数据结构

教学做一体化教程

刘鑫 陈恒 王雅轩◎编著

清华大学出版社

数据结构

教学做一体化教程

刘鑫 陈恒 王雅轩◎编著

清华大学出版社
北京

内 容 简 介

本书采用“教学做”一体化的方式撰写，合理地组织学习单元，并将每个单元分解为核心知识、能力目标、任务驱动、实践环节4个模块。全书共8章，第1章介绍数据结构的相关基本概念，以及与算法和算法分析相关的基础知识；第2章和第3章介绍了普通线性表、栈和队列等几种线性结构；第4章至第6章依次介绍了矩阵和广义表、树、图等几种常见的非线性结构；第7章和第8章分别介绍了查找和排序这两种常见的数据处理技术。书中采用C语言作为算法描述语言，所有程序均在VC++ 6.0环境下验证并调试通过，且代码中包含详细的注释，便于学生在编程实践时进行参考。全书既重视理论又重视实践，理论讲解生动有趣、图文并茂，实践环节例题丰富、讲解细致、针对性强，具有很强的实用性和可操作性。

本书适合作为全日制高等院校“数据结构”课程的专业基础课教材或教学参考书，也适合作为相关专业高职高专学生的参考教材，还可以作为计算机等级考试的参考书，供广大从事计算机应用工作的管理人员和技术人员学习与参考。

图书在版编目(CIP)数据

数据结构教学做一体化教程/刘鑫，陈恒，王雅轩编著. --北京：清华大学出版社，2014
ISBN 978-7-302-36594-5

Ⅰ. ①数… Ⅱ. ①刘… ②陈… ③王… Ⅲ. ①数据结构－高等学校－教材 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2014)第168号

责任编辑：田在儒
封面设计：王跃宇
责任校对：袁　芳
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795764

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.75　　**字　　数**：496千字
版　　次：2014年7月第1版　　**印　　次**：2014年7月第1次印刷
印　　数：1～2500
定　　价：39.80元

产品编号：045582-01

前言
FOREWORD

本教材按照“教学做”一体化模式精编了数据结构的核心内容，以核心知识、能力目标、任务驱动和实践环节为单元组织本教材的体系结构。核心知识体现最重要和实用的知识，是教师需要重点讲解的内容；能力目标提出学习核心知识后应具备的能力；任务驱动给出了教师和学生共同来完成的任务；实践环节给出了需要学生独立完成的实践活动。

全书共8章，第1章介绍数据结构的相关基本概念，以及与算法和算法分析相关的基础知识；第2章和第3章介绍了普通线性表、栈和队列等几种线性结构；第4章至第6章依次介绍了矩阵和广义表、树、图等几种常见的非线性结构；第7章和第8章分别介绍了查找和排序这两种常见的数据处理技术。书中采用C语言作为算法描述语言，所有程序均在VC++ 6.0环境下验证并调试通过，且代码中包含详细的注释。

全书既重视理论又重视实践，一方面通过形象有趣的例子和图形、图像细致地讲解抽象理论，通俗易懂、深入浅出；另一方面包含大量的习题和例题，并配有详细的解释和完整的参考答案，针对学生在学习中遇到的重点和难点有的放矢，反复训练。通过细致严谨的理论学习和大量实际的动手练习，使学生深刻理解算法的实质和基本思想，培养学生严谨的科学态度和学以致用的良好学风，提高软件设计和编程能力，为今后的专业课程学习打下坚实良好的基础，从而培养专业理论知识扎实、创新实践能力突出的高素质、复合型的应用人才，以满足社会经济、国家科技发展的需要。

本书适合作为全日制高等院校“数据结构”课程的专业基础课教材或教学参考书，也适合作为相关专业高职高专学生的参考教材，还可以作为计算机等级考试的参考书，供广大从事计算机应用工作的管理人员和技术人员学习与参考。

由于作者水平有限，书中的缺点和错误在所难免，殷切希望广大读者批评、指正。

编　者

2014年3月

目 录
CONTENTS

第1章 概 述

主要内容

- 数据结构的定义
- 数据的逻辑结构
- 数据的存储结构
- 算法和算法分析

许多人认为程序设计是一件非常高深复杂的事情，其实它和写作文的道理差不多。尽管汉语的语法和基本句型是固定的，可是当面对同一题目时，每个人写出的作文却风格迥异，有人写得生动优美，有人写得晦涩乏味。同样道理，尽管一种编程语言的语法和设计规则是固定不变的，可是当面对同一题目时，每个人写出的程序却差别很大，有的程序逻辑清晰、高效健壮，有的程序逻辑混乱、低效低能。

提高写作水平的方法有很多，除了多思考多练习之外，还有一条捷径——多阅读前人名家的优秀作品。同样的方法对于提高程序设计水平也大有好处，除了练好基本功、多多动手实践、多多独立思考之外，还应该多阅读前人写过的优秀程序，从中发现并总结更多更有效的解决问题的方法。其实，数据结构正是这样一门学问，它是针对待处理对象的特征及其相互关系的分析，是对计算机存储、组织数据方式的研究，更是对前人思索和解决问题方法的总结。

在学习本课程之前，读者应当学习过 C 语言，熟悉计算机的一些基础知识。当读者学习过数据结构之后，一方面可以通过 C 语言编写更加强大的程序，解决更多的实际问题；另一方面还可以学习更多种类的编程语言（如 Java、C#），并在学习的过程中融入数据结构的思想精髓。

1.1 数据结构的基本定义

1.1.1 核心知识

1. 生活中与数据结构相关的实例

数据（Data）是信息的载体，是对客观事物的符号表示。通俗地说，凡是能被计算机识别、存取和加工处理的符号、字符、图形、图像、声音和视频等一切信息都可以称为数据。其中，整数、实数、浮点数和复数等属于数值数据，主要用于科学计算、金融、财会等；而文字、

符号、图形、动画、语音、视频等属于非数值数据，主要用于如文字处理、图形图像处理、信息检索和日常办公管理等。

随着计算机应用领域的不断扩大，计算机开始处理越来越多的非数值问题，如学生个人信息的查询、某家族家谱信息的存储、城市之间交通路径的规划。这类问题的数学模型无法使用数学方程式加以描述，因此需要分析和研究问题中包含的所有数据之间的关系，进而合理地组织数据，最终建立有效的数据结构对其进行描述。

为了更加准确地理解数据结构的定义，请仔细观察并分析下面几个例子。

【例 1-1】 学生信息登记表。

为了加强信息化管理，某学校将所有学生信息通过表格方式存入计算机，表 1-1 是学生信息统计表。其中，每一行是一名学生的个人信息记录，每条记录由学号、姓名、性别、年龄、籍贯组成。所有记录的排列顺序如何，以及它们在计算机中采用何种存储方式，这些都是数据结构需要研究的问题。

表 1-1 学生信息登记表

学号	姓名	性别	年龄	籍 贯
0101	赵大	男	19	江苏省
0102	钱二	女	18	重庆市
0103	孙三	男	19	广东省
0104	李四	男	20	山东省
0201	周五	女	17	辽宁省
0202	吴六	男	18	北京市
0203	郑七	女	20	江苏省

【例 1-2】 家庭成员关系的表示和存储。

为了记录某家族所有成员的个人信息及其相互关系，现将所有数据通过树状图的方式存入计算机，图 1-1 是部分家族成员信息的记录。我们可以对这些记录进行查询、插入、删除和修改等操作，还可以研究如何将这些信息存储到计算机，能“既节省空间，又便于对信息进行检索”。这些问题都属于数据结构的研究范畴。

【例 1-3】 国内部分城市之间的交通路径图。

图 1-2 是国内部分城市之间的交通路径图，针对该图可以研究几个城市之间的“路程最短路径”、“路费最经济路径”、“行驶时间最短路径”等，这些问题都属于数据结构的研究领域。

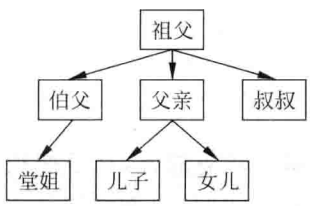

图 1-1 家庭成员关系图

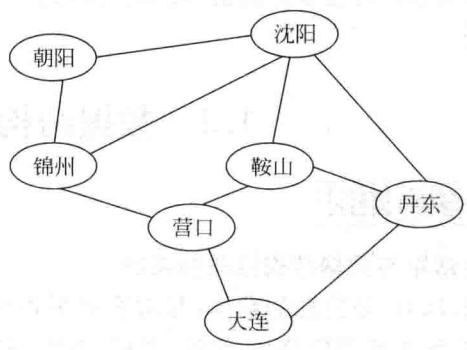

图 1-2 城市之间的交通路径

2. 基本概念和术语

数据元素(Data Element)是数据的基本单位,在计算机程序中通常作为一个整体进行考虑和处理。数据元素也称为结点(Node),如例 1-1 中每个学生的信息是一个数据元素,例 1-2 的家谱中每个家庭成员称为一个数据元素,例 1-3 的交通图中每个城市称为一个数据元素,且这三个例子中数据元素的数量均为 7 个。

数据项(Data Item)是数据不可分割的、具有独立意义的最小数据单位,是对数据元素属性的描述。数据项也称为**域**或**字段**(Field)。例如,例 1-1 中每个学生基本信息包含 5 个数据项:学号、姓名、性别、年龄和籍贯。

数据、数据元素、数据项反映了数据组织的 3 个层次,即数据可以由若干个数据元素组成,数据元素又由若干数据项组成。

3. 逻辑结构

通过前面的三个例子,我们发现一个**数据结构**(Data Structure)是由数据元素依据某种逻辑联系组织起来的。比如,例 1-1 中的 7 个结点(数据元素)除了第一个(赵大)和最后一个(郑七)之外,其余的结点都有且仅有一个直接前驱结点和一个直接后继结点(如钱二的直接前驱是赵大,直接后继是孙三),因此结点之间是“一对一”的逻辑关系;例 1-2 中祖父结点与伯父、父亲、叔叔三个结点相对应,父亲结点与儿子、女儿相对应,因此结点之间是“一对多”的逻辑关系;例 1-3 中每个城市结点都和若干个相邻城市结点相对应,因此结点之间是“多对多”的逻辑关系。

对数据元素间逻辑关系的描述称为数据的**逻辑结构**。根据数据元素之间的关系的不同特性,逻辑结构可分为如图 1-3 所示的四种。

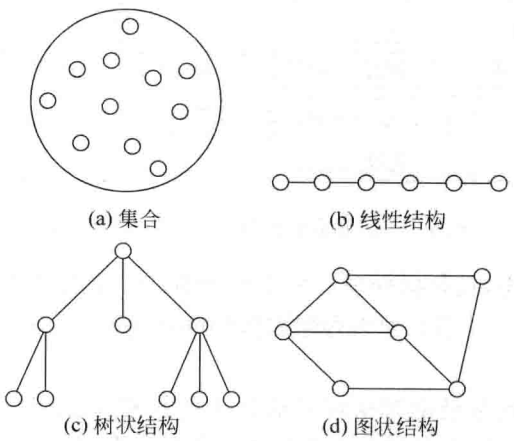

图 1-3 四种基本结构示意图

(1) 集合

数据元素之间除“同属于一个集合”的关系之外,别无其他关系。比如所有的中国人构成了中华民族这个集合。此外,集合中的元素还应该具备以下三个特征。

① 确定性:具有明确的判断标准。比如,“高富帅”就无法构成一个集合,因为这三个字都没有明确的标准。

② 互异性：集合元素互不相同。比如，{1,1,2}就不是一个集合。

③ 无序性：集合元素之间没有顺序。比如，{1,2,3}和{3,2,1}就是同一个集合。

(2) 线性结构

数据元素之间存在“一对一”的关系，例 1-1 是一个典型的线性结构。

在线性结构中，集合中的元素除了开始结点和终端结点之外，其余结点都有且仅有一个直接前驱结点和一个直接后继结点。

(3) 树状结构

数据元素之间存在“一对多”的关系，例 1-2 是一个典型的树状结构。

树状结构除了起始结点(即根结点)以外，各个结点都有唯一的直接前驱结点；所有结点都有 0 个到多个直接后继结点。

(4) 图状结构

数据元素之间存在“多对多”的关系，例 1-3 是一个典型的图状结构。

图状结构的每个结点都可以有多个直接前驱结点和多个直接后继结点。

4. 存储结构

数据的逻辑结构在计算机存储空间中的存放形式叫做数据的**存储结构**(或物理结构)。数据结构这门学科除了研究数据的逻辑结构之外，还研究数据的存储结构。

下面通过一个例子说明两种结构之间的关系：一个班级中每个学生都有学号，按照学号由小到大的顺序打印出的学生名单反映了该班学生的逻辑结构，如表 1-1 所示；标识每个学生实际位置的座位分布图则反映了该班学生的存储结构，如图 1-4 所示。可见，逻辑结构和存储结构之间没有必然的联系，相同的逻辑结构完全可以对应不同的存储结构。

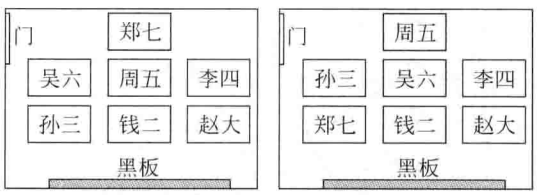

图 1-4　某班学生两幅不同的座位分布图

采用不同的存储结构，其数据处理效率是不同的。常用的存储结构有：顺序存储(参见 2.3 节)、链接存储(参见 2.5 节)、索引存储和散列存储(参见 7.5 节)等。

(1) 顺序存储

顺序存储结构的特点是借助元素在存储器中的相对位置表示数据元素之间的逻辑关系。它是把逻辑上相邻的结点存储在物理位置相邻的存储单元，结点间的逻辑关系由存储单元的邻接关系体现。

例如，图 1-5(a)表示复数 Z=1.2+3.4i 的顺序存储结构，其中地址 1000 存放实部 1.2，地址 1004 存放虚部 3.4(假定每个浮点数占 4 个存储单元)。即存放同一个复数的实部和虚部在逻辑上相邻，在存储器中的地址也相邻。

顺序存储结构是一种最基本的存储方法，通常借助于程序设计语言(如 C、Java)中的数组实现。

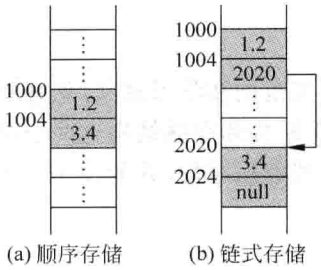

图 1-5　顺序存储结构与链式存储结构

(2) 链式存储

链式存储结构的特点是借助指示元素存储地址的指针(Pointer)表示数据元素之间的逻辑关系。它不要求逻辑上相邻的结点在物理位置上也相邻,结点间的逻辑关系由附加的指针域表示。

例如,图 1-5(b)为表示复数 Z=1.2+3.4i 的链式存储结构,其中地址 1000 存放实部 1.2,地址 2020 存放虚部 3.4,地址不连续。实部和虚部之间的关系用存放在地址 1004 中的值为"2020"的指针表示。

链式存储结构通常借助于程序设计语言(如 C、C++)中的指针类型实现。用指针实现链式存储时,数据元素不一定存放在地址连续的存储单元,存储处理的灵活性较大。

(3) 索引存储

索引存储除了建立存储结点信息外,还建立附加的索引表。索引表中的每一项都由关键字(能唯一标识一个结点的数据项)和地址组成。索引表反映了所有结点信息按某一个关键字递增或递减排列的逻辑次序。采取索引存储的主要作用是为了提高数据的检索速度,类似于一本字典中的索引目录。

(4) 散列(Hash)存储

散列技术是一种将数据元素的存储位置与关键字之间建立确定对应关系的查找技术。散列技术除了可以用于查找外,还可以用于存储。

散列存储是通过构造散列函数确定数据存储地址。散列存储的内存存放形式称为散列表,也称为**哈希(Hash)表**。

例如,要存储某地区新中国成立后每年出生的人口数,可以用"出生年份－1948＝存储地址"构造出散列函数,存放地址如表 1-2 所示。这样,若要查 2008 年出生的人数,只要查表中存储地址 60(2008－1948)对应的存放数据(15200)即可。

表 1-2　某地区年出生人口散列存储表

存储地址	出生年份	人数/人	存储地址	出生年份	人数/人
01	1949	10000	31	1979	17000
02	1950	10100	⋮	⋮	⋮
03	1951	14600	60	2008	15200
⋮	⋮	⋮	61	2009	14500

1.1.2 能力目标

了解数据结构这门学科的研究范围和学习意义，从感性上理解数据结构各名词、术语的含义和相关的基本概念，理解逻辑结构和存储结构这两个词的含义与区别，能够用表格或画图的方式从生活中简单抽象出一些数据结构，并能使用二元组的方式（在任务驱动中介绍）描述其数据元素之间的逻辑关系。

1.1.3 任务驱动

一个数据的逻辑结构可以用二元组描述：G=(D,R)，其中，G 表示数据逻辑结构的名称，第一元 D 表示数据元素的有限集合，第二元 R 表示 D 上所有元素之间关系的有限集合。

1. 任务 1

(1) 任务的主要内容

某二元组的逻辑关系如下：

数据结构五行 =(D, R)

其中：D = {金，木，水，火，土}，R = {<金，木>，<木，水>，<水，火>，<火，土>}。

请画出对应逻辑图形，并指出它属于哪种数据结构。

(2) 任务参考答案

对应逻辑图形如下：

分析：五行逻辑结构的二元组共由两部分组成：数据元素集合 D 和数据元素之间关系的集合 R。D = {金，木，水，火，土}说明五行共包含五个数据元素：金，木，水，火，土；R = {<金，木>，<木，水>，<水，火>，<火，土>}则说明："金"是"木"的前驱结点，"木"是"金"的后继结点，"木"是"水"的前驱结点，"水"是"木"的后继结点，……通过观察可以发现：五个数据元素中除了开始结点"金"和终端结点"土"之外，其余结点都有且仅有一个直接前驱结点和一个直接后继结点，数据元素之间存在"一对一"的关系，因此它符合线性结构的逻辑特点。

2. 任务 2

(1) 任务的主要内容

请使用二元组的方式表示例 1-2（家谱）中的树状结构。

(2) 任务参考答案

数据结构家谱=(D,R)

其中：D = {祖父，伯父，父亲，叔叔，堂姐，儿子，女儿}，R = {<祖父，伯父>，<祖父，父亲>，<祖父，叔叔>，<伯父，堂姐>，<父亲，儿子>，<父亲，女儿>}。

分析：R 是关系集合，尖括号表示数据元素的关系是有向的，如<祖父，伯父>表示从祖父指向伯父。结点"祖父"无直接前驱，称为根结点，其余结点都只有一个直接前驱；结点"叔叔"、"堂姐"、"儿子"和"女儿"无直接后继，称为叶子结点，其余结点都有一个或多个直接后继。结构中的数据元素之间存在"一对多"的关系，这就是树状结构的逻辑特点。

3. 任务3

(1) 任务的主要内容

请使用二元组的方式表示例1-3(城市交通图)中的图状结构。

(2) 任务参考答案

数据结构城市交通图=(D,R)

其中：D = {沈阳,锦州,鞍山,丹东,大连,营口,朝阳},R = {(沈阳,朝阳),(沈阳,锦州),(沈阳,鞍山),(沈阳,丹东),(锦州,朝阳),(锦州,营口),(丹东,鞍山),(营口,鞍山),(丹东,大连),(营口,大连)}。

分析：R是关系集合,圆括号表示数据元素之间的关系是双向的(或没有方向的),例如,(丹东,大连)表示从丹东到大连之间的边是双向的,因此(丹东,大连)和(大连,丹东)这对关系完全等价。此图中每个结点都可以有多个直接前驱或多个直接后继,即结构中的数据元素之间存在"多对多"的关系,这就是图状结构的逻辑特点。

1.1.4 实践环节

(1) 请列举出几个生活中与数据结构相关的实例,并使用图表的方式对其进行描述。同时,指出这些实例中数据元素的个数,以及每个数据元素中包含的数据项的个数。

(2) 请画出下面二元组表示对应的逻辑图形,并指出它属于哪种数据结构。

数据结构 X =(D,R)

其中：D = {A, B, C, D, E, F, G},R = {<A,B>,<A,C>,<B,D>,<B,E>,<C,F>,<F,G>}。

1.2 算法和算法分析

1.2.1 核心知识

1. 算法概述

算法(Algorithm)是对特定问题求解步骤的一种描述,是由若干操作构成的指令的有限序列。生活中算法的例子处处可见,在厨房里,"鱼香肉丝"的菜谱就是一个算法,厨师可以按照该算法描述的炒菜步骤做出可口的菜肴。在游戏中,"生化危机"的攻略是一个算法,玩家可以按照该算法描述的通关步骤完成整个游戏。

在计算机领域,算法的实质是针对所处理的问题,在数据结构的基础上施加的一种运算。因此,数据结构和算法就像一对亲密的伴侣,谁也离不开谁：缺少了算法,数据结构就丧失了灵魂,变得毫无意义；而缺少了数据结构,算法又如同一座空中的楼阁,无法实现。通常,在算法设计之前先要明确相应的逻辑结构和物理结构,而在研究某一种数据结构时也必然会涉及相应的算法。任何一个算法的设计都取决于选定数据的逻辑结构,而算法的实现则依赖于数据所采用的存储结构。

2. 算法的特性

算法具有4个基本特性：输入输出、有穷性、可行性和确定性。

(1) **输入输出**：为了能够解决实际问题,算法应该具备一定的交互能力。比如当计算

100 以内的所有素数时，用户无须输入，但算法需要输出 0～100 的所有素数作为运算结果。再比如：当计算 m 和 n 这两个正整数的最大公约数时，用户需要输入 m 和 n 的具体数值，而算法需要输出 m 和 n 的最大公约数作为运算结果。因此，一个正常的算法应该至少具有零个或多个输入，一个或多个输出。

（2）**有穷性**：如果一个算法运行了很长时间，让用户等到“花儿都谢了”，仍然无法输出结果，该算法显然没有意义。因此，一个合理的算法应该在执行有限的步骤之后自动结束，每一个步骤都在可接受的时间内完成，而且不应该出现无限循环。

（3）**可行性**：如果一个算法极其复杂，无法转换为程序在计算机上运行，那么该算法显然用处不大。因此，一个可用的算法每一步都必须是可行的，也就是说，每一步都能够通过执行有限次数完成。

（4）**确定性**：如果一个算法面对同样一个问题，两次运行却给出了不同的答案，那么该算法显然是无法被信任的。因此，一个可靠算法的每个步骤应该被精确定义而无歧义，不会出现二义性。在相同条件下，只有一条执行路径，相同的输入只能有唯一的输出结果。比如假设“鱼香肉丝”的菜谱上要求“加盐少许”，这就是不确定的步骤，不同的人对“少许”理解不同，最终做出的菜口味也不一样。如果菜谱上要求“加盐 10 克”，就不会造成歧义了。

3. 算法的评价标准

相同的一个问题往往可以采用多种算法解决，尽管算法不唯一，但这些算法中还是有好有坏的，那么什么样的算法是好算法呢？

通常一个优秀的算法设计应该满足以下条件。

（1）**正确性**。好的算法，运行结果必须是正确的，否则其他方面再优秀也毫无意义。因此，一个好算法的前提条件是，其执行结果应当满足具体问题的功能和要求。

（2）**可读性**。如果一个算法写得像火星文，除了作者和极少数天才之外无人能懂，那么该算法就很难普及，更难以被编写成程序在计算机上运行。因此，一个好算法应当层次分明、思路清晰、易于理解。

（3）**健壮性**。同样是吃了不卫生的食物，健壮的人也许只是闹闹肚子，可是羸弱的人就可能大病一场，甚至一命呜呼。程序也是如此，当发生误操作或输入非法数据时，“健壮”的算法能够作适当的反应和处理，而较差的算法则会输出莫名其妙的结果，甚至崩溃。

（4）**时间高效和存储量低**。在生活中，人们总希望用最短的时间，花最少的钱，办最大的事；算法也是相同的思想，解决同一个问题时，计算机花费的运行时间和占用的存储空间越少越好。

当然，一个算法很难做到十全十美，上述因素有时会相互抵触。例如，强调算法的可读性就可能降低算法的高效性，而追求算法的高效性可能要以牺牲一定存储空间为代价。因此实际操作中应以算法的正确性为前提，根据实际情况权衡，有所侧重。

4. 算法的时间复杂度

我们喜欢时间效率高的算法，但究竟应该用什么样的标准才能准确地量化一个算法时间效率的高低呢？

（1）事后统计法

最简单直接的想法是：利用计算机计时器对不同算法编制程序的运行时间进行比较，

从而确定算法效率的高低。但是该方法存在以下几大缺点。

① 相同的程序在不同的计算机软硬件环境下运行，其运行时间很有可能是不同的。我们很难确定单核 CPU 计算机测试的结果更合理，还是四核 CPU 计算机测试的结果更合理。

② 如果同一问题存在 A 和 B 两种算法：当输入第一批数据时，A 算法所用的时间更少；但是当输入第二批数据时，B 算法所用的时间更少。此时无法确定哪一组数据测试的结果更合理。

③ 须依据算法提前编制好程序，如果编写并运行完毕之后才发现算法很糟糕，就会白辛苦一场，浪费大量的时间和精力。

(2) 事前分析法

为了节约时间和精力，需要在计算机程序编制前，依据统计方法对算法进行估算。通过分析可以得出结论：抛开与计算机硬件、软件有关的因素，一个特定算法的运行时间只依赖于问题输入数据的规模(通常用整数量 n 表示)，或说是问题规模的函数。

通常我们将对运行时间有消耗的基本操作称为原操作，由于一个算法的运行时间与原操作重复执行次数成正比，因此通常以原操作总的执行次数作为算法的时间量度。

【例 1-4】 请分析下面程序的时间复杂度。

```
int n;                                          /* 频度为 1 */
printf("请输入一个整数:\n");                    /* 频度为 1 */
scanf("%d",&n);                                 /* 频度为 1 */
printf("%d 的平方是: %d\n",n,n*n);              /* 频度为 1 */
```

分析：该程序共包含 4 条语句，每条语句的执行次数均是 1，总的执行次数是 4，因此，该问题的执行时间是与问题规模 n 无关的常数，算法的时间复杂度为常数阶，表示方法为 $T(n)=O(1)$。

【例 1-5】 请分析下面程序的时间复杂度。

```
int i, n;                                       /* 频度为 1 */
printf("请输入一个整数:\n");                    /* 频度为 1 */
scanf("%d",&n);                                 /* 频度为 1 */
for(i=1; i<=n; i++)
{
    printf("当前是第 %d 次循环\n",i);           /* 频度为 n */
}
```

分析：该程序共包含 4 条基本语句，前三条语句的执行次数是 1，最后一条语句由于不断循环执行了 n 次，因此，该问题原操作总的执行次数为 n+3，其执行时间与问题规模 n 成正比，算法的时间复杂度标记为 $T(n)=O(n)$。

【例 1-6】 下面程序用于输出一个 n×n 大小的图形，请分析其时间复杂度。

```
int i,j, n;                                     /* 频度为 1 */
printf("请输入一个整数:\n");                    /* 频度为 1 */
scanf("%d",&n);                                 /* 频度为 1 */
for(i=1; i<=n; i++)
{
```

```
    for(j = 1; j <= n; j++)
    {
        printf(" * ");                          /* 频度为 n² */
    }
    printf("\n");                               /* 频度为 n */
}
```

分析：该程序共包含 5 条基本语句，原操作总的执行次数为 n^2+n+3。由于随着 n 值的不断增大，n^2 比 $n+3$ 对执行时间的影响更大，因此算法的时间复杂度标记为 $T(n)=O(n^2)$。

【例 1-7】 请分析下面程序的时间复杂度。

```
int i,n;                                        /* 频度为 1 */
printf("请输入一个整数:\n");                    /* 频度为 1 */
scanf(" %d",&n);                                /* 频度为 1 */
for(i = 1; i < n; i = i * 2)
{
    printf("翻倍后等于 %d\n",i);                /* 频度为 log₂n */
}
```

分析：该程序共包含 4 条基本语句，其中前三条语句各执行了 1 次，第四条语句执行了 $\log_2 n$ 次（假设该条语句执行了 x 次，则 for 循环必定执行了 x 次，每次 i 的值都会翻倍，直至超过 n 的值，故 $2^x=n$，$x=\log_2 n$）。所以，原操作总的执行次数为 $\log_2 n+3$，此算法的时间复杂度标记为 $T(n)=O(\log_2 n)$。

常见的算法时间复杂度有 $O(1)$、$O(n)$、$O(n^2)$、$O(\lg n)$和 $O(2^n)$等。其增长率如图 1-6 所示。由图 1-6 可见，当 n 逐渐增大时，各种时间复杂度的关系如下：

$$O(1) < O(\lg n) < O(n) < O(n\lg n) < O(n^2) < O(2^n)$$

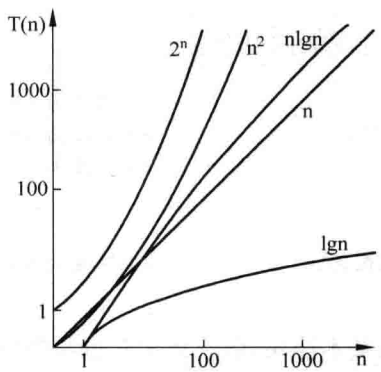

图 1-6　常见函数的增长率

5. 算法的空间复杂度

与算法的时间复杂度类似，算法的空间复杂度用于度量算法所需存储空间的大小，它也可以使用一个与问题规模 n 相关的函数 S(n)来表示。

一个程序上机执行时，除了需要存储空间存放本身所用的指令、变量、常量和输入数据以外，还需要一些实现算法所必需的辅助空间。若输入数据所占空间只取决于问题本身，和