高等学校计算机基础教育规划教材

数据结构

邵斌 叶星火 樊艳芬 朱绍军 编著

清华大学出版社

高等学校计算机基础教育规划教材

数据结构

邵斌 叶星火 樊艳芬 朱绍军 编著

清华大学出版社
北京

内 容 简 介

本书结合编者多年教学经验，全面系统地介绍了数据结构的基本概念和知识，条理清晰、重点突出，内容循序渐进、深入浅出，既注重理论知识的讲解，又注重算法设计的训练，突出了理论性与实用性。全书共分 9 章，第 1 章作为全书的综述和基础，介绍了数据结构、算法的相关概念和算法分析方法等，其后各章分别讨论了线性表、栈和队列、串、多维数组和广义表、树和二叉树、图、数据结构的定义、表示和实现，最后两章介绍了查找和内部排序的各种方法。在重点章节中，还结合精心编写的应用实例，介绍了应用数据结构和算法解决实际问题及进行程序设计的方法，增强了读者对基本知识的理解与掌握，有利于提高分析问题的能力和程序设计的能力。全书采用 C 语言作为数据结构和算法的描述语言。

本书可作为高等学校计算机类、信息类及相近专业本科生的数据结构课程教材，也可供从事计算机软件开发和工程应用的人员学习和参考。

图书在版编目(CIP)数据

数据结构/邵斌等编著. —北京：清华大学出版社，2018
(高等学校计算机基础教育规划教材)
ISBN 978-7-302-49332-7

Ⅰ. ①数… Ⅱ. ①邵… Ⅲ. ①数据结构－高等学校－教材 Ⅳ. ①TP311.12

中国版本图书馆 CIP 数据核字(2018)第 004238 号

责任编辑：颜廷芳
封面设计：常雪影
责任校对：刘　静
责任印制：丛怀宇

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，010-62770175-4278
印 装 者：三河市少明印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：17.25　　**字　　数**：390 千字
版　　次：2018 年 8 月第 1 版　　**印　　次**：2018 年 8 月第1次印刷
定　　价：45.00 元

产品编号：077657-01

前言

Foreword

数据结构是计算机程序设计的重要理论基础，是介于数学、计算机硬件和计算机软件三者之间的一门核心课程。数据结构的内容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。它不仅是计算机专业的核心课程，也是其他理工专业的热门选修课。在计算机的应用领域中，数据结构有着广泛的应用。

计算机的程序是对信息进行加工处理，而信息的表示和组织又直接关系到处理信息程序的效率。随着计算机的普及、信息量的增加、信息范围的拓宽，使许多系统程序和应用程序的规模很大，结构又相当复杂，因此，为了编写出一个“好”的程序，必须分析待处理对象的特征及各对象之间存在的关系。这就是数据结构这门课所要研究的问题。数据的结构，直接影响算法的选择和效率。

本书共分9章，第1章介绍了数据结构的基本概念和算法分析的初步知识；第2章到第5章介绍了线性表、栈和队列、串、多维数组和广义表等线性结构的基本概念及常用算法的实现；第6章和第7章介绍了非线性结构的树、二叉树、图等数据结构的存储结构和不同存储结构上的一些操作的实现；第8章介绍了各种查找表及查找方法；第9章介绍了各种排序算法。本书计划学时为64学时左右，其中上机实习为12学时左右。教师可根据专业情况，选讲或不讲目录中带*的章节。

本书是作者根据自己的教学经验总结，为地方本科院校计算机类学生编写的教材。作者在教学过程中发现，大多数学生在初学数据结构时，容易误解算法与程序之间的关系，经常会把书中的算法当作程序直接在编译器上进行运行测试。为了解决这个问题，本书采用C语言作为数据结构和算法的描述语言，并且对关键的算法都安排了比较完整的C语言程序供学生上机实习参考，只要添加上主函数，程序即可运行。本书力求做到选材精练、叙述简洁、通俗易懂，尽量避免抽象理论的介绍和复杂公式的推导。对各种数据结构均从实际出发，通过对实例的分析，使学生理解数据结构的基本概念。在每章后面带有适量的习题，习题中编排了较多的选择题和填空题，并配有习题答案，方便学生自学参考。

由于作者水平有限，书中难免会有不足之处，敬请广大读者批评指正。

编著者

2018年4月

目录

Contents

第 1 章　概论 …… 1

1.1　什么是数据结构 …… 1
1.1.1　数据和数据元素 …… 1
1.1.2　数据对象与数据类型 …… 2
1.1.3　数据结构 …… 2
1.2　为什么要学习数据结构 …… 5
1.2.1　学习数据结构的重要性 …… 5
1.2.2　数据结构的应用举例 …… 5
1.3　算法和算法分析 …… 7
1.3.1　什么是算法 …… 7
1.3.2　算法的描述和设计 …… 7
1.3.3　算法分析 …… 8
本章小结 …… 10
习题 …… 10

第 2 章　线性表 …… 12

2.1　线性表的基本概念 …… 12
2.1.1　线性表的定义 …… 12
2.1.2　线性表的基本操作 …… 13
2.2　线性表的顺序存储 …… 13
2.2.1　顺序表 …… 13
2.2.2　顺序表的基本操作 …… 14
2.2.3　顺序存储方式举例 …… 17
2.3　线性表的链式存储 …… 20
2.3.1　单链表的基本概念 …… 20
2.3.2　单链表的基本操作 …… 22
2.3.3　链式存储举例 …… 25
2.3.4　循环链表 …… 28
2.3.5　双向链表 …… 30
2.3.6　双向循环链表 …… 33

2.3.7 静态链表 …… 34
2.4 线性表顺序存储与链式存储的比较 …… 35
2.5 线性表的应用 …… 36
2.5.1 约瑟夫问题 …… 36
2.5.2 多项式加法 …… 38
2.5.3 电文加密 …… 40
本章小结 …… 42
习题 …… 43

第 3 章 栈和队列 …… 45

3.1 栈 …… 45
3.1.1 栈的定义与基本操作 …… 45
3.1.2 顺序栈的存储结构和操作的实现 …… 47
3.1.3 链栈的存储结构和操作的实现 …… 50
3.2 栈的应用 …… 52
3.2.1 数制转换 …… 52
3.2.2 括号匹配问题 …… 54
3.2.3 子程序的调用 …… 55
3.2.4 利用一个顺序栈逆置一个带头节点的单链表 …… 56
3.2.5 后缀表达式 …… 59
3.3 队列 …… 61
3.3.1 队列的定义与基本操作 …… 61
3.3.2 链队列的存储结构和操作的实现 …… 62
3.3.3 顺序队列的存储结构和操作的实现 …… 64
3.4 队列的应用 …… 68
3.4.1 打印杨辉三角形 …… 68
3.4.2 迷宫问题：寻找一条从迷宫入口到出口的最短路径 …… 71
3.5 递归 …… 74
3.5.1 递归的定义与实现 …… 74
3.5.2 递归消除 …… 77
本章小结 …… 81
习题 …… 81

第 4 章 串 …… 85

4.1 串的定义和基本操作 …… 85
4.1.1 串的定义 …… 85
4.1.2 串的基本操作 …… 87
4.2 串的表示和实现 …… 88

4.2.1 串的定长顺序存储 …… 88
4.2.2 串的堆存储结构 …… 91
4.2.3 串的块链存储结构 …… 93
4.3 串的模式匹配算法 …… 97
4.3.1 基本的模式匹配算法 …… 97
4.3.2 模式匹配的改进算法——KMP 算法 …… 100
本章小结 …… 102
习题 …… 102

第 5 章 多维数组和广义表 …… 104

5.1 多维数组 …… 104
5.1.1 多维数组的定义 …… 104
5.1.2 数组的存储结构 …… 105
5.2 矩阵的压缩存储 …… 106
5.2.1 特殊矩阵 …… 106
5.2.2 稀疏矩阵 …… 108
5.3 广义表 …… 114
本章小结 …… 116
习题 …… 117

第 6 章 树和二叉树 …… 118

6.1 树的概念与基本操作 …… 118
6.1.1 树的定义 …… 118
6.1.2 树的一些基本概念 …… 119
6.1.3 树的基本操作 …… 120
6.2 二叉树 …… 120
6.2.1 二叉树的定义和基本操作 …… 120
6.2.2 二叉树的性质 …… 121
6.2.3 二叉树的存储结构 …… 123
6.3 二叉树的遍历与线索化 …… 124
6.3.1 二叉树的遍历 …… 124
6.3.2 线索二叉树 …… 127
6.3.3 基于遍历的应用与线索二叉树的应用 …… 129
6.3.4 标识符树 …… 134
6.4 树和森林 …… 134
6.4.1 树的存储结构 …… 134
6.4.2 树、森林和二叉树之间的转换 …… 137
6.4.3 树和森林的遍历 …… 140

6.5 哈夫曼树及其应用 …… 142
6.5.1 与哈夫曼树相关的基本概念 …… 142
6.5.2 哈夫曼树的应用 …… 144
6.5.3 哈夫曼编码算法的实现 …… 146
*6.6 树的计数 …… 147
本章小结 …… 150
习题 …… 151

第 7 章 图 …… 154

7.1 图的基本概念 …… 154
7.1.1 图的定义 …… 154
7.1.2 图的相关术语 …… 155
7.2 图的存储结构 …… 157
7.2.1 邻接矩阵表示法 …… 157
7.2.2 邻接表表示法 …… 159
7.3 图的遍历 …… 163
7.3.1 深度优先搜索法 …… 163
7.3.2 广度优先搜索法 …… 165
7.3.3 非连通图的遍历 …… 167
7.4 生成树与最小生成树 …… 167
7.4.1 生成树的概念 …… 167
7.4.2 构造最小生成树的普里姆(Prim)算法 …… 168
7.4.3 构造最小生成树的克鲁斯卡尔(Kruskal)算法 …… 171
7.5 最短路径 …… 173
7.5.1 从某个源点到其余各顶点的最短路径 …… 174
7.5.2 每一对顶点之间的最短路径 …… 178
7.6 拓扑排序 …… 181
7.7 关键路径 …… 184
本章小结 …… 190
习题 …… 190

第 8 章 查找 …… 195

8.1 查找的基本概念 …… 195
8.1.1 查找表和查找 …… 195
8.1.2 查找表的数据结构表示 …… 196
8.1.3 平均查找长度 ASL …… 196
8.2 线性表的查找 …… 196
8.2.1 顺序查找 …… 196
8.2.2 二分查找 …… 198

8.2.3　分块查找 …… 201
8.3　树表的查找 …… 203
8.3.1　二叉排序树 …… 203
*8.3.2　平衡二叉树 …… 208
*8.3.3　B－树 …… 212
8.4　散列表的查找 …… 221
8.4.1　散列表的概念 …… 221
8.4.2　散列函数的构造方法 …… 222
8.4.3　处理冲突的方法 …… 223
8.4.4　散列表上的运算 …… 227
本章小结 …… 230
习题 …… 230

第 9 章　排序 …… 232

9.1　排序的基本概念 …… 232
9.1.1　关键字与排序 …… 232
9.1.2　排序的稳定性 …… 233
9.1.3　排序方法的分类 …… 233
9.1.4　排序算法性能评价 …… 233
9.1.5　不同存储方式的排序过程 …… 233
9.2　插入排序 …… 234
9.2.1　直接插入排序 …… 234
9.2.2　希尔排序 …… 237
9.3　交换排序 …… 239
9.3.1　冒泡排序 …… 239
9.3.2　快速排序(霍尔排序) …… 240
9.4　选择排序 …… 244
9.4.1　直接选择排序 …… 244
9.4.2　堆排序 …… 245
9.5　归并排序 …… 250
9.6　基数排序 …… 253
9.6.1　桶排序 …… 253
9.6.2　多关键字的排序 …… 253
9.6.3　链式基数排序 …… 254
9.7　内部排序算法比较 …… 257
9.8　外部排序简介 …… 259
本章小结 …… 259
习题 …… 260

参考文献 …… 263

第1章 概 论

Chapter 1

本章要点

- 什么是数据结构
- 为什么要学习数据结构
- 数据对象和数据类型
- 算法和算法分析

本章学习目标

- 了解数据结构的基本概念，理解常用术语。
- 掌握数据元素间的4类结构关系。
- 掌握算法的定义及特性，掌握算法设计的要求。
- 掌握分析算法的时间复杂度和空间复杂度的方法。

1.1 什么是数据结构

计算机科学是一门研究信息表示和处理的科学。信息的表示和组织直接关系到处理信息程序的效率。由于许多系统程序和应用程序的规模很大，结构又相当复杂，因此需要对程序设计方法进行系统的研究，这不仅涉及研究程序的结构和算法，同时也涉及程序加工对象（数据）的结构，因为数据的结构直接影响算法的选择和效率。

1.1.1 数据和数据元素

数据（Data）是信息的载体，是对客观事物的符号表示，它能够被计算机识别、存储和加工处理。可以说，数据是计算机程序加工的“原料”。例如，一个求解代数方程的程序所处理的对象是整数、实数或复数，一个编译程序或文本编辑程序所处理的对象是字符串。随着计算机科学和技术的发展，计算机应用领域的扩大，数据的含义也随之越来越广。目前，图像、声音、视频等都可以通过编码而由计算机处理，因此它们也属于数据的范畴。

数据元素（Data Element）是数据中具有独立意义的个体，是数据的基本单位，通常在计算机程序中作为一个整体进行考虑和处理。例如，成绩表中的学生成绩信息、通讯录中的个人或组织的通信信息等，数据元素也称为元素、节点或记录。有时，一个数据元素可以由若干个数据项（也称字段、域）组成，数据项是数据不可分割的最小单位。

1.1.2 数据对象与数据类型

在数据结构中往往涉及数据类型与数据对象的概念。

数据对象(Data Object)是性质相同的数据元素的集合,它是数据的一个子集。例如,整数数据对象是集合 $N=\{0,\pm1,\pm2,\pm3,\cdots\}$;大写字母字符数据对象是集合 $C=\{'A','B',\cdots,'Z'\}$。需要注意的是,计算机中的整数数据对象集合 N_1 应该是上述集合 N 的一个子集,$N_1=\{0,\pm1,\pm2,\cdots,\pm\text{maxint}\}$,其中 maxint 是依赖于所使用的计算机和语言的最大整数。

数据类型(Data Type)是计算机程序中的数据对象以及定义在这个数据对象集合上的一组操作的总称。例如,C 语言中的整数类型是区间[$-$maxint,$+$maxint]上的整数,在这个集合上可以进行加、减、乘、整除、求余等操作。

数据类型可以分为原子数据类型和结构数据类型,原子数据类型是由计算机语言所提供的,如 C 语言中的整型、实型、字符型;结构数据类型是利用计算机语言提供的一种描述数据元素之间逻辑关系的机制,由用户自己定义而成,如 C 语言中的数组类型、结构体类型等。

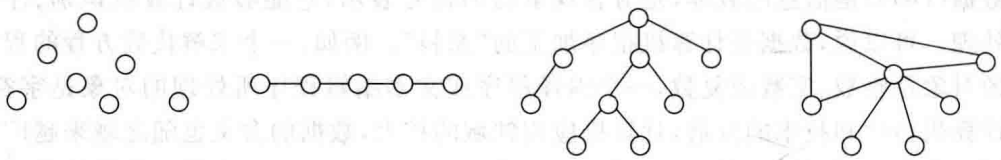
1.1.3 数据结构

数据结构不同于数据类型,也不同于数据对象,它不仅要描述数据类型的数据对象,而且要描述数据对象各元素之间的相互关系。比如需要描述数据对象元素,并使这些运算能合法地用于数据对象的各元素上。

数据结构(Data Structure)是指数据对象以及该数据对象集合中的数据元素之间的相互关系(即数据元素的组织形式)。数据结构的研究范围主要包括研究数据的逻辑结构和物理结构(数据结构在计算机中的表示),而且对每种结构定义相适应的运算,并使用某种高级程序设计语言给出各种运算的算法,分析算法的效率,同时还研究各种数据结构在计算机科学和软件工程中的某些应用,讨论数据分类、检索等方面的技术。

数据元素的组织形式一般包含下列内容。

(1) 数据元素之间的逻辑关系,也称为数据的**逻辑结构**。数据的逻辑结构通常有 4 类(见图 1-1)。



(a) 集合　(b) 线性结构　(c) 树状结构　(d) 图状结构

图 1-1　4 类基本逻辑结构关系图

① **集合**:其中的数据元素之间除了"属于同一个集合"的关系以外,别无其他关系。

② **线性结构**:其中的数据元素之间存在一对一的关系。

③ **树状结构**:其中的数据元素之间存在一对多的关系。

④ **图状结构**(或称**网状结构**)：其中的数据元素之间存在多对多的关系。

(2) 数据元素以及它们之间的相互关系在计算机存储器内的表示(又称映象)，称为数据的**存储结构**，也称数据的**物理结构**。

(3) 数据元素之间的**运算**，即对数据元素施加的**操作**，有时也直接称为数据的运算或操作。

由于数据的**逻辑结构**是从逻辑关系上描述数据的，它独立于计算机，与数据的存储无关，因此，数据的逻辑结构可以看作是从具体问题抽象出来的数学模型。而数据的**存储结构**是逻辑结构用计算机语言来实现的，它依赖于计算机语言，对机器语言来说，存储结构是具体的，但我们在高级语言的层次上讨论存储结构。数据的**运算**是定义在数据的逻辑结构上的，每一种逻辑结构都有一个运算的集合。例如，常用的运算有插入、删除、查找、排序等，这些运算实际上是对数据所施加的一系列抽象的操作，所谓抽象的操作，是指我们只知道这些操作是“做什么”，而不必考虑“怎么做”。只有在确定了数据的存储结构以后，我们才考虑如何具体地实现这些运算。本书所讨论的数据运算，均以 C 语言描述的算法来实现。

例 1-1　学生成绩表(见表 1-1)是一个数据结构。

表 1-1　学生成绩表

学　号	姓　名	计算机导论	高等数学	普通物理	平均成绩
08051101	陈玉洁	90	99	81	90
08051102	马哲丽	85	68	78	77
08051103	田春英	92	68	66	75
08051104	卢华娟	70	79	93	81
⋮	⋮	⋮	⋮	⋮	⋮
08051138	张晓祥	89	88	75	84

表 1-1 被称为一个数据结构，表中的每一行是一个节点(或记录)，由学号、姓名、各科成绩和平均成绩等数据项(或字段)组成。

首先，表 1-1 中数据元素之间的逻辑关系是：与表中任一个节点相邻且在其前面的节点(又称**直接前趋**)有且只有一个；与表中任一个节点相邻且在其后的节点(又称**直接后继**)有且只有一个。表中只有第一个节点没有直接前趋，故称为开始节点；也只有最后一个节点没有直接后继，故称为终端节点。例如，表中“马哲丽”所在节点的直接前趋节点是“陈玉洁”节点，其后继节点是“田春英”节点，上述节点之间的关系构成了这一张学生成绩表的逻辑结构。

其次，表 1-1 的存储结构是指如何用计算机语言表示各节点之间的关系。这种关系可以是表中的节点按顺序邻接地存储在一些连续的单元中，也可以是用指针将这些节点链接在一起。

最后，在该表中，经常要查看一些学生的成绩；转入新学生时需要增加节点；学生退学时需要删除相应的节点。至于如何进行查找、插入、删除，这就是数据的运算问题。搞清

楚上述三个问题，也就弄清了学生成绩表这一数据结构。

说明：数据结构可以理解为按某种逻辑关系组织起来的一批数据，应用计算机语言，按一定的存储表示方式把数据存储在计算机的存储器中，并在这些数据上定义一个运算的集合。

在不会产生混淆的前提下，可以将数据的逻辑结构简称为数据结构。本书的第 2 章至第 4 章介绍的是线性结构；第 5 章至第 7 章介绍的是非线性结构。

数据的存储结构可采用以下 4 种基本的存储方法得到。

(1) 顺序存储方法。该方法是将逻辑上相邻的节点存储在物理位置相邻的存储单元中，节点之间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储结构称为**顺序存储结构**，通常顺序存储结构用计算机高级语言中的数组来描述。

该方法主要应用于线性的数据结构，而非线性的数据结构可以通过某种线性化的方法来实现顺序存储。

(2) 链接存储方法。该方法不要求逻辑上相邻的节点在物理位置上也相邻，节点之间的逻辑关系是由附加的指针来表示的。由此得到的存储结构称为**链式存储结构**，通常链式存储结构用计算机高级语言中的指针来描述。

(3) 索引存储方法。该方法在存储节点信息的同时，还建立了附加的**索引表**。索引表中的每一项称为索引项。索引项的一般形式是：(关键字，地址)，所谓**关键字**(key)是指能够唯一标识一个节点的数据项。若每个节点在索引表中都有一个索引项，则该索引称为**稠密索引**。若一组节点在索引表只对应一个索引项，则该索引称为**稀疏索引**。稠密索引中索引项地址指出了节点所在的存储位置，而稀疏索引中索引项地址则指出了一组节点的起始存储位置。

(4) 散列存储方法。该方法的基本思想是根据节点的关键字直接计算出该节点的存储地址。

上述 4 种基本的存储方法，既可以单独使用，也可以组合起来对数据结构进行存储映象。同一种逻辑结构，若采用不同的存储方法，可以得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构，应该根据具体要求而定，主要是考虑运算便捷和算法的时间、空间需求。

注意：不管怎样定义数据结构，都应该将数据的逻辑结构、存储结构和运算(操作)这三方面看成一个整体。学习时不要孤立地去理解其中的某一个方面，而应该注意它们之间的联系。

由于存储结构是数据结构不可或缺的一个方面，因此常常将同一逻辑结构的不同存储结构分别冠以不同的数据结构名称来标识。例如，线性表是一种逻辑结构，若采用顺序存储方法表示，则称为顺序表；若采用链式存储方法表示，则称为链表；若采用散列存储方法表示，则称为散列表。

同理，由于数据的运算也是数据结构不可分割的一个方面，因此，在给定了数据的逻辑结构和存储结构之后，按定义的运算集合及其运算性质的不同，也可以导出完全不同的数据结构。例如，若对线性表上的插入、删除运算限制仅在表的固定一端进行，则该线性表称为栈；若对插入运算限制在表的一端进行，删除运算限制在表的另一端进行，则该线

性表就称为队列。进一步而言，若线性表采用顺序表或链表作为存储结构，则对插入、删除运算做了上述限制以后，可以分别得到顺序栈或链栈以及顺序队列或链队列。

1.2 为什么要学习数据结构

1.2.1 学习数据结构的重要性

数据结构是一门研究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和操作的学科。由于数据结构的研究不仅涉及计算机硬件，而且与计算机软件的研究有着密切的关系，因此它不只是一般程序设计的基础，也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和大型应用程序的重要基础。

目前，数据结构不仅是计算机科学与技术专业的核心课程之一，而且是其他非计算机专业的主要选修课程之一。由于在计算机系统软件和应用软件中都要用到各种数据结构，因此，仅仅掌握几种计算机语言是难以应付众多复杂问题的，要想更有效地使用计算机，就必须学习数据结构的有关知识。

在计算机发展初期，人们主要利用计算机处理数值计算问题。例如，在建筑设计时计算梁架结构的应力，需要求解线性方程组；预报人口增长情况，需要求解微分方程等。因为此阶段所涉及的运算对象比较简单（为整数、实数或布尔型数据），所以进行程序设计时主要考虑设计技巧，并不重视数据结构。随着软件、硬件的不断发展，计算机的应用领域也日益扩大，解决“非数值计算”问题显得越来越重要。据统计，目前处理非数值计算问题大约占用了90%以上的计算机时间。由于非数值计算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来描述，因此，解决此类问题的关键是设计出合适的数据结构。

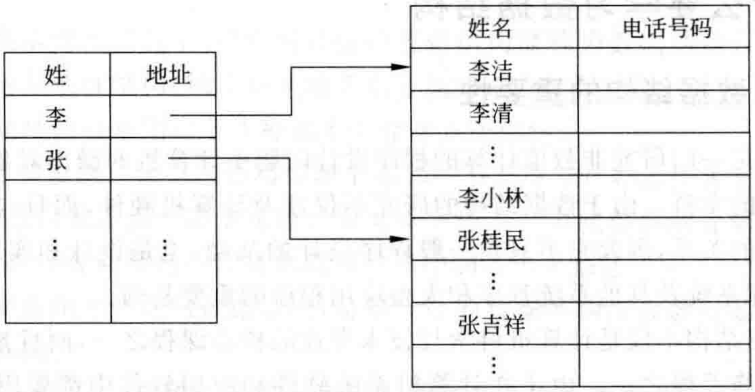
著名的瑞士计算机科学家 N. Wirth 曾指出：算法＋数据结构＝程序。这里的数据结构是指数据的逻辑结构和存储结构，而算法则是指对数据运算的描述。由此可见，程序设计的实质是对所提出的问题选择一种好的数据结构，加之设计一个好的算法，而好的算法在很大程度上取决于描述该问题的数据结构。请见 1.2.2 小节应用举例。

1.2.2 数据结构的应用举例

例 1-2 电话号码的查询问题。

编写一个电话号码的查询程序，要求任意给出一个姓名，如果此人留有电话号码，那么就找出他的电话号码；否则就指出这个人没有电话号码。要解决这个问题，首先构造一张电话号码登记表，表中的每个节点存放姓名和电话号码两个数据项。设计的查找算法，取决于该表的结构及存储方式。第一种算法是将表中节点顺序地存储在计算机中，查找时从头开始依次核对姓名，若找到正确的姓名则可获得相应的电话号码，若找遍整个表均无所找姓名，则表示此人无电话号码。此算法对于一个人数不多的单位是可行的，但对一个大单位或城市来说是不实用的。第二种算法是将电话号码登记表按姓氏排序，另外构造一张姓氏索引表，存储结构如图 1-2 所示。查找时首先在索引表中核对姓氏，然后根据

索引表中的地址到电话号码登记表中查对姓名，注意这时已经不需要查找其他不同姓氏的名字了。相比之下，在新的结构上产生的第二种查找算法比第一种算法更为有效。在第 8 章中将进一步讨论查找策略。

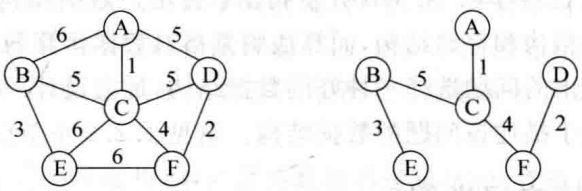


(a) 姓氏索引表　　(b) 已按姓氏排序的电话号码表

图 1-2　电话号码查询问题的索引存储

例 1-3　n 个城市之间铺设光缆的问题。

假设需要在 n 个城市之间铺设光缆，并且任意两个城市之间都可以铺设。大家知道，在 n 个城市之间只要铺设 $n-1$ 条光缆，就能将这 n 个城市连成网络，只是由于地理位置的不同，所需经费也不同，问题是采用什么样的设计方案才能使总投资最少。这个问题的数学模型如图 1-3(a)所示，图中“顶点”表示城市，顶点之间的连线及其上面的数值表示可以铺设的光缆及所需经费。求解该问题的算法为：在可以铺设的 $m(m>n)$ 条光缆中选取 $n-1$ 条，既能连通 n 个城市，又能使总投资为最少。实际上，这是一个“求图的最小生成树”的问题，如图 1-3(b)所示，将在第 7 章中进一步讨论。



(a) 城市示意图　　(b) 铺设光缆设计图

图 1-3　图及其最小生成树示例

从上述例子中可以看出，解决问题的关键有两点：①选取合适的数据结构表示问题；②写出有效的算法。

注意：由于在计算机系统软件和应用软件中都要用到各种数据结构，因此，仅仅掌握几种计算机语言是难以应付众多复杂问题的，要想更有效地使用计算机，就必须学习数据结构的有关知识。希望读者充分了解数据结构的重要性，下功夫学好这门课程。

1.3 算法和算法分析

1.3.1 什么是算法

由于数据的运算是通过算法来描述的，因此，讨论算法是数据结构课程的重要内容之一。

算法(Algorithm)是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或多个操作。此外，一个算法还具有下列5个特性。

(1) **有穷性**。一个算法必须在执行有穷步之后结束，即算法必须在有限时间内完成。

(2) **确定性**。算法中每一步必须有确切的含义，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**。一个算法是可行的，即算法中的每一步都可以通过已经实现的基本运算执行有限次得以实现。

(4) **输入**。一个算法有零个或多个输入，它们是算法开始时对算法给出的初始量。

(5) **输出**。一个算法有一个或多个输出，它们是与输入有特定关系的量。

在一个算法中，有些指令可能重复执行，因而指令的执行次数可能远远大于算法中的指令条数。由有穷性和可行性得知，对于任何输入，一个算法在执行了有限条指令后一定要终止，而且必须在有限时间内完成。因此，一个程序如果对任何输入都不会产生无限循环，则它就是一个算法。

尽管算法的含义与程序非常相似，但两者还是有区别的。首先，一个程序不一定满足有穷性，因此它不一定是算法。例如，系统程序中的操作系统，只要整个系统不遭受破坏，它就永远不会停止，即使没有作业要处理，它仍处于等待循环中，以待一个新作业的进入，因此操作系统就不是一个算法。其次，程序中的指令必须是计算机可以执行的，而算法中的指令却无此限制。如果一个算法采用机器可执行的语言来书写，那么它就是一个程序。

1.3.2 算法的描述和设计

算法是一个十分古老的研究课题，然而计算机的出现为这个课题注入了青春和活力，使算法的设计和分析成为计算机科学中最为活跃的研究热点之一。有了计算机的帮助，许多过去靠人工无法计算的复杂问题都有了解决的希望。

一个算法可以采用自然语言、数学语言或者约定的符号语言(如伪码、框图等)来描述。为了方便读者的阅读和实践，本书中的算法和数据结构均使用C语言来描述。

在例1-2中，介绍了电话号码查询问题的两种算法，那么如何来评价这些算法质量的优劣呢？一般来说，设计一个“好”的算法应该考虑以下几点。

(1) 正确性。算法应当满足具体问题的需求。

(2) 健壮性。当输入数据非法时，算法也能适当地作反应或进行处理，而不会产生莫名其妙的输出结果或出错信息，并中止程序的执行。

(3) 可读性。算法主要是为了方便人们的阅读和交流，其次才是机器执行。

(4) 执行算法所耗费的时间。

(5) 执行算法所耗费的存储空间，其中主要考虑辅助存储空间。

1.3.3 算法分析

评价一个程序优劣的重要依据是看这个程序的执行需要占用多少机器资源。在各种机器资源中，最重要的是时间资源和空间资源。因此，在进行程序分析时，大家最关心的就是程序所用算法在运行时所要花费的时间代价和程序中使用的数据结构所占有的空间代价。通常就称为时间复杂度(时间代价)和空间复杂度(空间代价)。

1. 算法的时间复杂度分析

当需要解决的问题的规模(以某种单位计算)由 1 增至 n 时，解决问题的算法所耗费的时间也以某种单位由 $f(1)$ 增至 $f(n)$，这时就称该算法的时间代价为 $f(n)$。

通常，一个算法是由控制结构(顺序、选择和循环)和“原操作”(指固有数据类型的操作)构成的，而算法时间取决于两者的综合效果。为了便于比较同一问题的不同算法，一般是从算法中选取一种对于所研究的问题(或算法类型)来说是基本操作的“原操作”，以该“原操作”重复执行的次数作为算法的时间度量。被称作问题的“原操作”应该是其重复执行次数和算法的执行时间成正比的“原操作”，大部分情况下它是最深层循环内的语句中的“原操作”，它的执行次数和包含它的语句的频度相同。语句的**频度**，是指该语句重复执行的次数。

例 1-4 有下列三个程序段：

```
(1) {x=x+1; s=0;}
(2) for(i=1; i<=n; i++){x=x+1; s=s+x;}
(3) for(j=1; j<=n; j++)
        for(k=1; k<=n; k++){x=x+1; s=s+x;}
```

它们含基本操作“x+1”的语句的频度分别为 1、n 和 n^2。

例 1-5 对 n 个记录进行升序排序的问题，采用最简单的选择排序方法。

每次处理时，先从 n 个未排序的记录中选出一个最小记录，则第一次要经过 $n-1$ 次比较，才能选出最小记录；第二次再从剩下的 $n-1$ 个记录中经过 $n-2$ 次比较，选出次小记录……如此反复，直到只剩两个记录时，经过 1 次比较就可以确定它们的大小。整个排序过程的基本操作(即“原操作”)是“比较两个记录的大小”，含“比较”的语句的频度是：

$$(n-1)+(n-2)+\cdots+2+1=n\cdot(n-1)/2$$

在同一个算法处理两个规模相同的问题，所花费的时间代价和空间代价也不一定相同。要全面分析一个算法，应该考虑它在最坏情况下的代价(即对同样规模的问题所花费的最大代价)、最好情况下的代价和平均情况下的代价等。然而，要全面准确地分析每个算法是相当困难的，因此，本书中在分析算法时将主要考虑它们在最坏情况下的代价，个别地方也涉及其他情况。

人们通常采用 ***O* 表示法**来描述算法分析的结果。如果存在正的常数 M 和 n_0，当问题的规模 $n\geqslant n_0$ 后，算法的时间量度 $T(n)\leqslant M\cdot f(n)$，那么就称该**算法的时间复杂度**为 $O(f(n))$。这种说法意味着：当 n 充分大时，该算法的时间复杂度不大于 $f(n)$ 的一个常