高职高专计算机系列规划教材

数据结构（C语言版）（第2版）

邓文华　李益明　主编

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

高职高专计算机系列规划教材

数据结构（C语言版）

（第2版）

邓文华　李益明　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，力求概念清晰，注重实际应用。全书共分 10 章，依次介绍了数据结构的基本概念、线性表、栈、队列串和数组、树结构和图结构，以及查找和排序等基本运算。第 9 章给出了几个综合应用实例程序，以供学生上机实习，第 10 章给出了 10 套模拟试题及相应的试题分析，供师生参考。全书用 C 语言作为算法描述语言并且每章后面均列举了典型应用实例。

本书主要面向高职高专院校计算机专业的学生，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用书。

图书在版编目(CIP)数据

数据结构：C 语言版 / 邓文华，李益明主编．—2 版．—北京：电子工业出版社，2007.8
（高职高专计算机系列规划教材）
ISBN 978-7-121-04502-8

Ⅰ．数… Ⅱ．①邓…②李… Ⅲ．①数据结构－高等学校：技术学校－教材②C 语言－程序设计－高等学校：技术学校－教材 Ⅳ．TP311.12 TP312

中国版本图书馆 CIP 数据核字（2007）第 075173 号

责任编辑：吕 迈
印 刷：
装 订：北京市铁成印刷厂
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：787×1 092 1/16 印张：14.75 字数：378 千字
印 次：2007 年 8 月第 1 次印刷
印 数：5 000 册 定价：19.00 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。

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。

服务热线：（010）88258888。

前　言

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

自本教材第一版 2004 年出版发行以来，受到了广大师生、读者的欢迎，至今已发行近 3 万册。在此对广大师生、读者表示衷心的感谢。为了更好地适应教学的需要，我们在广泛收集读者意见的基础上对本书进行了修订。本版在保留原教材特点的基础上，主要做了以下修改。

（1）增加了综合应用实例一章，给出了 4 个综合应用例子，如约瑟夫环问题、迷宫问题、短信促销活动和保龄球记分系统，供学生实习，以进一步加强对学生应用能力的培养。

（2）对原书中的某些章节进行了小幅调整与修改，使内容更实用。

（3）加了部分例题与习题，更有针对性。

本书共 10 章。第 1 章叙述数据、数据结构和算法等基本概念；第 2 章至第 6 章分别讨论了线性表、栈和队列、串和数组、树与二叉树，以及图等基本数据结构的实现及其应用；第 7、8 章分别讨论了查找和排序操作的各种实现方法及其特点；第 9 章给出了几个综合应用的实例程序；第 10 章给出了 10 套模拟试卷及参考答案，以供学生练习、参考。

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，淡化算法的设计分析和复杂的时空分析。

（2）各章（除 1，8 章外）都配有相应的应用实例，列举分析了许多实用的例子。大多数算法都直接给出了其相应的 C 语言程序，以便学生上机练习、实践。

（3）为了便于学生复习及掌握每章的重点、难点，本书在每章的结束处对该章进行小结。

本教材讲课时数为 60 学时左右，上机时数在 20 学时左右。教师可根据学时数、专业和学生的实际情况选讲应用举例中的例子。

本书第 1、5、6、9 章由李益明编写，第 2、3、4、7、8 章由邓文华编写，第 10 章及附录由李益明、邓文华老师共同编写，全书由邓文华统稿、修改、定稿。

江汉大学的乔维声老师对本教材也提出了许多中肯的意见和建议，在此一并表示衷心的感谢！

编　者

2007 年 4 月

目　录

第1章 绪 论

数据作为计算机加工处理的对象，在计算机中如何表示、存储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，它是计算机可以直接处理的最基本和最重要的对象。科学计算、数据处理、过程控制、文件存储、数据库技术等，都是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及数据间的相互关系及其对应的存储表示，并利用这些特性和关系设计出相应的算法和程序。

1.1 从问题到程序

“数据结构”作为计算机科学与技术专业的专业基础课，是十分重要的核心课程，其主要的研究内容就是数据之间的逻辑关系和物理实现，探索有利的数据组织形式及存取方式。所有的计算机系统软件和应用软件的设计、开发都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，仅掌握几种计算机程序设计语言是难以应付众多复杂的课题的。要想有效地使用计算机、充分发挥计算机的性能，还必须学习和掌握数据结构的有关知识。

在计算机技术发展的初期，人们使用计算机的目的主要是处理数值计算问题。当人们使用计算机解决具体问题时，一般需要经过以下几个步骤：首先从该具体问题抽象出一个适当的数学模型，然后设计或选择一个解此数学模型的算法，最后编出程序进行调试、测试，直至得到最终的结果（如图 1.1 所示）。例如，求解梁架结构中应力的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。

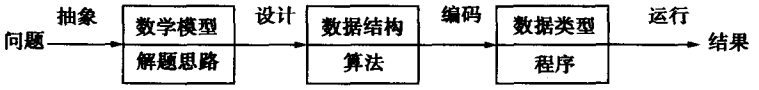


图 1.1 计算机解决问题的一般过程

由于当时所涉及的运算对象是简单的整型、实型或布尔型数据，所以程序设计者的主要精力集中在程序设计的技巧上，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，非数值计算问题越来越显示出重要性。据统计，当今处理非数值计算性问题占用了 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，而是要设计出合理的数据结构，才能有效地解决问题。下面所列举的就是属于这一类的具体问题。

【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，人们需要将图书

信息按不同分类进行编排，建立合适的数据结构进行存储和管理，按照某种算法编写相关程序，就可以实现计算机自动检索。由此，一个简单的信息检索系统就是建立一张按图书分类号和登录号顺序排列的图书信息表，以及分别按作者名、出版社等顺序排列的各类索引表，如图 1.2 所示。由这 3 张表构成的文件就是图书信息检索的数学模型，计算机的主要操作就是按照用户的要求（如给定书名）通过不同的索引表对图书信息进行查询。

图书分类号	登录号	书　名	作者	出 版 社
B259.1	3240	梁启超家书	张品兴	中国文联出版社
C52	5231	探寻语碎	李泽厚	上海文艺出版社
D035.5	6712	市政学	张永桃	高等教育出版社
G206	1422	传播学	邵培仁	高等教育出版社
H319.4	1008	英语阅读策略	李宗宏	兰州大学出版社
K825.4.00	5819	围棋人生	聂卫平	中国文联出版社
P1.00	8810	通向太空之路	邹惠成	科学出版社
TN915	7911	通信与网络技术概论	刘云	中国铁道出版社
TP312	7623	计算机软件技术基础	王宇川	科学出版社
TP393.07	8001	网络管理与应用	张琳	人民邮电出版社
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。在这类文档管理的数学模型中，计算机处理的对象之间通常存在着一种简单的线性关系，这类数学模型可称为线性的数据结构。

【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出现的情况以及响应的对策。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。由此，计算机操作的对象

（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，因此常用树形结构来表示。如图 1.3 所示是井字棋的对弈树。

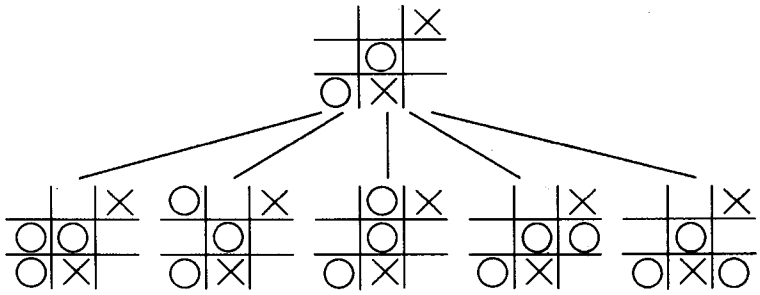
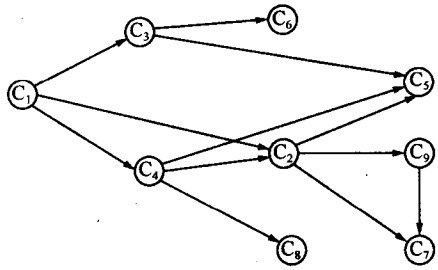


图 1.3　井字棋对弈树

【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。即有些课程之间有先修和后续的关系。这种各个课程之间先修和后续的次序关系可用一个称做“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 C_i 到 C_j 之间存在有向边<C_i,C_j>，则表示课程 i 必须先于课程 j 进行。

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	操作系统	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

由以上几个例子可见，描述这类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，可以说数据结构课程主要是研究非数值计算的程序设计问题中所出现的计算机处理对象以及它们之间的关系和操作的学科。

学习数据结构的目的是为了了解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。与此同时，通过算法训练提高学生的思维能力；通过程序设计的技能训练促进学生的综合应用能力，提高学生的专业素质。

1.2 有关概念和术语

在系统地学习数据结构知识之前，先掌握一些基本概念和术语的定义。

1. 数据（Data）

数据是信息的载体，它能够被计算机识别、存储和加工处理，它是计算机程序加工的原料。应用程序可以处理各种各样的数据，它可以是数值数据，也可以是非数值数据。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（Data Element）

数据元素是数据的基本单位，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，数据元素又可称为元素、结点、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（Data Item）

数据项指不可分割的、含有独立意义的最小数据单位，数据项有时也称字段（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：一种叫做初等项，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；另一种叫做组合项，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是把每个学生记录当做一个基本单位进行访问和处理的。

4. 数据结构（Data Structure）

数据结构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，数据元素之间都不会是孤立的，在它们之间都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。

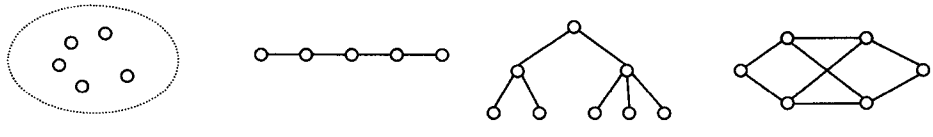
（1）集合结构：在集合结构中，数据元素之间的关系是“属于同一个集合” 的关系。数据元素之间除了同属一个集合外，不存在其他关系。

（2）线性结构：该结构中的数据元素之间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，图状结构也称网状结构。

图 1.5 为表示上述 4 类基本结构的示意图。



（a）集合结构　（b）线性结构　（c）树形结构　（b）图状结构

图 1.5　4 类基本结构示意图

由于集合是数据元素之间关系极为松散的一种结构，因此也可用其他结构来表示它，本书不做专门讨论。撇开集合关系，可以把数据的逻辑结构分为线性结构（表、栈、队、串）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。一个是数据元素的集合，另一个是数据元素之间关系的集合。在形式上，数据结构通常可以用一个二元组来表示：

Data_Structure =（D,R）

其中，D 是数据元素的有限集合，R 是 D 中元素之间关系的有限集合。

【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> }

则该结构的逻辑示意如图 1.6 所示，显然是一个树形结构。

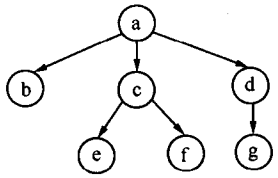


图 1.6 数据结构逻辑示意图

数据结构包括数据的逻辑结构和数据的物理结构。数据的逻辑结构可以看做是从具体问题抽象出来的数学模型，它与数据的存储无关。我们研究数据结构的目的是为了在计算机中实现对它的操作，为此还需要研究如何在计算机中表示和存储一个数据结构。数据的逻辑结构在计算机中的表示（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表示及数据元素之间关系的表示。

数据的存储结构可采用顺序存储或链式存储的方法。

（1）顺序存储方法是通过数据元素在计算机中存储位置上的关系来表示出元素逻辑上的关系，通常把逻辑上相邻的元素存储在物理位置相邻的存储单元中，由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于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元素不要求其物理位置相邻，元素间的逻辑关系通过附设的指针字段来表示，由此得到的存储表称为链式存储结构，链式存储结构通常借助于程序设计语言中的指针类型来实现。

除了通常采用的顺序存储方法和链式存储方法外，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为了在计算机中实现对数据的操作，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操作（或称运算）。事实上，数据结构就是专门研究某一类数据的表示方法及其相关操作的实现算法。

5. 数据类型（Datatype）

数据类型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操作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操作（运算）。

数据类型可分为两类：一类是非结构的原子类型，如C语言中的基本类型（整型、实型、字符型等）、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。结构类型的成分中可以是非结构的，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。

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，进一步在确定的数据类型上实现各种操作。

1.3 算法及算法分析

算法与数据结构的关系紧密，在算法设计时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、算法性能分析与度量等3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操作。一个算法应该具有以下特性。

（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惟一一条执行路径，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。例如操作系统，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，操作系统不是一个算法。另一方面，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解，而程序则是算法在计算机上特定的实现。如果用程序设计语言来描述一个算法，那么它就是一个程序。

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定不同的数据结构，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的要求。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的。

（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。可读性要求表明算法主要是人与人之间交流解题思路和进行软件设计的工具，因此可读性必须强。同时一个可读性强的算法，其程序的可维护性、可扩展性都要好许多，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，应能做适当处理，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出现的边界情况，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。高效性要求主要是指时间效率，即解决相同规模的问题时间应尽可能短。

一般来说，数据结构上的基本操作主要有以下几种。

（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经流行过。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，它是可在计算机上运行并获得结果的算法描述，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】　下面以求两个整数 m,n(m≥n)的最大公因子，该算法的不同描述方法如下：

（1）非形式算法描述（自然语言算法描述）如下。

a．求余数：以 n 除 m，并令 r 为余数（0≤r<n）；

b．余数是否为零：若 r=0，则结束算法，n 就是最大公因子；

c．替换并返回 a：若 r≠0，则 m←n，n←r 返回 a。

（2）该问题的框图描述（程序流程图）如图 1.7 所示。

（3）该问题的 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 ;
}
```

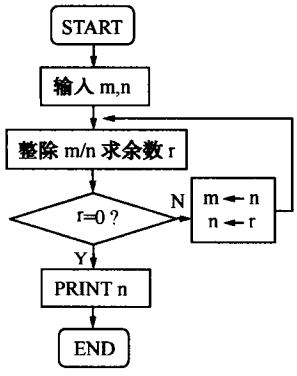


图 1.7　程序流程图

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外，还必须以较少的时间和空间的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的运行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以下几个因素：

（1）硬件的速度。即主机本身的运行速度，例如使用 CPU 是 486 的微机还是使用 CPU 为 P4 的微机。

（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相互矛盾的两个方面。

（5）问题的规模。例如，求 100 以内的素数与求 1000 以内的素数其执行时间必然是不同的。

显然，在各种因素都不能确定的情况下，很难比较出算法的执行时间。也就是说，使用执行算法的绝对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，针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操作构成的，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操作，以该原操作重复执行的次数作为算法的时间度量。一般情况下，算法中原操作重复执行的次数是规模 n 的某个函数 T(n)。

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

原操作语句的执行频度

```
for  (i=0; i<n; i++)
   for  (j=0; j<n; j++)
     {   s[i][j]=0;                                    n^2
         for  (k=0; k<n; k++)
             s[i][j]=s[i][j]+a[i][k]*b[k][j];          n^3
     }
```

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）。

许多时候要精确地计算 T(n)是困难的，我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（$n \geq n_0$）有

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

则有

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

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，它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：

$O(\log_2 n)$，对数级复杂度；

$O(n)$，线性复杂度；

$O(n^2)$，$O(n^3)$，分别为平方级、立方级复杂度；

$O(2^n)$，指数级复杂度。

它们随问题规模 n 的扩大而增长且增长的速度是不同的，其大小次序如下：

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

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，记做

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

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、变量和输入数据外，也需要一些对数据进行操作的工作单元和存储一些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，则只需要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。又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。

1.4 关于数据结构的学习

“数据结构”作为一门独立的课程在国外是从 1968 年开始的，但在此之前其有关内容已散见于编译原理及操作系统之中。20 世纪 60 年代中期，美国的一些大学开始设立有关课程，但当时的课程名称并不叫数据结构。1968 年美国唐·欧·克努特教授开创了数据结构的最初体系，他编著的《计算机程序设计技巧》第一卷《基本算法》是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作。从 20 世纪 60 年代末到 70 年代初，出现了大型程序，软件也相对独立，结构化程序设计成为程序设计方法学的主要内容，人们越来越重视数据结构。认为程序设计的实质是确定数据的结构，加上设计一个好的算法，也就是人们所说的“程序=数据结构+算法”。

作为一门课程，“数据结构”比较系统地介绍了软件设计中常用的数据结构以及相应的实