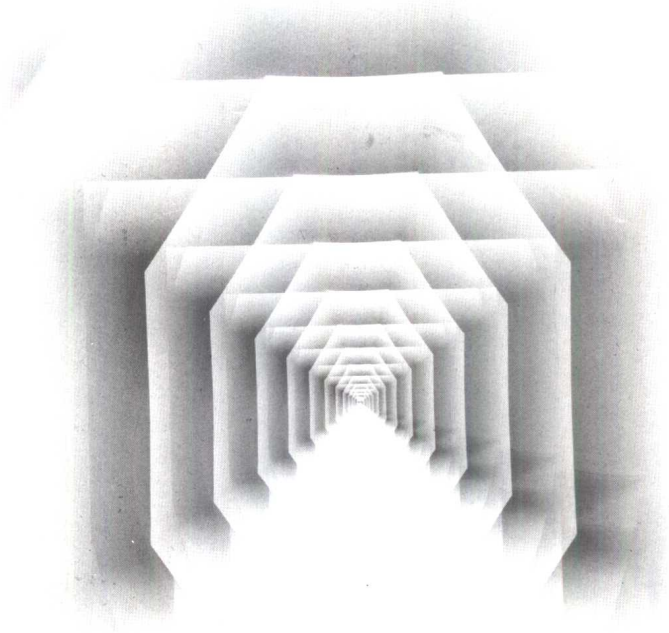
数据结构与算法

Shuju Jiegou Yu Suanfa

黄 定 黄煜廉 刘贤兴 编著
李吉桂 主审



广东科技出版社

数据结构与算法

黄定　黄煜廉　刘贤兴　编著
李吉桂　主审

广东科技出版社
·广　州·

图书在版编目（CIP）数据

数据结构与算法/黄定，黄煜廉，刘贤兴编著．—广州：广东科技出版社，2000．1
ISBN 7－5359－2432－8

Ⅰ．数…　Ⅱ．①黄…②黄…③刘…　Ⅲ．①数据结构②算法理论　Ⅳ．TP311.12

中国版本图书馆 CIP 数据核字（1999）第 56855 号

出版发行：广东科技出版社
（广州市环市东路水荫路 11 号　邮码：510075）
E－mail：　gdkjzbb@21cn．com
出 版 人：黄达全
经　　销：广东省新华书店
排　　版：广东科电有限公司
印　　刷：广东肇庆新华印刷有限公司
（广东肇庆市郊狮岗　邮码：526060）
规　　格：787mm×1 092mm　1/16　印张 18.75　字数 410 千
版　　次：2000 年 1 月第 1 版
2000 年 1 月第 1 次印刷
印　　数：1～6 200 册
定　　价：30.00 元

内 容 简 介

本书覆盖了数据结构和算法两门计算机软件主干课程的教学内容，包括数据结构和算法的基本概念，抽象数据类型，四种逻辑结构（集合与文件、线性结构、树、图）的定义，存储结构（顺序、链接、散列、索引）的实现，基本运算及常用算法（查找、排序等）的实现，并结合算法讲解，总结了算法设计方法和算法分析技术，介绍可计算性理论的一般知识。

本书采用目前最广泛使用的C++语言描述所有抽象数据类型及算法，强调了数据结构中数据抽象和信息隐藏的概念，体现了面向对象程序设计思想。

本书内容丰富，包含CAI教学软件和远程教学系统，并将配套出版相应的习题集、实验指导，因而特别适合作为《数据结构》和《算法设计与分析》课程的教材，也可作为研究生、大学生和相关教师学习数据结构和算法设计与分析的参考资料。

本书可作为大专院校计算机科学技术专业或相关专业《数据结构》和《算法设计与分析》课程的教材。

前　言

数据结构和算法设计与分析（以下简称数据结构与算法）是计算机专业的重要基础课程。计算机科学各个领域及各种应用软件都要使用相关的数据结构和算法。

本书覆盖了原教育部颁布的大纲及计算机全国教育学会推荐大纲的数据结构课程的内容，以及算法设计与分析的基本内容。为使这两部分内容能有机融合在一本教材中，在第一章的引论中介绍常用的非数值算法设计的基本思想、方法和策略，为以后各章内容的展开做准备；各类算法的设计和应用分散在第二章至第九章；关于算法分析的内容，在引论中介绍了算法分析的基础知识，在第十一章介绍了算法分析的技术和各类算法分析实例。

教材在传统的数据结构中引入抽象数据类型的概念，把数据的逻辑结构及运算定义为抽象数据类型，将数据抽象与过程抽象结合为一体，使用时把抽象数据类型看成一个普通的初等类型引用。引入抽象数据类型可提高读者的抽象能力和编程水平。

教材中数据结构和算法的描述均采用面向对象的C++语言，利用了C++语言隐蔽实现细节的特性，例如类、私有成员、构造函数、析构函数等。该特性能充分体现抽象数据类型中逻辑结构与存储实现的分离，从而更好地体现了抽象数据类型的概念，更本质地描述数据结构和算法。

本书内容分四部分。第一部分是引论，介绍了抽象数据类型和数据结构的基本概念和术语，以及算法的概念、常用非数值算法的设计方法、算法分析的基础知识。第二部分为第二章至第八章的部分内容以及第十章，介绍了基本的数据结构，依次为线性表、栈和队列、串、数组和广义表、树、图、集合和文件。每种数据结构都从数学特性入手，先介绍抽象数据类型，然后讨论不同存储实现方法以及基于不同存储实现的各种算法。第三部分包括第八章部分内容和第九章，介绍了各种查找和排序算法。这两个内容是数据结构的重要算法。第四部分是第十一章和第十二章。第十一章介绍了算法分析技术和算法的多项式时间可计算性。算法分析技术讨论了递归关系的分析和递归方程的求解技术，并分析了数据结构中一些典型算法的复杂度。第十二章简明介绍了是否存在非多项式可计算性问题的有关理论、P和NP的概念、NP完全性、COOK定理，并给出一批NP完全性问题的实例。

本书在内容取舍上力求体现计算机科学的发展，尽量体现目前流行的观点。例如在时空效率与算法可读性的权衡上更注重后者。

本书主要面向师范院校，因此，在如何体现师范院校的特点，即“师范性”方面进行了大胆的尝试。概括说来，师范性主要体现在以下几方面：本书内容是对教育部推荐的中学计算机学科大纲规定内容的指导、深化和提高；本书覆盖和深入讨论了非计算机专业本、专科计算机学科的教学内容，强调编程实践和上机指导；在教学过程中强调教学方法的改革，为进行网上教学提供了丰富的参考资料。

本书配有CAI教学软件和网上教学系统，并将配套出版习题集和实验指导。

本书的编写是在李吉桂教授的具体指导下进行的，他对全书进行了认真的审阅。李冠英教授和陈卫东同志对初稿提出了宝贵意见。作者在此表示诚挚的谢意。

本书的第一、十一、十二章由刘贤兴同志执笔；第二、三、四、六、九章由黄煜廉同志

执笔；第五、七、八、十章由黄定同志执笔。黄定同志对全书做了统编和修订工作。

由于编者水平有限，书中难免存在误漏。殷切希望读者给予批评指正。

黄定　黄煜廉　刘贤兴

1999 年 12 月 16 日

目　录

第一章　引　　论

内容提要本章的主要内容有：抽象数据类型；数据结构与数据类型的概念及其相互关系；数据的逻辑结构及其分类；数据的存储表示；算法设计的基本方法；算法分析的基础知识。本章内容是全书的导引。

第一节　抽象数据类型

本节介绍程序设计语言发展的过程，计算的抽象，数据抽象、数据类型、抽象数据类型的概念。

一、程序设计的一个基本原则是抽象

抽象是对一个系统的简化描述，它强调了系统的某些特性，而把其他性质**隐蔽**起来。对用户来说，所关心的是程序能做什么，应把系统的设想实现隐蔽起来。**程序设计语言的发展过程就是程序的抽象层次不断提高的过程**。程序中的计算和操作是按下列阶段发展的：二进制代码与机器指令→符号化的汇编语句→高级语言的语句→高级语言的子过程（过程、函数）与调用。这里的每一层次都是一种抽象，并且后一层次是在前一层次的基础上经过进一步的抽象建立起来的。

与计算和操作的抽象类似，计算的处理对象也随程序设计语言的发展，不断抽象、发展。其发展阶段为：

①机器语言的二进制序列，是非类型化的0，1串。

②汇编语言引入整数的表示。整型数据经过汇编系统的处理转换成二进制，这些处理细节被隐蔽了。

③过程语言中引进了数据类型，把非类型的0，1串分解成为若干子集，同一子集中每个成分有相同的特性行为，构成了不同的类型。不仅引入了整型、实型、布尔型、字符型等基本数据类型，而且引入了数组类型、记录类型等结构类型。

程序设计语言中的**数据从无类型发展到带有类型**，而且类型的结构的丰富程度已成为语言优劣的评价标准之一。还要指出的是，**数据抽象的发展和计算的抽象的发展是相互促进的**。

二、抽象数据类型

在程序设计语言中，数据类型是一个常用的主要概念。前面已指出，一个**数据类型**是若干对象和在这些对象上可以执行的各种操作的集合体，即一个数据类型给出了一个值集（对象集）和一个作用于值集的操作集合的集合体。如整数类型，它的值集是…，－3，－2，－1，0，1，2，…，操作集为＋，－，＊，div，mod，＞，＝，＜＞等。

程序设计语言提供了预定义的数据类型供用户使用，并且提供用户自定义所需类型的构造新类型的能力。这样可使程序设计的抽象原则更上一个层次，也因此引入抽象数据类型的

概念

抽象数据类型（Abstract Data Type，简称 ADT）是用户用以下方式构造的数据类型：定义一个数据逻辑模型（数学模型），以及在其上定义一组操作。每个操作由它的输入、输出定义

一个 ADT 的定义并不涉及它的实现细节，这些 ADT 的实现细节对于 ADT 的使用者是隐蔽的 **隐蔽实现细节的过程称为封装。**ADT 的实现包括首先给出数据的物理存储方式，然后通过子程序实现 ADT 的每一个操作。

我们始终强调 ADT 的定义与其实现之间的区别，对每一种数据对象，首先给出 ADT 定义及解释，然后再进入 ADT 的实现的讨论。ADT 的描述采用 C++ 的语法规则来表示。ADT 的实现也用 C++ 语言实现

例 1 **抽象数据类型**

ADT NaturalNumber is

objects: An ordered subrange of the integers starting at zero and ending at the maximum integer (MAXINT) on the computer.

functions:

for all x, y∈NaturalNumber; TRUE, FALSE∈Boolean and where +, −, <, ==, and = are the usual integer operations

```
Zero( ): NaturalNumber          :: = 0
IsZero(x): Boolean              :: = if(x == 0) IsZero = TRUE
                                   else IsZero = FALSE
Add(x, y): NaturalNumber        :: = if(x + y≤MAXINT) Add = x + y
                                   else Add = MAXINT
Equal(x, y): Boolean            :: = if(x == y) Equal = TRUE
                                   else Equal = FALSE
Successor(x): NaturalNumber     :: = if(x == MAXINT) Successor = x
                                   else Successor = x + 1
Subtract(x, y): NatrualNumber   :: = if(x < y) Subtract = 0
                                   else Subtract = x − y
```

end NaturalNumber

上例是对自然数的 ADT 定义。第一行是抽象数据类型的名称。定义分两个部分：objects 和 functions。objects 定义为非负整数，但并不给出明确的表达。functions 的定义稍复杂一些。定义中的符号 x、y 表示 NaturalNumber 的元素，而 TRUE 与 FALSE 是 Boolean 集的两个元素。另外，在定义中使用了整数集上的操作，即 +、−、==、<、= 等，因为定义一种 ADT 需要引用其他类型的操作。符号“:: =”读作“定义为”。对于每个函数，其返回类型放在“:: =”符号的左边，而函数的定义则放在该符号的右边。

第二节 数 据 结 构

本节先介绍基本术语，然后介绍数据的逻辑结构、存储结构、数据结构、数据的运算等内容

一、基本术语

数据：数据是信息的载体。能够被计算机识别、存储和加工处理的对象通称为数据。它是计算机程序加工的“原料”。

数据元素：数据元素是组成数据的基本单位。在有些情况下，数据元素简称为元素、结点、顶点、记录。

数据项：组成数据元素的一个成分。数据项是具有独立含义的最小标识单位。

数据对象：是性质相同的数据元素的集合，是数据的一个子集。

二、数据的逻辑结构

在实际问题中数据元素不是孤立存在的，而是存在一定的关系。数据以及数据元素之间的相互关系称为数据的逻辑结构。此逻辑关系又称为元素间的结构。

数据的逻辑结构可表示为：$B=(K, R)$，其中 K 是结点（数据元素）的有穷集合，R 是 K 上的一个关系。

例 2　设 $K=\{k_1, k_2, k_3, k_4, k_5, k_6\}$，

$R_1=\{\langle k_1, k_2\rangle, \langle k_2, k_3\rangle, \langle k_3, k_4\rangle, \langle k_4, k_5\rangle, \langle k_5, k_6\rangle\}$，

$R_2=\{\langle k_1, k_2\rangle, \langle k_1, k_3\rangle, \langle k_2, k_4\rangle, \langle k_3, k_5\rangle, \langle k_3, k_6\rangle\}$，

$R_3=\{\langle k_1, k_2\rangle, \langle k_2, k_3\rangle, \langle k_2, k_4\rangle, \langle k_3, k_5\rangle, \langle k_4, k_5\rangle, \langle k_5, k_6\rangle\}$。

相应的三个逻辑结构是 $B_1=(K, R_1)$，$B_2=(K, R_2)$，$B_3=(K, R_3)$，如图 1-1、图 1-2、图 1-3 所示

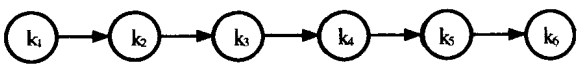


图 1-1　B_1 逻辑结构图示

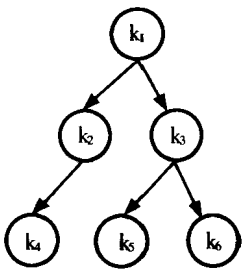


图 1-2　B_2 逻辑结构图示

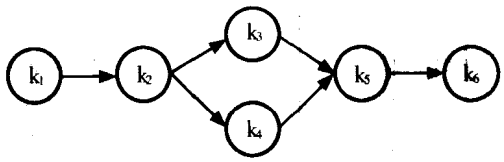


图 1-3　B_3 逻辑结构图示

对上述图示引入如下术语：设 $B=(K, R)$ 是一个逻辑结构，若 k、k'$\in$K 且 $<k, k'>\in R$，则称 k 是 k'的**前趋**，k'是 k 的**后继**，k 与 k'是**相邻**的。如 k'在 R 中无前趋结点，则称 k'为关于 R 的**开始结点**（无前趋结点）；如果 k 在 R 中无后继，则称 k 为**无后继结点**，也称 k 关于 R 为**终端结点**。如果 k 既不是开始结点又不是终端结点，则称 k 为**内部结点**。

根据 R 的特性，数据的逻辑结构可分为四类：集合结构（$R=\phi$）、线性结构（如 B_1）、树型结构（如 B_2）、图型结构（如 B_3）。

三、数据的存储结构

数据的逻辑结构是操作对象的数学描述，是用户对操作对象抽象得到的**数学模型，**是用户面前呈现的方式，因而它与数据的存储无关，是面向问题而独立于计算机的。而要实现在计算机中处理对象，必须首先把数据存储在计算机中。

数据的逻辑结构在计算机中的表示（映象）称为数据的存储结构，又称物理结构。

设有逻辑结构 B＝（K，R），要把 B 存储在计算机中，首先要建立一个从 K 到内存 M 的映象 f：K→M，使得：

①对每一个 $k\in K$，有唯一的 $z\in M$，使 $f(k)=z$。z 为存储 k 的变量。

②对每一个 $<k_1, k_2>\in R$，$f(k_1)=z_1$，$f(k_2)=z_2$，$<z_1, z_2>\in R$。

从而存储结构可表述为 B＝（M，R）。因而，存储结构是用各种方式联系在一起的变量的集合。

数据的存储结构是计算机中表示的数据的逻辑结构。数据的逻辑结构、存储结构是描述数据的两种形式，ADT 给出的数据定义是它的**逻辑形式，**而 ADT 的实现所选择的数据结构指的是对数据的实现的**物理形式。**因而，常把数据的逻辑结构、存储结构统称为**数据结构。**

数据的存储结构有四种基本的映象方法：

（1）顺序的方法

该方法把逻辑上相邻的结点存储在物理位置上相邻的存储单元里。结点的逻辑关系由存储单元的邻接关系来实现。这种存储方法称为顺序存储结构，在高级语言中用一维数组来描述。

（2）链接的方法

这个方法不要求逻辑上相邻的结点在物理上也相邻。结点间的关系是由附加指针字段来表示的。由此达到的存储表示称为链接存储结构，在高级语言中用指针类型来描述。

（3）索引的方法

索引的方法是建立索引表，索引表里第 i 项的值是第 i 个结点的地址，或用索引号 i 的函数值指出结点的存储地址。其基本思想是通过结点的索引号来确定结点的存储地址。

（4）散列的方法

散列方法的基本思想是根据结点的关键字的值直接计算出该结点的存储地址。

一般数据结构的存储映象都是四种基本映象之一，或是它们的组合。同一个逻辑结构可以有几种不同的映象方法。选择哪一种要根据运算的方便及算法的时间、空间要求来确定。

存储结构的描述是在高级语言的层次上给出的。用高级语言的“数据类型”给出存储结构的定义，如用一维数组来描述顺序存储结构，指针类型定义链式存储结构等。

第三节　算法的概念

在计算机科学中，算法的概念是基本的且是至关重要的。计算机是执行算法的机器，许多问题的解决途径是基于算法的。在软件系统的开发中，设计出有效的算法将起到重大的作用。本节将介绍算法的概念、算法与程序的关系和算法的描述等内容。

算法的意义十分类似于规程。算法是用于执行一个特定的任务或求解一个特定类型问题

的一个指令序列，该序列是有限的，每条指令都有一个明确的意义，每条指令的动作是可付诸实施的。这是一个非形式的定义。

D. E. Knuth 在他的专著中将算法定义如下：

一个**算法**，就是一个有穷规则（指令）的集合，它为某个特定类型问题提供了解决问题的运算序列，且具有五个特性：

(1) 有穷性

一个算法总是在执行有穷步之后结束。这并不是意味着在人们可容忍的时间内结束。有穷性只是说明算法要能结束。例如，利用下列公式计算 Π 值：

$\Pi = 4(1 - 1/3 + 1/5 - 1/7 + \cdots)$，精确到小数点后 15 位。这就是一个算法。如果不规定“精确到小数点后 15 位”，就不是一个算法，而只能是计算方法。

(2) 确定性

算法的每一个步骤必须是确切定义的。对于每一种情况，执行的动作必须严格地和无歧义地规定。例如，选择语句是任何算法描述语言的组成部分，它允许对下一步语句进行选择，但选择过程必须是确定的、无歧义的。

(3) 输入

一个算法有 0 个或多个输入。在算法开始时，一般要给出运算的初始数据。这里的 0 个输入指在算法内部给出了初始数据，不需要从算法外部输入初始数据。

(4) 输出

一个算法有一个或多个输出，没有输出的算法是没有意义的。输出与输入有特定的关系，是算法对输入进行加工的结果。因而算法可比拟为一个函数 f，f（输入）= 输出。

(5) 可行性

算法中所定义的每个动作都必须是相当基本的，能够精确地付诸实施，而且人们用笔和纸做有穷次计算即可完成。

具有上述五个特征的指令序列才能称为算法。任何一个算法都可用高级程序设计语言表述为一个程序。但是，一个程序不满足有穷性时，它就不是一个算法。一个典型的例子是，操作系统是一个程序，可以管理作业的执行。在没有作业运行时，它并不终止，而是处于等待状态，一直等到新的作业进入。因而操作系统程序不是算法。算法和程序是两个独立的不同的概念。但本书中所出现的程序都是可终止的，故在本书中对这两个概念不作严格区分。

算法的描述通常有三种形式：

①用自然语言描述算法的执行步骤；

②用流程图表示算法的步骤；

③用程序设计语言表示算法。

例 3　欧几里得算法：求两个正整数 n、m 的最大公约数。欧几里得算法在 50 年代初，经常与算法一词联系在一起，是一个经典的例子。

用自然语言描述的指令系列为：

E1［求余数］以 n 除 m 并令 r 为所得余数（$0 \leqslant r < n$）

E2［余数为 0?］若 r == 0，算法结束，n 即为答案，否则执行 E3

E3 令 m = n，n = r，并接着执行 E1

采用流程图表示欧几里得算法，得到下面的框图：