高等院校计算机教育系列教材

数据结构与算法

（C语言版）（第3版）

陈琳琳　李建林　主　编
孙启虎　李　橙　郭龙源　副主编

- 结构清晰，知识完整
- 入门快速，易教易学
- 实例丰富，实用性强
- 学以致用，注重能力

赠送实例代码和电子课件

清华大学出版社

高等院校计算机教育系列教材

数据结构与算法(C 语言版)
(第 3 版)

陈琳琳　李建林　主　编
孙启虎　李　橙　郭龙源　副主编

清华大学出版社
北 京

内 容 简 介

“数据结构与算法”是计算机学科研究的主题之一。本书采用类C语言描述，系统地介绍了各种数据结构和排序、查找算法。全书共分为9章，主要内容包括数据结构与算法简介、线性表、栈和队列、串、数组及广义表、树和二叉树、图、查找和排序等。对于各种数据结构，本书给出了基本概念、抽象数据类型以及相关的操作，并且对各种算法的运行时间进行了分析。

本书对数据结构中的重点和难点内容进行了深入的剖析，着重培养学生的动手能力，对经典算法、重点算法及应用算法进行了详细的讲解，以使学生更好地掌握数据结构的应用。

本书可作为计算机及相关专业的大学本科教材，也可作为应用型专业以及成人教育、工程技术人员的培训教材。

图书在版编目(CIP)数据

数据结构与算法(C语言版)/陈琳琳，李建林主编. --3版. --北京：清华大学出版社，2015
(高等院校计算机教育系列教材)
ISBN 978-7-302-40253-4

Ⅰ. ①数… Ⅱ. ①陈… ②李… Ⅲ. ①数据结构—高等学校—教材 ②算法分析—高等学校—教材 ③C语言—程序设计—高等学校—教材 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆CIP数据核字(2015)第106328号

责任编辑：章忆文　杨作梅
装帧设计：杨玉兰
责任校对：周剑云
责任印制：王静怡

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,010-62791865

印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：25　　**字　　数**：605千字
版　　次：2005年11月第1版　2015年7月第3版　　**印　　次**：2015年7月第1次印刷
印　　数：1～3000
定　　价：42.00元

产品编号：057599-01

前　言

本书订正了第 2 版中的笔误、描述不规范等问题，简化了使用不多的内容；每章的编程项目都有完整的 C 程序实现代码，并都在 VC++6.0 环境下编译通过，运行正确。

数据结构是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。它侧重于体系和思想上的训练，是程序设计的灵魂，为计算机语言课程设计提供了方法性的理论指导，所有的计算机系统软件和应用软件都要用到各种类型的数据结构。算法与数据结构之间存在着相辅相成的关系。解决一个问题既可以选择不同的数据结构，也可以选择不同的算法。数据结构的选择决定了算法执行所需要的时间与空间，影响算法的效率；反之，算法的优劣又决定了某个数据结构是否适合解决这个问题。

全书共分为 9 章，由浅入深、系统地讨论了各种数据结构的基本概念和相关操作，还介绍了查找和排序算法。各章的具体内容介绍如下。

第 1 章是绪论，介绍了学习数据结构和算法的意义、数据结构和算法的基本概念，并且指出了数据结构和算法之间的关系，给出了分析算法的方法。

第 2 章主要介绍了线性表的概念、抽象数据类型及其基本操作，最后列举了一些线性表的应用实例。

第 3 章主要介绍了受限制的线性表，即栈和队列；重点介绍了栈和队列的抽象数据类型及其实现，并列举了几个应用实例。

第 4 章主要介绍了串这一非数值数据结构，包括串的抽象数据类型及其基本操作与串的模式匹配算法。

第 5 章介绍了数组这一数据类型在计算机中的存储，重点介绍了稀疏矩阵在计算机中的压缩存储及其操作的实现，同时还介绍了广义表的概念。

第 6 章讨论了树形结构及其相关算法。

第 7 章讨论了图形结构及其相关算法。图形结构是一种复杂的数据结构，其应用也非常广泛，该章在介绍了图的遍历算法以后还采用遍历算法推导出了其他常用算法。

第 8 章讨论了在线性表和树上的查找算法，还介绍了期望查找复杂度为 $O(1)$的哈希表查找算法。

第 9 章重点介绍了插入排序、选择排序、交换排序、归并排序和基数排序等算法及其思想和分析，在最后还介绍了外部排序算法。

本书示例较多，讲解细致，对数据结构中的重点和难点内容进行了深入的剖析，着重培养学生的动手能力，突出实用性；对经典算法、重点算法及应用算法进行了详细的讲解，以使学生更好地掌握数据结构的应用。本书采用类 C 语言来描述算法。类 C 语言是伪码和 C 语言组合而成的一个描述工具，采用了 C 语言的核心部分，并为描述方便进行了扩充。

本书可作为计算机及相关专业的大学本科教材，也可作为应用型专业以及成人教育、

工程技术人员的培训教材。

本书主要由陈琳琳、李建林任主编，由孙启虎、李橙、郭龙源任副主编。何光明、赵传申、许娟、王珊珊、石雅琴、郑爱琴、许悦、陈珍、陈凤、卢振侠、曹冬梅、杨橙、陈莉萍等人员在内容编写、程序测试、文字校对等工作中也付出了辛勤劳动，在此一并表示衷心的感谢！

本书配有电子教案，并提供程序源代码，以方便读者自学，请到清华大学出版社网站下载。

限于作者水平，书中难免存在不当之处，恳请广大读者批评指正。任何批评和建议请发至 Book21Press@126.com 邮箱。

编　者

目　录

第1章 绪 论

Data Structure + Algorithm=Program.

Nikiklaus Wirth (1976)

本章初步讲解数据结构与算法，主要明确为什么要学习数据结构与算法、数据结构与算法的基本概念以及如何进行算法分析。

1.1 学习数据结构与算法的意义

1.1.1 学习数据结构的意义

“为什么要学习数据结构？”这是每个初学者都会提出的问题，为了能更好地回答这个问题，首先来看下面两个例子。

例 1.1 八皇后问题。这是一个古老而著名的问题。该问题是 19 世纪著名的数学家高斯于 1850 年提出的。问题描述：在 8×8 格的国际象棋盘上摆放着 8 个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。

这个问题的求解过程不是根据某种确定的计算法则，而是利用试探和回溯技术求解。为了能得到合理的布局，在计算机中要存储布局的当前状态。为了方便起见，以四皇后为例，图 1.1 给出了计算机存储这些布局的方式。

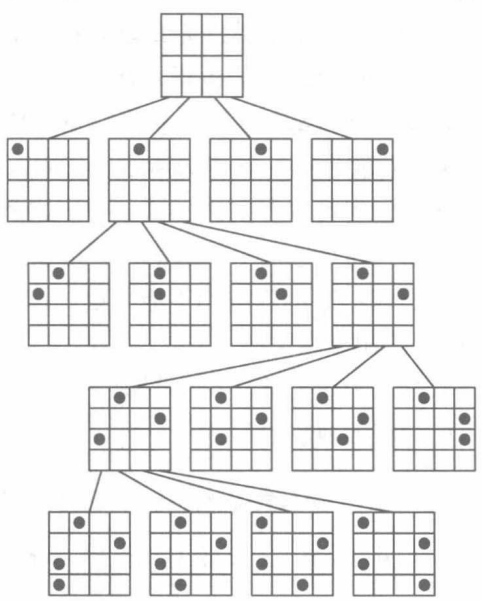


图 1.1 四皇后问题的状态树

如图 1.1 所示的结构在数据结构中称为树，在四皇后问题中又称为状态树。树的根结点是棋盘布局的最初状态，其余的结点是每一步试探时棋盘的布局状态。只需要遍历这棵树，寻找合理的分支，就可以得到问题的解。

八皇后问题中的状态树就是一种数据结构，它是数据结构中较为常用的树形结构。例 1.1 是一个典型的非数值问题，对这些非数值问题的求解不再是数学方程，而是一些诸如树之类的数据结构。由此可以看出，数据结构为研究非数值计算问题提供了数据的表示与操作途径。数据结构是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。所有的计算机系统软件和应用软件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课题的。要想有效地使用计算机，充分发挥计算机的功能，还必须学习和掌握好数据结构的有关知识。扎实地打好“数据结构”这门课程的基础，对于学习计算机专业的其他课程，如操作系统、编译原理、数据库管理系统、软件工程及人工智能等都是十分有益的。

学习数据结构的意义

有了数据结构知识，可以帮助我们分析数据的特征和数据之间的关系，从而更好地组织数据，更加有效地使用计算机，充分发挥计算机的功能。

1.1.2 学习算法的意义

算法也是一门很重要的课程，其重要性可以通过例 1.2 看出。

例 1.2 选择问题。设有一组 N 个数，确定其中的第 k 个是最大者。

这是一个比较简单的问题。大多数学习过一两门计算机程序设计语言的学生都可以不费劲地写出解决这个问题的程序。例如，一个较直接的解决方法是先将这 N 个数读进一个数组中，再通过某一简单的算法，如冒泡排序法，将这些数以递减顺序进行排序，然后返回第 k 个位置上的元素。另一种稍好一点的算法是可以先把前 k 个数读入数组中并将它们递减排序；接着，将剩下的元素再逐个读入。当新元素被读到时，如果它小于数组中的第 k 个元素，则忽略；否则就将其放到数组中正确的位置，同时将数组中的一个元素挤出数组，当算法终止时，位于第 k 个位置上的元素即为所求的元素。

上面的解法都很简单，建议读者可以上机试一试。有了这两个解法后我们就会问：哪一个算法更好？哪个算法更重要？还是这两个算法都好？可以发现，当 N 增大到 1000000，k 为 500000 时，这两个算法在合理的时间范围均不能结束。本书将在第 9 章讨论另外一种算法，这种算法可以在较短的时间内给出结果。

从上面的例子可以看到，学习和研究算法可以明确分析所得到的算法的好坏，寻找能满足要求的较优算法，从而更加高效地解决问题。

学习算法的意义

学习算法设计的方法和算法分析的技术后，可以帮助我们设计较好的算法，分析算法的优、缺点，从而找出解决某一问题的最好方法。

➢ 自测题

1. 学习数据结构的意义是什么？
2. 学习算法的意义是什么？

1.2 数 据 结 构

Less than 10% of the code has to do with the ostensible purpose of the system; the rest deals with input-output, data validation, data structure maintenance, and other housekeeping.

Mary Shaw

1.2.1 数据结构概述

数据结构是在整个计算机科学与技术领域广泛使用的术语。它用来反映一个数据的内部构成，即一个数据由哪些成分构成？以什么方式构成？呈现什么样的结构？数据结构有逻辑上的数据结构和物理上的数据结构之分。逻辑上的数据结构反映数据之间的逻辑关系，而物理上的数据结构反映数据在计算机内部的存储安排。数据结构是数据存在的形式。

数 据 结 构

数据结构(data structure)指的是数据之间的相互关系，即数据的组织形式。

1.2.2 基本概念

在系统地学习数据结构之前，首先来了解本课程的基本概念及相关术语的确切含义。

数据(data)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并被计算机程序识别和处理的符号的集合。数据是计算机程序加工的“原料”，共分为两类：数值型数据(主要用于数学计算等)和非数值型数据(文字、图形、图像、音频和视频等)。

数据元素(data element)是数据的基本单位。在不同的条件下，数据元素又可称为元素、结点、顶点、记录等，如八皇后问题中状态树的一个状态。一个数据元素可由不可分割的若干个**数据项(data item)**组成。例如，在图书馆管理系统中，通常将每本书的信息存储在一个表中，这样一本书的书目信息就为一个数据元素。而书目信息中的每一项，如书名、作者等就称为数据项。数据项是数据不可分割的最小单位。

数据对象(data object)是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合 $\mathbf{Z}=\{0, \pm1, \pm2, \cdots\}$，字母字符数据对象是集合 $C=\{\text{'A'}, \text{'B'}, \cdots, \text{'Z'}\}$。

数据结构(data structure)是指相互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，数据元素之间总是存在联系的。把某一数据对象及该数据对象中所有数据成员之间的关系组成的实体叫作数据结构。数据结构有以下 4 种基本结构。

(1) 集合结构：数据元素之间存在着“属于同一个集合”的关系，如图 1.2(a)所示。

(2) 线性结构：数据元素之间存在着“一对一”的关系，如图 1.2(b)所示。

(3) 树形结构：数据元素之间存在着“一对多”的关系，如图 1.2(c)所示。

(4) 图形结构：数据元素之间存在着“多对多”的关系，如图 1.2(d)所示。

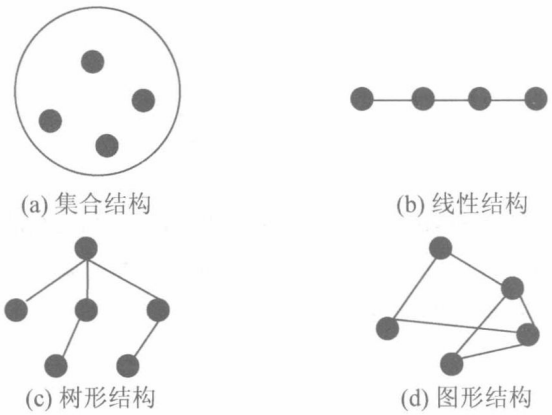


图 1.2　4 类基本数据结构示意图

数据结构的形式定义为

$$\text{Data\_Structure} = (D, R)$$

其中：D 是数据元素的有限集；R 是 D 上关系的有限集。

例 1.3　定义集合 $D = \{3, 6, 9, 18, 27\}$的数据结构。

$DS_1=(D, R_1)$，其中 R_1 定义为 D 上的“>”(大于)关系，则数据结构 DS_1 可以表示为如图 1.3(a)所示的形式。$DS_2=(D, R_2)$，其中 R_2 定义为 D 上的“整除”关系，则 $R_2=\{(3, 3), (3, 6), (3, 9), (3, 18), (3, 27), (6, 18), (9, 18), (9, 27)\}$，数据结构 DS_2 可以表示为如图 1.3(b)所示的形式。

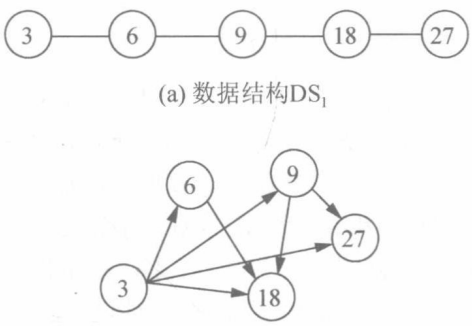


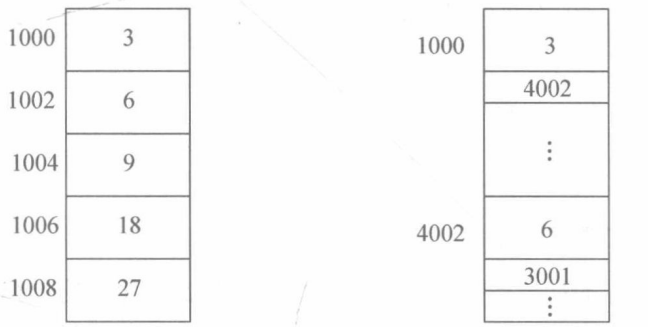
图 1.3　集合 D 上定义的两个数据结构

从上面的例子可以看出，即使是由相同元素构成的集合，只要定义的关系不同，也不是同一数据结构。数据结构不仅描述了结构中的元素，还描述了这些元素之间的关系。数据结构的定义仅是对操作对象的一种数学描述，结构中定义的关系是数据元素之间的逻辑关系。

前面已经提到过数据结构可以分为逻辑上的数据结构和物理上的数据结构。数据结构的形式化定义为逻辑结构；物理结构为数据在计算机中的表示，它包括数据元素的表示和关系表示。学过计算机程序设计语言的人都知道，在计算机中表示信息的最小单位是二进制的一位，可以用一个由若干个位组合起来形成的一个位串表示一个数据元素。因此可将

这些位串看成数据元素在计算机中的存储形式。

数据元素之间的关系在计算机中有两种不同的表示方法：顺序存储和非顺序存储。因此，就可以得到两种不同的存储结构，即顺序存储结构和链式存储结构。顺序存储结构是把逻辑上相邻的元素存储在物理位置相邻的两个单元中，它是一种最基本的存储方法，一般采用数组来实现。链式存储结构对逻辑上相邻的元素不要求其物理位置也相邻，元素间的逻辑关系通过指针来表示，一般采用链表来实现。图 1.4 给出了图 1.3 中数据结构 DS_1 的不同存储方式。



(a) 顺序存储结构　　(b) 链式存储结构

图 1.4　数据结构 DS_1 的存储结构示意图

➢ 自测题

3. 什么是数据结构？

4. 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

5. 设有数据逻辑结构为

$B=(K, R)$

$K=(k_1, k_2, \cdots, k_9)$

$R=\{<k_1, k_2>, <k_1, k_8>, <k_2, k_3>, <k_2, k_4>, <k_2, k_5>, <k_3, k_9>, <k_5, k_6>, <k_8, k_9>, <k_9, k_7>\}$

试画出这个逻辑结构的图示。

1.3　抽象数据类型

It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.

John von Neumann, circa 1949

抽象数据类型(Abstract Data Type，ADT)是指一个数学模型以及定义在此数学模型上的一组操作。抽象数据类型可以使人们更容易描述现实世界。使用它的人可以只关心它的逻辑特征，而不需要了解它的存储方式。定义它的人同样不必关心它如何存储。

抽象数据类型由元素、关系及操作 3 种要素来定义。抽象数据类型用三元组来表示：

$$(D, R, P)$$

其中：D 是数据对象；R 是 D 上的关系集；P 是对 D 的基本操作集。

抽象数据类型名称定义的一般形式如下：

```
ADT 抽象数据类型名称{
    数据对象：
        ⋮
    数据关系：
        ⋮
    操作集合：
        操作名 1;
            ⋮
            ⋮
        操作名 n;
}ADT 抽象数据类型名称
```

例如，线性表这样的抽象数据类型，其数学模型是数据元素的集合，该集合内的元素有这样的关系：除第一个和最后一个外，每个元素有唯一的前趋和唯一的后继。可以有这样的一些操作：插入一个元素、删除一个元素。那么线性表的抽象数据类型就可以定义为：

```
ADT list
{
    数据对象：任意数据元素的集合
    数据关系：除第一个和最后一个外，每个元素有唯一的直接前驱和唯一的直接后继
    基本操作：
        ListInsert(&L,i,e);          //元素的插入
        ListDelete(&L,i,e);          //元素的删除
        ⋮
}ADT list
```

通过以上定义可以看出，抽象数据类型只是数学的抽象，在 ADT 的定义中没有涉及如何实现操作的集合。对于每个 ADT，并不存在说明必须要有哪些操作的法则，这只是一个设计决策。本书还会在后续的章节中详细讨论不同数据结构的 ADT。

➢ **自测题**

6. 什么是抽象数据类型？

1.4 算　　法

The function of good software is to make the complex appear to be simple.

Grady Booch(1955—)

1.4.1 算法概述

算法(Algorithm)是解题的步骤，是指令的有限序列。它们规定了解决某一特定类型问题的一系列运算，是对解题方案的准确与完整的描述。制定一个算法，一般要经过设计、确认、分析、编码、测试、调试和计时等阶段。

一个算法应该具有以下特征：

(1) 有穷性。对于任何合法的输入值，一个算法必须保证执行有限步之后结束。

(2) 确定性。算法的每一步必须有确切的含义，无二义性，并且在任何条件下，算法只有唯一的一条执行路径，即对相同的输入只能得出相同的输出。

(3) 输入。一个算法有零个或多个输入，以刻画运算对象的初始情况，所谓零个输入是指算法本身设定了初始条件。

(4) 输出。一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

(5) 可行性。算法原则上能够精确地运行，即算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现。

对算法的学习包括下面5个方面的内容：

(1) 设计算法。设计一个好的算法，通常要考虑正确性、可读性、健壮性、高效性这几个方面的要求。正确性是指算法的执行结果应当满足预先规定的功能和性能要求。可读性是指算法应该思路清晰、层次分明、简单明了、容易阅读和理解。健壮性是指算法在接收不合理的输入时，应该能进行适当的处理，不至于引起不好的后果。高效性是指算法应使用相对较少的额外存储空间和有较高的时间效率。要想设计一个好的算法，首先需要对有待解决的问题有一个彻底、深入的分析，还需要多学习和了解一些已经被实践证明是有用的一些基本的算法设计方法，这些算法不但会涉及计算机科学，还会涉及其他的领域。

(2) 描述算法。描述算法的方法有多种形式，例如自然语言和算法语言，各自有适用的环境和特点。本书采用类C语言来描述算法，类C语言是伪码和C语言组合而成的一个描述工具，采用了C语言的核心部分，并为描述方便进行了扩充。

(3) 确认算法。算法确认的目的是使人们确信这一算法能够正确无误地工作，即该算法具有可计算性。正确的算法是用计算机算法语言构成计算机程序，计算机程序在计算机上运行，得到算法运算的结果。

(4) 分析算法。算法分析是对一个算法需要多少计算时间和存储空间做定量分析。分析算法可以预测这一算法适合在什么样的环境中有效地运行，对解决同一问题的不同算法的有效性做出比较。

(5) 验证算法。用计算机语言描述的算法是否可计算、有效合理，须对程序进行测试，测试程序的工作包括调试程序和对算法运行所需时间和空间进行分析。

算　法

算法是一系列准确的指令，一个问题的解决方案要以算法为基础。

1.4.2 算法与数据结构之间的关系

著名的计算机科学家沃思(Nikiklaus Wirth)提出一个公式：

数据结构 + 算法 = 程序

这个公式说明了算法与数据结构的重要性，也说明了算法与数据结构之间存在着相辅相成的关系。解决一个问题可以选择不同的数据结构，也可以选择不同的算法。数据结构的选择决定着算法执行时所需要的时间与空间，影响着算法的效率。反之，算法的优劣又决定着某个数据结构是否适合解决这个问题。

例 1.4 求 3 个数中的最大者。

算法 1(不需要额外的空间)：

```
int Max1(int a[3])
{
   if(a[0]>a[1])
   {
     if(a[0]>a[2]) return a[0];
     else return a[2];
   }
   else
   {
     if(a[1]>a[2]) return a[1];
     else return a[2];
   }
}
```

算法 2(需要额外的空间)：

```
int Max2(int a[3])
{
   c=a[0];
   for(i=1;i<3;i++)
     if(a[i]>c) c=a[i];
   return c;
}
```

在例 1.4 中，算法 1 不需要额外的空间，而且只需要比较两次；算法 2 需要额外的空间，而且需要比较 3 次。对于问题规模不太大的情况，算法 1 比算法 2 好。但是当问题规模增加到 100 个整数时，算法 1 的程序立即就会变得很长，并且代码重复严重；算法 2 不会增加程序的长度，这时算法 2 明显比算法 1 好。

1.4.3 算法的度量

下面主要通过两个方面对算法进行度量。

1. 算法的时间复杂度

算法的时间复杂度是指运行算法时所需要消耗的时间资源。其定义是：如果一个问题的规模是 n，解决这一问题的某一算法所需要的时间是 n 的某一函数，记为 $T(n)$。$T(n)$称为算法的时间复杂度。当输入量 n 逐渐增大时，时间复杂度的极限情形称为算法的渐近时间

复杂度。渐近时间复杂度评估了函数的增长率，常用来表示算法的时间复杂度。

算法的时间复杂度比下面将要讲的空间复杂度更为重要，在考虑算法的效率时一般只关注处理数据所花费的时间。即使是效率最低的算法，在 Cray 机上运行时，也会比在一般的 PC 上运行的效率要高，所以运行时间通常与系统相关。因此要评估算法的效率，不能使用微秒(μs)或纳秒(ns)这样的实际时间单位，而应该采用某种逻辑单位，来描述文件或数组的大小 n 同处理数据所需的时间 t 之间的关系。

2. 算法的空间复杂度

算法的空间复杂度是指算法在计算机内执行时所需存储空间的度量。存储空间具体是指编写程序时，程序的存储空间、变量占用空间、系统堆栈的使用空间等。

例 1.5 将两个数交换的算法。

算法 1(需要额外的空间)：

```
void swap(int &a, int &b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

算法 2(不需要额外的空间)：

```
void swap(int &a, int &b)
{
    a = a + b;
    b = a - b;
    a = a - b;
}
```

算法 1 中交换 a 和 b 两个数时，用到了第 3 个变量 temp；而算法 2 没有用到第 3 个变量。如果需要执行 N 次 swap 函数，则算法 1 的空间复杂度为 $O(N)$，算法 2 的空间复杂度为 $O(1)$。

➢ 自测题

7. 什么是算法？
8. 算法应具有什么特征？
9. 算法的衡量标准或依据是什么？

1.5 算 法 分 析

1.5.1 数学基础

估计算法资源消耗所需的分析是一个理论问题，需要一套理论体系。下面首先从一些