高等院校信息技术规划教材

算法与数据结构习题精解与实验指导（第2版）

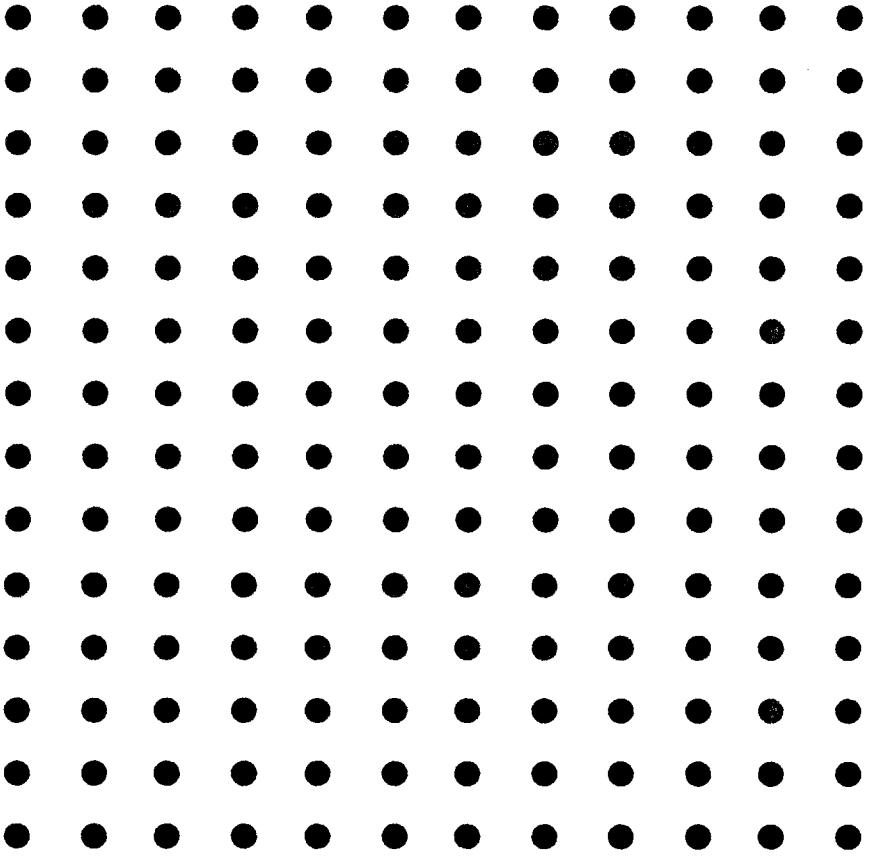
宁正元 赖贤伟 宁静 编著

清华大学出版社

高等院校信息技术规划教材

算法与数据结构习题精解与实验指导（第2版）

宁正元 赖贤伟 宁静 编著



清华大学出版社
北京

内 容 简 介

本书是与同期出版的教材《算法与数据结构(第2版)》配套的习题解析和实验辅导教材。

全书共分8章,与《算法与数据结构(第2版)》教材相对应,对各章中的习题、上机实验题作了详细的解答。对于算法题的描述主要包括解题思路、算法程序描述(用C语言)。算法程序均已在Turbo C环境中调试通过,可以直接引用。书后附有3套模拟练习题,供读者复习巩固教学效果使用。

本书可以作为不同院校计算机科学与技术学科及相关专业"数据结构"课程的辅导教材或参考书,也可作为考研考生的复习用书。

图书在版编目(CIP)数据

算法与数据结构习题精解与实验指导/宁正元,赖贤伟,宁静编著. --2版. --北京:清华大学出版社,2012.3

(高等院校信息技术规划教材)

ISBN 978-7-302-27652-4

Ⅰ.①算… Ⅱ.①宁… ②赖… ③宁… Ⅲ.①算法分析—高等学校—教学参考资料 ②数据结构—高等学校—教学参考资料 Ⅳ.①TP301.6 ②TP311.12

中国版本图书馆CIP数据核字(2011)第273899号

责任编辑: 袁勤勇 顾 冰
封面设计: 傅瑞学
责任校对: 白 蕾
责任印制: 何 芊

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,010-62795054

印 装 者: 北京鑫海金澳胶印有限公司
经 销: 全国新华书店
开 本: 185mm×260mm **印 张:** 15 **字 数:** 344千字
版 次: 2012年3月第2版 **印 次:** 2012年3月第1次印刷
印 数: 1～3000
定 价: 25.00元

产品编号:044579-01

前言 foreword

“算法与数据结构”课程是计算机科学与技术学科中一门十分重要的专业基础课程和专业核心课程。中国计算机学会教育专业委员会和全国高等学校计算机教育研究会在《计算机学科教学计划1993》中，把“算法与数据结构”课程列为计算机科学与技术学科公共要求的9个主科目之一。中国计算机科学与技术学科教程2002研究组在《中国计算机科学与技术学科教程2002》中，把“算法与数据结构”课程列为计算机科学与技术学科16个专业核心课程之一。计算机科学与技术学科的各个领域中，都要求科技工作者具备良好的算法与数据结构基础。

然而，“算法与数据结构”课程的学习难度较大。除了课程自身的内容多、介绍的方法技术多及在先修课程中涉及的专业基础知识少之外，“算法与数据结构”课程有着它自身的一些特点和规律：①学生不易理解和接受数据的逻辑结构的抽象数据类型表示；②动态存储结构的动态性和递归技术的抽象性，使得学生对相应的知识不易掌握；③算法描述的形式化和程序设计语言化使许多学生望而生畏；④算法设计的灵活多样性以及算法分析等内容使得许多学生较难掌握。所有这些，都是学生感觉到这门课程难度大的原因。许多学生在课堂上一听就明白，拿起课本认真一看也能基本弄懂，但做作业，尤其是做算法设计题目时总觉得无从下手。究其原因，首先主要是听懂数据结构的内容和应用数据结构知识解决实际问题之间存在着相当长的距离；其次是理解掌握算法分析和设计的各种方法、技术与灵活运用这些方法解决各种具体问题之间也存在着较大距离。逐步缩短进而消除这些距离是解决“算法和数据结构”课程学习难的关键所在，其根本途径在于加强实践环节，多学、多用、多做及多练，熟能生巧，以期达到对所学知识和方法技能的融会贯通。

在认真总结了“算法和数据结构”课程30年教学实践的基础上，结合应用型大学的教学特点和学生学习的实际需求，我们编写了《算法与数据结构习题精解与实验指导(第2版)》一书，以期通过该书中的习题解答和具体的实践环节来给学生一些解答示范和启发，

帮助学生更好地学习和掌握课程内容,理解和掌握算法设计所需的方法和技术,为整个专业学习打下坚实的基础。考虑到部分学生本科毕业后考研的需要,该书也选择了若干典型的研究生入学试题做了精细解析。

本书是同期出版的教材《算法与数据结构(第2版)》的配套用书。为了便于对照阅读,本书的章序与《算法与数据结构(第2版)》一书的章序保持一致,且一一对应。各章内容分别介绍《算法与数据结构》相应章节所涵盖的知识和技能的概括总结,归纳出本章知识主线,有利于学生复习总结和理解掌握;对各章之后的习题做精细解答,对典型题目提出多种解题思路;以及对上机实验题也作了详细解答和指导,通过实验使学生了解并学会如何运用数据结构知识去解决现实世界中的实际问题,具备较复杂程序的初步设计能力。书后增添了3套模拟练习题作为附录,是为了便于学生在课程结束后复习巩固主要教学内容。

组织这本辅助教材的主要目的是为了帮助学生学好"算法与数据结构"这门课程,所以在使用过程种要注意以下几点:①与课程学习内容同步使用。这样有利于教材中知识点的理解和掌握,有利于巩固和提高课堂教学效果。②切忌照抄照搬。算法的设计具有不唯一性,对算法设计类题目,书中给出了一种或多种解答方法,要在学习、理解及领会的基础上自己动手设计算法并编写程序,这样才能获得更好的效果。③遵循循序渐进的原则。书中内容按知识结构分类组织,同类内容按典型到一般、由易到难的次序排列,读者最好能按次序阅读学习。④学会举一反三,触类旁通。课程内容中的知识点是有限的,但运用所学知识和方法技术解决实际问题则是无限的。重在掌握基本原理、基本方法和基本技术,并学以致用和灵活运用。

本书是经宁正元教授统稿,由多人共同编著的,其中宁静完成第1～3章,赖贤伟完成第4～8章和附录。王秀丽教授、林大辉副教授、易金聪副教授、黄思先副教授、刘雄恩副教授、黄建实验师及林敏讲师等在本书的成稿前期工作或精品课程建设中做了部分工作或提供有益帮助。高等教育出版社和清华大学出版社对本书的出版自始至终都给予了极大的支持和鼓励,作者在此一并表示最诚挚的感谢。

鉴于时间仓促和作者水平所限,书中内容虽经反复讨论与仔细推敲,仍难免存在着错、谬、疏及漏之处,敬请同行专家和广大读者不吝赐教,我们将不胜感激。

编　者

2011年12月

目录 Contents

第1章 chapter 1

算法与程序

本章主要介绍算法的基本概念、特性、表示、评价和设计方法，并简要介绍程序的概念、特性、与算法的关系，以及对实际问题的求解方法。同时对章后习题做了较详细的解答。

1.1 基本知识点

1. 算法的基本概念

算法就是求解问题的方法和步骤。这里的方法和步骤是一组严格定义了运算顺序的规则；每一个规则都是有效的，且是明确的；按此顺序将在有限步骤后终止并获得结果。算法的形式化定义为：

算法是一个四元组，即(Q,I,Ω,F)。其中：

(1) Q是一个包含子集I和Ω的集合，它表示计算的状态。

(2) I表示计算的输入集合。

(3) Ω表示计算的输出集合。

(4) F表示计算的规则，它是由Q至它自身的函数，且具有自反性，即对任何一个元素$q\in Q$，有$F(q)=q$。

算法具有5个重要特性：输入(Input)、输出(Output)、确定性(Definiteness)、有穷性(Finiteness)和有效性(Effectiveness)。

2. 算法的表示

常用的算法表示方法有自然语言、流程图、N-S图及伪代码和计算机程序设计语言。每种表示方法都有各自的优缺点和相应的应用场合。

3. 算法的设计与评价

评价一个算法优劣的标准有5条，即正确性、可读性、健壮性、高效性和简洁性。

算法复杂性的评价方法：环路复杂度方法是利用程序图中线性无关的环路个数表示算法的复杂性，它是一种简单的定量地评价算法复杂度的方法；大O方法是把算法中总

的操作次数表示为问题规模 n 的函数，利用该函数的渐近阶表示算法的复杂性，它是一种常用的定性评价算法复杂度的方法。

常用的算法设计方法和技术包括穷举法、迭代法、递推法、递归法、回溯法、贪婪法和分治法等。

4. 算法与程序

算法与程序是密切相关的两个概念。研究和讨论算法是为了设计出更好的程序，设计好的算法都要转化为某种语言描述的程序才能在计算机上运行；或者说程序是算法表示的最终形态，程序只有装入计算机中运行(即程序的执行)时才能够起到对实际问题求解的作用。在低级语言中，程序表现为一组指令和有关数据；在高级语言中，程序表现为一组说明和语句。

程序具有 6 个特征，即程序是算法的程序设计语言描述；程序描述解决某一问题的特定任务与功能；程序遵循一定的规则和步骤；程序是由人来编写或设计；程序的执行者是计算机；程序的运行是自动完成的。

解决一个实际问题的一般过程：明确问题要求、建立数学模型、算法设计、编写程序、调试程序、运行及结果分析和编写程序文档。

程序调试技术主要有如下 3 类：输出存储器内容、在程序中插入打印语句和借助调试工具。

程序调试策略主要有以下 5 种：试探法、回溯法、对分查找法、归纳法和演绎法。

程序的查错策略主要有两种：黑盒子测试和白盒子测试。

程序设计方法可以分为两类。一类是全局性的；另一类则是局部性的。主要的程序设计方法与技术：结构化程序设计、软件工程方法、面向对象的程序设计、多媒体程序设计、可视化编程、函数程序设计、逻辑程序设计、并行程序设计、分布式程序设计和文化程序设计等。

1.2 习题解答

1. 何谓算法？简述算法的基本特性和表示方法。

解答：算法(Algorithm)就是求解问题的方法和步骤。这里的方法和步骤是一组严格定义了运算顺序的规则，每一个规则都是有效的，且是明确的；按此顺序算法将在有限次数下终止。算法也可以定义为，所谓算法就是对特定问题求解方法和步骤的一种描述，它是指令的一组有限序列，其中每个指令表示一个或多个操作。

一个算法必须具备以下 5 个重要特性。

(1) 输入(Input)：一个算法必须具有零个或多个输入，这些输入是算法开始前对运算法给出的最初量。

(2) 输出(Output)：一个算法必须有一个或多个输出。

(3) 有穷性(Finiteness)：一个算法在执行有穷步骤之后必然结束，并且每步都必须在有穷时间内完成。

(4) 确定性(Definiteness)：算法中每一个步骤必须有确切的定义，不会使人在读算法时产生二义性。

(5) 有效性(Effectiveness)：组成算法的每一个操作都应该是在特定环境下允许使用的、可以执行的，并能在有限时间内完成，最后得出确定的结果。

常用的算法表示方法有以下几种。

(1) 自然语言表示：即用人们日常使用语言，如汉语、英语及日语等来表示算法。

(2) 流程图表示：即采用一组图形符号来表示算法。

(3) N-S 图表示：N-S 图独立于任何计算机和计算机语言，全部用矩形框来描述算法，它仍是图形工具，阅读起来直观、明确及容易理解。

(4) 伪代码表示：即用文字和符号描述问题的求解方法和步骤，而不使用图形符号来描述算法。

(5) 程序语言表示：即用计算机程序设计语言来描述算法。

2. 如何评价一个算法？简述环路复杂度、空间复杂度和时间复杂度的概念。

解答：评价一个算法的优劣的标准有以下 5 条。

(1) 正确性。即所设计出来的算法要能够正确求解给定的问题，算法要能经得起一切可能的输入数据的考验。此外，在将算法用程序语言表示为特定语言的程序后还必须注意以下几点：

① 程序中不含有语法错误；

② 对于一切合法的输入数据，程序能够产生满足要求的输出结果；

③ 对于一切非法的输入数据，程序能够得出满足规格说明的结果；

④ 对于精心选择的，甚至是带有刁难性的典型测试数据，程序都有满足要求的输出结果。

(2) 可读性。即表示出来的算法要能够方便地供人们阅读、理解和交流。

(3) 健壮性。即算法对意外情况的反映能力要强，当输入数据非法时，算法应能适当地作出反应或进行处理。

(4) 高效性。即算法的执行效率要高，要求算法的执行时间要尽可能地短，对于存储空间要尽可能地少，要做到既省时又节省空间。

(5) 简洁性。即所设计出来的算法要尽可能地简洁。

环路复杂度是一种定量地评价算法复杂度的方法。以图论为工具，把算法(或程序)流程图中的每个处理框都退化成为一个点，原来连接不同框之间的流程线变成连接不同点的有向弧，画出程序图，增加一条由出口到达入口的虚弧，使得程序图变为强连通的有向图，然后可以用该程序图的环路作为算法(或程序)复杂性的度量值。计算环路复杂度的公式：

$$V(G) = e - n + p$$

其中，$V(G)$是图 G 中环路的个数；

e 是图 G 中有向弧的条数，包括出口到入口增加的一条虚弧；

n 是图 G 中结点的个数；

p 是图 G 中连通分量的个数，对于连通图而言 p 恒为 1。

时间复杂度是一种适应面更广泛的定性评价算法复杂度的方法，即大O方法，它是将算法执行时间的讨论转化为对算法中所有语句的执行次数(即频度)的讨论，并引入大O记号表示的算法的时间耗费$T(n)$。时间复杂度实质上是把算法的基本操作总数表示为问题规模n的函数之后，寻找出当问题规模n趋于无穷大时该函数的同阶最简形式，即渐近性态下的同阶最简函数，有时也简称为渐近阶；在计算算法时间复杂度时，只需考虑算法中频度最大的语句的频度就可以了。

空间复杂度也是用相对于问题规模函数的渐近阶形式给出，它是度量一个算法或程序在执行过程中所花费的额外存储开销(即临时存储工作单元)大小，包括为了在计算机上存储程序本身所占用的存储空间；算法或程序中的输入和输出数据所占用的存储空间；算法或程序在执行过程中临时占用的存储空间，也是用大O方法表示。

3. 简述算法与程序的联系与区别，并列举常见的算法设计方法。

解答：算法与程序的区别如下：

(1) 算法是指完成一个任务所需要的具体步骤和方法的描述，通俗地说，就是计算机解题的过程。算法设计的任务是对各类具体问题设计良好的算法及研究设计算法的规律和方法。常用的算法有穷举搜索法、递归法、回溯法、贪婪法和分治法等。算法分析的任务是对设计出的每一个具体的算法，利用数学工具，讨论各种复杂度，以探讨某种具体算法适用于哪类问题，或某类问题宜采用哪种算法。

(2) 程序是对所要解决的问题的各个对象和处理规则的描述，在低级语言中，程序表现为一组指令和有关数据，在高级语言中，程序表现为一组说明和语句；程序的执行者是计算机，计算机能在无须人工干预的情况下，连续自动地执行程序，最终给出运行结果；程序是算法的程序设计语言描述，但程序并不一定就是算法，因为程序没有有穷性的要求。

两者的联系：我们可以说程序是对数据结构和算法的具体描述，因此有人说，数据结构＋算法＝程序。算法是程序设计的精髓，程序设计的实质就是将算法表示为计算机语言，设计好的算法都要转化为某种语言描述的程序才能在计算机上运行；或者说程序是算法表示的最终形态，程序只有装入计算机运行(即程序的执行)时才能够起到对实际问题求解的作用。

常见的算法设计方法有如下几种：

(1) 穷举法。其基本思想：首先根据求解问题的部分条件确定答案的大致范围；然后在此范围内对所有可能的情况逐一验证，从而得出求解问题的完整解。

(2) 迭代法。其基本思想：由一个量的原值求出它的新值，不断地再用新值替代原值求出它的下一个新值，直到得到满意的解。

(3) 递推法。其基本思想：从已知的初始条件出发，逐次推出所需要求解的各中间结果和最终结果。

(4) 递归法。其基本思想：将一个复杂问题归结为若干较为简单的问题，然后将这些较为简单的问题进一步归结为更简单的问题，这个过程一直进行下去直到归结为最简单的问题时为止。

(5) 回溯法。其基本思想：通过对问题的分析找出一个解决问题的线索，然后沿着

这个线索逐步试探。

(6) 贪婪法。其基本思想：通过一系列的选择来得到问题的一个解，在每一步所做出的选择，都是在当前状态下看来是最好的选择，并希望通过每次所做的贪婪选择导致最终结果是求解问题的一个最优解。

(7) 分治法。其基本思想：求解一个复杂问题时，尽可能地把这个问题分解为若干较小的子问题，在找出各个较小问题的解之后再组合成为整个问题的解。

4. 简述程序调试与查错策略。

解答：

(1) 现有的程序调试技术有如下 3 类：

① 输出存储器内容。常以八进制或十六进制形式打印出存储器内容并检查分析。

② 在程序中插入打印语句。

③ 借助调试工具。调试工具可以提供程序动态行为的有关信息，但不需要修改原程序。

推断错误原因的调试策略主要有以下 5 种：

① 试探法。分析错误征兆，猜测大致位置，获取程序中被怀疑有错位置附近的信息。

② 回溯法。检查错误征兆，确定最先发现“症状”的地方，然后人工沿程序的控制流往回追踪原程序代码，直到找出错误根源或确定故障范围。

③ 对分查找法。如果已知每个变量在程序中若干关键点的正确值，则可在这些点附近注入正确值，检查程序的输出；若输出结果正确则故障在关键点之前，否则在关键点之后。

④ 归纳法。从错误的征兆(即线索)出发，通过分析这些线索之间的关系而找出故障，是一种从个别推断一般的系统化思考方法。

⑤ 演绎法。是一种从一般原理或前提出发，经过删除和精化的过程推导出结论的方法。

(2) 发现程序中潜藏的错误的查错策略主要有两种，即黑盒子测试和白盒子测试：

① 黑盒子测试是指已知产品的功能，可以测试它的每一个功能是否都达到了预期的要求。

② 白盒子测试是指已知产品的内部活动方式，可以测试它的内部活动是否都符合设计要求。

5. 试证明大 O 表示法中的加法准则。

解答：加法法则的内容：若两个程序段的时间复杂度为 $T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$，那么这两个程序段依次执行的时间复杂度为：

$$T(n) = T_1(n) + T_2(n) = O(\max(f(n), g(n)))$$

上式可以证明如下：

由于 $T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$

则存在常数 c 和 n_0，使得当 $n \geqslant n_0$ 时，有

$$T_1(n) + T_2(n) \leqslant c \times \max(f(n), g(n))$$

例如，$T_1(n)=O(n^2)$，$T_2(n)=O(n^3)$

当 $n_0=1$ 时,对所有 $n \geqslant n_0$ 都有

$$n^3+n^2 \leqslant n^3+n^3=2n^3$$

当取 $c=2$ 时,则有 $T(n) \leqslant c \times n^3$,取 $T(n)=O(n^3)$

所以有

$$T(n)=T_1(n)+T_2(n)=O(\max(f(n), g(n)))$$

成立。

6. 常见的时间(或空间)复杂度量级有 $O(1)$、$O(n)$、$O(n^2)$、$O(n^3)$、$O(n\log_2 n)$、$O(\log_2 n)$和 $O(2^n)$等,试将它们按增长率由小到大排列。

解答:上面的几种类型的数量级中 $O(1)$为常量型,$O(n)$为线性型,$O(n^2)$为平方型,$O(n^3)$为立方型,$O(\log_2 n)$为对数型,$O(2^n)$为指数型,$O(n\log_2 n)$为线性对数型,它们按增长率从小到大的顺序:$O(1)$,$O(\log_2 n)$,$O(n)$,$O(n\log_2 n)$,$O(n^2)$,$O(n^3)$,$O(2^n)$。

7. 已知输入 x、y、z 3 个不等的整数,欲将它们按从小到大的顺序排列后输出,请完成下列工作:

(1) 设计一个算法完成题目要求。

(2) 计算算法的环路复杂度。

(3) 估算算法的比较次数和元素的移动次数。

解答:

(1) 依题意将此题的算法用 N-S 图表示如图 1-1 所示。

具体程序如下所示:

```
#include "stdio.h"
main()
{
  int x,y,z,t;
  scanf("%d%d%d",&x,&y,&z);
  if(x>y)
    { t=x; x=y; y=t;}
  if(x>z)
    { t=x; x=z; z=t;}
  if(y>z)
    { t=y; y=z; z=t;}
  printf("x=%d,y=%d,z=%d\n",x,y,z);
}
```

(2) 依题意得出本题算法流程图的程序图如图 1-2 所示。

由计算环路复杂度的公式 $V(G)=e-n+p$,可知本程序的环路复杂度为 $V(G)=11-8+1=4$。

(3) 从以上算法可知需要进行 3 次比较,在最坏情况下需要移动元素 9 次。

8. 用 C 语言编写在输入的 20 个数中找最大数或最小数的程序,并讨论算法的复杂度(环路复杂度、时间复杂度和空间复杂度)。

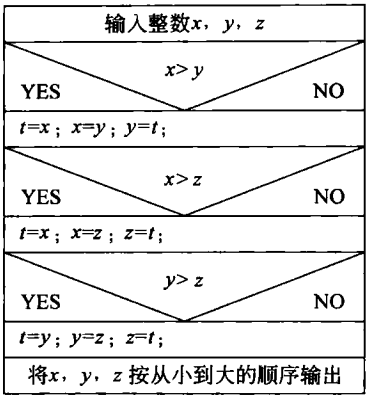


图 1-1 N-S 图表示法

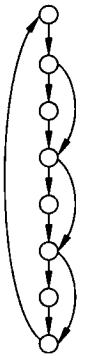


图 1-2 N-S 算法流程图的程序图

解答：依题意具体程序描述如下：

```
#include "stdio.h"
main()
{
  float a[20],min,max;
  int i;
  for(i=0;i<20;i++)
    {
     printf("Please input the %dth number.",i+1);
     scanf("%f",&a[i]);
    }
  min=a[0];
  max=a[0];
  for( i=1;i<20;i++)
    {
     if(a[i]<min)
        min=a[i];
     else if(a[i]>max)
        max=a[i];
    }
  printf("min=%f,max=%f",min,max);
}
```

由以上程序得出本题算法流程图的程序图如图 1-3 所示。

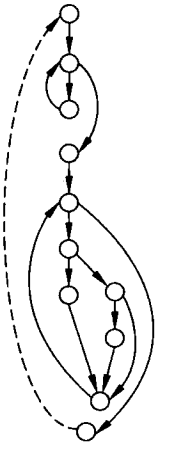


图 1-3 算法流程图的程序图

由计算环路复杂度的公式 $V(G)=e-n+p$，可知本程序的环路复杂度为 $V(G)=15-11+1=5$；时间复杂度为 $O(n)$；空间复杂度为 $O(1)$。

9. 试分析下列各程序段的时间复杂性：

```
(1) i=1;
    k=0;
    n=100;
    do{k=k+10*i;
       i++;
      }while(i!=n);
(3) for(i=0;i<m;i++)
      for(j=0;j<n;j++)
        A[i][j]=i*j;
(5) i=1;j=0;
    while((i+j)<=n)
        if(i>j) j++;
        else i++;
(7) x=n;     /*n>1*/
    y=0;
    while(x>=(y+1)*(y+1))
      y=y+1;
(9) i=0; s=0;
    while(s<n)
      {i++;
       s=s+i;}
```

```
(2) m=91; n=100;
    while(n>0)
      if(m>0)
        {m=m-10; n=n-1;}
      else
        m=m+1;
(4) i=1; k=1;
    while(i<=n-1)
      {k=k*10*i;i++;}
(6) for(i=0;i<n;i++)
      for(j=0;j<i;j++)
        for(k=0;k<j;k++)
          x=x+1;
(8) i=-1;s=0;
    while(s<n)
        {i=i+2;
         s=s+i;}
(10) for(i=0;i<n-1;i++)
       for(j=n;j>i;j--)
         y=y+j;
```

解答：

(1) $T(n)=O(1)$

(2) 本程序段是著名的 McCarthy 函数

$$f(m)=\begin{cases} m-10, & m>10 \\ f(f(m+1)), & x\leqslant 100 \end{cases}$$

对任何的 $m\leqslant 100$，$f(x)=91$，所以此程序段实质上是一个二重循环，对每个 $n(n>0)$值，if 语句执行 11 次，其中 10 次是执行 m++语句，但它们与 n 无关，所以 $T(n)=O(1)$。

(3) $T(n)=O(m\times n)$

(4) $T(n)=O(n)$

(5) $T(n)=O(n)$

(6) $T(n)=O(n^3)$

(7) $T(n)=O(\sqrt{n})$

(8) $T(n)=O(\sqrt{n})$

(9) $T(n)=O(\sqrt{n})$

(10) $T(n)=O(n^2)$

第2章 chapter 2

常用数据结构

本章主要介绍在前导课程“高级语言程序设计”中学习和使用过的数据类型、数组和字符串等常用数据结构，从数据结构的角度作进一步引申，为在后续章节介绍数据结构的新内容作一些必要的准备。同时，对章后习题做了较详细的解答。

2.1 基本知识点

1. 数据类型与数据结构

数据(Data)是信息的载体，是对自然界客观事物的符号表示。数据的基本单位是数据元素(Data Element)，有时也称作元素、结点、顶点及记录等。一个数据元素也可以由若干个数据项(Data Item)组成。数据项是具有独立含义的数据的不可再分割的最小标识单位。数据对象(Data Object)是指具有相同性质的数据元素的集合，是数据的一个子集。

数据类型(Data Type)是对在计算机中表示的同一数据对象及其在该数据对象上的一组操作的总称。数据类型的概念，具有 6 个显著的特征：①类型决定了变量或表达式所有可能取值的全体成员集合。②每一个值隶属于且仅隶属于某一类型。③任何常量、变量或表达式的类型，都可以从其形式上或所处的上下文关系中推断出来，无须了解在程序运行时计算出的具体值。④每一种操作都要求一定类型的操作数据，且得出一定类型的操作结果。⑤一种类型的值及其在该类型上规定的基本操作的性质可由一组公理来阐明。⑥高级程序设计语言使用类型信息来防止程序中出现无意义的结构，又由类型信息确定在计算机中的数据表示和数据处理方法。

数据结构(Data Structure)是指计算机程序中所操作的对象——数据以及数据元素之间的相互关系和运算。数据结构包含以下 3 个方面的内容：数据的逻辑结构、数据的存储结构及数据的运算及实现。线性结构中数据元素之间存在着一对一的次序关系。非线性结构中数据元素之间不存在一对一的次序关系，其中，树型结构中是一对多的层次关系，图形结构中是多对多网状关系，而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。数据的存储表示方式主要有 4 种，即顺序存储、链式存储、索引存储及散列存储。

抽象数据类型(Abstract Data Type,ADT)是指一个数据模型以及定义在该数据模型上的一组操作。抽象数据类型的定义,仅取决于它的一组逻辑特性,而与它在计算机内部如何表示和实现无关。

2. 数组

数组(Arrays)是把具有相同性质的一组元素组织在一起形成一种数据结构。数组结构是一个线性的、均匀的、其元素可以随机访问的数据结构。数组结构的运算操作通常只有赋值和读取数组元素两种操作。顺序存储结构是数组存储结构的最佳选择。所谓顺序存储结构,是指在计算机内存中安排一片地址连续的空间存储数据。对于特殊矩阵(如对角矩阵、三对角矩阵、上三角矩阵、下三角矩阵及对称矩阵等)可采用压缩存储。所谓压缩存储,是指为多个值相同的元素分配一个元素的存储空间,对零元素不分配存储空间。

3. 串

所谓串(String),是由零个或多个字符组成的有限序列。串中任意多个连续字符组成的子序列称作该串的子串(Substring),包含子串的串称作该子串的主串。字符在串中的序号称作该字符在串中的位置,子串在主串中的位置用子串的第一个字符在主串中的位置来表示。两个串相等是指这两个串的值相等,即两个串长度相等且对应字符都相同时才相等。要注意空串与空格串的区别。

串的基本运算有 6 种:串赋值 StrAssign(s,chars)、串复制 StrCopy(s,t)、串比较 StrCompare(s,t)、求串长 StrLenth(s)、串联接 StrConcat(l,s,t)和求子串 SubString(t,s,i,len)。

串的存储可分为定长顺序存储和堆式动态存储。

定长顺序存储分配是指为字符串分配一个固定长度的一组地址连续的存储单元的存储分配方法。定长顺序存储分配的实现,要结合具体的计算机编址方式进行。通常有如下 3 种实现方式。

(1) 紧缩方式:计算机按字编址,每个单元存放 4 个字符。

(2) 非紧缩方式:计算机按字编址,每个单元存放一个字符。

(3) 单字节存储方式:计算机按字节编址,每个字节存放一个字符。

堆式动态存储分配的思想:应用系统在内存中开辟一个容量很大且地址连续的一片存储空间,作为存放所有串值的可利用空间即堆空间。

4. 模式匹配

子串的定位操作通常称作模式匹配,子串 t 称作模式,在主串 s 中确定 t 的位置的过程称作匹配过程。模式匹配在文本编辑、文件检索和各种串处理系统中有着广泛的应用,因此它是最重要的串运算之一。虽然模式匹配不是串的基本运算,可以借用串的基本运算来实现;然而由于其有着重要的广泛应用,一直有人不断地尝试设计出更高效的模式匹配算法。