21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

数据结构与算法教程

Data Structures and Algorithms

朱明方 吴及 编著

- 取材精炼，易学易懂
- 数据结构与算法相融
- 实例丰富，突出应用

精品系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

数据结构与算法教程

Data Structures and Algorithms

朱明方 吴及 编著

精品系列

人民邮电出版社

北京

图书在版编目（CIP）数据

数据结构与算法教程 / 朱明方，吴及编著. -- 北京
：人民邮电出版社，2011.11
21世纪高等学校计算机规划教材
ISBN 978-7-115-25404-7

Ⅰ. ①数… Ⅱ. ①朱… ②吴… 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(2011)第157564号

内 容 提 要

本书以清华大学电子系数据结构讲义为蓝本，主要针对高等院校非计算机专业开设“数据结构”课程的需要而编写的。全书从应用的角度，重点介绍数据处理中常用的数据结构——线性表、树与二叉树、图，以及基本的数据处理技术——查找和排序方法，同时通过实例把回溯法、分治法、贪心法、动态规划法等常用的算法设计思想的应用融入其中，把数据结构的介绍和常用算法设计的讨论紧密结合，并且辅之以充足的练习题，从而使读者更具体、更深刻地理解各种常用的数据结构，及它们与算法之间的关系，以达到学以致用的目的。

本书可以作为大专院校数据结构课程的教材，也可以作为从事计算机应用开发的科技人员的参考书。

21 世纪高等学校计算机规划教材

数据结构与算法教程

◆ 编　　著　朱明方　吴　及
责任编辑　武恩玉
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷
◆ 开本：787×1092　1/16
印张：20.5　　2011 年 11 月第 1 版
字数：541 千字　　2011 年 11 月北京第 1 次印刷

ISBN 978-7-115-25404-7

定价：39.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前 言

当今时代是计算机统领发展的时代，计算机应用已经深入到国民经济的各个领域，成为人们学习、工作和生活的重要支撑。对于高等院校的学生特别是理工科专业的学生来说，计算机应用能力是必须培养的重要能力。作为计算机应用的重要基础——数据结构，已经不只是计算机专业的重要专业基础课，也是许多非计算机专业，特别是工科专业的重要技术基础课。但就实际情况而言，非计算机专业安排学习该课程的学时比计算机专业要少，先修课程的基础也不一样，而且一般都不再单独设置算法设计类课程。因此，如何针对非计算机专业的情况，能够在学时和基础有限的前提下，让学生较快地掌握数据结构和算法设计的基本知识，有效提升他们的计算机应用能力，是非计算机专业的数据结构课程教学中非常重要而又十分具有挑战性的课题，教材建设则是其中要解决的重要问题。

本书是为高等院校非计算机专业的数据结构课程而写的，是作者基于多年来为清华大学电子信息相关专业讲授“数据结构与算法”课的教学实践，经过不断地修改编写而成。本书的特点是：以高等院校非计算机专业学生已有的基础为前提，从应用的角度介绍常用的数据结构。同时，考虑到非计算机专业通常不再单独设置算法设计类课程，而数据结构又与算法设计及实现密切相关的实际情况，本书将数据结构的应用和常用算法设计方法的讨论紧密结合。例如，把栈结构与回溯法的讨论、递归算法设计结合起来；把树与二叉树的应用与减治法、回溯过程的讨论相结合；把最小生成树结构与贪心法的应用结合在一起；把求最短路径问题的讨论与动态规划方法的应用紧密联系等。通过具体的例子，使常用的算法设计思路较自然地融入其中，从而使读者对所学的抽象方法和设计、实现思路有更深入的掌握和理解。书中对各部分内容的介绍和讨论突出重点，力求少而实用但又不漏掉重要内容，对稍微复杂的数据结构和算法设计方法的讨论力求做到深入浅出，并通过应用例子使其与实际问题相结合，从而使读者能够把握书中的脉络，掌握和理解课程的重点、难点。每章给出的习题给读者留有充分的选择余地，通过书面作业和上机练习，可以使读者进一步巩固所学的重要概念和重点知识。同时本书以面向对象的观点来讨论数据结构，使其与该课程发展的新要求同步。

由于非计算机专业的学生通常都具有 C 语言基础，教材中采用 C 作为算法描述语言。只要有 C 语言程序设计基础的学生就能阅读书中的算法。因为书中对每

种数据结构都给出了抽象数据类型，因此，对于有C++语言基础的学生很容易转换为以类为基础的算法描述。

本书建议授课课时不少于48学时；标有“*”的章节，为可选内容，由教师视具体情况选定。若这些内容不作为课程要求，则可以适当减少授课学时，同时应该安排足够的实习学时相配合，包括上机作业、上机实验和大作业。

本书第1章至第4章由朱明方编写，第5章至第7章由吴及编写，朱峰做了部分算法的修改调试工作，施迎难整理提供了参考资料。全书由朱明方统稿。

限于作者的水平，书中疏漏和错误之处在所难免，敬请读者批评指正。

作　者

2010年11月

目　录

第 1 章 绪论

1.1 预 备 知 识

“关系”是数据结构课程中要涉及的概念，作为后续讨论的基础，本节首先介绍二元关系及其相关的必要知识。

1.1.1 集合的笛卡儿积

集合是现代数学中一个非常重要的概念，集合的并、交、差是集合的最基本的运算。除了最基本的运算，集合还有一种很重要的运算，这就是集合的笛卡儿积(Cartesian Product)。

为了说明什么是集合的笛卡儿积，首先给出有序对的定义：

由两个元素 x、y 按一定的顺序排列构成的二元组称为一个有序对或序偶，记作$<x, y>$或(x, y)，其中 x 是有序对的第一个元素，y 是有序对的第二个元素。

有序对具有以下特点：

① 当 $x \neq y$ 时，有$<x, y> \neq <y, x>$。

② 两个有序对相等的充分必要条件是两个分量分别相等，即只有当 $x=u$ 且 $y=v$ 时，才能有$<x, y>=<u, v>$。

基于有序对的概念，下面给出集合的笛卡儿积的定义：

设有集合 A 和 B，以 A 中的元素为第一分量，以 B 中元素为第二分量，构成有序对，所有这样的有序对组成的集合称为 A 和 B 的笛卡儿积，记作 $A \times B$。可以表示为

$$A \times B = \{<a, b> | a \in A \text{ 且 } b \in B\} \text{ 。}$$

根据上述定义，若设 $A = \{ a_1, a_2 \}$，$B = \{ b_1, b_2 \}$，则有

$$A \times B = \{<a_1,b_1>,<a_1,b_2>,<a_2,b_1>,<a_2,b_2>\} \text{ 。}$$

根据有序对的特点，不难理解：

$$A \times B \neq B \times A \text{。}$$

例如，上述例子中，有：$B \times A = \{<b_1,a_1>,<b_1,a_2>,<b_2,a_1>,<b_2,a_2>\}$，显然，它不等于 $A \times B$ 。

也就是说，笛卡儿积运算不适合交换率。

同时，根据有序对的特点可知：$<<x, y>, z> \neq <x, <y, z>>$，因此有

$$(A \times B) \times C \neq A \times (B \times C) \text{。}$$

也就是说，笛卡儿积运算也不适合结合率。

当然，笛卡儿积与集合的并、集合的交运算结合起来，还可以得到其他的性质。

在实际问题中，经常会用到多个集合的笛卡儿积。实际上，有了两个集合的笛卡儿积的定义，也就很容易推出 n 个集合的笛卡儿积的定义了。下面给出 n 个集合的笛卡儿积的定义。

设有 n 个集合 $D_1, D_2, \cdots, D_n$，它们的笛卡儿积定义为

$$D_1 \times D_2 \times \cdots \times D_n = \{<d_1, d_2, \cdots, d_n> \mid d_i \in D_i,\ i=1, 2, \cdots, n\}。$$

其中，$<d_1, d_2, \cdots, d_n> = << d_1, d_2, \cdots, d_{n-1}>, d_n>$ ，称为 n 元组，d_i 为元组的第 i 个分量。

1.1.2 二元关系

1. 二元关系的数学定义

二元关系是一个数学概念，其定义为：

设有集合 M、N，则 $M \times N$ 的任意一个子集 R 称为 M 到 N 的一个二元关系。

如果 $M=N$，则称 R 为 M 上的一个二元关系，简称关系。此时，它表示的是 M 中元素之间的某种关联情况。

例如，有集合 $M=\{m_1, m_2, m_3, m_4\}$，$N=\{n_1, n_2, n_3\}$，则

$$R_1=\{<m_1, n_1>, <m_1, n_2>, <m_2, n_2>, <m_3, n_2>\}$$

$$R_2=\{<m_1, n_1>, <m_2, n_2>, <m_3, n_1>, <m_2, n_1>, <m_4, n_2>\}$$

是集合 M 到集合 N 的两个二元关系。

$$R_3=\{<m_1, m_2>, <m_1, m_3>, <m_2, m_3>, <m_3, m_4>\}$$

则是集合 M 上的一个二元关系。

$M \times N$ 可以有若干子集，也就是说集合 M 到集合 N 可以定义若干不同的关系，其数目由集合 M 和 N 的元素个数决定；在实际应用中我们所取的关系是其中的很少一部分，它们是对所讨论的问题有用的也是我们所关心的关系，而对那些与所讨论的问题无关的关系，此时我们不会去关心它。

为了描述关系中元素之间的关联情况，对于关系 R 中的任意一个有序对$<a, b>$（可记为$<a, b> \in R$ 或记为 aRb）定义：

a 是 b 的关于 R 的前件（直接前驱）

b 是 a 的关于 R 的后件（直接后继）

例如，在上面的关系 R_1 中，因为包含了有序对$<m_1, n_2>$，因此，m_1 是 n_2 的关于 R_1 的前件。同样，n_2 是 m_1 的关于 R_1 的后件。

在上述关系 R_3 中，有序对$<m_1, m_2>$表明 m_1 是 m_2 的关于 R_3 的前件，同样，m_2 是 m_1 的关于 R_3 的后件。

2. 二元关系表示的实际意义

二元关系的数学定义是在笛卡儿积的基础上给出的，但从实际意义来说，二元关系表示了集合中两个元素之间的某种相关性。这两个元素可以来自两个不同集合，也可以同属一个集合。实际上这也就是客观事物之间关系的抽象。以下通过两个简单例子说明二元关系所表示的实际意义。

例 1.1 有甲、乙、丙、丁、戊 5 个毕业班学生将进入课题组做毕业设计，课题组有 A、B、C、D、E 5 个老师可以指导学生的毕业设计。为了使每个学生在毕业设计阶段能够得到更多的锻炼，也为了使毕业设计工作更好地与实际科研工作相结合，每个学生的毕业设计题目通过学生和导师之间的双向选择来确定。学生可以结合自己的兴趣，通过公布的毕业设计题目选择导师，导师则可以根据学生的特点来选择学生。假设选择结果为：学生甲的指导老师为 C，学生乙的指导

老师为 A，学生丙的指导老师为 B，学生丁的指导老师为 E，学生戊的指导老师为 D。现要求简单、明了地表示出学生与其指导老师的对应关系，为此可以用一个二元关系来达到此要求：

$$R_4 = \{<甲, C>, <乙, A>, <丙, B>, <丁, E>, <戊, D>\}$$

这里 R_4 是一个二元关系，其中每一个有序对表示每个学生及其对应的导师，有序对的第一分量是学生，第二分量是该学生所对应的导师。这个二元关系清晰地表示了“学生”集合｛甲、乙、丙、丁、戊｝中的元素“学生”和“导师”集合｛A、B、C、D、E｝中的元素 “导师”之间的关联。因此，R_4 也就是对这个问题中学生与导师之间联系的数学抽象。

例 1.2　有编号为 1、2、3、4 的 4 个足球队进行循环赛，比赛的结果为：1 队胜 2 队，3 队胜 2 队，3 队胜 4 队，3 队胜 1 队，1 队胜 4 队，4 队胜 2 队。这里可以用一个二元关系很清楚地表示上述比赛结果：

$$R_5 = \{<1, 2>, <3, 2>, <3, 4>, <3, 1>, <1, 4>, <4, 2>\}$$

这里 R_5 也是一个二元关系，其中每个有序对表示一场比赛的结果，每个有序对的第一分量是该场比赛的胜队，第二分量是该场比赛的负队。这个二元关系表示了同一个集合{1, 2, 3, 4}中的元素 “球队”之间的比赛胜负关系。这里 R_5 是本问题中球队比赛胜负关系的数学抽象。

从上述例子中可以看出，利用二元关系可以简单、明了、确切地表示出集合中两个元素之间的某种相关性。因而可以利用它来描述问题中事物之间的联系，也就是说二元关系是对实际问题进行数学抽象的重要结果。

1.1.3　二元关系的基本性质和几种重要关系

了解二元关系的基本性质，对分析数据元素之间的关系是有意义的；以下给出二元关系的几个主要性质。

1. 二元关系的基本性质

设 R 是集合 M 上的一个关系，则：

（1）若对于每一个 $a \in M$，都有$<a, a> \in R$，那么称 R 是自反关系（自反性）；

（2）若对于任何 $a \in M$，都有$<a, a> \notin R$，那么称 R 是反自反关系（反自反性）；

（3）如果有$<a, b> \in R$，则必有$<b, a> \in R$，那么称 R 是对称的（对称性）；

（4）如果$<a, b> \in R$ 且 $a \neq b$ 时，必有$<b, a> \notin R$，那么称 R 是反对称关系（反对称性）；

（5）如果$<a, b> \in R$ 且$<b, c> \in R$ 时，必有$<a, c> \in R$，那么称 R 是传递关系（传递性）。

假设有集合 M={1, 2, 3, 4, 5}，M 上的一个关系为

R={<1, 2>, <2, 3>, <3, 4>, <3, 5>, <1, 3>, <1, 4>, <1, 5>, <2, 4>, <2, 5>}，

根据上述性质定义，很容易看出该关系 R 是反自反的、反对称的、传递的。

在实际问题中，利用二元关系的基本性质，可以更方便、更确切地分析和描述数据集合中数据元素之间的关联规律和特点，从而有利于对它们的处理。

2. 等价关系与等价类

等价关系是一类重要的二元关系，在处理问题时经常会用到它。等价关系的定义是：

如果非空集合 S 上的关系 R 是自反的、对称的和传递的，则称 R 为 S 上的一个等价关系。

如果 R 是 S 上的等价关系，a, b 是 S 的任意元素，若有$<a, b> \in R$，则称 a 等价于 b，记作 $a \sim b$。

等价关系在现实世界中广泛存在。例如，假设有志愿者集合 S={a, b, c, d, e, f}，S 上的“同组”关系 R 就是等价关系。因为：

① 任何一个人都和自己同一个组，即具有自反性；

② 若 a 和 b 是同一个组的，那么 b 当然也就和 a 同在一个组，即具有对称性；

③ 如果 a 和 b 在一个组，b 与 c 是同组的，那么 a 和 c 一定是同组的，亦即具有传递性。

由此可以看出，这个“同组关系”是一个等价关系。

假设 a、b、c 是同组的，d、e、f 在一个组，则同组关系为

$$R=\{<a,a>,<b,b>,<c,c>,<a,b>,<b,a>,<b,c>,<c,b>,<a,c>,<c,a>,<d,d>,<e,e>,<f,f>,<d,e>,<e,d>,<e,f>,<f,e>,<d,f>,<f,d>\}$$

。

由等价关系可以进一步给出等价类的概念。假设 R 是非空集合 S 上的等价关系，则 S 上互相等价的元素构成了 S 的若干不相交的子集，这些子集称之为 S 的 R 等价类。下面给出等价类的一般定义：

设 R 是非空集合 S 上的等价关系，对任意的 $x\in S$，由 $[x]_R=\{y \mid y\in S\wedge xRy\}$ 给出的集合 $[x]_R$，称为 x 关于 R 的等价类，简称为 x 的等价类，简记为 $[x]$。

例如，对于上述志愿者集合的同组关系，根据“同组”关系得到的不相交的志愿者子集：$\{a,b,c\}$ 和 $\{d,e,f\}$ 分别是志愿者 a 和 d 的等价类。当然，它们也是志愿者 b、c 和志愿者 e、f 的等价类。即有

$$[a]=[b]=[c]=\{a,b,c\},\qquad [d]=[e]=[f]=\{d,e,f\};$$

在实际问题中，当需要根据某种关系来划分集合中的元素时，划分等价类是常用的方法。在后面的章节中，将结合具体应用讨论等价类的划分方法。

3. 偏序关系和全序关系

偏序关系和全序关系也是重要的关系，它们提供了比较集合中元素的工具。

偏序关系的定义是：如果集合 S 上的关系 R 是自反的、反对称的和传递的，则称 R 是集合 S 上的偏序关系。称序偶 $<S,R>$ 为偏序集合，也记为 $<S,\leqslant>$。

例如，集合 $S=\{2,4,6,8\}$ 和其上的整倍数关系 $R=\{<2,2>,<4,4>,<6,6>,<8,8>,<4,2>,<6,2>,<8,2>,<8,4>\}$，则构成偏序集合 $<S,R>$。

对于偏序集 $<S,\leqslant>$ 中的 $x,y\in S$，如果有 $x\leqslant y$ 或 $y\leqslant x$ 成立，就说 x 和 y 是可比较的。由偏序集合的定义可知，偏序集合中的元素不一定都是可比较的，也就是说，它们在偏序中不一定都有确定的位置上的先后关系。若要使集合中的元素之间都可比较，则需要构造下面定义的全序关系。

全序关系的定义是：在偏序集合 $<S,\leqslant>$ 中，如果对于任意的 $x,y\in S$，x 和 y 都是可比较的，则称此偏序关系 $\leqslant$ 为 S 上的全序关系，称序偶 $<S,\leqslant>$ 为全序集合。

例如，集合 $S=\{1,2,3,4\}$ 上的小于等于关系：$R_1=\{<1,1>,<2,2>,<3,3>,<4,4>,<1,2>,<1,3>,<1,4>,<2,3>,<2,4>,<3,4>\}$，即是全序关系，$<S,R_1>$ 为全序集合。

由全序集合的定义可以看出，全序集合中所有元素之间都是可以比较的，这为我们处理实际问题时从给定的一些“前题条件”（即偏序关系）出发，确定集合中各成员之间的先后位置关系提供了办法。因为根据偏序集合与全序集合的关系，我们能够以非空有限偏序集合 $<S,\leqslant>$ 为基础，在集合 S 的元素上构造一个包含了原来的偏序关系的全序，从而使得集合中的全体成员在满足给定的偏序关系的前题下都可比较，这就是在第 5 章中将专门讨论的拓扑排序（也叫拓扑分类）问题。

1.2 什么是数据结构

1.2.1 从实际问题理解数据结构

数据结构学科的形成与发展是和程序设计技术的发展、计算机应用的日益广泛联系在一起的。

数据结构是对实际问题中的数据元素及相互间的联系的抽象。对于数据处理问题，在计算机中的处理效率和处理结果与数据的组织形式是密切相关的。

例 1.3　为了更好地支援和帮助边远地区和有困难的人民群众脱贫致富，暑期实践中，大家都积极报名参加学校组织的扶贫志愿者队伍。若报名时以报名的先后顺序登记报名者的基本信息，即得到扶贫志愿者登记表，如表 1-1 所示。

表 1-1　　扶贫志愿者登记表

姓　名	学号	性别	专　业	学　历	所在院系	服务支持
张　三	20074098	男	通信工程	硕士	电子系	系统维修、开发
李　四	20076323	女	英语	硕士	外语学院	英语教学、培训
⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 1-1 列出了报名者的基本情况，同时也直接反映了报名者的先后顺序，它是对扶贫志愿者报名情况的数据抽象，这个“表”就是从该问题中抽象出来的一个数据结构。表中包含的元素是每个扶贫志愿者的信息，表中各元素之间的关系是报名者之间的报名先后关系。从这个表中，很容易得到：志愿者情况、报名先后、报名人数等信息。因此，它可以为处理问题所用。

当然上述扶贫志愿者登记表的结构不是唯一的形式，根据处理问题的需要可以进行不同的组织。例如，如果处理问题中需要统计各院系的报名情况，那么可以按学院或系为顺序来排列表中志愿者的信息，如表 1-2 所示。

表 1-2　　扶贫志愿者统计表

姓　名	学号	性别	专业	学历	所在院系	服务支持
张　三	20074098	男	信息与通信工程	硕士	电子系	信号处理系统开发
王　五	20074102	男	信息与通信工程	硕士	电子系	嵌入式系统开发
⋮	⋮	⋮	⋮	⋮	⋮	⋮
李　四	20076323	女	英　语	硕士	外语学院	英语教学、培训
⋮	⋮	⋮	⋮	⋮	⋮	⋮

显然，这时更容易反映出各专业特长志愿者的人数，也就更有利于协调各支援地需求和志愿者数量之间的关系。

表 1-1 和表 1-2 都是反映参加扶贫志愿者情况的表，是从同一问题中抽象出来的数据结构，但是由于它们表示了数据元素之间的不同关系，因此它们是两个不同的数据结构。这个例子说明，对于相同的问题，因为处理要求不同，就可能抽象出不同的数据结构。也就是说，数据结构是与问题处理要求有直接关系的。

例 1.4　操作系统对文件的管理是通过对文件目录项的管理来进行的，一个文件的目录项包含表征该文件的信息，包括文件名、文件类型、文件属性、文件长度、建立或修改文件的日期、文件存放的位置等。早期的操作系统，因为硬件条件的限制，所管理的文件少，因此把全部文件目录项放在一个目录表中，简称目录，它就是一个数据结构，其中的数据元素是各文件目录项，数据元素之间的关系是各目录项的某种次序排列，如图 1-1 所示。在这种目录组织方式下，当调用某个文件时，需要按照文

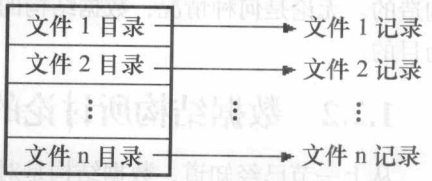


图 1-1　早期的文件目录结构

件名依次查找该文件的目录项，然后才能读出该文件的内容。也就是说，对文件的查找是采用依次查找的方式。

显然，这样的文件目录项组织方式下查找文件的效率低，而且当目录区大小确定以后，也就限制了所能存放的文件数。当需要管理的文件数量急剧增加时，这种文件目录组织形式也就不能适应文件管理的要求了。于是就出现了分层的文件目录管理技术，其目录结构如图 1-2 所示。

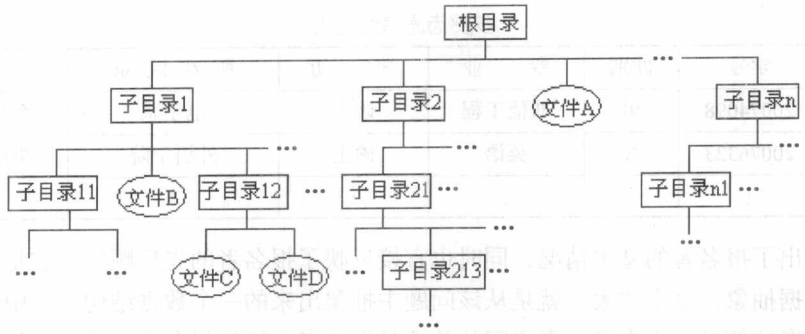


图 1-2　树型目录结构示意图

分层的文件目录结构也叫做树型目录。它使各文件目录之间形成“分支”结构，也就是说，不是把所有文件的目录项都“并列”存放，而是“分路”存放，每一条“路径”可以存放不同内容或不同用户的文件目录。第一层为根目录，根目录下的每一个目录又可以用同样的方式包含其下的子目录，这样层层细分下去一直到某一个具体文件的目录项，从而形成了从“树根”到“树枝”的层层分支，这就构成了另一种数据结构。与图 1-1 的结构不同，这个结构中数据元素之间的关系是层次关系，或者说是一对多的关系，因为每一个目录下包含了多个子目录或文件。这种树型目录结构的好处是，一方面，因为文件目录可以放在不同层上，因此它不会因为根目录区的大小限制所存放的文件个数；另一方面，每一个分支实际上是一条“路径”，查找所要的文件时，不必搜索全部文件目录，而只要根据文件的性质或用户所在的目录，逐层按照“路径”查找即可。比如，要查找图 1-2 中的文件 C，只要按照“根目录→子目录 1→子目录 12→文件 C”这样一条路径查找即可，而不必搜索其他的目录。显然，当存储的文件数量很大时，这种查找文件的方式，要比搜索全部文件目录快得多，从而大大提高了文件读写的效率。

并列的目录结构和分层的目录结构，都是对文件目录管理问题的数据抽象，由于抽象的结果不同，得到的数据结构也就不同。这个例子说明，同样的问题因为所得到的数据结构不一样，处理问题所采用的方法也就不同。也就是说，数据结构与处理问题所采用的算法是密切相关的。自然地，处理问题的效率也会不一样。

通过上述两个例子，我们对数据结构及其重要作用可以有以下的初步认识：数据结构是对实际问题的数据抽象，它描述的是问题中所涉及的数据元素及其相互之间的关系。其所描述的数据元素之间的关系，可以直接反映事物之间客观存在的联系，也可以是根据处理问题的需要而人为构造的。无论是何种情况，数据结构的选取或设计都是以有利于实际问题的处理和提高处理效率为目的。

1.2.2　数据结构所讨论的内容

从上一节已经知道，数据结构是对实际问题中的操作对象及其相互之间的关系的抽象。讨论和研究数据结构的目的，是高效地解决非数值计算问题。对数据结构的讨论涉及数据的逻辑结构、

数据的存储结构、运算的实现（算法设计）三个方面。

1. 数据的逻辑结构

如前所述，数据结构（Data Structure）讨论的是数据（Data）以及数据元素（Data Element）之间的关系，实际上，它表示的是客观事物及其相互之间的联系。计算机中的数据是一个广泛的概念，它包括所有能够被输入到计算机中，并且可以进行存储、处理和输出的信息。例如，字母、数字、声音、图像等都可能是计算机中的数据。整体数据中相对独立的单位称之为数据元素，它是数据处理的基本单位。例如，“字符串”数据中的每个字符，“数值”数据中的每个数等都是其所属数据的数据元素。有时候，一个数据元素可以由若干数据项组成，例如，记录是“文件”的数据元素，而一个记录可以包含若干数据项，此时的数据元素就不是一个简单的数据项。

客观事物之间的联系是各种各样的，反映到数据上来，数据元素之间的联系也各不相同。例 1.1 和例 1.2 就是反映了不同问题中数据元素之间的不同联系。还可以举出许多常见的例子。

例如，一年 12 个月（1 月、2 月、…、12 月）的月份表，表示了月份之间的前后关系。

又如，对一个年级的学生信息按平均成绩由高到低排列，每个学生的信息包括学号、姓名、班级、平均成绩四项信息，则学生信息表中各信息项之间构成了如表 1-3 所示的平均成绩高低的关系。

表 1-3　　学生信息表

班级	学号	姓名	成绩
W 21	10120	张旭	95
W 24	10048	李力坤	92
W 22	10851	王仿固	90
W 21	10252	徐畅	88
W 23	10469	林致意	86
⋮	⋮	⋮	⋮

还有，一个城市的行政区域划分，其关系是城市下辖多个市区，每个市区下辖多个街道和单位，每个街道下辖多个居委会，…。各级行政区域之间是如图 1-3 所示的包含关系或上下级关系。

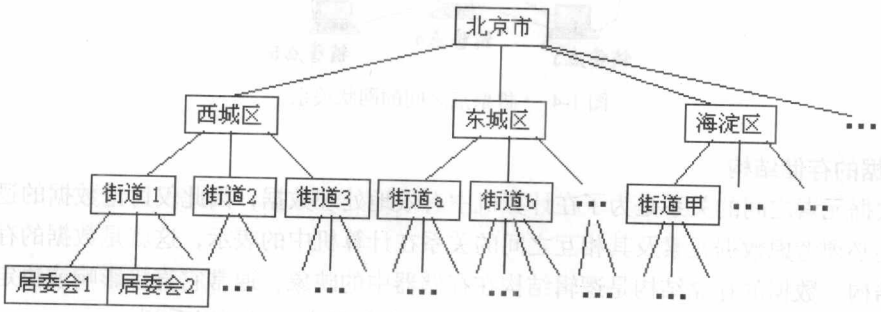


图 1-3　城市的行政区域划分

数据元素之间的关系，有的是客观存在的，比如前面例子中，月份表中各月份之间的先后关系；有的则是为解决问题的需要人为组织的，如学生成绩表中各学生信息之间的关系，文件目录之间的层次关系，城市行政区域的包含关系等。无论是客观存在的或是人为组织的，它们都是表示数据元素之间逻辑上的关系，称之为数据的逻辑结构，简称为数据结构。也就是说，数据结构

反映的是数据元素以及它们之间的相互关系，换句话说，数据结构是互相有关联的数据元素的集合。可以用一个二元组来表示一种数据结构：

$$B=(D,R)$$

其中 B 是一种数据结构，D 是数据元素的集合，R 是 D 上的二元关系的集合。本书讨论的是 R 为某一特定关系的情况。

如前所述，数据元素之间的关系 R 可能是各种各样的，但人们经过分析把它们归纳为两大类，一类为线性关系，另一类为非线性关系。

线性关系反映的是数据元素之间一对一的联系。这类结构称为线性结构，其特点是：

① 有一个元素无前件只有一个后件（头元素）；

② 有一个元素无后件仅有一个前件（尾元素）；

③ 其余的元素有且只有一个前件，有且仅有一个后件。

可见，在线性结构中，数据元素按某种次序排成一个序列，元素之间有着位置上的线性关系。

例如，有数据结构 $B_1=(D_1, R_1)$，其中：

$$D_1=\{d_1, d_2, \cdots, d_n\},\quad R_1=\{(d_i, d_{i+1}) \mid 1\leqslant i\leqslant n-1\}$$

则数据结构 B_1 是线性结构，元素之间的关系可表示为：$(d_1, d_2, \cdots, d_n)$。

除了线性结构之外的数据结构，统称为非线性结构。在非线性结构中，每个数据元素可能与 0 个或多个其他元素有联系，反映的是数据元素之间一对多或多对多的联系。例如：前述操作系统管理下的树型文件目录，因为根目录包含多个子目录，每个子目录又包含多个下一层的子目录，各层目录之间反映的是一对多的关系；城市行政区域划分也是一对多的关系。又如：某城市的火车票售票网，在城区的不同地点设有售票点，各售票点之间通过计算机网络连接起来，则这些售票点之间构成了一种网状关系，如图 1-4 所示。这些售票点之间的联系是多对多的联系。

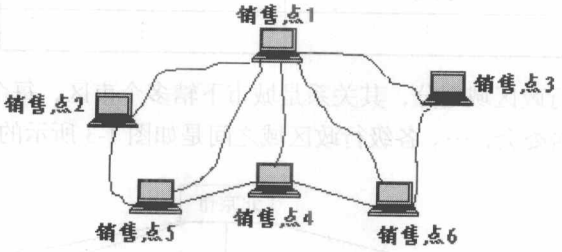


图 1-4　售票点之间的网状关系

2. 数据的存储结构

讨论数据元素之间的关系是为了在计算机中有效地处理数据，因此仅讨论数据的逻辑结构是不够的，还必须考虑数据元素及其相互之间的关系在计算机中的表示，这就是数据的存储结构或者叫物理结构。数据的存储结构是逻辑结构在存储器中的映象，通常它直接影响或决定了数据处理的方法和数据处理的效率。这一点从 1.2.1 节的两个例子中已经可以看到。

为了提高数据处理的效率，可以采用不同的存储结构，经常采用的有四种存储结构。

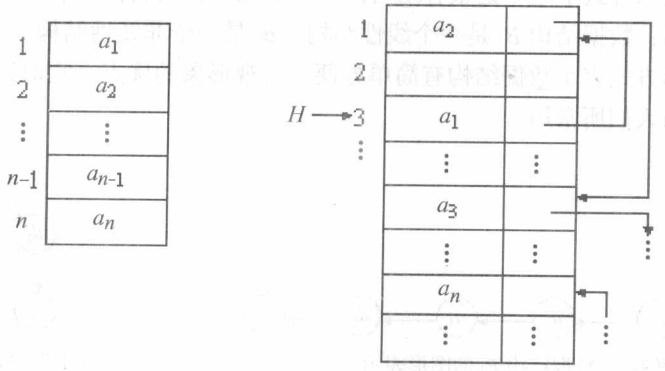
（1）顺序存储方式。在这种存储方式中，数据结构中的元素按其逻辑顺序，依次存放于一组等长的、连续的内存单元中，此时，数据元素之间的关系由存储单元的邻接关系唯一确定。

例如，对于线性结构$(a_1, a_2, \cdots, a_n)$采用顺序存储结构的情况，如图 1-5（a）所示。

顺序存储方式明显的优点是表示数据元素之间的关系简单、直观。

（2）链式存储方式。顺序存储结构虽然有实现简单的优点，但是它需要一组连续的存储单元。因此，它也有存储空间利用不灵活、扩充困难等局限性，为此可以采用链式存储结构。

在链式存储结构中，通过指示数据元素存储地址的指针来表示元素之间的逻辑关系。因此，每个存储单元除了存放数据元素本身之外，还要有指示该元素后件或前件的指针，我们称这样的存储单元为“结点”。各个结点之间在物理位置上可以连续也可以不连续，各个结点的物理顺序与数据元素之间的逻辑顺序可以不一致。例如，线性结构($a_1, a_2, \cdots, a_n$)的链式存储方式如图 1.5（b）所示。显然，这种存储方式增强了数据元素存储的灵活性。



(a) 线性结构的顺序存储方式　　(b) 线性结构的链式存储方式

图 1-5　线性结构的顺序存储和链式存储方式

顺序存储和链式存储是最基本的，也是最常用的两种存储方式，除此之外还有索引存储和散列存储方式。

（3）索引存储方式和散列存储方式。索引存储方式是通过数据元素在线性序列中的位置反映数据元素之间的关系。散列存储方式也就是 HASH 表技术，是通过结点值确定数据元素的存储位置。关于索引存储方式和散列存储方式将在第 6 章中结合查找结构进行讨论。

利用计算机进行数据处理，除了要考虑数据的逻辑结构和存储结构，还必须解决数据处理有关的数据操作问题。也就是说，数据的逻辑结构、数据的存储结构、数据处理中的相关操作及其实现是紧密相关的问题，必须联系起来进行分析和讨论，这就是数据结构所要讨论的内容。

1.2.3　如何表示数据结构

前面我们对数据结构的描述都是采用集合的方式来表示的。数据结构 B 的二元组表示为：$B=(D, R)$。从数据结构的概念出发，用集合的方式描述了数据结构的两个基本要素——数据元素的集合和这个集合上的关系，因而表示了一种确定的数据结构。但是这种表示方式有时不是很方便，也不太直观。为了更直观、形象地表示一种数据结构，在讨论数据结构时，经常采用图形表示方式。

数据结构的图形表示方法是：对结构中的数据集合 D 中的数据元素用中间标有元素值的圆圈（或方框）表示，称之为数据结点，简称结点。亦即一个结点代表 D 中的一个数据元素。用带箭头的连线从前件结点指向后件结点来表示关系 R 中的有序对。

由这两种图素构成的图形可以非常直观、形象地表示各种数据结构。