张晓莉 罗文劼 刘振鹏 许百成 编著

数据结构与算法

机械工业出版社
CHINA MACHINE PRESS

数据结构与算法

张晓莉　罗文劼　刘振鹏　许百成　编著

机 械 工 业 出 版 社

本书详细讲述了线性结构、树形结构和图形结构、查找表、排序表等常见数据结构的数据表示及数据处理的方法。

本书始终围绕易讲、易懂、易学这一原则来进行编写。在教材中配有大量算法设计的例子，以便于读者理解和掌握数据结构中数据表示和数据处理的方法。

本书可作为计算机科学与技术和信息类相关专业的本（专）科“数据结构”课程的教材或学习参考书。

图书在版编目（CIP）数据

数据结构与算法/张晓莉等编著. —北京：机械工业出版社，2002.10
ISBN 7-111-10898-1

Ⅰ.数… Ⅱ.张… Ⅲ.①数据结构②算法分析 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字（2002）第 067037 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策　　划：胡毓坚
责任编辑：王冰飞
责任印制：

北京忠信诚胶印厂印刷·新华书店北京发行所发行

2002 年 9 月第 1 版·第 1 次印刷
787mm×1092mm 1/16·16.75 印张·410 千字
0001－6000 册
定价：22.00 元

前　言

“数据结构”课程是计算机科学与技术、信息类相关专业的一门重要的专业基础课程。当用计算机来解决实际问题时，就要涉及到数据的表示及数据的处理，而这正是数据结构课程的主要研究对象。通过本课程的学习，能为后续课程，特别是软件方面的课程打下厚实的知识基础，同时也提供了必要的技能训练。因此，数据结构课程是计算机科学与技术专业的核心课程之一。当然本课程也是有一定难度的，因此，选择一本合适的教材对学好这门课至关重要。因为该课程大多安排在二年级讲授，前面只有程序设计和离散数学的基础，过分抽象和简单都不利于学生的学习和教师的讲授，因此根据我们多年的教学经验，在多年讲义的基础上编写了本教材。

本书的编写以“内容丰富，语言通俗易懂，概念讲解清楚，逻辑性强，可读性好、应用算法多”为宗旨。编者全部是多年讲授“数据结构”课程的教师。为了加强读者对“数据结构”的理解和应用，教材中含有 130 余例算法，而且大部分算法都进行了调试，这在同类教材中是少见的。数据结构课程是一门实践性较强的课程，因此本书在每章后面选配了难易程度不同的基本题、算法题和实践题。

本书共分 9 章：第 1 章和第 5 章的 5.4 节由刘振鹏编写，第 2 章、第 3 章、第 4 章和第 5 章的 5.1～5.3 节由张晓莉编写，第 6 章、第 7 章由罗文劼编写，第 8 章、第 9 章由许百成、张晓莉编写，最后由张晓莉、刘振鹏统一定稿。考虑到动态存储管理和文件部分会在其他课程中讲授，故这部分内容没有纳入本教材之中。

在编写本书的过程中，参考了一些国内外优秀教材（见参考文献），王苗、石强、徐建民、张风声、杨成等老师对本书内容提出了很多修改意见，彭澎老师在百忙中审阅了全书，在此表示衷心感谢。

由于编者的水平有限和时间的仓促，书中肯定会存在着错误之处，恳请各位专家、读者批评指正。

本书可作为计算机科学与技术和信息类相关专业的本（专）科“数据结构”课程的教材和教学参考书。

编著者

目　录

第1章 绪 论

计算机科学是一门研究数据表示和数据处理的科学。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。无论是进行科学计算或数据处理、过程控制以及对文件的存储和检索及数据库技术等计算机应用，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而且效率高的程序，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 数据结构的概念

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，还必须学习和掌握好数据结构的有关知识。学好“数据结构”这门课程，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、人工智能等都是十分有益的。

1.1.1 为什么要学习数据结构

在计算机发展的初期，人们使用计算机的目的主要是处理数值计算问题。当我们使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先要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，直至得到最终的解答。

由于当时所涉及的运算对象是简单的整型、实型或布尔类型数据，所以程序设计者的主要精力是集中于程序设计的技巧上，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得重要。据统计，当今用计算机处理的非数值计算性问题约占90%以上的机器时间。这类问题涉及到的数据结构相当复杂，数据元素之间的相互关系一般无法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，而是要设计出合适的数据结构，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】 学生信息检索系统。当我们需要查找某个学生的有关情况时；或者想查询某个专业或年级的学生的有关情况时，只要我们建立了相关的数据结构，按照某种算法编写了相关程序，就可以实现计算机自动检索。对此，可以在学生信息检索系统中建立一张按学号顺序排列的学生信息表和分别按姓名、专业、年级顺序排列的索引表（如图1-1所示），由这四张表构成的文件便是学生信息检索的数学模型。计算机的主要操作便是按照某个特定要求（如给定姓名）对学生信息文件进行查询。

诸如此类的还有电话自动查号系统、考试查分系统、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的是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。

学号	姓名	性别	专　业	年级
980001	吴承志	男	计算机科学与技术	98 级
980002	李淑芳	女	信息与计算科学	98 级
990301	刘　丽	女	数学与应用数学	99 级
990302	张会友	男	信息与计算科学	99 级
990303	石宝国	男	计算机科学与技术	99 级
000801	何文颖	女	计算机科学与技术	2000 级
000802	赵胜利	男	数学与应用数学	2000 级
000803	崔文靖	男	信息与计算科学	2000 级
010601	刘　丽	女	计算机科学与技术	2001 级
010602	魏永鸣	男	数学与应用数学	2001 级

a) 学生信息表

崔文靖	8
何文颖	6
李淑芳	2
刘　丽	3, 9
石宝国	5
魏永鸣	10
吴承志	1
赵胜利	7
张会有	4

b) 姓名索引表

计算机科学与技术	1, 5, 6, 9
信息与计算科学	2, 4, 8
数学与应用数学	3, 7, 10

c) 专业索引表

2000 级	6, 7, 8
2001 级	9, 10
98 级	1, 2, 3
99 级	4, 5

d) 年级索引表

图 1-1　学生信息查询系统中的数据结构

【例 1-2】　八皇后问题。在该问题中，处理过程不是根据某种确定的计算法则，而是利用试探和回溯的探索技术求解。为了求得合理布局，在计算机中要存储布局的当前状态。从最初的布局状态开始，一步步地进行试探，每试探一步形成一个新的状态，整个试探过程形成了一棵隐含的状态树，如图 1-2 所示（为了描述方便，将八皇后问题简化为四皇后问题）。回溯法求解过程实质上就是一个遍历状态树的过程。在这个问题中所出现的树也是一种数据结构，它可以应用在许多非数值计算的问题中。

【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，有些必须按规定的先后次序进行，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，有些课程可以任意安排次序。这种各个课程之间的次序关系可用一个称作图的数据结构来表示，如图 1-3 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。

由以上三个例子可见，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操作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，通过算法训练来提高学生的思维能力，通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

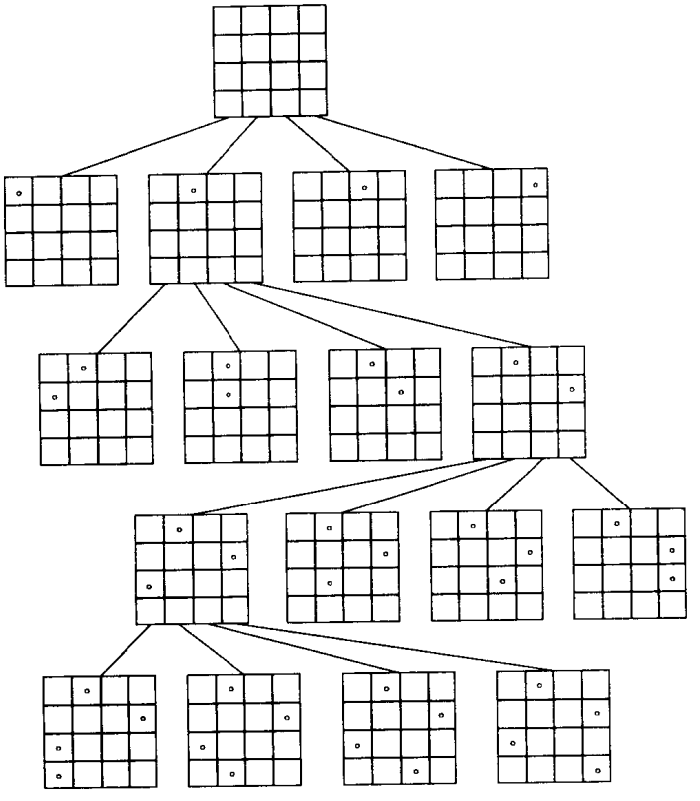
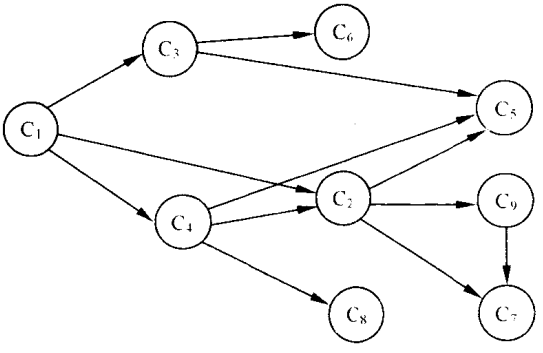


图 1-2　四皇后问题中隐含的状态树

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 1-3　教学计划编排问题的数据结构

1.1.2 有关概念和术语

在系统地学习数据结构知识之前，先对一些基本概念和术语赋予确切的含义。

数据(Data) 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。计算机科学中，所谓数据就是计算机加工处理的对象，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，主要用于工程计算、科学计算和商务处理等；非数值数据包括字符、文字、图形、图像、语音等。

数据项（也称项或字段） 是具有独立含义的标识单位，是数据不可分割的最小单位。如学生信息表中的“学号”、“姓名”、“年级”等。数据项有名和值之分，数据项名是一个数据项的标识，用变量定义，数据项值是它的一个可能取值，学生信息表中“20010983”是数据项“学号”的一个取值。数据项具有一定的类型，依数据项的取值类型而定。

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。例如，学生信息检索系统中学生信息表中的一个记录、八皇后问题中状态树的一个状态、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。

有时，一个数据元素可由若干个数据项(Data Item)组成，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作一个基本单位进行访问和处理的。

数据对象(Data Object)或数据元素类(Data Element Class)是具有相同性质的数据元素的集合。在某个具体问题中，数据元素都具有相同的性质(元素值不一定相等)，属于同一数据对象(数据元素类)，数据元素是数据元素类的一个实例。例如，在交通咨询系统的交通网中，所有的顶点是一个数据元素类，顶点 A 和顶点 B 各自代表一个城市，是该数据元素类中的两个实例，其数据元素的值分别为 A 和 B。

数据结构 (Data Structure) 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，这种数据元素之间的关系称为结构。根据数据元素间关系的不同特性，通常有下列四类基本结构：

(1) 集合结构。在集合结构中，数据元素间的关系是“属于同一个集合”。集合是元素关系极为松散的一种结构。

(2) 线性结构。该结构的数据元素之间存在着一对一的关系。

(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) 图形结构。该结构的数据元素之间存在着多对多的关系，图形结构也称作网状结构。图 1-4 为表示上述四类基本结构的示意图。

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。

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，另一个是关系的集合。在形式上，数据结构通常可以采用一个二元组来表示。

数据结构的形式定义为：数据结构是一个二元组。

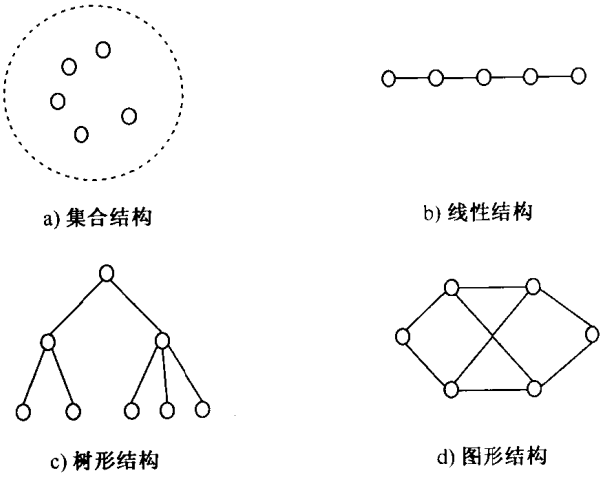


图 1-4　四类基本结构的示意图

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

其中,D 是数据元素的有限集,R 是 D 上关系的有限集。

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作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一个数据结构。数据结构在计算机中的标识(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,包括数据结构中元素的表示及元素间关系的表示。

数据的存储结构可采用顺序存储或链式存储的方法。

顺序存储方法是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。

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,由此得到的存储表示称为链式存储结构,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,有时为了查找的方便还采用索引存储方法和散列存储方法。

1.1.3　数据结构课程的内容

数据结构与数学、计算机硬件和软件有十分密切的关系。数据结构是介于数学、计算机硬件和计算机软件之间的一门计算机科学与技术专业的核心课程,是高级程序设计语言、编译原理、操作系统、数据库、人工智能等课程的基础。同时,数据结构技术也广泛应用于信息科学、系统工程、应用数学以及各种工程技术领域。

数据结构课程集中讨论软件开发过程中的设计阶段、同时设计编码和分析阶段的若干基本问题。此外,为了构造出好的数据结构及其实现,还需考虑数据结构及其实现的评价与选择。因此,数据结构的内容包括三个层次的五个“要素”,如图 1-5 所示。

层次 \ 方面	数据表示	数据处理
抽象	逻辑结构	基本运算
实现	存储结构	算法
评价	不同数据结构的比较及算法分析	

图 1-5　数据结构课程内容体系

数据结构的核心技术是分解与抽象。通过对问题的抽象，舍弃数据元素的具体内容，就得到逻辑结构。类似地，通过分解将处理要求划分成各种功能，再通过抽象舍弃实现细节，就得到运算的定义。上述两个方面的结合使我们将问题变换为数据结构。这是一个从具体（即具体问题）到抽象（即数据结构）的过程。然后，通过增加对实现细节的考虑进一步得到存储结构和实现运算，从而完成设计任务。这是一个从抽象（即数据结构）到具体（即具体实现）的过程。熟练地掌握这两个过程是数据结构课程在专业技能培养方面的基本目标。

数据结构作为一门独立的课程在国外是从 1968 年才开始的，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，他所著的《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。从 70 年代中期到 80 年代，各种版本的数据结构著作相继出现。目前，数据结构的发展并未终结，一方面，面向各专门领域中特殊问题的数据结构得到研究和发展，如多维图形数据结构等；另一方面，从抽象数据类型和面向对象的观点来讨论数据结构已成为一种新的趋势，越来越被人们所重视。

1.2　抽象数据类型

首先我们回顾一下在程序设计语言中出现的各种数据类型。

1.2.1　数据类型

数据类型是和数据结构密切相关的一个概念。它最早出现在高级程序设计语言中，用以刻划程序中操作对象的特性。在用高级语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。类型显式地或隐含地规定了在程序执行期间变量或表达式所有可能的取值范围，以及在这些值上允许进行的操作。因此，数据类型（Data Type）是一个值的集合和定义在这个值集上的一组操作的总称。

在高级程序设计语言中，数据类型可分为两类：一类是原子类型，另一类则是结构类型。原子类型的值是不可分解的。如 C 语言中整型、字符型、浮点型、双精度型等基本类型，分别用保留字 int、char、float、double 标识。而结构类型的值是由若干成分按某种结构组成的，因此是可分解的，并且它的成分可以是非结构的，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，也可以是数组等。在某种意义上，数据结构可以看成是“一组具有相

同结构的值”，而数据类型则可被看成是由一种数据结构和定义在其上的一组操作所组成的。

1.2.2 抽象数据类型

抽象数据类型(Abstruct Data Type，简称 ADT)是指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关。即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各种计算机都拥有的整数类型就是一个抽象数据类型，尽管它们在不同处理器上的实现方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

但在另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型，还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的重用性，在近代程序设计方法学中，要求在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作，并在模块的内部给出这些数据的表示及其操作的细节，而在模块的外部使用的只是抽象的数据及抽象的操作。这也就是面向对象的程序设计方法。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作组成，而数据结构又包括数据元素及元素间的关系，因此抽象数据类型一般可以由元素、关系及操作三种要素来定义。

抽象数据类型的特征是使用与实现相分离，实行封装和信息隐蔽。就是说，在抽象数据类型设计时，把类型的定义与其实现分离开来。

1.3 算法和算法分析

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。下面从算法特性、算法描述、算法性能分析与度量这三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。一个算法应该具有下列特性：

(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。算法的执行对应着的相同的输入仅有惟一的一条路经。

(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的特定的实现。一

个算法若用程序设计语言来描述，则它就是一个程序。

算法与数据结构是相辅相承的。解决某一特定类型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来体现。

要设计一个好的算法通常要考虑以下的要求。

(1) 正确。算法的执行结果应当满足预先规定的功能和性能要求。

(2) 可读。一个算法应当思路清晰、层次分明、简单明了、易读易懂。

(3) 健壮。当输入不合法数据时，应能作适当处理，不至引起严重后果。

(4) 高效。有效使用存储空间和有较高的时间效率。

1.3.2 算法描述

算法可以使用各种不同的方法来描述。

最简单的方法是使用自然语言。用自然语言来描述算法的优点是简单且便于人们对算法的阅读。缺点是不够严谨。

可以使用程序流程图、N-S图等算法描述工具。其特点是描述过程简洁、明了。

用以上两种方法描述的算法不能够直接在计算机上执行，若要将它转换成可执行的程序还有一个编程的问题。

可以直接使用某种程序设计语言来描述算法，不过直接使用程序设计语言并不容易，而且不太直观，常常需要借助于注释才能使人看明白。

为了解决理解与执行之间的矛盾，人们常常使用一种称为伪码语言的描述方法来进行算法描述。伪码语言介于高级程序设计语言和自然语言之间，它忽略高级程序设计语言中一些严格的语法规则与描述细节，因此它比程序设计语言更容易描述和被人理解，而比自然语言更接近程序设计语言。它虽然不能直接执行但很容易被转换成高级语言。

1.3.3 算法性能分析与度量

我们可以从一个算法的时间复杂度与空间复杂度来评价算法的优劣。

当我们将一个算法转换成程序并在计算机上执行时，其运行所需要的时间取决于下列因素：

(1) 硬件的速度。例如使用486机还是使用586机。

(2) 书写程序的语言。实现语言的级别越高，其执行效率就越低。

(3) 编译程序所生成目标代码的质量。对于代码优化较好的编译程序其所生成的程序质量较高。

(4) 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，这样一个特定算法的运行工作量的大小就只依赖于问题的规模(通常用正整数n表示)，或者说它是问题规模的函数。