青少年信息学奥林匹克竞赛培训教材

数据结构与算法设计

——Pascal 语言

（第2版）

张文双　王学红　郭连凤　主编

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

青少年信息学奥林匹克竞赛培训教材

数据结构与算法设计

——Pascal 语言

（第 2 版）

主编　张文双　王学红　郭连凤

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内 容 简 介

本书按照全国青少年信息学奥林匹克竞赛要求编写，立足于 Free Pascal 程序设计语言的普及和应用。

本书包含数据结构与算法的引入、队列、栈、树、图、数据结构的综合应用、排列和组合、高精度计算、排序法、搜索策略、分治策略、贪心法、动态规划和算法的综合应用等内容，各章配备 A、B 两级习题，并附习题参考答案。

本书结构严谨，语言简练，可以作为中小学校信息学奥赛的培训用书，也适合读者选作自学资料。

图书在版编目（CIP）数据

数据结构与算法设计：Pascal 语言/张文双，王学红，郭连凤主编. —2 版. 北京：北京理工大学出版社，2010.12

青少年信息学奥林匹克竞赛培训教材

ISBN 978-7-5640-0743-0

Ⅰ. ①数…　Ⅱ. ①张… ②王… ③郭…　Ⅲ. ①数据结构-技术培训-教材②电子计算机-算法设计-技术培训-教材③PASCAL 语言-程序设计-技术培训-教材　Ⅳ. ①TP311.12②TP312

中国版本图书馆 CIP 数据核字（2010）第 062671 号

出版发行／北京理工大学出版社

社　　址／北京市海淀区中关村南大街 5 号

邮　　编／100081

电　　话／(010)68914775(办公室)　68944990(批销中心)　68911084(读者服务部)

网　　址／http：// www.bitpress.com.cn

经　　销／全国各地新华书店

印　　刷／天津紫阳印刷有限公司

开　　本／787 毫米×1092 毫米　1/16

印　　张／18.5

字　　数／434 千字

版　　次／2010 年 12 月第 2 版　2010 年 12 月第 6 次印刷

印　　数／13001～17000 册　　责任校对／陈玉梅

定　　价／32.00 元　　责任印制／边心超

编写委员会

主　编：张文双　王学红　郭连凤

编　委：侯启明　战久成　李　文

王　宇　杨　伟　安文君

周　敏　王亚平　刘萍萍

第 2 版前言

在联合国教科文组织的倡导下，自 1989 年至今国际信息学奥林匹克学科竞赛（IOI）已经举办了 21 届。在世界各国青少年优秀选手竞展雄姿的舞台上，中国代表队战绩辉煌。

与 IOI 同步的全国青少年信息学奥林匹克分区联赛（NOIP）的开展，提高了我国青少年的科学素养，促进了信息科技活动的普及，选拔出了大量的计算机拔尖人才，受到了众多信息学爱好者的关注。

目前在竞赛中多数选手选用 Pascal 语言。Pascal 语言功能强大，数据类型丰富，程序结构严谨，便于阅读和理解。应用 Pascal 语言程序设计求解问题，核心是数据结构和算法的整合。因此，系统地研究数据结构和算法，将会使选手的编程技能如虎添翼。

在目前的图书市场上，有关 Pascal 语言数据结构和算法的竞赛辅导教材极少，而且其中一些是写给大学生的，不适合中小学生阅读。为了帮助中小学生学习数据结构和算法知识，特聘请具有丰富竞赛辅导经验的一线教师和曾在国际信息学奥赛中获得金牌的优秀选手共同编写了这本书。本书是 Pascal 语言（小学版）和 Pascal 语言（中学版）的后继教材，内容紧扣信息学竞赛大纲，结构严谨，语言简练。希望本书能为提高读者竞赛技艺奉献绵薄之力。

本书第 1 版自 2006 年出版至今，受到广大读者的关注和厚爱，在此深表谢意。

近年来，Free Pascal 语言已替代 Turbo Pascal 语言成为我国青少年信息学奥林匹克竞赛（NOI）和分区联赛（NOIP）的复赛语言之一。为了适应竞赛的需要，我们对书中内容进行了修订。第 2 版中增加了队列、栈、数据结构的综合应用和贪心法 4 章，所有的例题和习题均能在 Free Pascal 环境中运行。

这本书的内容共分 14 章，主要内容包括：数据结构与算法的引入、队列、栈、树、图、数据结构的综合应用、排列和组合、高精度计算、排序法、搜索策略、分治策略、贪心法、动态规划和算法的综合应用等。具体编写分工如下：第 1、6 章由李文编写，第 2 章由安文君编写，第 3 章由刘萍萍编写，第 4 章由张文双编写，第 5 章由王学红编写，第 7 章由战久成编写，第 8、13 章由郭连凤编写，第 9 章由周敏编写，第 10 章由王宇编写，第 11 章由杨伟编写，第 12 章由王亚平编写，第 14 章由侯启明编写。全书由张文双统稿审定。

由于编者的水平有限，新版中若有疏漏之处，恳请各位读者指正。

编　者

目　录

第1章　数据结构与算法的引入

“程序=数据结构+算法”，这是著名计算机科学家 N. Wirth 提出的程序公式。从中可以看出，编程解决各种各样的实际问题，只学习掌握好诸如 Pascal、C 等程序设计语言是远远不够的。另外还必须具备数据结构和算法的相关知识。前者主要讲述客观世界中纷繁复杂的事物及其内在的联系如何有效地在计算机中进行表示，后者主要讲解建立在这种表示基础上的一些常用的解决实际问题的思想、方法和步骤。二者是一个相辅相成的统一体，是缺一不可的两个方面。有了它们以后，才能比较顺利地编写程序。

比如针对表 1-1 的学生档案，如果想编写一个优雅而高效的程序，让计算机帮助我们迅速、准确地从中查找是否有某个人，此人的性别、民族、语文成绩是什么，或求出所有学生的总成绩，或统计少数民族的学生人数，或按总成绩排列顺序等，需要做的事情主要有以下 3 点：

① 把表中数据及数据之间的关系转换成计算机所能处理的形式，输入到计算机中。

② 用适当的方式，完成数据及其关系在计算机中高效的存储，以方便问题求解和算法的优化。

③ 依据数据及其关系和它们在计算机中的存储形式构造一个高效的算法，完成问题求解。

前两点是选择适当的数据结构，第三点是设计算法。可见，所谓程序设计实质上就是数据结构的选用和算法设计这两者的有机组合。

表 1-1　学生档案

学号	姓名	性别	民族	语文	数学	英语	总成绩
101	张华	男	汉族	104.5	124	115	
102	郝博	男	汉族	113	125	94.5	
103	张天恩	男	苗族	97	139	87.5	
104	陈晓峰	男	回族	94	130	101	
105	李晨静	女	朝鲜族	100	99	134	
201	刘红梅	女	回族	132	117.5	75.5	
202	宋子荣	男	汉族	127	93	112	
203	王静	女	汉族	107	111	85.5	
301	孟晓立	男	苗族	112	74.5	133	
302	李立威	男	汉族	88	116	113	
303	白晓燕	女	朝鲜族	96	105	135	

1.1 数据结构的概念

1.1.1 数据结构的基本概念

为了方便讲述，先以表 1-1 的学生档案为例，介绍几个基本的概念和术语。

数据（data） 数据是计算机化的信息，是计算机处理的基本对象。凡是能输入到计算机中的描述客观事物的符号，都叫数据。数据的范围是非常广泛的，它不仅指整数、实数等数学中的数，其他诸如字符、声音、图形、图像、视频等也都是数据。表 1-1 中，学号、姓名、性别、民族等列的内容为字符型（字符串）数据，语文、数学、英语等列的内容为数值型（实型）数据。

数据元素（data element） 数据元素也叫结点（node）或记录（record），是数据的基本单位。表 1-1 中的一行就是一个数据元素。在实际处理中，一般把它看做一个整体来考虑和处理。

数据项（data item） 数据项是由一个或多个字符组成的具有独立含义的数据标识单位，是数据不可分割的最小单位。如表 1-1 中的“学号”、“姓名”等。若干个数据项综合起来构成一个数据元素。

数据对象（data object） 指性质相同的数据元素的集合，是数据的一个子集。比如表 1-1 中，所有的男生就构成了一个数据对象。

数据结构（data structure） 指相互之间存在一种或多种特定关系的数据元素的集合，数据之间的关系称作结构。数据结构包括数据的逻辑结构和数据的存储结构。

1.1.2 数据的逻辑结构

计算机所处理的数据一般都存在着某种内在的、特殊的关系，绝对孤立或杂乱无章的数据是不存在的。这种数据本身以及它们内在的相互之间的逻辑关系，叫做数据的逻辑结构，它是进行数据处理的依据，同时也是在计算机中对数据进行存储和加工处理的基础。

数据的逻辑结构可以表示成一个二元组：

$$B=(D, S)$$

其中 D 是由结点（数据元素）构成的集合；S 是关于 D 中结点之间关系的集合。二者都必须是有穷的，即两个集合中的元素都必须是有限多个，而不能是无限的。

D 集合中结点的数据类型，有初等类型和组合类型两种。只包含一个初等项的属于初等类型，包含多个初等项的属于组合类型。

1. 初等类型

初等类型实质上就是 Pascal 中的基本数据类型，具体包括：

① 整型。包括短整型（ShortInt）、整型（Integer）、长整型（LongInt）、字节型（Byte）等。

② 实型。包括单精度（Single）、实型（Real）、双精度（Double）、扩展型（Extented）等。

③ 布尔型（Boolean）。

④ 字符型（Char）。

⑤ 指针型（将在本书的第 2 章讲述）。

2. 组合类型

组合类型则是指 Pascal 中的构造类型，包括集合、数组、记录等。

S 集合中的关系是根据对实际问题的分析，并经过高度抽象化后所得到的结点之间的相互关系，通常分为图 1-1 所示的 4 类。

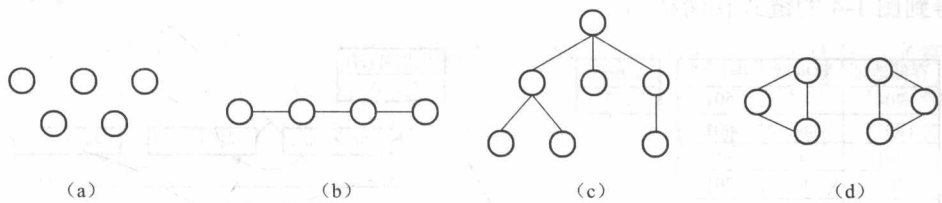

图 1-1

（a）集合结构；（b）线性结构；（c）树形结构；（d）图结构（网状结构）

① 集合结构。数据元素之间除了同属于一个集合的关系外，无任何其他关系。

② 线性结构。数据元素之间存在着一对一的线性关系。

③ 树形结构。数据元素之间存在着一对多的层次关系。

④ 图结构或网状结构。数据元素之间存在着多对多的任意关系。

数据结构既不同于数据类型，也不同于数据对象，它不仅要描述数据对象的数据类型，而且要描述数据对象各元素之间的相互关系。

1.1.3 数据的存储结构

存储结构也叫物理结构，是数据的逻辑结构在计算机中的存储方式。它不仅要实现数据元素本身的存储，还要实现数据之间逻辑关系的存储。方法主要有顺序存储和链式存储两种。

1. 顺序存储结构

顺序存储结构指借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系。该存储方式使得在逻辑上相互关联的数据元素，在物理存储结构中是相邻的。在 Pascal 中一般采用数组类型实现。

如对表 1-1 中的学生信息进行处理时，表中每个学生的信息就是一个数据元素（在表中占一行），其中包括学号、姓名、性别、民族、语文成绩等多个数据项。实际处理时，一般把表中的一行（一个学生的信息）抽象为一个结点，于是表 1-1 可抽象成（a_1，a_2，a_3，…，a_i，a_{i+1}，…，a_n），用 a_i 抽象地表示第 i 个学生的信息。数据元素 a_i 和 a_{i+1} 在逻辑关系上是相邻的，在计算机中可以把它们存储为图 1-2 所示的顺序存储方式。设元素 a_1 的存储地址为 s，每个元素占用 m 个存储单元（字节），则第 i 个数据元素 a_i 的存储位置 LOC(a_i)与第 i+1 个数据元素 a_{i+1} 的存储位置是相邻的，并且第 i 个数据元素 a_i 的存储位置 LOC(a_i)可按如下公式计算出来：

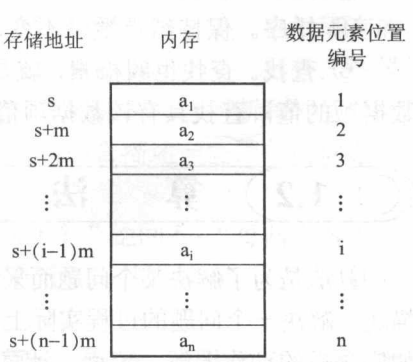

图 1-2 顺序存储结构

$$LOC(a_i) = LOC(a_1) + (i-1)m$$

2. 链式存储结构

借助元素存储地址的指针（pointer）表示数据元素之间逻辑关系的存储方式称为链式存储结构。链式存储结构使得在逻辑上相互关联的数据元素，在物理存储结构中不一定相邻，其逻辑关系用指针变量（一个或多个）指示，所以需要在数据元素中增加一个或多个指针域。

如对表 1-1 中的学生信息采用链式存储结构，假定有 6 个学生信息在内存中按图 1-3 存储，可得到图 1-4 的链式存储结构。

存储地址	数据域	指针域
201	a_1	501
301	a_2	601
401	a_3	nil
501	a_4	701
601	a_5	401
701	a_6	301

头指针
201

图 1-3　结点结构及存储地址

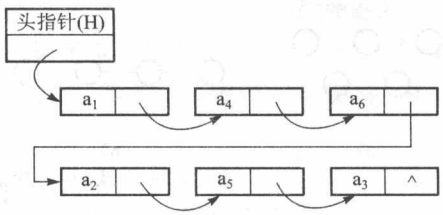

图 1-4　链式存储结构

实际应用中，究竟采用哪种存储结构，要根据具体情况进行细致分析，需要综合考虑哪种结构更易于理解和实现，可以得到更加高效的算法，能使算法和程序设计更加简单，以及计算机内存的限定等情况。

逻辑结构是数据元素之间相互关系的抽象表示，存储结构是数据元素本身及其内在关系在计算机中的存储方式，是在计算机中的具体实现，两者综合起来构成了数据元素之间的结构关系。

1.1.4　数据的运算

为了更加有效地处理数据，提高数据运算的效率，一般首先把数据按一定的逻辑结构组织起来，然后选择适当的存储方式把按逻辑结构组织好的数据存储到计算机中。数据的运算是定义在数据的逻辑结构上的，而运算的具体实现要在数据的存储结构上进行。数据的各种逻辑结构有相应的各种运算，每种逻辑结构都有一个运算的集合，其中常用的运算主要有：

① 插入。在数据结构中增加新的结点。

② 删除。把指定的结点从数据结构中删除掉。

③ 更新。改变指定结点的一个或多个数据项的值。

④ 排序。保持结点数量不变，把结点按照某种指定的顺序重新排列顺序。

⑤ 查找。查找也叫检索，就是在数据结构里查找满足一定条件的结点，一般是给定某个数据项的值，查找具有该数据项值的结点。

1.2　算　法

算法是为了解决某个问题而采用的一系列方法和步骤的总称，是关于问题解决方法的精确描述。解决一个问题的过程实际上也就是一个算法的实现过程。算法的含义是十分广泛的，诸如广播操的动作图解、菜谱、计算机的操作步骤等都可称作“算法”。但我们此处所说的算法是计算机的算法，即计算机可以实现的算法，比如我们让计算机计算 1+2 是可以的，也是我们

接下来要研究的，而让计算机完成“西红柿炒鸡蛋”的算法则显然不可行，至少目前尚不可行。

1.2.1 算法的特点

计算机算法大体可分为数值计算和非数值计算两大类。比如求若干个数之和、求方程的根、求n!等，都属于数值计算类，而图书检索、按考试成绩排列名次等则属于非数值计算类。无论哪一类算法，也无论算法多么简单或多么复杂，都必须满足以下特点：

1. 确定性

算法的每一步都必须是明确无误的，不能含糊其辞，不能存在歧义（具有二义性或多义性），否则就会使执行者无所适从。如在算法中出现“计算3/0”或“将3或4与x相乘”等是不允许的，原因是前者的计算结果不确定，后者则无法确定究竟是哪个数与x相乘。

2. 有效性

算法中的每一步运算都必须能够在计算机上有效地执行，整个算法执行完毕后必须得到确定的结果。例如求–5的平方根，求所有自然数的和，就无法在计算机上有效地执行。

3. 有穷性

一个算法必须包含有限个操作步骤，即执行若干步之后，算法能够终止，而不能无限地执行下去。当然，这种有穷性还应该在一个合理的范围之内，比如一个算法要执行10 000年才能得到结果，尽管它不是无限的，但显然没有什么实际意义。

4. 输入

一个算法可以含有0个或多个输入，用于提供算法执行所需要的外界信息。

5. 输出

设计算法的目的是得到问题的解，所以一个算法应包含一个或多个输出，用于描述问题求解后所得的结果。比如求100个数中最大的数，算法执行完毕应该输出最大的数。

1.2.2 算法的表示

描述算法的方法很多，常用的主要有自然语言、流程图、结构化流程图、PAD图、伪代码等。

1. 自然语言

用日常生活中使用的汉语、英语等自然语言来描述算法，优点是通俗易懂，缺点首先是比较烦琐冗长，经常需要一段很长的文字才能把操作说清楚，比如“在变量x原来的基础上增加10”，显然不如“x+10→x”简单明了；其次是容易产生“歧异”，这是由于自然语言的表达不一定十分严格，经常需要根据上下文理解和判断才能确定其正确的含义。另外，在描述顺序执行某一部分的步骤时，采用自然语言比较方便，但若是算法中包含判断和转移时，采用自然语言描述就显得比较麻烦和困难。所以人们一般用自然语言描述一些简单的问题，当问题较复杂时则采用其他的方法。

例1-1 一位妇女在河边洗碗，邻居问：“家里来了多少客人？”她回答：“每两个客人合用一个菜碗，每三个客人合用一个汤碗，每四个客人合用一个饭碗，共用碗65只。”问共来了多少客人？

算法分析：设客人的数量为x，显然x应是一个整数，菜碗、汤碗、饭碗这三种碗的数量也都必须是一个整数，所以根据题目条件，x必定是12的倍数。采用自然语言描述的算法

如下：

① x 赋初始值 0；

② x 在原来的基础上增加 12；

③ 求 x/2、x/3、x/4 这三个数的和（即菜碗、汤碗、饭碗的总数量），结果赋值给 y；

④ 如果 y 等于题目给定的碗的总数量 65，则输出客人数量 x，否则转到步骤（2）继续执行。

2. 流程图

流程图是采用图形方式来描述算法，事先规定不同形状的几何图形代表不同类型的操作。目前比较通用的是 ANSI（美国国家标准化协会）制订的一套流程图符号标准如图 1-5 所示。

例 1-1 的算法采用流程图描述，结果如图 1-6 所示。

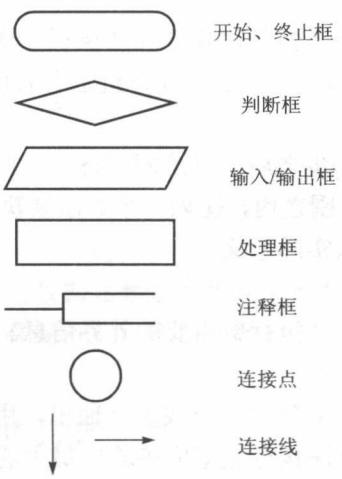

图 1-5　流程图符号标准

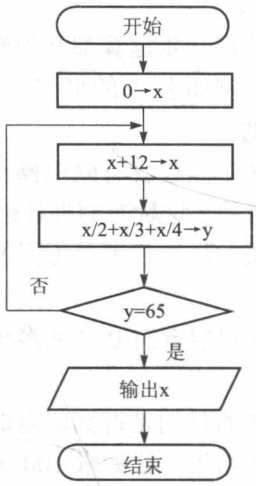

图 1-6　例 1-1 算法流程图

用流程图描述算法比较形象，且逻辑关系也清晰易懂，但流程图一般所占的篇幅较大。在算法较复杂时，每一步画一个框很费事，尤其是转向线较多时不仅不好安排，而且不容易读懂。

3. N-S 结构流程图

传统的流程图虽然直观、灵活，但是由于允许流程线指向任何一个框，即对流程的顺序无任何限制，使流程能任意转向，很容易造成逻辑思路的混乱，所表示的算法往往使人难以阅读理解和修改，降低了算法的可靠性和可维护性。为此经过研究，人们提出了一种全新的流程图——N-S 结构流程图。在这种形式的流程图中，规定了顺序、选择和循环三种基本的逻辑结构，如图 1-7 所示，并省略了基本结构之间的流程线。

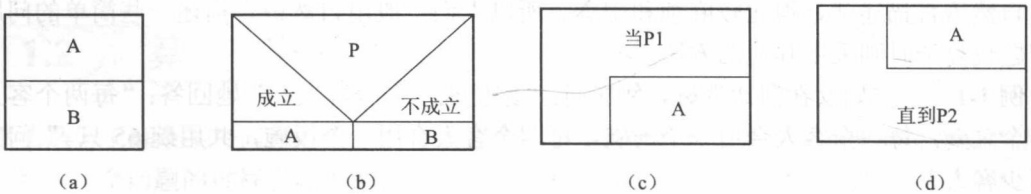

图 1-7　程序的逻辑结构

（a）顺序结构；（b）选择结构；（c）循环结构；（d）循环结构

在图 1-7 中，(a) 表示顺序结构，A 和 B 两个框组成一个顺序结构；(b) 表示选择结构，如果条件 P 成立则执行 A，若 P 不成立则执行 B；(c) 和 (d) 表示循环结构，(c) 表示 WHILE 循环，即当条件 P1 成立时反复执行 A，直到 P1 不成立为止，而 (d) 表示 UNTIL 循环（对应 Pascal 中的 REPEAT-UNTIL），反复执行 A，直到 P2 成立为止。

例 1-2 用键盘输入一个正整数 n，输出 1～n 之间所有的偶数。

算法分析： 设变量 x 的初始值为 1，当 x≤n 时，循环反复执行，如果 x 能被 2 整除，则输出 x，x 在原来的基础上增加 1。对应的 N-S 结构流程图为图 1-8。

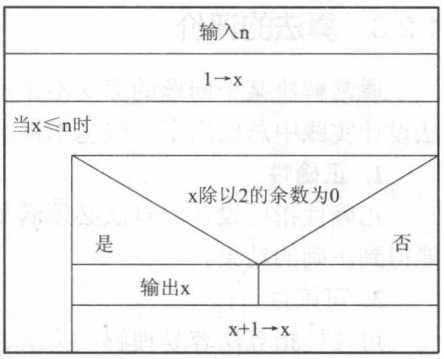

图 1-8 例 1-2 的 N-S 结构流程图

4. 伪代码

伪代码是采用一种介于自然语言和计算机语言之间的文字和符号来描述算法，其中的自然语言可以是英语、汉语等。例如求某数是否为负数，采用伪代码描述的算法如下：

```
if  x<0
  then  输出  '是'
  else  输出  '否'
```

采用伪代码描述的算法，其结构紧凑且易懂，写法比较灵活，没有严格的规则，只要能表达清楚算法含义即可。相比流程图，伪代码可随手写出，且容易修改。另外，伪代码的书写格式与高级程序设计语言（如 Pascal、C 等）非常相似，能很容易地转化为高级语言的源程序。

5. 类 Pascal 语言

这是一种采用伪代码描述算法的方式，它以标准 Pascal 中的基本语句为基础，在其中增加了若干不规则的语句。类 Pascal 语言的基本语句如表 1-2 所示。

表 1-2 类 Pascal 语言的基本语句

语句	表达方式	语句	表达方式
输入	READ	当循环	WHILE <条件表达式> DO S
输出	WRITE	记数循环	FOR-DO
赋值语句	变量←<表达式>	直到循环	REPEAT S UNTILE <条件表达式>
条件语句	IF <条件> THEN <语句> 或 IF <条件> THEN <语句 1> ELSE <语句 2>		

以上语句的使用方法与 Pascal 语言类似。

比如例 1-2 用类 Pascal 语言描述如下：

```
read(n)
x←1
```

```
while x<=n do
  if x mod 2=0 then write(x:5)
  x←x+1
```

在进行程序设计时，究竟采用哪种方法描述算法，可根据个人的习惯和实际需要来选择。

1.2.3 算法的评价

通常解决某个问题的算法不止一个，那么如何评价一个算法的优劣呢？人们在长期的算法设计实践中总结出了一些基本标准，一个“好”算法应满足以下要求：

1. 正确性

正确性指所设计的算法必须满足问题的要求，依据问题给定的条件，通过算法求解后，能得到正确的结果。

2. 可读性

可读性指算法容易理解，以方便人们的阅读和交流。晦涩难懂的算法不仅容易隐藏错误，而且还会给程序的调试和维护带来极大的困难和麻烦。

3. 健壮性

健壮性指一个算法除了能对合法的输入数据得到正确的结果外，还应对非法的输入数据作出正确合理的处理。比如火车售票处理，除了能在输入车票的数量为正整数时，得到正确的结果外，还应在车票数量为 0 或负数时，给出诸如“车票数不能为 0 或负数”这样的错误处理信息，而不能得到一些莫名其妙的输出结果。

4. 高效性

算法效率有两方面的含义，一是时间方面的效率，也叫算法的时间复杂度，指算法执行的时间，所用时间越少，则算法效率越高；二是空间方面的效率，也叫算法的空间复杂度，指算法所占用的存储空间，占用存储空间越少，效率越高。

1.2.4 算法效率的评价

1. 算法的时间复杂度

若要比较不同算法的时间效率，首先需要确定一个度量标准，最直接的办法是将算法转化为程序，在计算机上运行，通过计算机内部的计时功能获得精确的运行时间，然后进行比较。但该方法受计算机的硬件、软件等因素的影响，会掩盖算法本身的优劣，所以一般采用事先分析估算的方法，即撇开计算机软硬件等因素，只考虑问题的规模（一般用自然数 n 表示），认为一个特定算法的时间复杂度，只取决于问题的规模，或者说它是问题规模的函数。

为了方便比较，通常的做法是，从算法中选取一种对于所研究的问题（或算法模型）来说是基本运算的操作，以其重复执行的次数作为评价算法时间复杂度的标准。该基本操作多数情况下是由算法中最深层循环内的语句表示的，基本操作的执行次数实际上就是相应语句的执行次数。

多数情况下，算法中基本操作重复执行的次数是问题规模 n 的某个函数 f(n)，所以一般把算法的时间量度记作 T(n)=O(f(n))。

它表示随着问题规模 n 的增大，算法执行时间的增长率和 f(n)的增长率相同，称作算法

的渐进时间复杂度，简称时间复杂度。

例如，在下列三个程序段中：

```
① x:=x+1;
② for i:=1 to n do x:=x+1;
③ for i:=1 to n do
    for j:=1 to n do x:=x+1;
```

都包含基本操作“x 增 1”的语句 x:=x+1，其执行频度分别是 1、n 和 n^2，则这三个程序段的时间复杂度分别为 O(1)、O(n)、$O(n^2)$，分别称为常量阶、线性阶和平方阶。算法还可能呈现的时间复杂度有对数阶 $O(\log_2 n)$，指数阶 $O(2^n)$等。在 n 足够大时，不同数量级时间复杂度满足 $O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)$。可以看出，在算法设计时，应当尽量选择时间复杂度量级低的算法。

对一个问题（或一类问题）只需要选择一种基本操作来讨论算法的时间复杂度即可。当然有时也需要同时考虑几种基本操作，甚至可以对不同的操作赋以不同的权值，以反映执行不同操作所需要的相对时间。这种做法便于综合比较解决同一问题的多种不同的算法。

由于算法的时间复杂度考虑的只是对于问题规模 n 的增长率，所以在难以计算基本操作执行次数（或语句频度）的情况下，只需要求出它关于 n 的增长率或阶即可。

例如下面的程序段：

```
for i:=2 to n do
    for j:=2 to i-1 do x:=x+1;
```

基本操作是内重循环中的语句 x:=x+1，对于内重 j 循环其最少执行次数为 0 次（当 i=2 时），最多执行次数为 n–2 次（当 i=n 时），平均次数为（n–2）/2，外重 i 循环的执行次数是固定的，为 n–1 次。所以基本操作 x:=x+1 语句频度表达式为（n–1）（n–2）/2=（n^2–3n+2）/2，显然该表达式中增长最快的一项是 n^2，故可认为语句 x:=x+1 执行的次数关于 n 的增长率为 n^2。

2．算法的空间复杂度

一个上机执行的程序，除了需要存储空间来存储本身所用指令、常量、变量和输入数据外，还需要一些额外的辅助存储空间，一部分用作对数据进行操作的工作单元，另一部分用于存储一些为实现计算所需的信息。如果输入数据所占空间只取决于问题本身，与算法无关，则只需要分析除输入和程序之外的额外空间，否则应同时考虑输入本身所需空间（和输入数据的表示形式有关）。如果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。如果所占空间量依赖于特定的输入，则除特别说明外，均按最坏情况来分析，即以所占空间可能达到的最大值作为其空间复杂度。

类似于算法的时间复杂度，用空间复杂度作为算法所需存储空间的度量，记作：

S(n)=O(f(n))　（其中 n 为问题的规模或大小）

例 1-3　水仙花数。如果一个 n 位自然数的各位数字的 n 次方之和等于它本身，则称这个数为水仙花数。例如 153（$153=1^3+5^3+3^3$）是一个 3 位的水仙花数，8 208 是一个 4 位的水仙花数。编程找出所有的 3 位数到 7 位数中的水仙花数。

算法分析：可以采用枚举的策略，逐个枚举出 100～9 999 999 这一范围内所有的数，依