

十一五规划计算机主干课程辅导丛书

数据结构

习题与解析

A级

第3版

李春葆 编著

Exercise

&
Analysis



清华大学出版社

► 十一五规划计算机主干课程辅导丛书

数据结构习题与解析

(A 级第 3 版)

李春葆 编著

清华大学出版社
北 京

内 容 简 介

本书是重点大学的资深教授根据高等学校计算机专业数据结构课程的教学大纲的要求,结合丰富的教学经验编写而成的,通过对概念和习题的讲解与分析,帮助读者了解、掌握数据结构的原理和算法。

本书按照课程的讲授顺序,阐述了线性表、栈和队列、串、数组和稀疏矩阵、递归、广义表、树形结构、图、查找、排序、文件等内容。每章都精选了大量习题,并对习题进行了详细、深入、透彻的分析,使学生充分掌握求解数据结构问题的思想和方法,深化对基本概念的理解,提高分析与解决问题的能力。

本书可以作为计算机专业本、专科学生数据结构课程的学习参考书,也是报考计算机专业硕士研究生的必读参考资料,同时适合于数据结构课程自学者和计算机等级(三级或四级)考试者研习。

版权所有,翻印必究。举报电话:010-62782989 13501256678 13801310933

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。

本书防伪标签采用特殊防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

数据结构习题与解析: A 级/李春葆编著. —3 版. —北京:清华大学出版社, 2006
(“十一五”规划计算机主干课程辅导丛书)

ISBN 7-302-13674-2

I. 数… II. 李… III. 数据结构—高等学校—解题 IV.TP311.12-44

中国版本图书馆 CIP 数据核字(2006)第 099486 号

出 版 者: 清华大学出版社

<http://www.tup.com.cn>

社总机: 010-62770175

地 址: 北京清华大学学研大厦

邮 编: 100084

客户服务: 010-62776969

组稿编辑: 夏非彼

文稿编辑: 王华

封面设计: 林陶

版式设计: 科海

印 刷 者: 北京市耀华印刷有限公司

发 行 者: 新华书店总店北京发行所

开 本: 787×1092 1/16 印张: 23 字数: 559 千字

版 次: 2006 年 10 月第 3 版 2006 年 10 月第 1 次印刷

书 号: ISBN 7-302-13674-2/TP·8250

印 数: 1~5000

定 价: 29.00 元

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话: (010) 82896445

丛 书 序

“习题与解析”系列计算机主干课程辅导丛书自 1999 年推出以来,一直被许多院校师生采用并受到普遍好评,其中也有很多人给我们反馈了不少中肯的改进建议。同时,随着计算机科学与技术的持续发展和演化,传统的计算机专业教学模式也在不断扩充与革新。在计算机教学和教材改革的不断深化过程中,如何促进学生将理论用于实践以及通过实践加深对理论的理解,如何提高其分析问题与动手解决实际问题的能力,都是我们 21 世纪计算机教育亟待解决的问题。

正是基于这样的需求,我们组织专家和一线教师对 2004 年修订的第 2 版“习题与解析”丛书的使用情况进行了深入调研,并对自身教学经验进行认真总结提炼之后,再次修订并隆重推出了这套“十一五规划计算机主干课程辅导丛书”。

丛书组成与新增内容

本套丛书根据计算机专业普遍采用的课程体系,在原丛书的基础上,针对教师备课、出题和学生考研需要,新增了《C 语言习题与解析》(B 级)、《数据结构习题与解析》(B 级)和《计算机组成原理习题与解析》(B 级)等专项分册,并将原先与之对应的各课程升级后称之为 A 级;其他课程根据各自的版本号依次升级。升级的原则是,依据各门课程的最新教学大纲,对原有图书内容进行全面的修订和扩充,使其更加完备、充实。升级之后的新版丛书几乎囊括了计算机专业的各个科目,且与当今计算机专业课程体系更加吻合。

丛书特色

本丛书除保留原丛书的体例风格外,还强化了如下几个方面的内容:

☒ 题型完整,内容丰富

从培养学生扎实的基本功入手,通过选择题、填空题、简答题、算法设计题和综合设计题等各种题目的分析讲解,巩固学生课堂学习的基本概念,扩展教材中讲授的知识,弥补课堂教学中因学时紧张带来的不足,从而帮助学生掌握学习各课程知识的基本技巧。

☒ 以典型题目分析带动能力培养

本丛书注重以典型题目的分析为突破口,点拨解题思路,强化各知识点的灵活运用,启发解题灵感。所有例题不仅给出了参考答案,还给出了详细透彻的分析过程,便于读者在解题过程中举一反三,触类旁通,从而提高分析问题和解决问题的能力。

☑ 全面复习，形成知识体系

本丛书以权威教材为依托，对各知识点进行了全面、深入的剖析和提炼，构成了一个完备的知识体系。在各类考试中，一个微小的知识漏洞，就有可能造成无法弥补的损失，因此复习必须全面扎实。本丛书以基本知识点归纳和例题分析为框架，提纲携领地为考生全程领跑。

☑ 把握知识点间的内在联系，拓展创新思维

把握知识点之间的关系，将掌握的知识变“活”。本丛书通过对知识点的分解，找出贯穿于各知识点之间的内在联系，并配上相关的例题，阐明如何利用这些内在联系解决问题，从而做到不仅授人以“鱼”，更重视授人以“渔”。本套丛书由长期坚持在教学第一线的教授和副教授编写，他（她）们结合自己的教学经验和见解，把多年的教学实践成果无私奉献给读者，希望能够提高学生素质，培养学生的综合分析能力。

☑ 扩充三门核心课程的 B 级版本，提高学习的针对性

“C 语言程序设计”、“数据结构”和“计算机组成原理”是计算机科学及相关专业的考研课程，考生学习难度更高。本丛书的 B 级以考研要求为依据，进一步细化各章的知识点，将相关知识点归类，并进行相应的例题剖析；特别是针对近两年研究生入学考试的真题和典型难解题，增加了“知识创新”和“创新应用”部分，前者是作者在总结相关知识点的基础上归纳总结的解题新方法，后者是应用这些新方法去解决问题的过程。因此，B 级更适合考研者和相关任课教师参考。

读者对象

本套丛书针对本科院校计算机及相关专业的学生作为教学参考书，也适合广大计算机专业本科生作为考研的学习与辅导教材。

如果说科学技术的飞速发展是 21 世纪的一个重要特征的话，那么，教学改革将是 21 世纪教育工作不变的主题，也是需要我们不断探索的课题。紧跟教学改革，不断更新教学内容和教材，真正满足新形势下的教学需求，还需要我们不断地努力实践和完善。本套丛书虽然经过细致地编写与审校，以及几个版次的修订，但仍然难免还有疏漏和不足之处，需要我们不断地补充、修订和完善。我们热情欢迎使用本套丛书的教师、学生和读者朋友提出宝贵意见和建议，使之日臻成熟完善。

本丛书作者的电子邮件：licb@public.wh.hb.cn

本丛书出版者的电子邮件：chiefeditor@khp.com.cn

2006 年 9 月

前 言

计算机编程中加工处理的对象是数据，而数据具有一定的组织结构，所以学习编写程序仅仅了解计算机语言是不够的，还必须掌握数据组织、存储和运算的一般方法，这便是数据结构课程中所学习和研究的内容，也是编写计算机程序的重要基础。由于它对计算机学科起到承前启后的作用，因此，本课程列为计算机等相关专业最重要的专业基础课程。

由于数据结构的原理和算法较抽象，而该课程一般在本科低年级开设，对于只具有计算机程序设计基础知识的初学者，理解和掌握其中的原理就更困难了，特别是在解答数据结构的习题时，往往感到无从下手。对此，作者在多年的教学中感受颇深。本人通过长期的实践、收集与整理，编写了本书，其目的是：通过对习题的解答，使学生充分掌握数据结构的原理以及求解数据结构问题的思路与方法，深化对基本概念的理解，提高分析与解决问题的能力。

本书遵循全国高等学校计算机专业本科数据结构课程的教学大纲的要求，从内容上分为 12 章：第 1 章概述，讨论数据结构的基本概念及相关题解；第 2 章线性表，讨论顺序表和链表的基本概念及相关题解；第 3 章栈和队列，讨论各种栈和队列的基本概念及相关题解；第 4 章串，讨论串的基本概念及相关题解；第 5 章数组和稀疏矩阵，讨论数组和稀疏矩阵的基本概念及相关题解；第 6 章递归，讨论基本递归设计方法及相关题解；第 7 章广义表，讨论广义表的基本概念及相关题解；第 8 章树形结构，讨论树和二叉树的基本概念及相关题解；第 9 章图，讨论图的基本概念及相关题解；第 10 章查找，讨论基本查找方法及相关题解；第 11 章排序，讨论常用的内排序、外排序方法及相关题解；第 12 章文件，讨论基本文件组织结构及相关题解。

每章由两部分组成，第 1 部分简要介绍本章的基本内容，第 2 部分精选了大量的习题，并予以详细解答，涵盖解题思路和求解的完整过程。这些习题中包含一些高校计算机专业招收硕士研究生的数据结构试题（习题题号后面加有“*”号）。

本书在第 2 版的基础上进行了修订和扩充。习题覆盖面广，既收集了较容易的题目，也收集了难度适中和难度较高的题目。因此，本书可以作为计算机专业本、专科学生数据结构课程的学习参考书，也适于报考计算机专业硕士研究生的考生阅读，还适合于数据结构课程自学者和计算机等级（三级或四级）考试者研习。

在编写本书时，作者力求从方法上提高解题的能力，例如，递归问题是学生较难理解的知识点，但在计算机专业知识中又是经常遇到的问题。为此，作者专门以一章的篇幅（第 6 章）深入地分析了递归的执行过程，提出了从递归模型到递归设计的步骤。在其他几章中，也采用了类似的解题方法。

由于习题较多，解答上可能存在不准确或不完美之处，内容编排上也可能存在不合理的地方，敬请广大读者批评指正。

作 者

2006 年 8 月

目 录

第 1 章 概述	1
1.1 基本知识点	1
1.1.1 数据结构的定义	1
1.1.2 存储方式	2
1.1.3 算法及评价	3
1.2 例题分析	6
1.2.1 单项选择题	6
1.2.2 填空题	8
1.2.3 简答题	9
1.2.4 算法设计题	12
第 2 章 线性表	19
2.1 基本知识点	19
2.1.1 线性表的定义	19
2.1.2 线性表的顺序存储结构	20
2.1.3 线性表的链式存储结构	23
2.2 例题分析	35
2.2.1 单项选择题	35
2.2.2 填空题	41
2.2.3 简答题	45
2.2.4 算法设计题	46
第 3 章 栈和队列	69
3.1 基本知识点	69
3.1.1 栈	69
3.1.2 队列	73
3.2 例题分析	77
3.2.1 单项选择题	77
3.2.2 填空题	81
3.2.3 简答题	82
3.2.4 算法设计题	91



第 4 章 串	107
4.1 基本知识点	107
4.1.1 串的定义	107
4.1.2 串的存储及其运算	107
4.1.3 串的模式匹配	114
4.2 例题分析	118
4.2.1 单项选择题	118
4.2.2 填空题	119
4.2.3 简答题	120
4.2.4 算法设计题	123
第 5 章 数组和稀疏矩阵	130
5.1 基本知识点	130
5.1.1 数组	130
5.1.2 稀疏矩阵	134
5.2 例题分析	139
5.2.1 单项选择题	139
5.2.2 填空题	142
5.2.3 简答题	143
5.2.4 算法设计题	146
第 6 章 递归	158
6.1 基本知识点	158
6.1.1 什么是递归	158
6.1.2 递归设计方法	158
6.1.3 递归设计	160
6.1.4 递归到非递归的转换	161
6.2 例题分析	164
6.2.1 单项选择题	164
6.2.2 填空题	164
6.2.3 简答题	167
6.2.4 证明题	168
6.2.5 编程题	169
第 7 章 广义表	184
7.1 基本知识点	184
7.1.1 什么是广义表	184
7.1.2 广义表的表示	184

7.1.3 广义表的基本运算.....	185
7.2 例题分析	189
7.2.1 单项选择题.....	189
7.2.2 填空题.....	190
7.2.3 简答题.....	191
7.2.4 编程题.....	193
第 8 章 树形结构	196
8.1 基本知识点	196
8.1.1 树	196
8.1.2 二叉树.....	199
8.1.3 树和森林.....	205
8.1.4 哈夫曼树.....	206
8.2 例题分析	207
8.2.1 单项选择题.....	207
8.2.2 填空题.....	212
8.2.3 简答题.....	217
8.2.4 证明题.....	226
8.2.5 算法设计题.....	229
第 9 章 图	245
9.1 基本知识点	245
9.1.1 图的基本术语.....	245
9.1.2 图的存储方式.....	246
9.1.3 图的遍历.....	250
9.1.4 最小生成树.....	252
9.1.5 最短路径.....	252
9.1.6 拓扑排序和关键路径.....	256
9.2 例题分析	258
9.2.1 单项选择题.....	258
9.2.2 填空题.....	262
9.2.3 简答题.....	265
9.2.4 证明题.....	271
9.2.5 算法设计题.....	272
第 10 章 查找	278
10.1 基本知识点	278
10.1.1 静态查找表.....	278
10.1.2 动态查找表.....	281



10.1.3 散列表查找.....	287
10.2 例题分析	289
10.2.1 单项选择题.....	289
10.2.2 填空题.....	294
10.2.3 简答题.....	297
10.2.4 证明题.....	308
10.2.5 算法设计题.....	310
第 11 章 排序	316
11.1 基本知识点	316
11.1.1 内排序.....	316
11.1.2 外排序.....	324
11.2 例题分析	326
11.2.1 单项选择题.....	326
11.2.2 填空题.....	332
11.2.3 简答题.....	335
11.2.4 证明题.....	343
11.2.5 算法设计题.....	344
第 12 章 文件	351
12.1 基本知识点	351
12.1.1 顺序文件.....	351
12.1.2 索引文件.....	351
12.1.3 散列文件.....	352
12.1.4 多关键字文件.....	353
12.2 例题分析	353
12.2.1 单项选择题.....	353
12.2.2 填空题.....	354
12.2.3 简答题.....	354
参考文献	358

第1章 概述

本章学习要点

- ☑ 深入掌握数据结构（包括数据的逻辑结构、数据的存储结构和数据的运算）的相关概念。
- ☑ 深入掌握各种逻辑结构的特点。
- ☑ 深入掌握各种存储结构的特点。
- ☑ 深入掌握算法的相关概念，并能分析各种算法的时间复杂度。

1.1 基本知识点

1.1.1 数据结构的定义

表示数据的基本单位是数据元素（或结点）。数据结构是指同一类数据元素中各数据元素之间存在的关系，数据元素通常由若干个数据项构成。数据结构一般包括三个方面的内容，它们分别是数据的逻辑结构、数据的存储结构和数据的运算。

数据的逻辑结构是对数据之间关系的描述，所以有时就把数据的逻辑结构简称为数据结构。逻辑结构形式上用二元组

$$B = (K, R)$$

来表示，其中 K 是结点（即数据元素）的有穷集合，即 K 是由有限个结点所构成的集合， R 是 K 上的关系的有限集合，即 R 是由有限个关系所构成的集合，而每个关系都是从 K 到 K 的关系。设 r 是一个 K 到 K 的关系， $r \in R$ ，若 $k, k' \in K$ ，且 $\langle k, k' \rangle \in r$ ，则称 k' 是 k 的后续， k 是 k' 的前驱，这时 k 和 k' 是相邻的结点（相对 r 而言）；如果不存在一个 k' 使 $\langle k, k' \rangle \in r$ ，则称 k 为 r 的终端结点；如果不存在一个 k' 使 $\langle k', k \rangle \in r$ ，则称 k 为 r 的开始结点；如果 k 既不是终端结点也不是开始结点，则称 k 是内部结点。

数据逻辑结构分为线性结构和非线性结构。在线性结构中有且仅有一个终端结点和一个开始结点，并且所有结点都最多只有一个前驱和后续，顺序表就是典型的线性结构。非线性结构中可能有多个终端结点和多个开始结点，每个结点可能有多个前驱和多个后续。非线性结构中最重要的是树形结构和图形结构。

数据的存储结构是数据的逻辑结构在计算机存储器中的实现，逻辑结构是从逻辑关系上观察数据，它与数据的存储无关，即独立于计算机，而存储结构是依赖于计算机的。计算机存储器是由有限多个存储单元组成的，每个存储单元有惟一的地址，各存储单元的地址是连续编码的，每个存储单元 Z 都有惟一的后续单元 $Z' = \text{succ}(Z)$ ， Z 和 Z' 称为相邻单元。

一片相邻的存储单元的整体叫做存储区域，记做 M 。把 B 存储在计算机中，首先必须建立一个从 K 的结点到 M 单元的映像 $S: K \rightarrow M$ ，即对于每一个 $k \in K$ ，都有惟一的 $Z \in M$ 使得 $S(k)=Z$ ， Z 为 K 结点所占存储空间中的起始单元。通常有四种基本的存储映像方法，即顺序方法、链式方法、索引方法和散列方法。

数据的运算是在数据的逻辑结构上定义的操作算法，如检索、插入、删除、更新和排序等。

1.1.2 存储方式

1. 线性结构的存储方式

线性结构的数据有顺序、链式、索引和散列等四种存储方式。

顺序存储方式是把逻辑上相邻的结点存储在物理上相邻的存储单元里，结点之间的关系由存储单元的邻接关系来体现。其优点是占用最少的存储空间。其缺点是由于每个结点只能使用一整块存储区域，因此可能产生较多的碎片；另外，在做插入或删除运算时需移动大量元素。

链式存储方式是将结点所占的存储单元分为两部分：一部分存放结点本身的信息，即为数据项；另一部分存放该结点的后续结点所对应的存储单元的地址，即为指针项。其优点是充分利用所有的存储单元，不会出现碎片现象。其缺点是每个结点占用较多的存储空间。

索引存储方式是用结点的索引号来确定结点存储地址。其优点是检索速度快。其缺点是增加了额外的索引表，会占用较多的存储空间；另外，在增加和删除数据时还要修改索引表，因而会花费较多时间。

散列存储方式是根据结点的值确定它的存储地址。其优点是检索、增加和删除结点的操作都很快。其缺点是采用不好的散列函数时可能出现结点存储单元的冲突，为解决冲突需要额外的时间和空间开销。

2. 树形结构的存储方式

在树形结构的数据中，每个结点可能有多个后续结点，因此一般只能采用链式的方式进行存储，链式的方式正好能表达树形结构中的父子和兄弟两种层次关系。由于链式的方式不能表达任意多个儿子结点，所以常常使用较规则的树（如二叉树），限制后续结点的最多个数。而其他几种存储方式则难以达到这种要求。

可以在特定规则的情况下，采用顺序结构（如一维数组）来存储树形结构。如图 1.1 所示的一棵二叉树，用以下数组来存储：

1	2	3	4	5	6	7
a	b	c	d	e		f

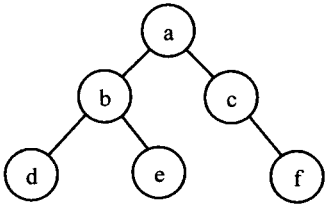


图 1.1 一棵二叉树

3. 图形结构的存储方式

在图形结构的数据中，每个结点可能有多个前驱结点和多个后续结点，因此一般只能采用链式的方式进行存储。同样，由于链式的方式不能表达任意多个后续结点，因此，将链式方式改进成邻接表，即图形结构中的每个结点对应有一个链表，该链表存储这个结点的所有相邻结点。

另外，还可以采用矩阵存储图形结构，即用矩阵表示图形结构中任意两个结点之间的关系，这种矩阵称为邻接矩阵。

1.1.3 算法及评价

算法是解决某一特定类型问题的有限运算序列。描述一个算法可以采用某一种计算机语言，也可以采用流程图等。本书的算法是采用 C/C++ 语言描述的。

算法具有 5 个基本特性：有穷性、确定性、可行性、输入和输出。

评价一个算法一般从 4 个方面进行：正确性、可读性、稳健性和算法效率。其中算法效率通过算法时间复杂度和空间复杂度来描述。

算法时间复杂度是衡量一个算法好坏的重要指标。所谓时间复杂度是指算法中包含简单操作的次数。一般不必要精确计算出算法的时间复杂度，只要大致计算出相应的数量级，如 $O(1)$ 、 $O(\log_2 n)$ 、 $O(n)$ 或 $O(n^2)$ 等。 O 的形式定义为：若 $f(n)$ 是正整数 n 的一个函数，则 $T(n)=O(f(n))$ 表示存在一个正的常数 M ，使得当 $n \geq n_0$ 时满足 $T(n) \leq M \times f(n)$ 。换句话说， $O(f(n))$ 给出了函数 $f(n)$ 的上界。其中， M 为常量。

 **注意：**考虑算法时间复杂度一般是在 n 相当大时的情况，即有当 $n \rightarrow \infty$ 时，

$$\frac{T(n)}{f(n)} = M$$

当算法时间复杂度 $T(n)$ 与 n 无关时， $T(n)=O(1)$ ；当算法时间复杂度 $T(n)$ 与 n 为线性关系时， $T(n)=O(n)$ ；当算法时间复杂度 $T(n)$ 与 n 为二次方关系时， $T(n)=O(n^2)$ ；依此类推。一般地，常用的时间复杂度有如下关系：

$$O(1) \leq O(\log_2 n) \leq O(n) \leq O(n \log_2 n) \leq O(n^2) \leq O(n^3) \leq \dots \leq O(n^k) \leq O(2^n)$$

类似于算法时间复杂度，空间复杂度作为算法所需存储空间的量度，主要考虑在算法运行过程中临时占用的存储空间的大小，一般以数量级形式给出。

【例 1】 分析以下程序段的时间复杂度。

```
for (i=0; i<n; i++)           //①
{
    y=y+1;                     //②
    for (j=0; j<=2*n; j++)     //③
        x++;                   //④
}
```



语句的频度指的是该语句重复执行的次数。一个算法中所有语句的频度之和构成了该算法的运行时间。在本例算法中, 语句①的频度是 $n+1$, 语句②的频度是 n , 语句③的频度是 $n(2n+2)=2n^2+2n$, 语句④的频度是 $n(2n+1)=2n^2+n$ 。于是, 该程序段的时间复杂度 $T(n)=(n+1)+n+(2n^2+2n)+(2n^2+n)=4n^2+5n+1=O(n^2)$ 。

实际上, 可以用算法中基本操作重复执行的频度作为度量标准。被视作基本操作的一般是最深层循环内的语句。

在上例中, 语句④为基本操作, 其频度是 $2n^2+n$ 。所以, 该算法复杂度 $=2n^2+n=O(n^2)$ 。

【例 2】 分析以下程序段的时间复杂度。

```
i=1;
while (i<=n)
    i=i*2;
```

上述算法中基本操作是语句 $i=i*2$, 设其频度为 $T(n)$, 则有

$$2^{T(n)} \leq n$$

即 $T(n) \leq \log_2 n = O(\log_2 n)$ 。所以, 该程序段的时间复杂度为 $O(\log_2 n)$ 。

【例 3】 分析以下程序段的时间复杂度。

```
s=0;
for (i=0; i<=n; i++)
    for (j=0; j<=i; j++)
        for (k=0; k<j; k++)
            s++;
```

该算法的基本操作是语句 $s++$, 其频度为

$$\begin{aligned} T(n) &= \sum_{i=0}^n \sum_{j=0}^i \sum_{k=0}^{j-1} 1 = \sum_{i=0}^n \sum_{j=0}^i (j-1-0+1) = \sum_{i=0}^n \sum_{j=0}^i j \\ &= \sum_{i=0}^n \frac{i(i+1)}{2} = \frac{1}{2} \left(\sum_{i=0}^n i^2 + \sum_{i=0}^n i \right) = \frac{1}{2} \left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} \right) \\ &= \frac{2n^3 + 6n^2 + 4n}{12} = O(n^3) \end{aligned}$$

故该程序段的时间复杂度为 $O(n^3)$ 。

【例 4】 有以下程序, 分析其中 `order()` 函数的时间复杂度。

```
int a[]={2,5,1,7,9,3,6,8};
order(int j,int m)
{
    int i,temp;
    if (j<m)
```

```

{
    for (i=j; i<=m; i++)
        if (a[i]<a[j])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    j++;
    order(j,m);
}
}
main()
{
    int i;
    order(0, 7);
    for (i=0; i<=7; i++)
        cout << a[i] << " ";
}

```

`order()`函数是一个递归排序过程, 设 $T(n)$ (这里 $n=m+1$) 是排序 n 个元素所需要的时间。在排序 n 个元素时, 算法的计算时间主要花费在递归调用 `order()` 上。第一次调用时, 处理的元素序列个数为 $n-1$, 也就是对余下的 $n-1$ 个元素进行排序, 这部分所需要的计算时间应为 $T(n-1)$ 。

又因为在 `for` 循环中, 需要 $n-1$ 次比较, 所以排序 n 个元素所需要的时间为

$$T(n)=T(n-1)+n-1 \quad n>1$$

这样得到如下方程:

$$T(n)=\begin{cases} 0 & n=1 \\ T(n-1)+n-1 & n>1 \end{cases}$$

求解过程为

$$\begin{aligned}
 T(n) &= [T(n-2)+(n-2)]+(n-1) \\
 &= [T(n-3)+(n-3)]+(n-2)+(n-1) \\
 &= \dots \\
 &= (T(1)+1)+2+\dots+(n-1) \\
 &= 0+1+2+\dots+(n-1) \\
 &= n(n-1)/2 \\
 &= O(n^2)
 \end{aligned}$$

故 `order()` 函数的时间复杂度为 $O(n^2)$ 。



1.2 例题分析

1.2.1 单项选择题

【例 1.1】 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 ① 以及它们之间的 ② 和运算等的学科。

- | | | | |
|-----------|---------|---------|---------|
| ① A. 数据元素 | B. 计算方法 | C. 逻辑存储 | D. 数据映像 |
| ② A. 结构 | B. 关系 | C. 运算 | D. 算法 |

解 数据结构主要研究数据元素 (并非数据项) 及其关系和施加在数据上的运算实现等三个方面。本题答案为: ①A ②B。

【例 1.2】 数据的逻辑结构被形式地定义为 $B=(K, R)$, 其中 K 是 ① 的有限集合, R 是 K 上的 ② 的有限集合。

- | | | | |
|---------|---------|---------|---------|
| ① A. 算法 | B. 数据元素 | C. 数据操作 | D. 逻辑结构 |
| ② A. 操作 | B. 映像 | C. 存储 | D. 关系 |

解 由数据的逻辑结构形式化定义可知本题答案为: ①B ②D。

【例 1.3】 在数据结构中, 从逻辑上可以把数据结构分成_____。

- | | |
|---------------|---------------|
| A. 动态结构和静态结构 | B. 紧凑结构和非紧凑结构 |
| C. 线性结构和非线性结构 | D. 内部结构和外部结构 |

解 逻辑结构反映数据元素之间的逻辑关系, 线性结构表示数据元素之间为一对一的关系, 非线性结构表示数据元素之间为一对多或多对一的关系。本题答案为 C。

【例 1.4*】 数据结构在计算机内存中的表示是指_____。

- | | |
|------------|--------------|
| A. 数据的存储结构 | B. 数据结构 |
| C. 数据的逻辑结构 | D. 数据元素之间的关系 |

解 数据的存储结构是数据结构 (实际上指逻辑结构) 在计算机内存中的表示, 它既保存数据元素也保存数据元素之间的关系。本题答案为 A。

【例 1.5*】 在数据结构中, 与所使用的计算机无关的是数据的_____结构。

- | | | | |
|-------|-------|----------|-------|
| A. 逻辑 | B. 存储 | C. 逻辑和存储 | D. 物理 |
|-------|-------|----------|-------|

解 物理结构即为存储结构。逻辑结构描述数据元素之间的关系, 与使用的计算机无关, 而存储结构是逻辑结构在计算机中的表示, 与计算机有关。本题答案为 A。

【例 1.6*】 算法分析的目的是 ①, 算法分析的两个主要方面是 ②。

- | | |
|-----------------|-------------------|
| ① A. 找出数据结构的合理性 | B. 研究算法中的输入和输出的关系 |
| C. 分析算法的效率以求改进 | D. 分析算法的易懂性和文档性 |

- ② A. 空间复杂度和时间复杂度
B. 正确性和简明性
C. 可读性和文档性
D. 数据复杂性和程序复杂性

解 算法分析是为了找出高效的算法, 算法分析最重要的是算法效率分析, 其中包含时间复杂度和空间复杂度分析。本题答案为: ①C ②A。

【例 1.7】 计算机算法指的是 ①, 它必须具备输入、输出和 ② 等 5 个特性。

- ① A. 计算方法
B. 排序方法
C. 解决问题的有限运算序列
D. 调度方法
② A. 可行性、可移植性和可扩充性
B. 可行性、确定性和有穷性
C. 确定性、有穷性和稳定性
D. 易读性、稳定性和安全性

解 由算法的定义和特性可知本题答案为: ①C ②B。

【例 1.8*】 在以下的叙述中, 正确的是_____。

- A. 线性表的线性存储结构优于链表存储结构
B. 二维数组是其数据元素为线性表的线性表
C. 栈的操作方式是先进先出
D. 队列的操作方式是先进后出

解 线性表的线性存储结构不一定优于链表存储结构, 例如在查找数据时, 线性存储结构优于链表存储结构; 栈的操作方式是先进后出; 队列的操作方式是先进先出。一维数组是典型的线性表, 而二维数组可以看成元素为一维数组的线性表。本题答案为 B。

【例 1.9】 在决定选取何种存储结构时, 一般不考虑_____。

- A. 各结点的值如何
B. 结点个数的多少
C. 对数据有哪些运算
D. 所用编程语言实现这种结构是否方便

解 存储结构是数据的逻辑结构在计算机存储器中的实现, 存储结构有顺序、链式、索引和散列等, 它们都可以存储各结点的值, 主要区别在于各结点之间的关系不同, 因此在决定选取何种存储结构时, 一般不考虑各结点值如何。本题答案为 A。

【例 1.10】 在存储数据时, 通常不仅要存储各数据元素的值, 而且还要存储_____。

- A. 数据的处理方法
B. 数据元素的类型
C. 数据元素之间的关系
D. 数据的存储方法

解 在存储数据时, 需要存储数据元素的值和数据元素之间的关系。本题答案为 C。

【例 1.11*】 下面说法错误的是_____。

- (1) 算法原地工作的含义是指不需要任何额外的辅助空间
(2) 在相同的规模 n 下, 复杂度 $O(n)$ 的算法在时间上总是优于复杂度 $O(2^n)$ 的算法
(3) 所谓时间复杂度是指最坏情况下, 估计算法执行时间的一个上界
(4) 同一个算法, 实现语句的级别越高, 执行效率越低