普通高等教育“十一五”国家级规划教材

高等院校“十三五”规划教材 · 软件技术系列

数据结构

（第5版）（C语言版）

邓文华 主编

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

数据结构

（第5版）（C语言版）

策划编辑：薛华强
责任编辑：徐建军
封面设计：张　昱

ISBN 978-7-121-36934-6

定价：49.80 元

普通高等教育“十一五”国家级规划教材

高等院校“十三五”规划教材·软件技术系列

数据结构（第5版）

（C语言版）

邓文华 主编

尹 魁 副主编

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书对常用的数据结构做了系统的介绍，概念论述清晰，注重实际应用。全书分为 9 章，前 8 章依次介绍了数据结构的基本概念、线性表、栈和队列、串和数组、树与二叉树、图、查找和排序；第 9 章为实验，共设计了 10 个实验项目，基本涵盖了数据结构的主要内容。全书以 C 语言为算法描述语言，第 2～8 章均附有典型例题与小结，便于总结与提高。

本书叙述简洁、深入浅出，注重实践和应用，可一书两用（理论与实验），主要面向应用型本科、高职高专计算机类专业的学生，也可作为大学非计算机专业学生的选修课教材和计算机应用技术人员的自学参考书。

图书在版编目（CIP）数据

数据结构：C 语言版 / 邓文华主编. —5 版. —北京：电子工业出版社，2019.8

ISBN 978-7-121-36934-6

Ⅰ. ①数… Ⅱ. ①邓… Ⅲ. ①数据结构－高等学校－教材 ②C 语言－程序设计－高等学校－教材
Ⅳ.①TP311.12 ②TP312.8

中国版本图书馆 CIP 数据核字（2019）第 128631 号

策划编辑：薛华强
责任编辑：徐建军　　特约编辑：田学清
印　　刷：三河市君旺印务有限公司
装　　订：三河市君旺印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：17　　字数：503 千字
版　　次：2004 年 9 月第 1 版
　　2019 年 8 月第 5 版
印　　次：2019 年 8 月第 1 次印刷
定　　价：49.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254569，xuehq@phei.com.cn，QQ1140210769。

前　言

“数据结构”是计算机程序设计的重要理论基础，也是计算机及其应用专业的一门重要基础课程和核心课程。它不仅是计算机软件专业课程的先导课程，而且逐渐被其他工科类专业所重视。

本教材自2004年出版第1版、2007年出版第2版、2011年出版第3版、2015年出版第4版以来，受到了广大师生、读者的热烈欢迎，至今已累计发行超过6万册。在此对广大师生、读者表示衷心的感谢。为了更好地适应新形势的发展与需要，特别是针对应用技术型院校的需要，在保留第4版特点的基础上对本书进行了修订和完善。

本版教材以第4版教材为基础，在广泛听取广大读者的意见与建议的情况下修改而成。本版教材主要对第4版中的一些描述、笔误、解释等进行了订正，去掉了一些过时的内容，简化了使用不多的内容，删除了原第9章的预备实验。

本教材有以下特点：

（1）基础理论知识的阐述由浅入深、通俗易懂。内容组织和编排以应用为主线，略去了一些理论推导和数学证明的过程，淡化算法的设计分析和复杂的时空分析。

（2）各章（除第1章、第9章外）都配有相应的典型例题，列举并分析了众多实用的例子。大多数算法都直接给出了其相应的C语言设计程序，以便学生上机练习、实践。

（3）一书两用，本教材配有实验一章，该章设计了10个实验项目，基本涵盖了数据结构的主要内容。所以本教材既可作为数据结构的理论教材，也可作为数据结构的实验教材。

（4）配有电子教案（PPT）、习题解答、实验源程序及实验参考答案，从而极大地方便了教师备课。

本教材讲课为60～90学时，上机为20～36学时。教师可根据学时数、专业和学生的实际情况选讲。

本书目录中带*号的部分为高职高专院校选讲内容。

本书由邓文华担任主编，尹魁担任副主编。第1、3、5、7、9章由邓文华编写和修改；第2、4、6、8章由尹魁编写和修改。

由于编者水平有限，书中难免存在不妥之处，敬请读者批评指正。

编　者

目 录

CONTENTS

第1章 绪 论

数据作为计算机加工处理的对象，如何在计算机中表示和存储数据是计算机科学研究的主要内容之一，更是计算机技术需要解决的关键问题之一。数据是计算机化的信息，是计算机处理的主要对象。科学计算、数据处理、过程控制、文件存储、数据库技术等，都涉及对数据进行加工处理的过程。因此，要设计出一个结构好、效率高的程序，必须研究数据的特性、数据间的相互关系及其对应的存储表示方法，并利用这些特性和关系设计相应的算法和程序。

1.1 从问题到程序

“数据结构”是计算机科学与技术专业的专业基础课，也是核心课程之一，其主要研究内容是数据之间的逻辑关系和物理实现，即探索有利的数据组织形式及存取方式。计算机系统软件和应用软件的设计、开发要用到各种类型的数据结构。因此，要想更好地运用计算机来解决实际问题，仅仅依赖几种计算机程序设计语言是不够的，还必须学习和掌握数据结构的有关知识。

在计算机发展的初期，人们使用计算机的目的主要是处理数值计算问题。使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先要从该具体问题中抽象出一个适当的数学模型，然后设计或选择一个求解此数学模型的算法，再编写程序并进行调试、测试，最后运行程序并得到答案（如图 1.1 所示）。例如，求解梁架结构中应力数学模型的线性方程组，该方程组可以使用迭代算法来求解。

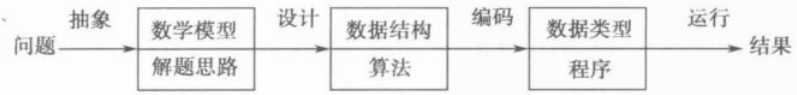


图 1.1 计算机解决问题的一般过程

由于当时所涉及的运算对象是简单的整型数据、实型数据或布尔型数据，所以程序设计者的主要精力集中于程序设计的技巧上，而无须重视数据结构。随着计算机应用领域的扩大和软/硬件的发展，非数值计算问题显得越来越重要。据统计，计算机在处理非数值计算性问题上占用了 90% 以上的机器时间。这类问题涉及的处理对象不再是简单的数据类型，其形式更加多样，结构更为复杂，数据元素之间的相互关系一般无法直接用数学方程式加以描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是设计出合适的数据结构，以便有效地解决问题。

【例 1.1】 图书信息检索系统。在现代图书馆中，人们往往借助计算机图书检索系统来查找需要的图书信息，或者直接通过图书馆信息系统进行图书借阅。为此，需要将图书信息分类编排，建立合适的数据结构对图书信息进行存储和管理，按照某种算法编写相关程序，实现计算机自动

检索。由此，一个简单的图书信息检索系统包括一张按图书分类号和登录号顺序排列的图书信息表，以及分别按作者、出版社等顺序排列的各类索引表，如图 1.2 所示。由这三张表构成的文件便是图书信息检索的数学模型，计算机的主要运算便是按照用户的要求（如给定作者）通过不同的索引表对图书信息进行检索、查询。

序　号	图书分类号	登 录 号	书　名	作　者	出 版 社
1	B259.1	3240	梁启超家书	张品兴	中国文联出版社
2	C52	5231	探寻语碎	李泽厚	上海文艺出版社
3	D035.5	6712	市政学	张永桃	高等教育出版社
4	G206	1422	传播学	邵培仁	高等教育出版社
5	H319.4	1008	英语阅读策略	李宗宏	兰州大学出版社
6	K825.4.00	5819	围棋人生	聂卫平	中国文联出版社
7	P1.00	8810	通向太空之路	邹惠成	科学出版社
8	TN915	7911	通信与网络技术概论	刘云	中国铁道出版社
9	TP312	7623	计算机软件技术基础	王宇川	科学出版社
10	TP393.07	8001	网络管理与应用	张琳	人民邮电出版社
11	Q3.00	2501	普通遗传学	杨业华	高等教育出版社

（a）图书信息表

姓　名	序　号
邵培仁	4
李泽厚	2
李宗宏	5
刘云	8
聂卫平	6
王宇川	9
杨业华	11
张琳	10
张品兴	1
张永桃	3
邹惠成	7

（b）作者姓名索引表

出 版 社	序　号
高等教育出版社	3,4,11
科学出版社	7,9
兰州大学出版社	5
人民邮电出版社	10
上海文艺出版社	2
中国铁道出版社	8
中国文联出版社	1,6

（c）出版社索引表

图 1.2　图书信息检索系统中的数据结构

诸如此类的还有电话自动查号系统、学生信息查询系统、仓库库存管理系统等。在这类数学模型中，计算机处理的对象之间通常存在着一种简单的线性关系，这类数学模型属于线性数据结构。

【例 1.2】　人机对弈问题。人机对弈是一个古老的人工智能问题，其解题思想是将对弈的策略事先存入计算机，策略包括对弈过程中所有可能的情况及响应的对策。在决定对策时，根据当前状态，考虑局势发展的趋势做出最有利的选择。因此，计算机运算的对象（数据元素）是对弈过程中的每一步棋盘状态（格局），数据元素之间的关系由比赛规则决定。通常，这个关系不是线

性的，因为从一个格局可以派生出多个格局，所以通常用树形结构来表示，如图 1.3 所示为井字棋对弈树。

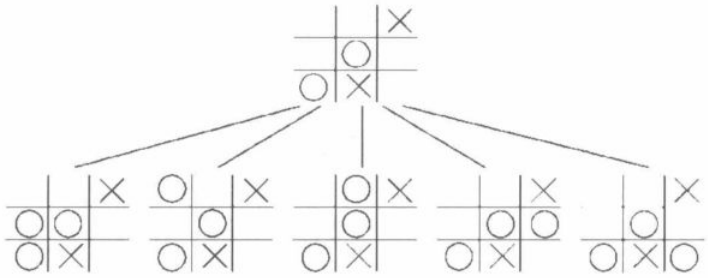
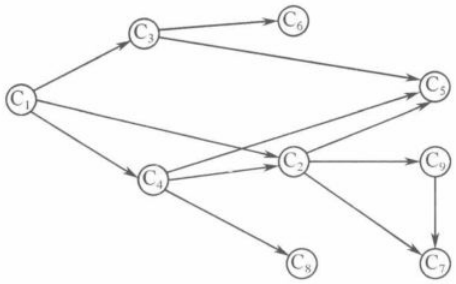


图 1.3 井字棋对弈树

【例 1.3】 教学计划编排问题。一个教学计划包含许多课程，在教学计划包含的许多课程之间，有些课程必须按规定的先后次序进行学习，有些则没有次序要求。课程之间先修和后修的次序关系可用一个称作图的数据结构来表示，如图 1.4 所示。有向图中的每个顶点表示一门课程，如果从顶点 v_i 到 v_j 之间存在有向边$<v_i,v_j>$，则表示课程 i 必须先于课程 j 进行学习。

课程编号	课程名称	先修课程
C_1	计算机导论	无
C_2	数据结构	C_1,C_4
C_3	汇编语言	C_1
C_4	C 程序设计语言	C_1
C_5	计算机图形学	C_2,C_3,C_4
C_6	接口技术	C_3
C_7	数据库原理	C_2,C_9
C_8	编译原理	C_4
C_9	运算系统	C_2

（a）计算机专业的课程设置



（b）表示课程之间优先关系的有向图

图 1.4 教学计划编排问题的数据结构

由以上几个例子可知，描述非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。因此，数据结构课程是研究非数值计算的程序设计问题中计算机处理对象及它们之间关系和运算的学科。

学习数据结构的目的是了解和掌握计算机处理对象的特性，将实际问题中所涉及的处理对象在计算机中表示出来并对它们进行处理。同时，通过算法训练来提高学生的思维能力，通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

1.2 有关概念和术语

在系统地学习数据结构知识之前，先对一些基本概念和术语赋予确切的定义。

1．数据

数据（Data）是信息的载体，它能够被计算机识别、存储和处理。数据是计算机程序加工的原料，应用程序能处理各种各样的数据，包括数值数据和非数值数据。数值数据是一些整数、实数或复数；非数值数据包括字符、文字、图形、图像、语音等。

2．数据元素

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。一个数据元素可由若干个数据项（Data Item）组成。在不同的条件下，数据元素又可称为数据元素、节点、顶点、记录等。例如，学生信息检索系统中学生信息表中的一个记录、教学计划编排问题中的一个顶点等，都被称为一个数据元素。

3．数据项

数据项（Data Item）指不可分割的、具有独立意义的最小数据单位，数据项有时也称为字段（Field）或域。例如，学籍管理系统中学生信息表的每一个数据元素就是一个学生记录。它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。这些数据项可以分为两种：一种叫作初等项，如学生的性别、籍贯等，这些数据项是在数据处理时不能再分割的最小单位；另一种叫作组合项，如学生的成绩，它可以再划分为数学、物理、化学等更小的项。通常，在解决实际应用问题时把每个学生记录当作一个基本单位进行访问和处理。

4．数据结构

数据结构（Data Structure）是指互相之间存在着一种或多种关系的数据元素的集合。在任何问题中，数据元素都不会是孤立的，在它们之间存在着这样或那样的关系，这种数据元素之间存在的关系称为数据的逻辑结构。根据数据元素之间关系的不同特性，通常有以下 4 类基本的逻辑结构。

（1）集合结构：在集合结构中，数据元素之间的关系是“属于同一个集合”。数据元素之间除了同属一个集合，不存在其他关系。

（2）线性结构：在该结构中，数据元素除了同属于一个集合，数据元素之间还存在着一对一的顺序关系。

（3）树形结构：该结构的数据元素之间存在着一对多的层次关系。

（4）图状结构：该结构的数据元素之间存在着多对多的任意关系，图状结构也称为网状结构。

上述 4 类基本结构的示意图如图 1.5 所示。

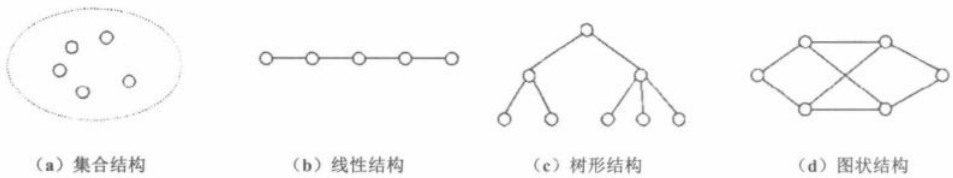


图 1.5　4 类基本结构的示意图

由于集合是数据元素之间极为松散的一种结构，本书不专门讨论。因此，本书主要讨论线性结构（表、栈、队、串等）和非线性结构（树、图或网）。

从上面所介绍的数据结构的概念中可以知道，一个数据结构有两个要素：一是数据元素，二是数据元素之间的关系。因此，数据结构通常可以采用一个二元组来表示：

```
Data_Structure = (D,R)
```

其中，D 是数据元素集合，R 是 D 中数据元素之间关系的集合。

【例 1.4】 假设一个数据结构定义如下：

$$DS = (D,R)$$

$$D = \{ a, b, c, d, e, f, g \}$$

$$R = \{ <a, b>,<a, c>,<a, d>,<c, e>,<c, f>,<d, g> \}$$

则该数据结构的逻辑示意图如图 1.6 所示，显然是一个树形结构。

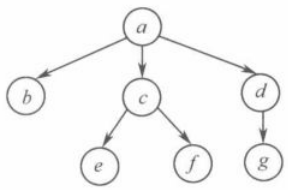


图 1.6 例 1.4 的数据结构逻辑示意图

数据结构包括数据的逻辑结构和物理结构。数据的逻辑结构可以看作从具体问题抽象出来的数学模型，它与数据的存储无关。数据的逻辑结构在计算机中的存储表示（又称映像）称为数据的物理结构（或称存储结构），它所研究的是数据结构在计算机中的实现方法，包括数据结构中数据元素的存储表示及数据元素之间关系的表示。

在计算机中，数据的存储方法包括顺序存储和链式存储。

（1）顺序存储方法通过数据元素在计算机中的存储位置关系来表示数据元素间的逻辑关系，通常把逻辑上相邻的数据元素存储在物理位置相邻的存储单元中。顺序存储是一种最基本的存储表示方法，通常借助程序设计语言中的数组来实现。

（2）链式存储方法对逻辑上相邻的数据元素不要求其物理位置相邻，数据元素间的逻辑关系通过指针字段来表示。链式存储结构通常借助程序设计语言中的指针来实现。

除了顺序存储方法和链式存储方法，有时为了查找方便还采用索引存储方法和散列表（Hash）存储方法。

讨论数据结构的目的就是在计算机中实现对数据的运算，因此在讨论数据的组织结构时必然要考虑在该结构上进行的运算（或称运算）。事实上，数据结构是专门研究某一类数据的表示方法及其相关运算实现算法的一门学科。

5．数据类型

数据类型（Data Type）是和数据结构密切相关的一个概念，在高级程序设计语言中用以限制变量取值范围和可能进行的运算的总和称为数据类型。因此，所谓数据类型，一是限定了数据的取值范围（实际上与存储形式有关）；二是规定了数据能够进行的一组运算（运算）。

数据类型可分为两类：一类是非结构的原子类型，原子类型的值是不可再分解的，如 C 语言中的基本类型（整型、实型、字符型及指针类型和空类型）；另一类是结构类型，它的成分可以由多个结构类型组成，并可以分解。结构类型的成分可以是非结构的，也可以是结构的。例如，数组的值由若干分量组成，每个分量可以是整数等基本类型，也可以是数组等结构类型。

6．抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型及定义在该模型上的一组运算。抽象数据类型的定义取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即无论其内部结构如何变化，只要它的数学特性不变，就不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各种计算机都拥有的整数类型就是一个抽象数据类型，尽管它们在不同处理器上的实现方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组运算组成，而数据结构又包括数据元素及数据元素间的关系，因此抽象数据类型一般可以由数据元素、关系及运算三个要

素来定义。

本书在讨论各种数据结构时，针对其逻辑结构和具体的存储结构给出相应的数据类型，并在确定的数据类型上通过各种算法实现各种运算。

1.3 算法及算法分析

算法与数据结构的关系非常紧密，在算法设计时总是先要确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。下面就从算法特性、算法描述和算法分析三个方面对算法进行介绍。

1.3.1 算法特性

算法（Algorithm）是对特定问题求解步骤的一种描述，是指令的有限序列，其中每一条指令表示一个或多个运算。一个算法应该具有下列特性。

（1）有穷性：一个算法必须在有穷步之后结束，即必须在有限时间内完成。

（2）确定性：算法的每一步必须有确切的定义，无二义性，且在任何条件下算法只有唯一一条执行路径，即对于相同的输入只能得出相同的输出。

（3）可行性：算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。

（4）输入：一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。

（5）输出：一个算法具有一个或多个输出，这些输出同输入之间存在某种特定的关系。

算法的含义与程序十分相似，但又有区别。一个程序不一定满足有穷性。例如，对于运算系统，只要整个系统不遭破坏，它将永远不会停止，即使没有作业需要处理，它仍处于动态等待中。因此，运算系统不是一个算法。另外，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的求解方法，而程序则是算法在计算机上的特定实现。一个算法若用程序设计语言来描述，就是一个程序。

算法与数据结构是相辅相成的。解决某一类特定问题的算法可以选择不同的数据结构，而且选择恰当与否直接影响算法的效率。反之，一种数据结构的优劣由各种算法的执行效果来体现。

在算法设计时通常需要考虑以下几个方面的要求。

（1）正确性：算法的执行结果应当满足预先规定的功能和性能要求。正确性要求表明算法必须满足实际需求，达到解决实际问题的目标。

（2）可读性：一个算法应当思路清晰、层次分明、简单明了、易读易懂。一个可读性强的算法，其程序的可维护性、可扩展性都要好得多，因此，许多时候人们往往在一定程度上牺牲效率来提高可读性。

（3）健壮性：当输入不合法数据时，应能适当处理，不至于引起严重后果。健壮性要求表明算法要全面细致地考虑所有可能的边界情况，并对这些边界条件做出完备的处理，尽可能使算法没有意外的情况。

（4）高效性：有效使用存储空间和有较好的时间效率。高效性主要是指时间效率，即解决相同规模的问题时间尽可能短。

一般来说，数据结构上的基本运算主要有以下几种。

（1）查找：寻找满足特定条件的数据元素所在的位置。

（2）读取：读出指定位置上数据元素的内容。
（3）插入：在指定位置上添加新的数据元素。
（4）删除：删去指定位置上对应的数据元素。
（5）更新：修改某个数据元素的值。

1.3.2 算法描述

算法的描述方法很多，根据描述方法的不同，大致可将算法描述分为以下 4 种。

（1）自然语言算法描述：用人类自然语言（如中文、英文等）来描述算法，同时还可插入一些程序设计语言中的语句来描述，这种方法也称为非形式算法描述。其优点是不需要专门学习，任何人都可以直接阅读和理解，但直观性很差，复杂的算法难写难读。

（2）框图算法描述：这是一种图示法，可以采用方框图、流程图、N-S 图等来描述算法，这种描述方法在算法研究的早期曾一度流行。它的优点是直观、易懂，但用来描述比较复杂的算法就显得不够方便，也不够清晰简洁。

（3）伪代码算法描述：如类 C 语言算法描述。这种算法描述很像程序，但它不能直接在计算机上编译、运行。这种方法很容易编写、阅读，而且格式统一，结构清晰，专业设计人员经常使用类 C 语言来描述算法。

（4）高级程序设计语言编写的程序或函数：这是直接用高级语言来描述算法，它可在计算机上运行并获得结果，使给定问题能在有限时间内被求解。通常这种算法描述也称为程序。

【例 1.5】 求两个整数 m、n（$m \geq n$）的最大公因子，该算法的不同描述方法如下。

（1）自然语言算法描述（非形式算法描述）如下。

① [求余数]以 n 除 m，并令 r 为余数（$0 \leq r < n$）；

② [判断余数是否为零]若 $r = 0$，则结束算法，n 就是最大公因子；

③ [替换并返回步骤①]若 $r \neq 0$，则 $m \leftarrow n$，$n \leftarrow r$，返回步骤①。

（2）算法的框图描述如图 1.7 所示。

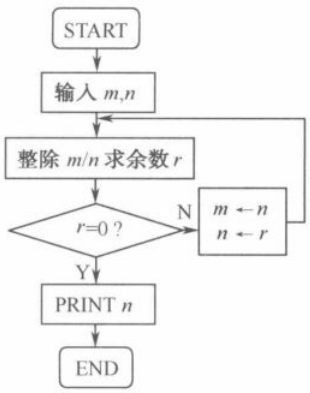


图 1.7 算法的框图描述

（3）C 语言函数描述如下。

```
int max_common_factor(int m, int n)
{
int  r;
```

```
    r=m%n;
    while(r!=0)
        {  m=n;n=r;r=m%n;}
           return n;
    }
```

本书主要介绍算法的思路和实现过程，且尽可能地给出算法对应的 C 语言函数或程序（或类 C 语言算法描述），方便读者阅读或上机运行，以便更好地理解算法。

1.3.3 算法分析

所谓好的算法，除了满足上文提到的几个基本要求，还必须以较少的时间与空间代价来解决相同规模的问题。因此，一个算法的优劣，可以从该算法在计算机上运行的时间和所占存储空间来衡量和评判。算法分析就是预先分析算法在实际执行时的时空代价指标。

当一个算法被转换成程序并在计算机上执行时，其运行所需要的时间一般取决于下列几个因素。

（1）硬件的速度。即主机本身运行速度，主要与 CPU 的主频和字长有关，也与主机系统采用的技术有关，如多机系统的运算速度一般比单机系统要快。

（2）实现算法的程序设计语言。实现算法的语言的级别越高，其执行效率相对就越低。

（3）编译程序所生成目标代码的质量。代码优化较好的编译程序所生成的程序质量较高。

（4）算法所采用的策略。采用不同设计思路与解题方法，其时空代价是不同的，一般情况下时间指标与空间指标常常是矛盾的两个方面。

（5）问题的规模。例如，求 100 以内的素数与求 1 000 以内的素数的执行时间必然不同。

显然，在各种因素都不能确定的情况下，很难比较算法的执行时间。也就是说，用算法的绝对执行时间来衡量算法的效率是不合适的。为此，可以将上述各种与计算机相关的软、硬件因素都确定下来，仅对采用不同策略的算法，分析其运行代价随问题规模大小变化的对应关系，即运行代价仅依赖于问题的规模（通常用正整数 n 表示），或者说它是问题规模的函数。这种函数被称为算法的时间复杂度和空间复杂度。

1. 时间复杂度

一个程序的时间复杂度（Time Complexity）是指该程序的运行时间与问题规模的对应关系。一个算法是由控制结构和原运算（所谓原运算是指从算法中选取对于所研究问题是基本运算的运算）构成的，其执行时间取决于两者的综合效果。为了便于比较同一问题的不同的算法，通常的做法是：从算法中选取一种对于所研究的问题来说是基本运算的原运算，以该原运算重复执行的次数为算法的时间度量。在一般情况下，算法中原运算重复执行的次数是该算法所处理问题的规模 n 的某个函数 $T(n)$。

【例 1.6】 两个 $n \times n$ 阶的矩阵相乘的程序中的主要语句及其重复次数如下。

原运算语句的执行频度如下。

```
    for  ( i = 0;i < n;i + + )                          //原运算语句的执行频度
      for  ( j = 0;j < n;j + + )
        {   s [ i ][ j ] = 0;                           //n²
          for  ( k = 0;k < n;k + + )
            s [ i ][ j ] = s [ i ][ j ] + a [ i ][ k ] * b [ k ][ j ];//n³
        }
```

则该段程序的时间复杂度 $T(n) = cn^3 + n^2$，其中 c 为常量，表示算术运算时间是简单赋值运算时间的常数倍。

通常情况下，精确地计算 $T(n)$是困难的，人们引入渐进时间复杂度在数量上估计一个算法的执行时间，也能够达到分析算法的目的。

定义（大 O 记号）：如果存在两个正常数 c 和 n_0，使得对所有的 n（$n \geqslant n_0$），有：

$$T(n) \leqslant c*f(n)$$

则 $T(n)=O(f(n))$。

例如，一个程序的实际执行时间为 $T(n)=2.7n^3+3.8n^2+5.3$，则 $T(n)=O(n^3)$。

使用大 O 记号表示的算法的时间复杂度称为算法的渐进时间复杂度（Asymptotic Time Complexity）。

通常用 $O(1)$表示常数级时间复杂度，表明这样的算法执行时间是恒定的，不随问题规模的扩大而增长，显然这是最理想的，但往往难以实现。此外，常见的渐进时间复杂度还有：

（1）$O(\log_2 n)$，对数级复杂度；

（2）$O(n)$，线性复杂度；

（3）$O(n^2)$和 $O(n^3)$，分别为平方级和立方级复杂度；

（4）$O(2^n)$，指数级复杂度。

上述时间复杂度随问题规模 n 的扩大其增长速度是不同的，其增长速度的快慢次序表示如下：

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

2．空间复杂度

一个程序的空间复杂度（Space Complexity）是指程序运行从开始到结束所需的存储量与问题规模的对应关系，记作：

$$S(n) = O(f(n))$$

其中 n 为问题的规模（或大小）。

一个上机执行的程序除了需要存储空间来寄存本身所用指令、常数、变量和输入数据，还需要一些对数据进行运算的工作单元和存储为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身，和算法无关，则只需分析除输入数据和程序之外的额外空间，否则应同时考虑输入数据本身所需空间（和输入数据的表示形式有关）。若额外空间相对于输入数据量来说是常数，则称此算法为原地工作，辅助空间为 $O(1)$。如果所占空间量依赖于特定的输入，则除特别指明外，均按最坏情况来分析。

算法执行时间的耗费和所占存储空间的耗费是相互矛盾的，难以兼得。即算法执行时间上的节省是以增加存储空间为代价的，反之亦然。不过，一般而言，常常以算法执行时间作为算法优劣的主要衡量指标。

1.4 关于数据结构的学习

计算机机器性能价格比持续提高，硬件发展如此之快，是否没有必要去追求提高算法的时间复杂度、没有必要去追求节省算法占用存储空间的数目呢？不是没有必要，而是要求越来越高。原因之一是，由于机器性能价格比的提高，人们所面临的处理问题的问题规模越来越大，要把过