩大,价格逐步下降。更重要的是,计算机的应用范围已远远超出了科学计算的范畴,信息检索、企业管理、系统控制、虚拟现实等多种应用,几乎渗透到了人类社会活动的一切领域。与此相应,计算机的处理对象由早期纯粹的数值信息发展到文字、声音、多媒体以及图像等多样化的非数值信息。这类问题通常被称为非数值计算问题。与数值计算问题相比,非数值计算问题需要处理的数据对象及其相关关系更为复杂,加工数据的程序规模也更加庞大。单凭程序设计人员的经验和技巧已经难以设计出效率高、可靠性强的程序。由于系统中数据的表示方法和组织形式直接影响到系统运行的效率。因此,为了设计出高效的程序,就需要对计算机程序加工的数据对象进行系统地研究,即研究数据的特性以及数据之间存在的关系,此即为**数据结构**(Data Structure)。

1968 年,在图灵奖得主高纳德所著的《计算机程序设计艺术》(The Art of Computer Programming)第一卷《基本算法》中,较系统地阐述了数据的逻辑结构和存储结构及其操作,开创了数据结构的最初体系。同年,“数据结构”作为一门独立的课程在国外正式开始设立。

20 世纪 60 年代末出现了大型程序,软件也相对独立,结构化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,加上设计一种好的算法。数据结构在程序设计中的重要地位也日益凸显。

如今,“数据结构”作为计算机科学中的一门重要的专业基础课,综合了数学、计算机硬件和计算机软件等多学科的研究。它不仅是一般程序设计(特别是非数值性程序设计)的基础,而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序的重要基础。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”已经成为构建计算机类专业课程群的核心课程。

值得注意的是,虽然通过系统的分析与研究,已经总结得到了几种基本类型的数据结构,但数据结构的发展并未终结。据有关统计资料表明,现在计算机用于数据处理的时间

比例达到 80% 以上,随着时间的推移和计算机应用的进一步普及,计算机用于数据处理的时间比例必将进一步增大。因此,针对各专业领域中的特殊问题和特定数据,有必要研究并构建高效的数据结构,如高维图形数据的数据结构和空间数据的数据结构等,以促使快速有效地数据分析与实时的数据处理。

1.2 基本概念和术语

既然数据结构主要研究数据的特性以及数据之间存在的关系,那我们就先来看什么是数据。

1. 数据

数据是能被输入到计算机中,能被计算机识别,且能被计算机加工处理的符号集合,是计算机操作对象的总称。现代计算机科学的观点,数据不仅包括数值信息,如整型、实型等,还包括非数值的信息,如图像、声音、文字等。

例如,游戏中的三维人物角色就是图形数据,而其中播放的声音就是声音数据。换句话说,所谓的数据,是对客观事物的符号表示,这些符号具备两个条件:

- (1) 能输入到计算机中;
- (2) 能被计算机程序处理。

对于数值型数据,可以直接对其处理;而对于非数值的数据,可以通过编码将其变成字符数据来处理。

2. 数据元素

数据元素是数据的一个基本单位,在计算机中通常作为一个整体进行考虑和处理。如整数 5 是整型数据中的一个数据元素。字符‘N’是字符型数据中的一个数据元素。如果数据元素被组织成表结构,也将其称为数据记录。例如,描述一个学生的数据元素可能包含学号、姓名、性别、出生日期等多个属性。于是,一条具体的学生信息:“10001、王小丽、女、1999/1/1”,即是一条数据记录,如表 1-1 所示。

表 1-1 学生信息表

学号	姓名	性别	出生日期
10001	王小丽	女	1999/1/1
10002	李东东	男	2000/10/23
10003	张一毛	男	1998/11/2
10004	于亮	男	2002/10/5
10005	赵小小	女	2003/4/23

3. 数据项

数据元素可以由若干项构成,这些项是构成数据元素的单位,即为数据项。数据项可以是原子项,也可以是组合项。如学生数据元素中包含多个数据项,其中,姓名、学号和性

别都是原子项,不能再进行分割;而出生日期可看作是由年、月、日构成的组合项,它可以分割为更小的数据项。

4. 数据对象(Data Object)

数据对象是性质相同的数据元素的集合,是数据的一个子集。例如,集合 $N=\{0,1,-1,2,-2,\dots\}$ 是整型数据对象, $C=\{'A','B','C',\dots,'Z'\}$ 是字符型数据对象。

5. 数据结构

结构,即关系。通常,实际应用问题中的数据元素之间并不是孤立存在的,而是存在一定的关系。数据元素之间的关系称为结构。所谓数据结构,简单地说,是指相互之间存在着某种逻辑关系的数据元素的集合。

综上所述,不难看出:数据包含数据对象;数据对象包含数据元素;数据元素包含数据项。

1.3 理解数据结构

计算机解决一个具体问题时,大致需要经过下列几个步骤:首先要从具体问题中抽象出一个适当的数学模型,然后设计一个求解此数学模型的算法(Algorithm),最后编出程序,测试和调整,直至得到最终解答。寻求数学模型的实质是分析问题,从中提取操作的数据对象,找出数据对象中的数据元素之间含有的关系,并加以描述。这实际上是对数据结构进行分析与描述。

为了更好地理解数据结构,先举一个简单的例子。

例如,求解一元二次方程 $ax^2+bx+c=0$ 的根,其中 a,b,c 作为已知输入。

这是一个典型的数值计算问题。需要处理的数据对象是 a,b,c ,三者之间的关系可用给定的方程来表示,即 a,b,c 之间的关系隐含在一元二次方程中。

下面,再给出几个例子。

例 1-1: 将 10 个整数进行求和运算。

例 1-2: 对一个班学生的基本信息进行管理。

例 1-3: 对一个单位的组织机构进行管理。

例 1-4: 对一个行政区域的地图进行着色,要求最多用 4 种颜色,相邻区域不可用相同的颜色。

很显然,上述 4 个问题,都不是数值问题,要操作的数据之间的关系也不可能用简单的数学方程表示。那么,对于这类非数值计算问题,其数据之间的关系如何描述呢?这就是数据结构要研究的问题。下面我们进行详细分析。

在例 1-1 中,要求和的 10 个数是需要操作的数据对象。这 10 个数除了属于一个数集之外,相互之间没有其他特定的关系。或者说,这 10 个数之间具有比较松散的关系。这 10 个数称为集合结构。

在例 1-2 中,表示学生基本信息的数据是该问题需要处理的数据对象。它们可以组织为如表 1-1 的表格形式。在表格中,每一条学生信息的记录依次线性排列。这些学生信息和他们的前后次序关系称为线性结构。

在例 1-3 中,需要操作的数据对象是该单位中所有的组织机构名。例如,一个高校的组织机构如图 1-1 所示。

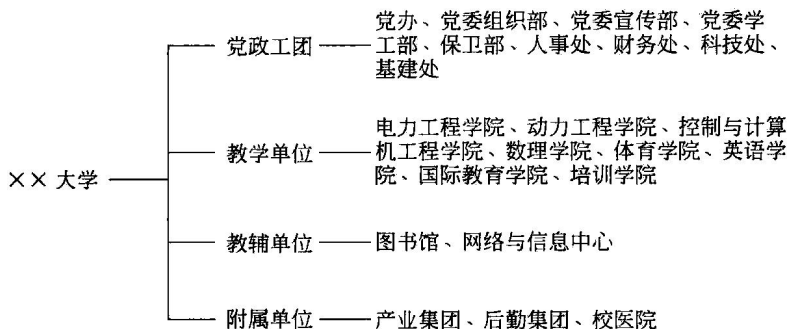


图 1-1 高校的组织机构图

这些组织机构之间具有更加复杂的层级关系：最高一级只有一个数据，是××大学，它包含党政工团、教学单位、教辅单位、附属单位多个类别的单位，这些类别可看作是第二层级，它们各自又分别包含了更低一级的子单位，如党政工团包括党办、党委组织部、党委宣传部等多个子单位。这些组织机构和它们之间的层次关系即为树形结构。

在例 1-4 中,行政区域示意图如图 1-2 所示。

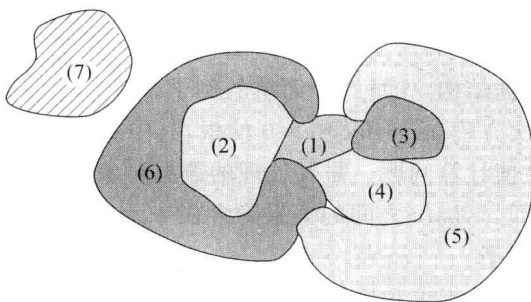


图 1-2 行政区域示意图

该问题需要操作的数据是这些行政区域,以及任何两个区域之间是否具有相邻接的关系。这些区域和它们的邻接关系我们称为图形结构。

从上述示例可知,数据结构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以及它们之间的关系。在此基础之上,进行数据对象上的操作。

1.4 数据的逻辑结构和存储结构

一般认为,一个数据结构是由数据元素依据某种逻辑关系组织起来的。对数据元素间逻辑关系的描述称为数据的逻辑结构;其次,数据必须在计算机内存储,数据的存储结构是数据结构的实现形式,是其在计算机内的表示;此外,讨论一个数据结构,必须同时讨论在该类数据上执行的运算才有意义。

1.4.1 逻辑结构

逻辑结构是对数据元素之间的逻辑关系的描述,是从具体问题抽象出来的数学模型,与数据的存储无关。通常所说的数据结构即指逻辑结构。由 1.3 节的分析可知,有 4 种类型的数据结构,即 4 种逻辑结构。逻辑结构可以用一个数据元素的集合和定义在此集合上的若干关系来表示,例如,图形表示、二元组表示、语言描述。其中,最常用的方法是图形表示法,习惯用小圆圈表示数据元素,而用圆圈之间的线表示数据元素之间的关系;如果数据元素之间的关系是有方向的,则用带箭头的线表示关系。

1. 集合结构

集合结构,数据对象中的数据元素之间除了同属于一个集合之外,没有任何其他的关系。这种结构类似于数学中的集合,如图 1-3 所示。

2. 线性结构

线性结构,数据对象中的数据元素之间存在一对一的线性关系,可以表示为 $1:1$,如图 1-4 所示。

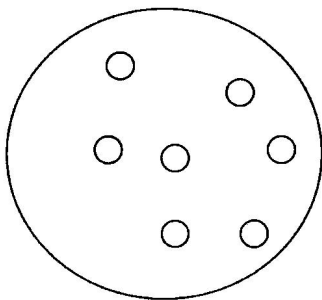


图 1-3 集合结构图

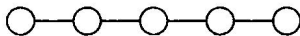


图 1-4 线性结构关系图

其中图 1-4 中的线段隐含了从左到右到关系,通常省略标志方向的箭头。

3. 树形结构

树形结构,数据对象中的数据元素之间存在一对多的层次关系,可以表示为 $1:n$,如图 1-5 所示,其中的线段隐含了从左到右的层次关系,通常省略标志方向的箭头。

4. 图形结构

图形结构,也称为网状结构,指数据对象中的数据元素之间存在多对多的任意关系,可以表示为 $m:n$,如图 1-6 所示。

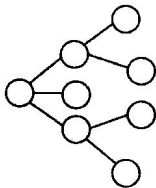


图 1-5 树形结构关系

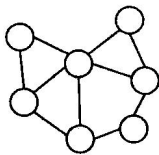


图 1-6 图形结构关系

图 1-6 中线段表示的关系是双向的,通常省略方向的标注。如果线段表示的关系是单向的,则必须标注方向。

在上述 4 种数据结构中,集合是数据元素之间关系极为松散的一种结构,因此在实际应用中往往用其他结构来表示。集合结构、树形结构和图形结构属于非线性结构。

数据的逻辑结构,除用图形法表示之外,也可用二元组进行形式定义:

$$\text{Data_structure} = (D, S) \text{ 或 } (D, R)$$

其中, D 是数据元素的集合; S 或 R 是 D 上关系的集合。下面举例说明。

例 1-5: 给出表 1-1 中数据结构的二元组形式描述。

$\text{StudentList} = (D, S)$, 其中:

$$D = \{a_i \mid i=1, \dots, 5\}$$

$$S = \{\langle a_i, a_{i+1} \rangle \mid i=1, \dots, 4\}$$

$\langle a_i, a_{i+1} \rangle$ 是一对序偶,表示 a_i 是 a_{i+1} 的直接前驱, a_{i+1} 是 a_i 的直接后继。

例 1-6: 给出以树形所示的数据结构的二元组形式描述,如图 1-7 所示。

$\text{Tree} = (D, S)$, 其中:

$$D = \{A, B, C, D, E, F, G, H\}$$

$$S = \{\langle A, B \rangle, \langle A, C \rangle, \langle A, D \rangle, \langle B, E \rangle, \langle B, F \rangle, \langle D, G \rangle, \langle D, H \rangle\}$$

例 1-7: 给出以图形所示的数据结构的二元组形式描述,如图 1-8 所示。

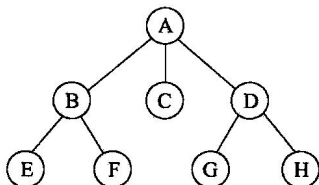


图 1-7 树形结构

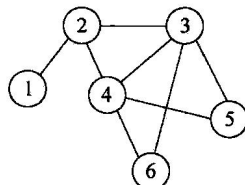


图 1-8 图形结构

该结构为一个图形结构,其对应的二元组形式可描述为:

$G = (D, S)$, 其中:

$$D = \{i \mid i=1, 2, \dots, 6\}$$

$$S = \{(1, 2), (2, 3), (2, 4), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6)\}$$

$(1, 2)$ 表示序偶 $\langle 1, 2 \rangle$ 和序偶 $\langle 2, 1 \rangle$ 同时存在。

1.4.2 存储结构

分析和研究数据结构的最终目的是为了使计算机能够对其进行处理。为此,仅有数据的逻辑结构是不够的,还必须研究这些结构在计算机内的存储方式,即存储结构(也称为物理结构)。需要注意的是,数据的存储结构应能正确反映数据元素之间的逻辑关系。换句话说就是,需要存储的内容包含:

- (1) “数据”的存储;
- (2) “关系”的存储。

常用的数据的存储结构形式有两种:顺序存储和链式存储。