HZ BOOKS
华章教育

计算机学科硕士研究生
入学统一考试课程参考教材

Detailed Solutions of the Data Structure's Exercises for Graduate Entrance Examination

数据结构习题精析与考研辅导

殷人昆 编著

- 考研大纲权威解释
- 考点解析透彻清楚
- 历年真题深入剖析
- 备考方法贴心提示
- 丰富教学阅卷经验
- 模拟试卷全面演练

机械工业出版社
China Machine Press

计算机学科硕士研究生
入学统一考试课程参考教材

数据结构习题精析与考研辅导

考研大纲权威解释
考点解析透彻清楚
历年真题深入剖析
备考方法贴心提示
丰富教学阅卷经验
模拟试卷全面演练

机械工业出版社
China Machine Press

本书是根据《全国硕士研究生入学统一考试计算机学科专业基础综合考试大纲》编写的学习数据结构的辅导教材。全书共分8章。第1章介绍数据结构课程的地位和主要知识点，数据结构和算法的基本概念和算法分析的简单方法，以及C语言编程的要点。第2～7章对应考试大纲的6个方面，包括线性表，栈、队列和多维数组，树与二叉树，图，查找，排序，分别进行详解。每个方面细分为若干知识点，每个知识点按照“知识点复习一关键问题点拨一选择填空题解析一综合应用题选讲”等4个步骤层层深入，有针对性地讲解和分析。在紧紧把握考试大纲的前提下，尽可能深入细节、扩展知识面、联想相关数据结构。第8章对历年联考的真题做了精确解析，细化了考试大纲各个知识点的要求，并提供了学习指导和应试指南。

本书融入作者30多年数据结构教学的经验，考虑了不同层次学生学习的需要，精选了630个例题，覆盖了相关知识点的方方面面，既可以作为大学计算机专业学习数据结构课程的辅助教材，也可以作为计算机专业考研的辅导教材。

图书在版编目（CIP）数据

数据结构习题精析与考研辅导/殷人昆编著. —北京：机械工业出版社，2011.1
ISBN 978-7-111-32283-2

Ⅰ.数…　Ⅱ.殷…　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（2010）第204129号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：张少波　刘立卿
北京诚信伟业印刷有限公司印刷
2011年1月第1版第1次印刷
185mm×260mm·23印张
标准书号：ISBN 978-7-111-32283-2
定　价：45.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

前　言

根据教育部办公厅教学厅（2008）11号文件要求，从2009年起，全国硕士研究生统一入学考试计算机学科专业基础综合考试全国联考，考试科目包括数据结构、计算机组成原理、操作系统和计算机网络。要求考生比较系统地理解相关科目的基本概念、基本原理和方法，能够运用所掌握的基本原理和方法分析、设计和解决相关的理论问题和实际问题。

数据结构科目占45分，从考研大纲可以看到，考核的主要知识点涵盖线性表，栈、队列与多维数组，树与二叉树，图，查找和排序等6个方面。从考试出题的点和面分析，基本覆盖了这6个方面。然而，与其他考试科目比较，数据结构是最不好把握，试题灵活性最强，最容易在细节上失分的科目。因此，如何提高数据结构学习的效果，全面掌握数据结构的相关知识点并能合理运用，是应考的关键。

许多学习数据结构课程的学生和正在复习数据结构课程的考生最感困惑的问题是如何抓住复习的重点，如何把握考核的范围。按道理讲，根据考试大纲复习不就可以了吗？遗憾的是，考试大纲只给出了一个大概的范围，到底考到何种深度和广度，考试大纲并未指明。例如，“栈和队列的应用”是一个必考的知识点，凡是在解决问题时涉及栈或队列的都算栈和队列的应用，其范围很难界定。所以在计算机学科专业基础综合考试中，数据结构是最难复习的课程。许多学生都希望通过选择一本权威的参考书来解决所有问题，殊不知任何一本优秀的参考书都有其优点和缺点。有的参考书覆盖范围广但不深，有的参考书对某些问题阐述比较清楚但覆盖面不够，有的因为出书较早不能反映学科的发展，有的因为著作者的教学经验不足不能深入挖掘由知识点拓展出来的知识，有的甚至想当然地划分重点而没有了解学生的感受和需要。从几次联考阅卷来看，我确实为某些考生感到可惜，他们看了不少书，也做了不少题，然而某些题还是得不了高分。

本书作者从1978年便开始学习数据结构。从1983年开始，曾为清华附中和北京四中的中学生开课。1987年从日本回国后走上大学数据结构课程的讲堂。1992年开始与严蔚敏老师合作为清华大学计算机系本科生开课，1996～2008年承担清华大学考研数据结构和程序设计课程的命题和批改任务。针对不同层次的学生，使用过严蔚敏、刘美纶、许卓群、张乃孝以及本人自己编写的数据结构教材，积累了较多的教学经验。特别是通过与清华大学本科生的互动，与清华大学夜大学大专生的互动，与高教自考培训学生的互动，与北京广播电视大学学生的互动，与清航考研培训网站学生的互动，作者对数据结构的许多知识点有较深层次的理解，愿意通过本书，将这些经验与广大读者或考生共享。

本书共分8章，第1章分3部分，首先较为概括地介绍数据结构课程的地位和主要考点，这是一个引子；然后介绍数据结构相关的基本概念和算法设计、分析、评价的简要知识和方法；最后简单介绍C语言中涉及算法编写的相关知识。这一章虽然不是考研大纲所要求的，但有助于后续章节相关数据结构和算法设计的理解。

第2～7章涵盖考研大纲的6个方面。每一个知识点都按照“知识点复习—关键问题点拨—选择填空题解析—综合应用题选讲”来复习，通过例题深入分析讲解，以期达到举一反三的目的。在“知识点复习”部分简单概括了相关知识点的主要内容和要点说明，部分算法给出了算法思路和源代码。在“关键问题点拨”部分介绍了许多教科书上没有讲到的但不可忽视的细节辨析，这些都是通过与学生互动而得到的。在“选择填空题解析”部分对相关知识点涉及的数据结构定义、特点、性质，存储实现，算法分析等进行考查，并详细解释和分析可能混淆的概念。在“综合应用题选讲”部分中有证明题、问答题和算法设计题。根据知识点的可能出题频率，这类题在各个知识点中分配的数量有所不

同，在某些知识点中题的数量较多，在有的知识点中题的数量较少，多数题都给出了求解思路和求解过程。

第8章也由3部分组成。首先对历年联考的真题做了精确解析，并总结了阅卷中发现的学生易犯的错误，对于算法设计题特别提示了应如何对其阅读和理解。接下来详细列出考研大纲各知识点的细化要求，最后导出考研的复习方法和应试指南，可供读者参考。

本书是作者本着作为一名教师，为考生“解惑”而编写的。它不但适合考研的学生使用，对在校学生学习数据结构课程也很有帮助。书中精选了各个大学和研究院的历年考研试题中的部分真题，也参考了部分国内外相关数据结构教材，还参考了网上的一些资料。由于时间有限，书中难免存在疏漏之处，恳请广大读者批评指正。

作者联系方式：yinrk@ tsinghua. edu. cn。

目　录

前言

第 1 章　引论 …… 1
1.1　数据结构课程的地位和考试要求 …… 1
1.1.1　数据结构课程的地位 …… 1
1.1.2　考试要求 …… 1
1.1.3　考查的知识点 …… 1
1.2　数据结构和算法的预备知识 …… 3
1.2.1　数据结构的主要概念 …… 3
1.2.2　算法及算法分析 …… 4
1.2.3　选择填空题解析 …… 6
1.2.4　综合应用题选讲 …… 8
1.3　使用 C/C++ 的几个规则 …… 11
1.3.1　算法结构 …… 11
1.3.2　函数参数 …… 11
1.3.3　条件运算 …… 11
1.3.4　动态存储分配 …… 12
1.3.5　标准输入/输出 …… 12
1.3.6　指针 …… 12
第 2 章　线性表 …… 13
2.1　线性表的定义和基本操作 …… 13
2.1.1　知识点复习 …… 13
2.1.2　关键问题点拨 …… 13
2.1.3　选择填空题解析 …… 14
2.2　线性表的存储表示 …… 14
2.2.1　知识点复习 …… 14
2.2.2　关键问题点拨 …… 17
2.2.3　选择填空题解析 …… 17
2.2.4　综合应用题选讲 …… 18
2.3　线性表的插入和删除运算 …… 19
2.3.1　知识点复习 …… 19
2.3.2　关键问题点拨 …… 21
2.3.3　选择填空题解析 …… 21
2.3.4　综合应用题选讲 …… 24
2.4　线性表的应用 …… 33
第 3 章　栈、队列和多维数组 …… 39
3.1　栈和队列的基本概念 …… 39
3.1.1　知识点复习 …… 39
3.1.2　关键问题点拨 …… 40
3.1.3　选择填空题解析 …… 40
3.1.4　综合应用题选讲 …… 42
3.2　栈的存储结构 …… 44
3.2.1　知识点复习 …… 44
3.2.2　关键问题点拨 …… 45
3.2.3　选择填空题解析 …… 46
3.2.4　综合应用题选讲 …… 47
3.3　队列的存储结构 …… 49
3.3.1　知识点复习 …… 49
3.3.2　关键问题点拨 …… 51
3.3.3　选择填空题解析 …… 52
3.3.4　综合应用题选讲 …… 53
3.4　栈和队列的应用 …… 58
3.4.1　知识点复习 …… 58
3.4.2　关键问题点拨 …… 60
3.4.3　选择填空题解析 …… 61
3.4.4　综合应用题选讲 …… 61
3.5　数组与特殊矩阵的压缩存储 …… 72
3.5.1　知识点复习 …… 72
3.5.2　关键问题点拨 …… 75
3.5.3　选择填空题解析 …… 77
3.5.4　综合应用题选讲 …… 78
第 4 章　树与二叉树 …… 87
4.1　树的基本概念 …… 87
4.1.1　知识点复习 …… 87
4.1.2　关键问题点拨 …… 88
4.1.3　选择填空题解析 …… 88
4.1.4　综合应用题选讲 …… 88
4.2　二叉树的定义和特性 …… 88
4.2.1　知识点复习 …… 88
4.2.2　关键问题点拨 …… 90
4.2.3　选择填空题解析 …… 90
4.2.4　综合应用题选讲 …… 91
4.3　二叉树的存储和遍历 …… 93
4.3.1　知识点复习 …… 93
4.3.2　关键问题点拨 …… 95
4.3.3　选择填空题解析 …… 96

4.3.4 综合应用题选讲 …… 101
4.4 线索二叉树 …… 110
4.4.1 知识点复习 …… 110
4.4.2 关键问题点拨 …… 112
4.4.3 选择填空题解析 …… 112
4.4.4 综合应用题选讲 …… 113
4.5 树与森林 …… 118
4.5.1 知识点复习 …… 118
4.5.2 关键问题点拨 …… 123
4.5.3 选择填空题解析 …… 124
4.5.4 综合应用题选讲 …… 125
4.6 二叉排序树 …… 131
4.6.1 知识点复习 …… 131
4.6.2 关键问题点拨 …… 134
4.6.3 选择填空题解析 …… 135
4.6.4 综合应用题选讲 …… 136
4.7 平衡二叉树 …… 144
4.7.1 知识点复习 …… 144
4.7.2 关键问题点拨 …… 147
4.7.3 选择填空题解析 …… 148
4.7.4 综合应用题选讲 …… 149
4.8 Huffman 树与 Huffman 编码 …… 151
4.8.1 知识点复习 …… 151
4.8.2 关键问题点拨 …… 153
4.8.3 选择填空题解析 …… 153
4.8.4 综合应用题选讲 …… 155
4.9 堆 …… 157
4.9.1 知识点复习 …… 157
4.9.2 关键问题点拨 …… 160
4.9.3 选择填空题解析 …… 161
4.9.4 综合应用题选讲 …… 162
第 5 章 图 …… 164
5.1 图的基本概念 …… 164
5.1.1 知识点复习 …… 164
5.1.2 关键问题点拨 …… 165
5.1.3 选择填空题解析 …… 165
5.1.4 综合应用题选讲 …… 166
5.2 图的存储及基本操作 …… 168
5.2.1 知识点复习 …… 168
5.2.2 关键问题点拨 …… 170
5.2.3 选择填空题解析 …… 171
5.2.4 综合应用题选讲 …… 173
5.3 图的遍历 …… 177
5.3.1 知识点复习 …… 177
5.3.2 关键问题点拨 …… 179
5.3.3 选择填空题解析 …… 180
5.3.4 综合应用题选讲 …… 182
5.4 最小（代价）生成树 …… 189
5.4.1 知识点复习 …… 190
5.4.2 关键问题点拨 …… 192
5.4.3 选择填空题解析 …… 193
5.4.4 综合应用题选讲 …… 195
5.5 最短路径 …… 200
5.5.1 知识点复习 …… 200
5.5.2 关键问题点拨 …… 202
5.5.3 选择填空题解析 …… 202
5.5.4 综合应用题选讲 …… 204
5.6 拓扑排序 …… 211
5.6.1 知识点复习 …… 211
5.6.2 关键问题点拨 …… 212
5.6.3 选择填空题解析 …… 213
5.6.4 综合应用题选讲 …… 214
5.7 关键路径 …… 217
5.7.1 知识点复习 …… 217
5.7.2 关键问题点拨 …… 218
5.7.3 选择填空题解析 …… 218
5.7.4 综合应用题选讲 …… 219
第 6 章 查找 …… 222
6.1 查找的基本概念 …… 222
6.1.1 知识点复习 …… 222
6.1.2 关键问题点拨 …… 222
6.2 顺序查找法 …… 223
6.2.1 知识点复习 …… 223
6.2.2 关键问题点拨 …… 225
6.2.3 选择填空题解析 …… 226
6.2.4 综合应用题选讲 …… 227
6.3 折半查找法 …… 230
6.3.1 知识点复习 …… 230
6.3.2 关键问题点拨 …… 231
6.3.3 选择填空题解析 …… 231
6.3.4 综合应用题选讲 …… 233
6.4 B 树与 B^+ 树 …… 237
6.4.1 知识点复习 …… 237
6.4.2 关键问题点拨 …… 243
6.4.3 选择填空题解析 …… 244
6.4.4 综合应用题选讲 …… 247

6.5 散列表及其查找 …… 252
6.5.1 知识点复习 …… 252
6.5.2 关键问题点拨 …… 257
6.5.3 选择填空题解析 …… 259
6.5.4 综合应用题选讲 …… 261
第7章 排序 …… 269
7.1 排序的基本概念 …… 269
7.1.1 知识点复习 …… 269
7.1.2 关键问题点拨 …… 269
7.2 四种简单的排序方法 …… 270
7.2.1 知识点复习 …… 270
7.2.2 关键问题点拨 …… 272
7.2.3 选择填空题解析 …… 274
7.2.4 综合应用题选讲 …… 276
7.3 希尔排序 …… 281
7.3.1 知识点复习 …… 281
7.3.2 关键问题点拨 …… 282
7.3.3 选择填空题解析 …… 283
7.3.4 综合应用题选讲 …… 283
7.4 快速排序 …… 285
7.4.1 知识点复习 …… 285
7.4.2 关键问题点拨 …… 286
7.4.3 选择填空题解析 …… 287
7.4.4 综合应用题选讲 …… 288
7.5 堆排序 …… 294
7.5.1 知识点复习 …… 294
7.5.2 关键问题点拨 …… 295
7.5.3 选择填空题解析 …… 296
7.5.4 综合应用题选讲 …… 297
7.6 二路归并排序 …… 299
7.6.1 知识点复习 …… 299
7.6.2 关键问题点拨 …… 301
7.6.3 选择填空题解析 …… 302
7.6.4 综合应用题选讲 …… 303
7.7 基数排序 …… 307
7.7.1 知识点复习 …… 307
7.7.2 关键问题点拨 …… 310
7.7.3 选择填空题解析 …… 310
7.7.4 综合应用题选讲 …… 310
7.8 各种内排序方法的比较和选择 …… 311
7.8.1 知识点复习 …… 311
7.8.2 关键问题点拨 …… 312
7.8.3 选择填空题解析 …… 313
7.8.4 综合应用题选讲 …… 314
第8章 试题分析与备考指南 …… 319
8.1 全国硕士研究生入学考试真题分析 … 319
8.1.1 2009年联考试题数据结构部分 …… 319
8.1.2 2010年联考试题数据结构部分 …… 325
8.2 考试复习建议 …… 331
8.2.1 试题难度分析 …… 331
8.2.2 风险和机遇 …… 332
8.2.3 主要知识点的难度级别和重点级别 …… 332
8.2.4 复习建议 …… 342
8.2.5 考试指导 …… 343
8.2.6 结束语 …… 345
模拟试题及参考答案 …… 346
参考文献 …… 356

第1章 引 论

1.1 数据结构课程的地位和考试要求

1.1.1 数据结构课程的地位

数据结构是计算机科学与技术专业本科生的专业基础课程之一，是程序设计系列课程中一个不可或缺的环节，对于信息系统的研究和开发起着重要的支撑作用。因此，国内外高等院校计算机和软件工程专业都把“数据结构”列为考研的必考科目。2009 年教育部更是把这门课程列为全国硕士研究生入学考试计算机专业基础综合考试的考试科目之一，在满分 150 分中占了 45 分。复习好“数据结构”课程，对于通过联考有着至关重要的作用。

1.1.2 考试要求

2010 年教育部指定的《全国硕士研究生入学统一考试计算机学科专业基础综合考试大纲》明确提出，对于数据结构部分，主要考查：

（1）理解数据结构的基本概念，掌握数据的逻辑结构、存储结构及其差异，以及各种基本操作的实现。

（2）在掌握基本的数据处理原理和方法的基础上，能够对算法进行时间复杂度和空间复杂度分析。

（3）能够选择合适的数据结构和方法进行问题求解。具备采用 C、C++ 或 Java 语言设计与实现算法的能力。

换句话说，考查的目标有两个：知识和能力。

1. 知识方面

从数据结构的结构定义和使用，以及存储表示和操作的实现两个层次，系统地考查：

（1）掌握常用的基本数据结构（包括顺序表、链接表、栈与队列、数组、二叉树、堆、树与森林、图、查找结构、索引结构、散列结构）及其不同的实现。

（2）掌握分析、比较和选择不同数据结构、不同存储结构、不同算法的原则和方法。

2. 能力方面

从解决问题的角度出发，系统地考查：

（1）掌握运用基本数据结构来设计算法的能力。

（2）掌握算法设计和分析的思考方式及技巧，提高分析问题和解决问题的能力。

知识方面在全国联考的试卷中占 20 分，主要通过选择填空题方式考查；能力方面在全国联考的试卷中占 25 分，主要通过综合应用题方式考查。

1.1.3 考查的知识点

分析 2010 年的考试大纲，对其主要条目细化和整理，总结出数据结构部分主要考查的知识点有 45 个，分布在 6 章内。

1. 线性表

包括 4 个知识点：

（1）线性表的定义、特点和基本操作（已考）。

（2）线性表的存储表示，包括顺序存储和链式存储（已考）。

（3）特殊链表的定义和基本运算的实现，包括循环链表和双向链表。

（4）线性表的应用，包括基于一维数组的一些算法、一元多项式的组织和操作等。

2. 栈、队列和数组

包括 7 个知识点：

（1）栈与队列的定义、特点和基本操作（已考）。
（2）栈的存储表示及其操作的实现，包括顺序栈和链式栈。
（3）队列的存储表示及其操作的实现，包括循环队列和链式队列。
（4）特殊队列的定义和存储表示，包括双端队列（已考）和优先队列。
（5）栈与队列的应用，包括递归改非递归（分治与回溯）、表达式计算、括号配对、数值转换、分层处理。
（6）多维数组的定义和特点，包括二维数组计算、基于矩阵的算法实现。
（7）特殊矩阵的定义和特点，包括对称矩阵、三对角线矩阵、稀疏矩阵。

3. 树与二叉树

包括9个知识点：
（1）树与二叉树的定义、性质（已考）。
（2）二叉树的存储表示，包括顺序存储和链式存储。
（3）二叉树的遍历及其应用，包括前序、中序、后序（已考）和层次序遍历。
（4）线索二叉树的定义（已考）、存储表示和寻找前驱、后继。
（5）树与森林的存储表示（已考）和遍历（已考）。
（6）二叉排序树的定义、存储表示，基于二叉排序树的算法实现。
（7）平衡二叉树的定义（已考）、存储表示、基本操作和平衡旋转（已考）。
（8）哈夫曼（Huffman）树的定义（已考）、存储表示和应用。
（9）堆的定义（已考）、存储表示和基本操作的实现（已考）。

4. 图

包括8个知识点：
（1）图的基本概念，包括顶点的度、路径、回路、连通图（已考）等。
（2）图的存储表示，包括邻接矩阵、邻接表。
（3）图的遍历，包括深度优先搜索和广度优先搜索的实现。
（4）最小生成树的定义（已考）、构成方法，包括Kruskal算法和Prim算法。
（6）最短路径的概念（已考）和求解方法，包括Dijkstra算法和Floyd算法。
（7）拓扑排序的定义、实现方法（已考）。
（8）关键路径的定义、求解方法。

5. 查找

包括6个知识点：
（1）查找的概念和性能评价标准。
（2）顺序查找算法和性能分析。
（3）折半查找算法和性能分析（已考）。
（4）m阶B树的定义和存储结构（已考）、查找、插入与删除的平衡化。
（5）m阶B^+树的定义和存储结构、查找、插入与删除的平衡化。
（6）散列法的概念、散列函数的选择、解决冲突的线性探测法（已考）、二次探测法、双散列法和链地址法、散列法的性能分析。

6. 内排序

包括11个知识点：
（1）排序的概念和性能评价标准，包括排序码比较次数、元素移动次数、附加存储数、稳定性。
（2）直接插入排序的思路、算法和性能分析（已考）、稳定性。
（3）折半插入排序的思路、算法和性能分析、稳定性。
（4）起泡排序的思路、算法和性能分析（已考）、稳定性。
（5）简单选择排序的思路、算法和性能分析（已考）、稳定性。
（6）快速排序的思路、算法和性能分析（已考）、稳定性。
（7）堆排序的思路、算法和性能分析、稳定性。

（8）二路归并排序的思路、算法和性能分析（已考）、稳定性。
（9）基数排序的思路、算法和性能分析。
（10）排序方法的性能比较（已考）。
（11）排序方法的应用。
在复习过程中要切实掌握这些知识点，才能对考试应付自如。复习方法参看第8章。

1.2　数据结构和算法的预备知识

本节介绍的知识是考试大纲没有明确要求，但贯穿于所有各章的必备知识，对于理解和掌握各章的内容起到了支持作用。

1.2.1　数据结构的主要概念

1．什么是数据

数据是信息的载体，是描述客观事物的数、字符以及所有能输入到计算机中，被计算机程序识别和处理的符号的集合。

数据主要分两大类：数值型数据和非数值型数据。数据结构主要研究的是非数值型数据。

2．什么是数据元素

数据的基本单位就是数据元素。例如，在学校的学籍管理系统中，学生文件是由一系列学生记录组成的，每个学生记录就是一个数据元素。在计算机程序中数据元素常被作为一个整体进行考虑和处理。数据元素又可称为元素、结点、记录。

有时一个数据元素可以由若干数据项组成。数据项是具有独立含义的最小标识单位。例如，每个学生记录又可由学号、姓名、性别等数据项组成。

数据元素的集合构成一个数据对象，它是针对某种特定的应用。

3．什么是数据结构

数据结构指某一数据元素集合中的所有数据成员之间的关系。完整的定义为：

$$\text{数据结构} = \{ D, R \}$$

其中，D 是某一数据元素的集合；R 是该集合中所有数据成员之间的关系的有限集合。

4．数据结构的三个要素

数据结构包括三个要素：逻辑结构、物理结构和作用于数据结构的运算。

数据结构是指数据元素间的逻辑关系，即数据的逻辑结构。数据逻辑结构的要点包括：

- 数据的逻辑结构从逻辑关系上描述数据，与数据如何存储无关；
- 数据的逻辑结构可以看作是从具体问题抽象出来的数据模型（面向应用的）；
- 数据的逻辑结构与数据元素本身的形式、内容无关；
- 数据的逻辑结构与数据元素的相对存储位置无关。

数据的物理结构是数据元素及其关系在计算机内存储的表示，也称为数据的存储结构。

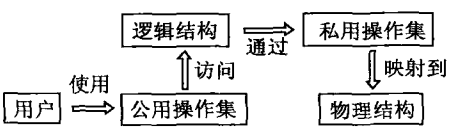

图1-1　数据结构的三个要素之间的关系

作用于数据结构上的运算是讨论数据结构的另一个重要方面。讨论数据结构必须讨论相关的操作，操作的实现依赖于相应的存储结构。图1-1展示了数据结构的三个要素之间的关系。

5．数据逻辑结构的分类

数据的逻辑结构通常划分为线形结构、集合结构、树形结构和图结构。如图1-2所示。后三种统称为非线性结构。

线形结构中元素之间的关系是一对一的关系，集合结构中元素之间的关系为空，树形结构中元素之间的关系是分层的一对多的关系，图结构中元素之间的关系是多对多的关系。

这4种关系中集合结构的实现往往采用其他逻辑结构的存储表示。

6．数据存储结构的分类

数据存储结构分为4类：顺序存储表示、链接存储表示、索引存储表示、散列存储表示。

通常在内存中组织各种数据结构，都可以采用顺序存储表示或链接存储表示。前者采用连续的存储区域相继存放数据元素，后者采用链表存储数据元素并通过各个元素附加的指针将它们按其逻辑顺序链接起来。但对于数量特别大的数据元素集合，一般需要存放于外存中。因此，有索引存储表示和散列存储表示。前者通过建立索引表来组织所有数据元素，后者通过散列函数直接把数据记录的关键码映射为该元素的存放地址。

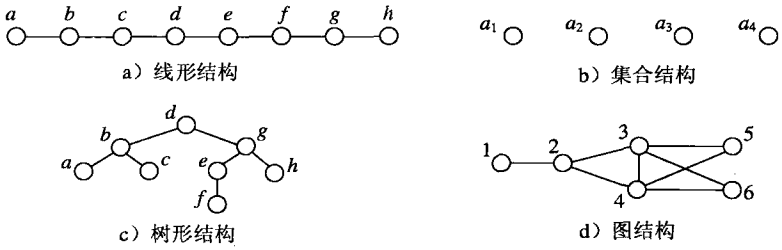

图 1-2 数据逻辑结构的分类

7. 数据类型与抽象数据类型

数据类型是一组性质相同的值的集合，以及定义于这个值集合上的一组操作的总称。数据类型可分为两大类：基本数据类型和构造数据类型。

基本数据类型可以看作是计算机中已实现的数据结构。例如，C 语言中的基本数据类型有字符型（char）、整型（int）、浮点型（float）、双精度型（double）和无值型（void），可直接使用由它们定义的变量和相应的操作。

构造数据类型由基本数据类型或构造数据类型组成，在应用中可视为一种数据模型。构造数据类型由不同成分的类型构成，在 C 语言中用 typedef struct 来定义。

数据类型和数据结构的共同点在于它们都有抽象性，并不特指适用于何处，可根据问题的需要和特定的使用环境，直接使用它们来表示相关的数据元素的构成或这些数据元素之间的关系。

数据类型与数据结构之间的区别在于，数据结构本身是一种数据的组织形式和使用形式，通过把数据结构定义成数据类型才能在计算机上使用。从这个意义上来看，数据类型是指从编程者使用的角度来看可由计算机实现的数据结构。注意，数据类型本身不能参与运算，必须定义属于某种数据类型的变量，使用这些变量来参与运算。

抽象数据类型由用户定义，用以表示应用问题的数据模型。抽象数据类型由构造数据类型组成，包括数据的组成和一组相关的服务（或称操作）。

抽象数据类型的三大特征是信息隐藏、数据封装、使用与实现相分离。面向对象方法中的类（class）完美地实现了抽象数据类型。

1.2.2 算法及算法分析

1. 算法的概念

所谓算法，就是基于特定的计算模型，在信息处理过程中为了解决某一类问题而设计的一个指令序列。算法的 5 个要素是：

- 输入：指待处理的信息，即用数据对具体问题的描述；
- 输出：指经过处理后得到的信息，即问题的答案；
- 确定性：对于相同的输入数据，算法执行确定的预设路线，得到确定的结果；
- 可行性：算法的每一基本操作都可以实施，并能够在常数时间内完成；
- 有穷性：对于任何输入，算法都能经过有穷次基本操作得到正确的结果。

2. 设计算法的三个阶段

设计算法通常经过三个主要阶段：

（1）从问题出发，寻找可能的解决方案，结合计算机选择合适的算法。

（2）建立解决问题的数据模型和程序框架，并用伪代码描述一系列步骤。

（3）细化程序框架，用程序设计语言写出完整的数据结构和程序代码。

3. 算法的性能分析

设计算法就是为了求解问题。算法的效率是衡量是否具有可计算性的关键。性能分析的目的就是要了解算法的效率。性能指算法实际执行的功效或表现如何，主要从算法执行的时间和空间效率进行分析。

时间复杂度

算法的时间复杂度度量是通过统计算法的每一条指令的执行次数和执行时间得到的。因此，算法的时间复杂度的计算公式为：

算法的执行时间 = 算法中每条语句的执行时间之和

每条语句的执行时间 = 该语句的执行次数（或频度）× 该语句执行一次所需时间

语句执行一次所需时间取决于机器的指令性能、速度和编译所产生的代码质量，很难确定。因此设每条语句执行一次所需时间为单位时间，则一个算法的时间复杂度 = 该算法中所有语句的执行次数（或频度）之和。

例如，有一个求两个 n 阶方阵的乘积 $C=A\times B$ 的算法，其执行次数如表 1-1 所示。

表 1-1

程　序　语　句	语句执行次数
`void MatrixMultiply(int A[][],int B[][],int C[][],int n)`	
`{`	
`int i,j,k;`	
`for(i=0;i<n;i++)`	$n+1$
`for(j=0;j<n;j++)`	$n(n+1)$
`{`	
`C[i][j]=0;`	n^2
`for(k=0;k<n;k++)`	$n^2(n+1)$
`C[i][j]=C[i][j]+A[i][k]*B[k][j];`	n^3
`}`	
`};`	
总　计	$2n^3+3n^2+2n+1$

渐近时间复杂度：大 O 表示法

算法中所有语句的频度之和是矩阵阶数 n 的函数：

$$T(n)=2n^3+3n^2+2n+1$$

称 n 是问题的规模，则时间复杂度 $T(n)$ 是问题规模 n 的函数。当 n 趋于无穷大时，称时间复杂度的数量级为算法的渐近时间复杂度。

$$T(n)=O(n^3)$$——算法的大 O 表示

大 O 表示表明当 $n\to\infty$ 时 $T(n)$ 的变化趋势。大 O 表示的较严格的定义是：

如果存在正常数 a、N 和一个函数 $f(n)$，使得对于任何 $n>N$，都有

$$T(n)<a\times f(n)$$

则可以认为在 n 足够大之后，$f(n)$ 给出了 $T(n)$ 的一个上界，记为

$$T(n)=O(f(n))$$

事实上，大 O 表示给出了算法执行效率的一个保守的估计，即对于规模为 n 的任意输入，算法的执行时间都不会超过 $O(f(n))$。对于表 1-1 所示的例子，其大 O 表示为

$$T(n)=O(n^3)$$

在分析一个程序的渐近时间复杂度时，需要抓关键操作，为此需要分解程序结构。在程序的许多并列的程序段中找出最复杂的程序段，特别是包括多层嵌套循环的程序段，其最内层循环中的执行语句即为关键操作。分析关键操作的执行次数，就可直接得到大 O 表示的结果。例如，表 1-1 所示程序最内层循环中执行语句 `C[i][j]=C[i][j]+A[i][k]*B[k][j]`就是关键操作，它的执行次数为

n^3，则整个程序的大 O 表示为 $O(n^3)$，这样不必再逐条语句进行计算。

空间复杂度

空间复杂度是对算法的存储效率的度量。算法所用存储空间可分为两大部分：固定部分和可变部分。

存储空间的固定部分主要包括程序指令代码所占用的空间，常数、简单变量、定长成分（如数组元素、结构成分、对象的数据成员等）变量所占用的空间等。这部分空间可以通过程序分析很容易地统计出来。存储空间的可变部分主要包括尺寸与问题规模有关的成分变量所占用的存储空间、递归栈所占用的存储空间、通过 malloc 和 free 命令动态使用的存储空间等。这部分存储空间需要分析程序的执行过程才能统计出来。

通常，空间复杂度的度量只是在相同功能的不同算法之间比较优劣时才使用。在这种情况下，一般比较它们完成功能时所需的附加存储空间即可。

不同大 O 表示的比较

表 1-2 显示了不同的函数（$=f(n)$）当 n 增大时的增长趋势。如果一个算法的时间复杂度达到 $O(\log_2 n)$，就说明这个算法具有很高的时间效率，如果一个算法的时间复杂度达到 $O(n!)$，一旦 n 变大，算法的运行时间将达到不可计算的程度。因此，如果用“<”表示优于，则有

$$c < \log_2 n < n < n\log_2 n < n^2 < n^3 < 2^n < 3^n < n!$$

表 1-2

n	$\log_2 n$	$n\log_2 n$	n^2	n^3	2^n	$n!$
4	2	8	16	64	16	24
8	3	24	64	512	256	80 320
10	3.32	33.2	100	1000	1024	3 628 800
16	4	64	256	4096	65 536	2.1×10^{13}
32	5	160	1024	32 768	4.3×10^{9}	2.6×10^{35}
128	7	896	16 384	2 097 152	3.4×10^{38}	∞
1024	10	10 240	1 048 576	1.07×10^{9}	∞	∞
10 000	13.29	132 877	10^8	10^{12}	∞	∞

1.2.3 选择填空题解析

题 1.2.1 顺序存储表示中数据元素之间的逻辑关系是由（ ① ）表示的，链接存储表示中数据元素之间的逻辑关系是由（ ② ）表示的。

A. 指针　　B. 逻辑顺序　　C. 存储位置　　D. 问题的上下文

【解析】 ①选 C，②选 A。顺序存储的存储位置与逻辑结构中的逻辑顺序是对应的；链接存储用指针把元素按照其逻辑关系链接起来，不要求数据元素的逻辑顺序与存储顺序有对应关系。选哪种存储结构是由问题的上下文确定的。

题 1.2.2 计算机所处理的数据一般具有某种关系，这是指（　　）。

A. 数据与数据之间存在的某种关系　　B. 数据元素与数据元素之间存在的某种关系

C. 元素内数据项与数据项之间存在的某种关系　D. 数据文件内记录与记录之间存在的某种关系

【解析】 选 B。数据由数据元素构成，特定上下文数据元素构成的集合即为数据对象。一般数据处理考虑的是数据对象内各元素之间的关系。数据元素是数据的基本标识单位，数据项是数据的最小处理单位。

题 1.2.3 以下关于数据结构的说法中，错误的是（　　）。

A. 数据结构相同，对应的存储结构也相同

B. 数据结构涉及数据的逻辑结构、存储结构和施加其上的操作共三个方面

C. 数据结构操作的实现与存储结构有关

D. 定义逻辑结构时可不考虑存储结构

【解析】 选 A。数据结构通常指的是逻辑结构。同一逻辑结构可对应不同的存储结构。例如字典，可用顺序表、链表、散列表、索引表来实现。数据结构的操作在不同存储下有不同的实现。

题 1.2.4 以下关于数据结构的说法，正确的是（　　）。

A. 数据结构的逻辑结构独立于其存储结构

B. 数据结构的存储结构独立于该数据结构的逻辑结构

C. 数据结构的逻辑结构唯一地决定了该数据结构的存储结构

D. 数据结构仅由其逻辑结构和存储结构决定

【解析】 选 A。数据的逻辑结构是面向问题的，是从应用的需求出发而建立的，至于如何在计算机上存储，这是设计时再考虑的内容，所以数据的逻辑结构可独立于存储结构来组织。反之，数据的存储结构是逻辑结构在计算机中的映像，它不能独立于逻辑结构存在；此外，同一逻辑结构在计算机中如何存储，要看是否易于访问，是否易于修改或增删，是否利于安全保密等，相应地有不同存储结构可以选用；数据结构不是孤立的，不但要考虑数据的组织，还要考虑作用在数据结构上的操作，即数据对象的行为，因此数据结构不是仅由其逻辑结构和存储结构决定的。

题 1.2.5 下面关于抽象数据类型的描述，不正确的是（　　）。

A. 数据封装　　B. 使用与实现分离　　C. 信息隐藏　　D. 用例驱动

【解析】 选 D。抽象数据类型（ADT）的三个特点是数据封装、信息隐藏、使用与实现分离，即把数据结构的实现封装在模块内部，使用者只能通过模块接口的操作来访问模块内存储的数据。其作用是使将来的变更局部化。目前 ADT 已成为系统实现的核心技术。用例是系统分析时描述功能的图示，与 ADT 无关。

题 1.2.6 算法的时间复杂度与（　　）有关。

A. 问题规模　　B. 计算机硬件的运行速度

C. 源程序的长度　　D. 编译后执行程序的质量

【解析】 选 A。算法的具体执行时间与计算机硬件的运行速度、编译产生的目标程序的质量有关，但这属于事后测量。算法的时间复杂度的度量属于事前估计，与问题的规模有关。

题 1.2.7 某算法的时间复杂度是 $O(n^2)$，表明该算法（　　）。

A. 问题规模是 n^2　　B. 问题规模与 n^2 成正比

C. 执行时间等于 n^2　　D. 执行时间与 n^2 成正比

【解析】 选 D。算法的时间复杂度是 $O(n^2)$，这是设定问题规模为 n 的分析结果，所以 A、B 都不对；它也不表明算法的执行时间等于 n^2，只表明算法的执行时间 $T(n) \leqslant c \times n^2$（$c$ 为比例常数）。有的算法，如 $n \times n$ 矩阵的转置，时间复杂度为 $O(n^2)$，不表明问题规模是 n^2。

题 1.2.8 下面说法中错误的是（　　）。

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 在相同问题规模 n 下时间复杂度为 $O(n)$ 的算法总是优于时间复杂度为 $O(2^n)$ 的算法

③ 所谓时间复杂度是指在最坏情形下估算算法执行时间的一个上界

④ 同一个算法，实现语言的级别越高，执行效率越低

A. ①　　B. ①②　　C. ①④　　D. ③

【解析】 选 A。算法原地工作的含义指空间复杂度为 $O(1)$。

题 1.2.9 在下列程序中：

```
void calc( int p1,int p2) {
    p2 = p2 *p2;  p1 = p1 - p2;  p2 = p2 - p1;
}   //calc
void main {
    int i = 2,j = 3;
    calc(i,j);  cout << j << endl;
}   //main
```

当参数传递采用引用方式（call by reference）时，所得结果 $j=$（　①　）；

A. 2　　B. 16　　C. 20　　D. 28

当参数传递采用赋值方式（call by value）时，所得结果 $j=$（　②　）。

A. 0　　B. 3　　C. 5　　D. 6

【解析】 ①选 B，②选 B。引用方式传递参数的实质是传送作为实际参数的变量的地址，这样函

数体内的运算将会直接对该变量进行操作，改变变量本身的内容。赋值方式传递参数只是将变量的内容赋给参数，函数体内只是对该变量的副本进行操作，这样不能改变作为实际参数的变量本身的内容。

题 1.2.10 设有以下三个函数：

$$f(n)=100n^3+n^2+1000,\quad g(n)=25n^3+4000n^2,\ h(n)=n^{2.01}+1000n\log_2 n$$

以下关系式中有错误的是（　　）。

A. $f(n)=O(g(n))$　　B. $g(n)=O(f(n))$　　C. $h(n)=O(n^{2.01})$　　D. $h(n)=O(n\log_2 n)$

【解析】 选 D。因为 $f(n)=O(n^3),g(n)=O(n^3)$，当 $n\to\infty$ 时，显然 $n^{2.01}$ 比 $n\log_2 n$ 增长得快，$h(n)=O(n^{2.01})$，所以，$f(n)=O(g(n)),g(n)=O(f(n))$，而 $h(n)=O(n\log_2 n)$ 不对。

题 1.2.11 下列函数中渐近时间复杂度最小的是（　　）。

A. $T_1(n)=n\log_2 n+1000\log_2 n$　　B. $T_2(n)=n^{\log_2 3}-1000\log_2 n$

C. $T_3(n)=n^2-1000\log_2 n$　　D. $T_4(n)=2n\log_2 n-1000\log_2 n$

【解析】 选 A。因为 $T_1(n)=O(n\log_2 n),T_2(n)=O(n^{\log_2 3}),T_3(n)=O(n^2),T_4(n)=O(n\log_2 n)$。虽然 $T_1(n)=O(T_4(n))$，但 $T_4(n)$ 中 $n\log_2 n$ 是 $T_1(n)$ 的 2 倍，所以 $T_1(n)$ 的渐近时间复杂度最低。

题 1.2.12 判断下列各对函数 $f(n)$ 和 $g(n)$，当 $n\to\infty$ 时，增长最快的函数是（　　）。

A. $f(n)=10^2+\ln(n!+10^n)$　　$g(n)=2n^4+n+7$

B. $f(n)=(\ln(n!)+5)^2$　　$g(n)=13n^{2.5}$

C. $f(n)=n^{2.1}+\sqrt{n^4+1}$　　$g(n)=(\ln(n!))^2+n$

D. $f(n)=2^{(n^2)}+(2^n)^2$　　$g(n)=n^{(n^2)}+n^5$

【解析】 选 D。一般情况下，指数级和阶乘级比多项式级的增长速度快得多。

1.2.4 综合应用题选讲

题 1.2.13 用归纳法证明：

(1) $\sum_{i=1}^{n} i=\frac{n(n+1)}{2},n\geqslant 1$

(2) $\sum_{i=1}^{n} i^2=\frac{n(n+1)(2n+1)}{6},n\geqslant 1$

(3) $\sum_{i=0}^{n} x^i=\frac{x^{n+1}-1}{x-1},x\neq 1,n\geqslant 0$

【参考答案】 证明过程如下。

(1) 当 $n=1$ 时，$\sum_{i=1}^{1} i=1=\frac{1\times(1+1)}{2}$ 成立；设当 $n=k$ 时公式成立，即 $\sum_{i=1}^{k} i=\frac{k(k+1)}{2}$，则当 $n=k+1$ 时，$\sum_{i=1}^{k+1} i=\sum_{i=1}^{k} i+(k+1)=\frac{k(k+1)}{2}+k+1=\frac{(k+1)(k+2)}{2}$，结论成立。

(2) 当 $n=1$ 时，$\sum_{i=1}^{1} i^2=1=\frac{1\times(1+1)\times(2\times 1+1)}{6}$ 成立；设当 $n=k$ 时公式成立，即 $\sum_{i=1}^{k} i^2=\frac{k(k+1)(2k+1)}{6}$，则当 $n=k+1$ 时，

$$\sum_{i=1}^{k+1} i^2=\sum_{i=1}^{k} i^2+(k+1)^2=\frac{k(k+1)(2k+1)}{6}+(k+1)^2$$

$$=\frac{(k+1)(2k^2+k+6k+6)}{6}=\frac{(k+1)(2k^2+7k+6)}{6}$$

$$=\frac{(k+1)(k+2)(2k+3)}{6}=\frac{(k+1)((k+1)+1)(2(k+1)+1)}{6}$$

结论成立。

(3) 当 $n=0$ 时，$\sum_{i=0}^{0} x^i=1=\frac{x^{0+1}-1}{x-1}$ 成立；设当 $n=k$ 时 $\sum_{i=0}^{k} x^i=\frac{x^{k+1}-1}{x-1}$ 成立，

则当 $n=k+1$ 时，$\sum_{i=0}^{k+1} x^i=\sum_{i=0}^{k} x^i+x^{k+1}=\frac{x^{k+1}-1}{x-1}+x^{k+1}=\frac{x^{k+1}-1+x^{k+1}(x-1)}{x-1}=\frac{x^{k+2}-1}{x-1}$

结论成立，这实际上就是求等比级数的和。

证毕。

题 1.2.14　设 n 为正整数，分析下列各程序段中加下划线的语句的执行次数。

(1)
```
int i,j,k;
for(i=1;i<=n;i++)
    for(j=1;j<=n;j++){
        c[i][j]=0.0;
        for(k=1;k<=n;k++)
        c[i][j]=c[i][j]+a[i][k]*b[k][j];
    }
```

(2)
```
int i,j,k,x=0,y=0;
for(i=1;i<=n;i++)
    for(j=1;j<=i;j++)
        for(k=1;k<=j;k++)
        x=x+y;
```

【参考答案】　当 n 给定后，各小题中加下划线的语句的执行次数如下。

(1) $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}1=n^3$

(2) $\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}1=\sum_{i=1}^{n}\sum_{j=1}^{i}j=\sum_{i=1}^{n}\left(\frac{i(i+1)}{2}\right)=\frac{1}{2}\sum_{i=1}^{n}i^2+\frac{1}{2}\sum_{i=1}^{n}i$

$$=\frac{1}{2}\frac{n(n+1)(2n+1)}{6}+\frac{1}{2}\frac{n(n+1)}{2}=\frac{n(n+1)(n+2)}{6}$$

题 1.2.15　有实现同一功能的两个算法 A_1 和 A_2，其中 A_1 的渐近时间复杂度是 $T_1(n)=O(2^n)$，A_2 的渐近时间复杂度是 $T_2(n)=O(n^2)$。仅就时间复杂度，具体分析这两个算法哪个好。

【参考答案】　比较算法好坏需比较两个函数 2^n 和 n^2。

- 当 $n=1$ 时，$2^1>1^2$，算法 A_2 好于 A_1；
- 当 $n=2$ 时，$2^2=2^2$，算法 A_1 与 A_2 相当；
- 当 $n=3$ 时，$2^3<3^2$，算法 A_1 好于 A_2；
- 当 $n=4$ 时，$2^4>4^2$，算法 A_2 好于 A_1；
- 当 $n>4$ 时，$2^n>n^2$，算法 A_2 好于 A_1；
- 当 $n\to\infty$ 时，算法 A_2 在时间上显然优于 A_1。

另一种解法是对算法 A_1 和 A_2 的时间复杂度 $T_1(n)$ 和 $T_2(n)$ 取对数，得 $n\log 2$ 和 $2\log n$，用大 O 表示，有 $T_1(n)=O(n)$，$T_2(n)=O(\log n)$。显然，算法 A_2 好于 A_1。

题 1.2.16　按增长率由小至大的顺序排列下列各函数：

2^{100}，$(3/2)^n$，$(2/3)^n$，$(4/3)^n$，n^n，$n^{3/2}$，$n^{2/3}$，$\sqrt{n}$，$n!$，n，$\log_2 n$，$n/\log_2 n$，$(\log_2 n)^2$，$\log_2(\log_2 n)$，$n\log_2 n$，$n^{\log_2 n}$。

【参考答案】　各函数的排列次序如下：

$(2/3)^n$，2^{100}，$\log_2(\log_2 n)$，$\log_2 n$，$(\log_2 n)^2$，$\sqrt{n}$，$n^{2/3}$，$n/\log_2 n$，n，$n\log_2 n$，$n^{3/2}$，$(4/3)^n$，$(3/2)^n$，$n^{\log_2 n}$，$n!$，n^n。

题 1.2.17　已知有实现同一功能的两个算法，其时间复杂度分别为 $O(2^n)$ 和 $O(n^{10})$，假设计算机可连续运算的时间为 10^7 秒（100 多天），又每秒可执行基本操作（根据这些操作来估算算法时间复杂度）10^5 次。试问在此条件下，这两个算法可解问题的规模（n 值的范围）各为多少？哪个算法更适宜？请说明理由。

【参考答案】　根据假设，计算机可连续运行 10^7 秒，每秒可执行基本操作 10^5 次，那么计算机可连续执行基本操作 $10^7\times10^5=10^{12}$ 次。在此能力限制下，算法 1 的时间复杂度为 $O(2^n)$，则可求得问题规模 $n\leqslant\log_2(10^{12})=12\log_2 10$；算法 2 的时间复杂度为 $O(n^{10})$，则其问题规模 $n\leqslant(10^{12})^{0.1}$。显