通信与计算机专业考研指导丛书

数据结构考研指导

SHUJU JIEGOU KAOYAN ZHIDAO

徐寒虹 顾懋柟 编

北京邮电大学出版社
www.buptpress.com

通信与计算机专业考研指导丛书

数据结构考研指导

徐寒虹　顾慰木井　编

北京邮电大学出版社

·北京·

图书在版编目(CIP)数据

数据结构考研指导/徐寒虹,顾懋枘编. —北京:北京邮电大学出版社,2001.10

ISBN 7-5635-0542-3

Ⅰ.数… Ⅱ.徐… Ⅲ.数据结构—研究生—入学考试—自学参考资料

Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2001)第 063956 号

书　　名:数据结构考研指导
编　　者:徐寒虹　顾懋枘
责任编辑:郑　捷
出 版 者:北京邮电大学出版社(北京市海淀区西土城路 10 号)
邮编:100876　电话:62282185　62283578
网址:http://www.buptpress.com
经　　销:各地新华书店
印　　刷:北京源海印刷厂
印　　数:3 000 册
开　　本:850 mm×1 168 mm　1/32　印张:12.25　字数:315 千字
版　　次:2001 年 10 月第 1 版　2001 年 10 月第 1 次印刷
书　　号:ISBN 7-5635-0542-3/TN·243
定　　价:25.00 元

内容简介

本书面向各类《数据结构》课程的应考者和在学者。读者既可以将此书用于考前的全面复习；也可以在学习中作为参考书使用。

书中内容涵盖了《数据结构》课程教学大纲的要求。全书共分十一章，每章按照知识点、内容精要、典型例题解析三个部分精心组织。知识点归纳应该掌握的知识重点；内容精要部分在一个个明确的标题下浓缩了各章的核心内容；典型例题解析部分则重在起到举一反三的作用。

本书还给出了两套模拟试题及参考答案，用于读者自检学习效果。

前　言

《数据结构》课程是计算机专业非常重要的专业基础课，它以大量非数值数据的组织、存储和运算的方法为研究内容，目的是使计算机能够有效地对此类数据进行处理。强调这一点，是希望读者能够把握课程的脉络，同时以“知其然还要知其所以然”的态度积极主动地学习。学习过程中，不妨多问自己“为什么要研究这种方法？”“这么做的好处何在？”“这种方法有无局限性？”……相信这样做你会发现《数据结构》课程的内容并不是枯燥的、抽象的、难于掌握的，而是具有有机联系的、有趣的和实用的。

本书旨在帮助各类《数据结构》课程的应考者和在学者，力求能使大家有事半功倍之效。读者应始终对基本概念、基本原理和基本方法有足够的重视，应考者也是如此，因为以不变应万变才是制胜的法宝。

本书内容共分十一章，依次为概论、线性表、栈和队列、串、多维数组和广义表、树和二叉树、图、查找、内部排序、外部排序、文件；每章又分为三个部分：知识点、内容精要、典型例题解析。同时还附有两套模拟试题及参考答案。

鉴于C/C++语言使用的普遍性，本书采用类C语言作为数据结构和算法的描述语言，但在算法描述时并不刻意追求表达和使用上的技巧，而尽量体现较好的可读性。为便于读者使用，附录中专门对类C语言的语法作了简要介绍。对于书中涉及的运算的实现方法，提醒读者，掌握算法思想是最为重要的，它是算法描述的依据，算法描述本身则具有个人化特点（可以选用不同的实现语言、不同的语句结构等）。

还有一点需要说明的是，在一个完整的算法描述中，应对算法

中使用的数据结构有明确的定义，本书的前几章为了给读者以正确的概念，都是这样做的。从第六章开始，为节省篇幅和避免烦琐，算法中若使用本章已特别给出的存储结构定义时，则将其略去，希望不致造成读者的误解。读者在完成算法，尤其是参加考试时，务必不要省略。

本书的第一章、第二章、第四章、第五章以及第七章至第十章由徐寒虹执笔；第三章、第六章、第十一章以及模拟试题由顾懋棣执笔。本书的编写得到谢楚屏教授和杨俊副教授的大力支持和帮助，在此表示由衷的感谢。

本书的疏漏之处欢迎读者赐教。

编　者

2001年8月

目　录

第一章　概　论

1.1　知识点

- 数据结构的研究目的和研究内容
- 数据结构中的几个重要概念和术语
- 算法设计的基本要求以及算法复杂度的分析和计算方法

1.2　内容精要

1.2.1　概念和术语

数据:是计算机程序所加工处理的描述客观事物的符号的总称。

数据元素(或称为记录、表目):数据的基本单位,是数据集合中的一个个体。一个数据元素可由若干个数据项组成。

数据对象:是具有相同性质的数据元素的集合,是数据的一个子集。

数据结构:具有结构的数据元素的集合。它包括三个方面的内容:数据元素的逻辑结构、存储结构和相适应的运算(操作)。

逻辑结构:数据元素之间的逻辑关系,可用一个二元组表示:

Data_Structure = (D,R)

其中,D——数据元素的有穷集合;

R——D 上关系的有穷集合。

存储结构(或称为物理结构):指数据的逻辑结构在计算机存储器中的映像表示,即在能够反映数据逻辑关系的原则下,数据在存储器中的存储方式。

抽象数据类型(Abstract Data Type, ADT):数据类型概念的引伸。指一个数学模型以及在其上定义的操作集合,面向逻辑层次。

算法:建立在数据结构基础上的、求解问题的一系列确切的步骤。

1.2.2 数据结构的研究目的和研究内容

数据结构的研究目的是寻求有效地组织和处理非数值数据的理论、技术和方法,这类数据具有量大且具有一定内在逻辑关系的特点。

数据结构的核心研究内容包括三个方面:数据的逻辑结构、存储结构以及相应的基本操作运算的定义和实现。

1.2.3 逻辑结构的四种基本形态

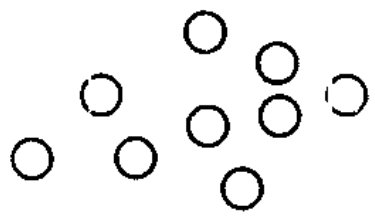
现实世界中非数值数据对象的逻辑关系可以划分为以下四种基本结构,如图 1.1 所示。它们的复杂程度依次递进(例如,以某班级学生作为数据对象,数据元素是学生的学籍档案为例,考察数据元素之间的关系)。

(1) 集合结构:数据元素之间除了"属于同一集合"的联系之外,没有其它关系。例如,认定一个学生是否为班级的成员。

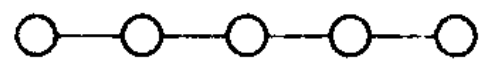
(2) 线性结构:数据元素之间存在一对一的关系。例如,以学生入学报到的时间先后顺序排列数据元素。

(3) 树型结构:数据元素之间存在一个对多个的关系。例如,

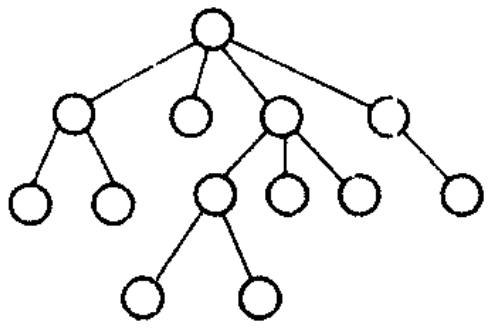
班级中的管理体系，班长管理多个组长，每个组长管理多个组员，……



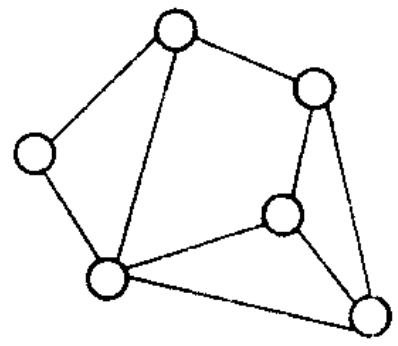
集合结构



线性结构



树结构



图结构

图 1.1　四类基本逻辑结构关系图

（4）图状结构（或称网状结构）：数据元素之间存在多对多的关系。例如，同学之间的朋友关系。

由于集合结构的简单性和松散性，可以用其它结构来表示，因此《数据结构》中通常只讨论后三种逻辑结构。其中树型结构和图结构都属于非线性结构。

1.2.4　数据存储结构的基本组织方式

一种观点认为数据元素之间的关系在计算机中有两种基本的存储结构，分别是顺序存储结构和链式存储结构。

（1）顺序存储结构：借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系，通常顺序存储结构是利用程序语言的数组来描述的。

（2）链式存储结构：在数据元素上附加指针域，借助指针来指示数据元素之间的逻辑关系，通常利用程序语言的指针类型来

描述。

另一种观点是将另外两种存储结构也纳入基本的存储结构类型中，它们是索引存储结构和散列存储结构。尽管实质上这两种结构亦可看作前述两种结构的变化。

(3) 索引存储结构：在存储所有数据元素信息的同时，建立附加的索引表，索引表项的一般形式是(关键字，地址)。关键字是数据元素的某个数据项的值，通过关键字可以找到相关的数据元素的存储地址。

(4) 散列存储结构(也称为哈希存储结构)：此方法的基本思想是根据数据元素的关键字直接计算出相应的存储地址。

上述基本的存储结构既可以单独使用，也可以组合使用，选择何种应视具体要求而定，主要考虑的是操作运算方便以及算法的时空要求。

1.2.5 数据逻辑结构上定义的基本运算

基本运算的种类和数目以及每个基本运算中参数的数目和类型，都应依据数据结构的实际用途和需要来设定，在定义阶段是有灵活性的，实际中不一定要照搬某本教材中的定义，理解这一点非常重要。基本的操作运算只有在一定的存储结构上具体实现之后才有真实的意义，这时使用者就可以按照定义的形式来使用它们了，使用时和高级语言程序设计中的系统函数有相似之处，不必关心该定义是如何实现的。基本的运算通常有以下几种：

(1) 建立数据结构：建立某种指定的数据结构。

(2) 清除数据结构：置某个指定的数据结构为空的结构。

(3) 插入：在数据结构的指定位置上增加一个新的数据元素。

(4) 删除：在数据结构的指定位置上去掉一个数据元素。

(5) 更新：改变数据结构中某个数据元素的值。

(6) 查找：在数据结构中寻找满足一定条件的数据元素。

(7) 排序:使数据元素按某种指定的次序重新排列(一般是在线性结构中)。

(8) 判空和判满:判定某个数据结构是否为空和判定数据结构是否已达到最大允许的容量。

(9) 求长:求指定的数据结构中的数据元素的个数。

1.2.6 在数据结构中引入抽象数据类型概念的好处

运用抽象数据类型的概念实际上是将数据结构的讨论分成两部分:一部分是其概念所涵盖的数据逻辑结构的说明和运算的定义,突出的是在某种关系的数据对象上可以做什么;另一部分是在计算机中如何存储这些数据并具体实现定义的运算,解决的是怎样做的问题。抽象数据类型面向使用层次,隐蔽了实现层次,主要的优点在于实现方法的改变,不会影响用户的使用,可以提高含有该抽象数据类型的软件模块的复用程度。抽象数据类型的概念与面向对象方法的思想是一致的。

1.2.7 算法的五个重要特性

(1) 确定性:其每一条指令必须有确切的含义,相同的输入在算法的多次执行时都一定按同一指令路径,得到相同的输出。

(2) 有穷性:算法指令是有限序列,且算法可以在某段时间内完成。

(3) 可行性:算法中描述的操作都是可以通过可用的基本运算实现的。

(4) 输入:一个算法有零个或多个输入。

(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。

1.2.8 评价算法优劣的基本标准

(1) 正确性:能够确保对于某种相对程度的随机输入有正确的输出。

(2) 可读性:算法描述清晰易懂,便于修改和移植。

(3) 健壮性:当输入非法数据时,算法能作出适当的反应和处理。

(4) 快速性:算法设计合理,执行时间效率高,可以用时间复杂度度量。

(5) 节省性:算法占用存储容量合理,可以用空间复杂度或者存储密度度量。

1.2.9 算法分析的目的

算法的时间复杂度和空间复杂度分析是算法分析的两个主要方面。其目的主要是考察算法的时间和空间效率,以求改进算法或对不同的算法进行比较。一般情况下,鉴于运算空间(内存)较为充足,所以把算法的时间复杂度分析作为重点。

1.2.10 算法时间复杂度的含义

某个指定语句的频度是指它在算法中被重复执行的次数。算法中所有语句的频度之和记做 $T(n)$,它是该算法所求解问题规模 n 的函数。当问题的规模趋向无穷大时,$T(n)$的数量级(阶)称为渐近时间复杂度,简称为时间复杂度,记作 $T(n)=O(f(n))$。

上述表达式中"O"的文字含义是 $T(n)$的量级,其严格的数学定义是:若 $T(n)$和 $f(n)$是定义在正整数集合上的两个函数,则存在正的常数 C 和 n_0,使得当 $n\geqslant n_0$ 时,都满足 $0\leqslant T(n)\leqslant C\cdot f(n)$。

常见的算法时间复杂度的形式按性能降序的排列如下:

$O(1)$, $O(\log_2 n)$, $O(n)$, $O(n\log_2 n)$, $O(n^2)$, $O(n^3)$,

$O(2^n)$、$O(n!)$、$O(n^n)$

1.2.11 算法空间复杂度的含义

空间复杂度是对一个算法在运行过程中临时占用的存储空间(或称辅助空间)大小的量度,一般也作为所求解问题规模 n 的函数,以数量级形式给出,记作 $S(n)=O(f(n))$。

若辅助空间是常数,即空间复杂度为 $O(1)$时,称算法为**就地工作**(原地工作)。

在计算空间复杂度不便的情况下,有时也以存储密度来研究算法的空间使用状况。存储密度 d 定义为:

$$d=\frac{\text{数据本身所占用存储容量}}{\text{数据实际存储所占用存储容量}}$$

1.3 典型例题解析

1.3.1 设有如下两组用二元组表示的数据结构,试分别画出其逻辑结构图。

(1) $DS1=(D,R)$,其中 $D=\{e1,e2,e3,e4,e5,e6,e7,e8\}$,$R=\{r\}$,$r=\{<e1,e2>,<e2,e3>,<e3,e4>,<e4,e5>,<e5,e6>,<e6,e7>,<e7,e8>\}$。

(2) $DS2=(D,R)$,其中 $D=\{e1,e2,e3,e4,e5,e6,e7,e8\}$,$R=\{r\}$,$r=\{(e1,e2),(e1,e3),(e1,e4),(e2,e5),(e2,e6),(e3,e6),(e4,e8),(e5,e7),(e6,e7),(e6,e8)\}$。

解 数据元素之间的逻辑结构可以有不同的表示方式。本题中采用的是二元组的表达式描述;为直观和分析方便起见,也可以用逻辑结构图来表达。逻辑结构图的一般画法是:采用小圆圈代表数据元素,线段代表两个数据元素的关系。数据元素之间的关

系表示在一对尖括号中时，表示其中的两个数据元素呈单向关系，可以采用带箭头的线段来表示；若数据元素之间的关系表示在一对圆括号中，则表示其中的两个数据元素呈双向关系，可以采用不带箭头的线段来表示。照此得到的对应逻辑图分别如图 1.2 和图 1.3 所示。可以看出，图 1.2 反映的是一种线性关系，图 1.3 反映的是图/网状关系。

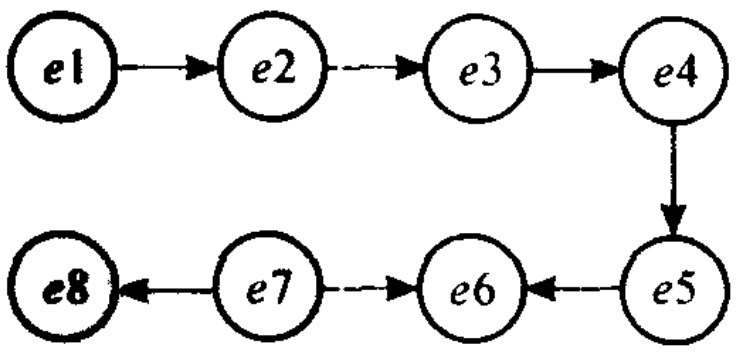


图 1.2　*DS*1 对应的逻辑结构图

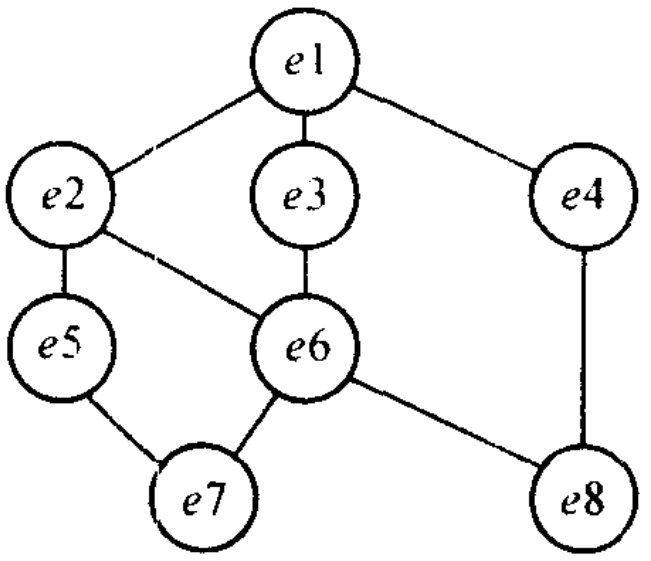


图 1.3　*DS*2 对应的逻辑结构图

1.3.2　求下面程序段的时间复杂度

```
(1) temp = x;                    //语句 1
    x = y;                       //语句 2
    y = temp;                    //语句 3

(2) sum = 0;                                    //语句 1
    for (i = 0; i < n; i++)                     //语句 2
        for (j = 0; j < n; j++)                 //语句 3
            sum = sum + i * j;                  //语句 4

(3) i = 0;                                      //语句 1
    while (i < n) && (a[i] != K)                //语句 2
        i++;                                    //语句 3
    return(i);                                  //语句 4
```