“十二五”普通高等教育本科国家级规划教材

数据结构：思想与实现

（第2版）

翁惠玉　俞　勇　编著

高等教育出版社

“十二五”普通高等教育本科国家级规划教材

数据结构：思想与实现

（第2版）

翁惠玉　俞　勇　编著

高等教育出版社·北京

内容提要

数据结构是计算机类专业最基础，也是最重要的课程之一，它和程序设计一起为计算学科其他后继课程的学习奠定了基础。

上海交通大学的“数据结构”课程是国家精品课程和精品资源共享课程，本书是该课程的教学成果之一，被列入“十二五”普通高等教育本科国家级规划教材。

本书条理清晰，严格按照线性结构、树状结构、集合结构和图状结构的次序来组织。除常规的数据结构内容之外，还介绍了一些高级的数据结构，如红黑树、AA 树和跳表等，并提供了大量的数据结构应用实例，让读者在学习数据结构的同时，逐步了解为什么要学数据结构以及数据结构对计算机类专业的重要性。

本书内容详实，既注重数据结构和算法的原理，又十分强调和程序设计课程的衔接。在讲授数据结构的同时，不断加强学生对程序设计的理解。书中的算法都有完整的 C++ 实现，这些程序结构清晰，构思精巧，且所有程序都在 VC 6.0 环境下编译通过，并能正确运行。它们既是学习数据结构和算法的示例，也是学习 C++ 程序设计很好的示例。

为便于读者学习与理解，本书为重点、难点内容提供了教学视频，读者可通过扫描二维码观看。完整的课程视频可见 http://www.icourses.cn/coursestatic/course_2945.html。本书可作为高等学校计算机类专业“数据结构”课程的教材，也可供相关技术人员参考。

图书在版编目（CIP）数据

数据结构：思想与实现 / 翁惠玉，俞勇编著. -- 2 版. -- 北京 : 高等教育出版社，2017.11
ISBN 978-7-04-048699-5

Ⅰ. ①数… Ⅱ. ①翁… ②俞… Ⅲ. ①数据结构 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2017)第246475号

数据结构：思想与实现（第2版）
Shuju Jiegou：Sixiang yu Shixian

策划编辑	倪文慧	责任编辑	倪文慧	封面设计	于文燕	版式设计	童　丹
插图绘制	杜晓丹	责任校对	高　歌	责任印制	尤　静		

出版发行	高等教育出版社	网　址	http://www.hep.edu.cn
社　址	北京市西城区德外大街 4 号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.hepmall.com.cn
印　刷	北京机工印刷厂		http://www.hepmall.com
开　本	787 mm×1092 mm　1/16		http://www.hepmall.cn
印　张	30	版　次	2009 年 8 月第 1 版
字　数	680 千字		2017 年 11 月第 2 版
购书热线	010-58581118	印　次	2017 年 11 月第 1 次印刷
咨询电话	400-810-0598	定　价	52.00 元

物 料 号　48699-00

数字资源使用说明

与本书配套的数字资源发布在高等教育出版社数字课程网站，请登录网站后开始课程学习。

一、注册 / 登录

访问 http://abook.hep.com.cn/18567917，单击“注册”，在注册页面输入用户名、密码及常用的邮箱进行注册。已注册的用户直接输入用户名和密码登录即可进入“我的课程”页面。

二、课程绑定

单击“我的课程”页面右上方“绑定课程”，正确输入教材封底防伪标签上的 20 位密码，单击“确定”完成课程绑定。

三、访问课程

在“正在学习”列表中选择已绑定的课程，单击“进入课程”即可浏览或下载与本书配套的课程资源。刚绑定的课程请在“申请学习”列表中选择相应课程并单击“进入课程”。与本书配套的数字资源包括知识点讲解视频、电子教案等素材，以便读者学习使用。

账号自登录之日起一年内有效，过期作废。

如有账号问题，请发邮件至 abook@hep.com.cn。

前　言

数据结构是计算机类专业最基础，也是最重要的课程之一，它和程序设计一起为计算学科其他后继课程的学习奠定了基础。

数据结构和程序设计是关系非常密切的课程。在教学安排中，通常程序设计课程后接着就是数据结构课程。在学习数据结构时，学生往往对程序设计还不是很熟练。通常学生的难点不在于对数据结构的理解，而在于如何用特定的程序设计语言来实现这些数据结构，特别是如何按照面向对象的思想将一个个数据结构设计成一个个类。本书十分强调和程序设计课程的衔接，在讲授数据结构的同时，不断加强学生对程序设计的理解。书中的算法都有完整的C++程序实现，这些程序结构清晰，构思精巧，且都在VC 6.0环境下编译通过，并能正确运行。它们既是学习数据结构和算法的示例，也是学习C++程序设计很好的示例。

本书既适用于过程化方法讲授，也适用于面向对象方法讲授。除第1章外，其余各章的结构基本一致。每章介绍一个数据结构，首先介绍该数据结构所处理的逻辑结构及其常用操作。其次介绍该数据结构的各种实现方法，以及如何将其封装成类。接着介绍C++中对应于该数据结构的工具，告诉读者如何应用现有的工具。最后介绍该数据结构的应用，让读者进一步了解数据结构的重要性。

本书共16章，第1章引言介绍什么是数据结构，什么是算法；回顾面向对象的程序设计方法；还介绍了本书的使用方法。第2~16章被分为五大部分：线性结构、树状结构、集合结构、图状结构和算法设计基础。其中，第2~5章为第1部分，主要讨论线性结构，包括线性表、栈、队列与字符串。第6、7章为第2部分，讨论树状结构，包括树和基于树实现的优先级队列。第8~12章为第3部分，讨论集合结构，这一部分是根据集合的查找和排序两个基本操作组织的，包括集合与静态查找表、动态查找表、排序、外部查找与排序，以及基于集合操作的、用于处理等价类的工具——不相交集。第13~15章为第4部分，讨论图状结构，主要包括图的基本概念、基本操作、实现方法、常见应用，以及图中的两个重要的操作——最小生成树和最短路径问题。最后一部分由第16章组成，介绍基本的算法设计方法，主要包括枚举法、贪婪法、分治法、动态规划和随机算法，使读者在遇到一些没有现成算法的问题时知道如何着手解决问题。

本书内容丰富，除介绍常规的数据结构内容之外，还介绍了一些高级的数据结构，如红黑树、AA树、并查集和跳表等，并提供了大量的数据结构应用实例。让读者在学习数据结构的同时，逐步了解学习数据结构课程的目的，以及该课程对计算机类专业的重要性。

本书第1版于2009年出版。在使用过程中编者收到了许多任课教师和学生的反馈信息，对本书提出了一些意见和建议。第2版就以下几点做了较大的修改。

(1) 进一步理顺框架。

(2) 从初学者的角度出发,尽量讲得通俗,突出基础,突出重点。

(3) 增加重点、难点的教学视频。读者可通过扫描二维码的方式进行在线学习。完整的课程视频可见 http://www.icourses.cn/coursestatic/course_2945.html。

相信修改以后的第 2 版会更加符合读者的需求。但因作者水平有限,书中肯定还会存在很多不足之处,敬请读者批评指正。

编者

2017 年 11 月

目　录

第 2 部分　树 状 结 构

第3部分　集合结构

第 4 部分　图 状 结 构

第 5 部分　算法设计基础

第1章 引 言

1.1 算法与数据结构

视频 1-1：算法与数据结构

在学习数据结构之前，各位读者想必已学过程序设计。在程序设计的学习中，读者已经知道，要编写出解决某一个问题的程序，必须先考虑如何存储待处理的数据以及根据数据处理的要求设计数据处理的算法。作为一门研究信息表示和处理的科学，计算学科设置了一门核心课程：数据结构，专门研究信息的存储和处理。

随着计算学科的发展，计算机的应用已经深入各个领域。计算机处理的数据已不再局限于整型、实型等数值型数据，还可以是字符、表格、声音、图像等非数值型数据。数值计算的特点是数据类型简单、算法复杂，所以更侧重于程序设计的技巧。而非数值计算的特点是数据之间的关系复杂，而且数据量庞大，要设计出好的非数值计算的程序必须解决下列问题。

(1) 理清所处理的数据之间的逻辑关系以及处理要求。非数值计算一般处理的是一批同类数据，如家谱管理系统处理的是家族中所有成员的信息，所需的操作有查找某人的父亲，或查找某人的孩子和所有的子孙。再如仓库管理系统处理的是一批库存信息，所需的操作有改变某一产品的库存或查找某一产品的库存。因此数据之间的逻辑关系包括两个层次：每个数据元素的组成，以及数据元素之间的关系。如家谱管理系统中，每个数据元素是家族中某位成员的信息。描述一个人必须包含姓名、性别、出生年月等信息。数据元素之间的关系主要就是父子关系，其他关系都可以从父子关系推导出来。

(2) 数据在计算机中的存储。数据的保存包含两个方面：数据元素的保存以及数据元素之间关系的保存。非数值计算中的数据元素很少是简单类型，一般都是由多个部分组成，因此每个数据元素可以用程序设计语言中的记录类型的变量或对象来表示，而如何保存数据元素之间的关系则是数据结构研究的内容。

(3) 数据处理的实现。数据元素之间的关系有各种保存方法，对每一种方法数据处理的过程是不同的。每个数据处理的过程就是一个算法。

应用系统可以千变万化，但数据元素之间逻辑关系的种类是有限的，数据结构抛开各种具体的应用和数据元素的内容，通过抽象的方法研究被处理的数据元素之间有哪些逻辑关系（称为逻

辑结构),对于每种逻辑关系可能有哪些操作。然后研究每种逻辑关系在计算机内部如何表示(称为物理结构),对于每一种物理结构其对应的操作如何实现。每个数据结构处理一类逻辑关系,包括逻辑关系的物理表示以及运算的实现。因此,数据结构的讨论可以分为两个层次:抽象层和实现层。抽象层讨论数据的逻辑结构和所需的运算,实现层讨论数据如何存储及运算如何实现。

掌握了数据结构以后,当要解决一个问题时,首先要分析被处理的数据元素之间是什么关系,它需要完成哪些操作,然后选择一个合适的数据结构来处理数据,而不用针对每个应用设计解决方法。

1.1.1　数据的逻辑结构

视频 1-2:
逻辑结构

从概念上讲,一个数据结构是由一组同类的数据元素依据某种联系组织起来的。数据元素间的逻辑关系描述称为数据的逻辑结构。不管应用如何变化,从抽象层面上看,数据的逻辑结构有下列 4 种。

(1) 集合结构:数据元素间的次序是任意的。元素之间除了"属于同一集合"关系外没有其他的关系。例如,某次聚会中的所有人员,公共汽车上的所有乘客,存放在仓库中的产品。由于集合结构的元素间没有固有的关系,因此它的存储往往可以借助于其他结构,不需要设计专门的集合结构。

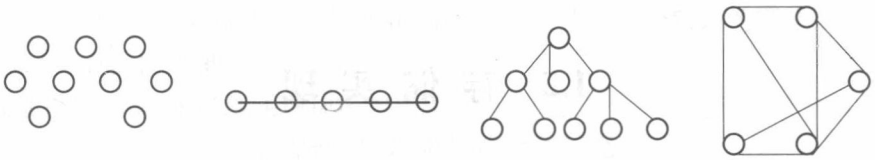
(2) 线性结构:数据元素之间构成一个有序序列。其中,第一个元素之前没有元素,最后一个元素之后没有元素。除了第一个和最后一个元素外,其余元素前面都有唯一的一个元素,后面也有唯一的一个元素。例如,水泊梁山上的 108 条好汉形成了一个数据集合,在他们之间是有次序的,宋江排第一,卢俊义排第二……排在某个元素之前的元素称为它的前驱,排在它后面的元素称为它的后继。

(3) 树状结构:数据元素之间形成一个层次关系。除了一个特殊的根元素外,每个元素有且仅有一个前驱,后继数目不限。根元素没有前驱。例如,一个家族就可以表示为树状结构。这家的老祖宗就是树根,老祖宗的儿子就是老祖宗的后继。每个人可以有多个儿子,因此后继数目不限,但每个人只能有一个父亲,因此只有一个前驱。老祖宗的儿子们又可以有他们的儿子,这样就形成了一棵树。老祖宗的父亲无历史记载,因此老祖宗就没有前驱。

(4) 图状结构:图是最一般的逻辑结构,图中每个元素的前驱和后继数目都不限。例如,在一个计算机网络中,各个网络设备之间是由线路连接起来的。如果把连接看成是网络设备之间的关系,A 设备和 B 设备之间有条通信介质,A 可以通过这条介质发信息给 B,则认为 A 是 B 的前驱,那么一个计算机网络的拓扑结构就形成了一个图状结构,因为每台网络设备都可以和多台其他设备相连接,向多台设备发送信息,也可以接收多台设备发来的信息。

这 4 种结构示意如图 1-1 所示。有时,也把线性结构以外的其他 3 种结构称为非线性结构。

一些表面上很不相同的数据可以有相同的逻辑结构。如前所述,一个计算机网络可以用一



(a) 集合结构　　(b) 线性结构　　(c) 树状结构　　(d) 图状结构

图 1-1　数据的逻辑结构示意

个图状结构来表示。如果用先导课程和后继课程作为课程之间的关系，那么一个课程管理系统中的课程也是一个图状的结构。一门课程可以有若干门先导课程，也可以有若干门后继课程。因此，逻辑结构是数据组织的本质。只要解决了这个本质问题，就可以把解决方案用到各种应用中去，而不必分别对每个问题进行研究。

关于逻辑结构，有几点需要特别说明。

(1) 逻辑结构与数据元素本身的内容无关。如在家谱系统中，不管如何描述一个人，人和人之间的关系总是树状的关系。当用姓名、性别、出生年月描述一个人时是树状关系，对每个人再加上一张照片后还是树状关系。

(2) 逻辑结构与数据元素的个数无关。如在家谱中增加或删去某些人，整个家谱还是一个树状结构。

1.1.2　数据结构的运算

每种逻辑结构都有一定的处理要求，这些处理要求称为数据结构的操作或运算。数据结构最常见的运算有以下几种。

(1) 创建运算(create)：创建一个空的数据结构。

(2) 清除运算(clear)：删除数据结构中的所有数据元素。

(3) 插入运算(insert)：在数据结构指定的位置上插入一个新数据元素。

(4) 删除运算(remove)：将数据结构中的某个数据元素删去。

(5) 搜索运算(search)：在数据结构中搜索满足特定条件的数据元素。

(6) 更新运算(update)：修改数据结构中的某个数据元素的值。

(7) 访问运算(visit)：访问数据结构中的某个数据元素。

(8) 遍历运算(traverse)：按照某种次序访问数据结构中的每一数据元素，使每个数据元素恰好被访问一次。

除了这些运算之外，每种数据结构还可以包含一些特定的运算。如树状结构中，需要找某个数据元素的儿子，或找某个数据元素的父亲的操作；线性结构中，需要找某个元素的前驱或后继的操作。

数据元素之间的逻辑关系和数据结构的运算是数据结构不可分割的两个方面。一个数据结构就是针对某一个逻辑结构讨论数据的存储以及运算的实现，通常称为**存储实现**和**运算实现**。

1.2　存储实现

视频 1-3：
存储实现

存储实现的基本目标是建立数据的机内表示，它包括两部分：数据元素的存储和数据元素之间关系的存储。有时为了方便运算的实现，还可能会增加一些辅助信息的存储。数据的机内表示称为数据的物理结构。按照上述思路，一个物理结构由以下 3 部分组成。

(1) 存储结点，每个存储结点存放一个数据元素。

(2) 数据元素之间关系的存储，也就是逻辑结构的机内表示。

(3) 附加信息，便于运算实现而设置的一些“哑结点”，如链表中的头结点。

其中，前两部分是每个存储实现都必须具备的，第三部分则是根据运算实现的需要而决定是否设置。由于数据元素本身比较简单，因此存储结点的组织也比较简单，可能是一个基本的数据类型，也可能是一个记录类型或用户定义的类类型。因此，物理结构主要讨论的是数据元素之间关系的表示。由于每个数据元素被表示为一个存储结点，所以逻辑结构就由存储结点之间的关联方式间接地表示。存储结点之间的关联通常有以下 4 种实现方式。

(1) 顺序实现：所有的存储结点存放在一块连续的存储区域中，结点之间的逻辑关系可以通过结点的存储位置来体现。即可以通过结点的存储位置找出与它有关系的结点的存储位置。在高级语言中，一块连续的存储空间通常可用一个数组来表示。因此，顺序实现通常用一个数据元素类型的数组来存储。

(2) 链接实现：存储结点可以分散地存放在存储器的不同位置，结点之间的关系通过一个指针显式地指出。因此，在链接存储中，每个存储结点包含两部分：数据元素部分和指针部分。数据元素部分保存数据元素的值，指针部分保存一组指针，每个指针指向一个与本结点有关系的结点。

(3) 散列存储方式：也称哈希存储，是专用于集合结构的数据存放方式。在哈希存储中，各个结点均匀地分布在一块连续的存储区域中，用一个哈希函数将数据元素和存储位置关联起来。

(4) 索引存储方式：所有的存储结点按照生成的次序连续存放。另外设置一个索引区域表示结点之间的关系。

这些基本的存储方式以及它们的组合可以实现数据的灵活存储。

1.3　算法分析

一般而言，对同一个问题可以设计出不同的解决方法。因此，对于一个运算也可以有不同的实现，即可以设计出实现该运算的不同算法。这样就产生了如何评价这些算法的问题。通过这

种评价，设计人员可以区分不同算法实现的相对优劣，从而选择一个较好的算法。通常可以从以下几个方面来评价算法的质量。

(1) 正确性：算法应能正确地实现预定的功能。

(2) 易读性：算法应易于阅读和理解，以便调试、修改和扩充。

(3) 健壮性：当环境发生变化(如遇到非法输入)时，算法能适当地做出反应或进行处理，不会产生不正确的运算结果。

(4) 高效率：具有较高的时间和空间性能。

这些指标往往是互相冲突的，如易读的算法一般效率较差。因此在实际评价中应根据需要有所侧重。数据结构课程着重讨论算法的时空性能，这并不意味着这一指标比其他指标更重要，而仅仅是由于学习阶段所限。

确定算法的时空性能通常称为**算法分析**。算法的时空性能是指算法的**时间性能**(或称**时间复杂度**)和**空间性能**(或称**空间复杂度**)。前者指算法包含的计算量，后者指算法需要的存储量。

1.3.1 时间复杂度的概念

时间性能最直观的度量指标是程序运行时间。但影响程序运行时间的因素有很多，主要包括以下几个方面。

视频 1-4：时间复杂度概念

(1) 问题规模和输入数据的分布。

(2) 编译器生成的目标代码的质量。

(3) 计算机系统的性能。

(4) 程序采用算法的优劣。

当在不同的计算机上运行同一个程序，输入同样的数据，所需运行的时间是不同的。这是因为计算机的硬件性能可能不同，如 CPU 的速度、内存的大小。即使硬件环境一样，但运行的软件环境不一样，运行时间也可能不一样。因为不同的编译器生成的目标代码的质量是不一样的，另外操作系统进程调度方法也会影响程序运行的时间。这就是说，程序运行所需要的时间取决于计算机软硬件系统。用运行时间作为时间性能的衡量指标是不客观的。

衡量一个算法的好坏不能看它在某一台计算机上的运行时间的长短，而要抽取算法的本质，即运算量与问题规模之间的关系。算法所需的运算量与问题规模之间的关系称为算法的**时间复杂度**。算法的时间复杂度是一种抽象的度量。它与所运行的计算机的软硬件无关。优化算法的时间复杂度才是减少运行时间的关键。

算法的运算量除了与问题规模有关以外，还与被处理的数据的分布情况有关。因此，很难用一个指标来描述算法的时间性能。在算法分析中，通常用**最好情况的时间复杂度**、**最坏情况的时间复杂度**以及**平均情况的时间复杂度**来描述。

例如，在一个 n 个元素的数组中查找某个元素是否出现，采用的算法是从第一个元素开始，依次往后查找，直到找到该元素或找遍整个数组都没有找到这个元素。如果被查找的元素出现

在第一个位置，则比较一次就找到了该元素。在这种情况下，算法所需的时间最短，这就是最好情况。如果被查找的元素出现在最后或根本没有出现，则需要比较所有的元素，即比较 n 次。在这种情况下，算法所需的运行时间最长，这就是最坏情况。如果每个元素被查找的概率是相同的，对于多次查找平均起来需要检查 $n/2$ 个元素，这就是平均情况。

在程序运行时不可能如此好运，每次都遇到最好情况，因此最好情况下的时间性能不足以说明问题。平均的时间性能比较难算，有时甚至定义平均本身就已经够难的。因此在衡量时间性能时，通常采用比较容易分析和计算，并且也最具有实际价值的最坏情况的时间性能。

1.3.2　算法运算量的计算

如果有一组正整数，存放在数组 array 中，要求设计一个算法求数组中的最大值与正整数 d 的乘积。对于这个问题，可以设计出如代码清单 1-1 和代码清单 1-2 所示的两个算法。

代码清单 1-1　算法一

```
int max1(int array[ ], int size, int d)
{
    int max = 0, i;
    for (i = 0; i<size ; ++i)  array[i] *= d;
    for (i = 0; i<size ; ++i)
        if (array[i] > max)  max = array[i];
    return max;
}
```

代码清单 1-2　算法二

```
int max2 (int array[ ], int size, int d)
{
    int max = 0, i;
    for (i = 0; i<size ; ++i)
        if (array[i] > max)  max = array[i];
    return max * d;
}
```

这两个算法的主要区别在于：max1 先将 d 分别与数组元素相乘，然后再求最大值；而 max2 先求出数组元素中的最大值，再与 d 相乘。显然对于任意的输入，算法 max1 的计算量比 max2 大，因为当数组中有 n 个元素时，max1 多做了 $n-1$ 次乘法和 n 次赋值。因此，这两个函数在相同的软硬件环境、相同的数据量、相同的数据分布下，前者的运行时间一定比后者多。也就是说，后者的时间性能更好。

从上述讨论可以看出，评价算法的性能并不需要知道精确的运算时间，只要能反映出求解同一问题的不同算法的计算量之间的差异即可。通常采用下述办法来估计算法在给定输入下的计算量。

(1) 根据问题的特点合理地选择一种或几种操作作为“标准操作”，将标准操作作为一个抽象的运算单位。

(2) 确定每个算法在给定的输入下共执行了多少次标准操作，并将它作为算法的计算量。

对于上述问题，如果用乘法、赋值和条件判断作为标准操作，则当输入的数组值为 1,2,3，d=4 时，max1 执行了 3 次乘法（出现在第一个 for 循环的循环体中），14 次赋值（第一个 for 循环的循环控制行中的表达式 1 执行了一个赋值，表达式 3 执行了 3 次，即 3 个赋值，循环体也执行了 3 次，也是 3 个赋值，第二个 for 循环中也是如此），和 11 次比较（第一个循环的循环控制行中的