计算机教学通用教材

计算机软件技术基础

JISUANJI RUANJIAN JISHU JICHU

钟 诚 主编

GUANGXI NORMAL UNIVERSITY PRESS
广西师范大学出版社

计算机教学通用教材

计算机软件技术基础

JISUANJI RUANJIAN JISHU JICHU

主编　钟　诚

编者(按姓氏笔画为序)

李春贵　陆向艳

杨　柳　唐培和

GUANGXI NORMAL UNIVERSITY PRESS

广西师范大学出版社

·桂林·

内容提要

本书针对程序设计、软件开发介绍了计算机科学和技术的基础知识，其中包括计算机技术核心课程《数据结构》和计算机科学技术专业课程《操作系统》、《软件工程》三方面的基本内容。重点讲授线性表、树、图、排序、查找、作业管理、进程管理、存储管理、设备管理、文件管理、主流操作系统简介、系统调查分析、软件计划、软件需求分析、软件设计、软件实现、软件测试、软件维护、面向对象的软件工程方法等知识。

本书内容全面、丰富、先进、科学、精练，重点突出，图文并茂，算法描述详尽，可读性强，理论和实践紧密结合。

本书可作为普通高等学校非计算机专业课程《软件技术基础》的教材，也可作为各类计算机技术培训班的教材以及广大工程技术人员学习软件技术基础知识的参考书。

图书在版编目（CIP）数据

计算机软件技术基础 / 钟诚主编. —2 版. —桂林：广西师范大学出版社，2005.3
计算机教学通用教材
ISBN 7-5633-2466-6

Ⅰ. 计… Ⅱ. 钟… Ⅲ. 软件—基本知识 Ⅳ. TP31

中国版本图书馆 CIP 数据核字（2005）第 014115 号

广西师范大学出版社出版发行
（广西桂林市育才路 15 号 邮政编码：541004
网址：http://www.bbtpress.com）
出版人：肖启明
全国新华书店经销
广西师范大学印刷厂印刷
（广西桂林市临桂县金山路 168 号 邮政编码：541100）
开本：787 mm ×1 092 mm 1/16
印张：20 字数：510 千字
2005 年 3 月第 2 版 2005 年 3 月第 1 次印刷
印数：0 001～1 000 册 定价：25.00 元

前　言

人类已经迈进21世纪。21世纪是信息的时代，是知识经济的时代。在这个充满机遇和挑战的时代，信息与我们息息相关。信息技术在国民经济建设、社会发展、国防、科学研究、教育等领域的作用日益重要。计算机技术与通信技术的融合极大地促进了信息化社会的发展，而信息技术的发展又为其他高新技术产业的发展起到带动和示范作用。信息技术的发展趋势是计算机、通信、信息应用三者的结合，软件则是结合的载体。因此，作为21世纪的大学生，学习、应用计算机软件技术显得尤为重要。

本书是在广西计算机等级考试委员会的组织和安排下，结合广西具体的实际情况和多年的教学实践编写而成。全书共17章，1～11章为数据结构，12～17为软件工程。第1章为数据结构概论；第2章详细介绍线性表、栈、队列和串等线性结构的存储表示及其相应操作的算法；第3章讲述树结构的数学性质、树的物理存储结构及树的应用；第4章讨论图的物理存储结构、图的遍历和连通算法；第5章重点介绍选择排序、交换排序、插入排序和归并排序技术；第6章讲授数据的顺序查找、折半查找、树型查找和Hash查找方法；第7章为操作系统概述；第8章讲述作业的建立、控制与调度，进程的组成、控制、调度、通信、死锁处理以及线程知识；第9章讨论存储器的分区管理、分页管理、分段管理机制，覆盖、交换以及虚拟存储技术；第10章介绍设备的控制、中断、缓冲、分配和假脱机技术，文件系统、文件目录管理、文件共享、文件保密与保护等问题；第11章介绍目前常用的Unix、Linux和Windows操作系统；第12章介绍软件工程的基本概念，软件可行性分析，人力资源、软硬件资源和软件成本计划的制定等内容；第13章讲述软件需求分析技术，包括数据流图、数据字典和加工逻辑的设计，以及软件需求规格说明书的编写方法；第14章主要讨论软件总体设计和详细设计的技术，以及优化软件结构的方法；第15章讲述如何选择程序语言和开发工具编写具有良好风格的程序代码，软件测试过程和测试方法，以及程序纠错技术等内容；第16章介绍软件维护的类型、问题和作用，以及软件维护的工作量估算、维护过程的管理；第17章简要介绍面向对象的软件工程方法。附录配上了相应的上机实验指导。

全书按照66学时的教学计划编写，其中打星号的章节为选学内容。本书由钟诚主编。第1～6章由钟诚和陆向艳编写，第7～10章由杨柳编写，第11章由陆向艳编写，第12～17章由李春贵编写，附录的上机实验指导由唐培和编写。全书由钟诚统稿、润色和校订。

本书的编写、出版，自始至终得到广西教育厅高教处、广西非计算机专业等级考试委

员会的指导和关怀，得到广西师范大学出版社的大力支持，我们深表谢意。本书的编写直接或者间接引用了专家、学者的有关文献，我们深表感谢。同时，感谢责任编辑张贻松先生以及林士敏教授、唐培和教授的热情鼓励和大力帮助，感谢我们的家人对本书写作的充分理解和大力支持，感谢其他有关人员为本书的规划、编辑、出版、发行所作出的卓有成效的工作。

我们希望本书的编辑、出版能为广西非计算机专业计算机教育事业的发展起到添砖加瓦的作用。由于编者学术水平有限，加之编写时间较紧，书中可能有错误、遗漏和不妥之处，敬请专家、学者和读者批评、指正。

编著者

2004 年 12 月

目　录

第1章　数据结构概论

自1946年第一台计算机研制成功以来，计算机科学与技术得到了飞速的发展，如今计算机技术已经在各行各业得到了广泛的应用。计算机的应用已不再仅仅是最初的科学计算，而更多地应用于信息管理、数据处理以及控制等非数值计算。计算机处理的对象已从数值发展到字符、声音和图像等。计算机应用范围的广泛性和所处理问题的复杂性对程序设计提出了新的要求。为了更有效地使用计算机设计出高效率、高可靠性的软件，人们不仅需要研究程序的结构与算法，也需要对数据的组织、数据的逻辑关系以及数据在计算机内部如何表示和操作进行深入的研究。

1.1　数据结构的定义

数据是计算机程序处理的对象。抽象地说，数据是客观事物的符号描述，而这种描述是采用计算机能够识别、存储和处理的形式来进行的。这种描述一般可以归结为数字、字符或符号的集合。例如，财务账本中的数量、单价、金额等是数值数据，而科目、摘要、凭证号、日期等则是字符数据。会计凭证是进行会计处理的原始依据。

数据元素是数据的基本单位，是计算机进行输入输出操作的最小单位。在大多数情况下，一个数据元素由若干个数据项组成。表1.1描述了某个班级学生的成绩，每个学生的数据为一条记录，每条记录包括学号、姓名、C语言、软件基础、数据结构五个数据项。

表1.1　学生成绩

学　号	姓　名	C语言	软件基础	数据结构
6201001	张　三	85	74	92
6201002	李　四	92	84	62
6201003	王　五	87	88	78
6201004	刘俐莉	90	80	88
…				

数据结构中要研究的数据不是某个具体的、孤立的数据，而是大量的相互关联的数据。数据元素之间存在的相互关系就叫作结构，而数据元素之间抽象化的相互关系称为数据结构。结构定义中的“关系”描述的是数据元素之间的逻辑关系，因此又称为数据的

逻辑结构。

数据结构的形式定义为数据结构是一个二元组

$$DS = (D, R)$$

这表示 DS 是一种数据结构，它由数据元素集合 D 和在 D 上定义的一种二元关系的集合 R 组成，其中：

$$D = \{D_i \mid 0 \leqslant i \leqslant n, \quad n \geqslant 0\}$$

$$R = \{R_j \mid 0 \leqslant j \leqslant m, \quad m \geqslant 0\}$$

D_i 表示第 i 个数据，n 为 DS 中数据元素的个数；R_j 表示数据元素之间的第 j 个关系，m 为 D 之关系的个数。

D 上的二元关系 R 是序偶的集合。对于 R 中的任一序偶 $<x, y>$ $(x, y \in D)$，称 x 为序偶的第一元素（或 y 的前驱），称 y 为序偶的第二元素（或 x 的后继）。

例如，我们要用计算机来处理复数，则可以定义复数是如下的一种数据结构：

$$\text{Complex} = (C, R)$$

其中：C 是包含两个实数的集合 $\{c1, c2\}$；$R = \{P\}$，P 是定义在集合 C 上的一种关系 $\{<c1, c2>\}$，其中有序偶 $<c1, c2>$ 表示 $c1$ 是复数的实部，$c2$ 是复数的虚部。

数据之间的结构关系是数据结构的重要研究内容。例如，对于某个城市的电话号码簿，它记录了 n 个用户的名字和相应的电话号码。假定它按如下形式安排：

$$(a_1, b_1)(a_2, b_2)\cdots(a_n, b_n)$$

其中 (a_i, b_i) $(i = 1, 2, \cdots, n)$ 分别表示第 i 个用户的名字和对应的电话号码。现要求设计一个算法（程序），当任意给定一个用户的名字时，算法能够输出此用户的电话号码，若电话号码簿中没有此用户，则算法报告没有此用户的信息。

如果用户名字和对应的电话号码在计算机中的排列次序没有任何规律，则只能采用从头到尾逐一比较的方法即顺序查找方法来解决此问题。这种方法是相当费时的，尤其是对于具有很多用户的系统更是如此。

但是，如果将电话号码簿经过适当的组织，比如按字典顺序排列用户名字及其电话号码，就可以大大节省查找时间。例如，若用户名字的第一个字母为 S，则只需查找以 S 开头的那些名字即可，而另外 25 个字母开头的名字就无需查找了。

由此可见，数据的结构直接影响着算法的选择和效率。

上述的电话号码簿就是一种数据结构问题。在具体实现时，可以将名字及对应的电话号码这种数据设计成二维数组形式，或者设计成记录数组结构，也可以设计成链表结构。

假定用户名字和对应的电话号码在数据逻辑上安排成 n 元向量形式：

$$((a_1, b_1)(a_2, b_2)\cdots(a_n, b_n))$$

对于这样的电话号码簿，当需要增加一个用户及其电话号码，或者注销某个用户的电话号码时，就需要在上述安排好的结构上进行插入或删除数据的操作。这样，就会产生一系列的问题：新增的用户名字和电话号码应插入到什么地方？是插入到前头，还是插入到末尾，或者插入到中间某个合适的位置？插入后，对原来的数据是否有影响？有何影响？另外，删除某个用户的名字及其电话号码之后，原来的数据是否移动？若需要移动，则应如何移动？这些问题说明为了适应数据的增长或减少的需要，还必须对这样的数据结构定义一些操作（运算）及相应的算法。

在设计算法(程序)时,不仅要考虑数据的逻辑结构及其操作(运算),还要考虑数据在计算机中的存储方式,即数据的物理结构(存储结构)。一种逻辑结构可以通过映象得到与它相应的存储结构。

这样,我们可以定义数据结构为研究数据的逻辑结构、物理结构以及它们之间的相互关系,对这种结构定义相应的操作(运算)和设计出相应的算法,并且确保经过这些操作(运算)后所得到的新结构仍然是原来的结构类型。

在数据结构中往往涉及数据类型和数据对象的概念,它们是不同的。数据类型是指某种程序设计语言所允许使用的变量种类,例如,在 FORTRAN 语言中,变量的数据类型有整型、实型、布尔型、双精度型、复型等;在 C 语言或者 PASCAL 语言中,除了标准的整型、实型、布尔型和字符型之外,还允许用户自己根据需要定义构造类型(记录类型)等。一个数据类型不仅定义了一个变量可以取值的范围,而且还规定了对变量允许进行的运算及其规则,例如,在 C 语言中布尔型变量的取值可为 TRUE 或 FALSE,而对布尔型变量进行的运算有 AND、OR、NOT 等。数据对象则是某种数据类型的数据元素的集合,例如,整数的数据对象是 $I = \{0, \pm 1, \pm 2, \cdots\}$,英文字符的数据对象是 $CHAR = \{A, B, \cdots, Z\}$。数据对象可以是有限的,也可以是无限的。

由此可见,数据结构、数据类型、数据对象是三个不同的概念。数据结构不仅要描述数据类型的数据对象,而且还要描述数据对象各元素之间的相互关系。数据结构类型包括数组、记录、栈、队列、链表、树、图和文件等。

数据结构是一门以程序设计、离散数学等为基础的理论性和实践性都较强的计算机专业核心课程。它的主要研究范围包括:研究各种数据结构的性质,即不仅研究数据的逻辑结构和物理结构,而且还要对每种结构定义相适应的操作(运算),并应用某种高级程序设计语言设计出各种操作(运算)的算法,分析算法的效率,同时也研究各种数据结构在计算机科学技术中的应用。

1.2 算法的描述与分析

1.2.1 算法的描述

研究数据结构,必定要研究其中定义的操作(运算)。简单地说,算法就是求解某个特定问题的步骤。通常所说的算法是指在计算机上执行的计算过程的抽象描述,它和计算机程序不同,程序是在特定的计算机上执行算法。算法是抽象的,它与具体的计算机无关。

为了使读者更好地理解和掌握算法的思想和实质,选择一种合适的程序设计语言描绘算法就是一个重要的问题。在本书中,我们选择 C 语言和类 C 语言相结合的方式来描述算法。这些算法稍加修改或扩充就可以变成能在机器上直接运行的 C 语言程序。

通常,求解一个问题 P 可能有几种算法,因此要考虑:

① 哪一个是求解问题 P 的“好”的算法?

为了回答这个问题,需要比较不同算法的有效性。

② 什么是求解问题 P 所需的基本运算的最少次数?

可以通过比较求解问题 P 的所有算法的基本运算次数来确定。

③ 在指定的适当时间内是否有求解问题 P 的算法?

上述问题一直是计算机科学技术研究的核心问题之一。

1.2.2 算法评价和算法分析

评价、判断一个算法优劣的标准有:

① 正确性。这是评价算法优劣的前提。一个算法是正确的,其意思是指当输入合法数据时,能在有限的运行时间内得到正确的结果。

② 简明性。算法思路清晰、层次分明、易读易用、简单明了。

③ 占用存储空间少。算法占用存储空间的大小称为算法的空间复杂度。

④ 运行速度快。运行算法所需的时间称为算法的时间复杂度。

如何比较两个算法的运行速度呢? 本节将着重讨论这个问题。

在 20 世纪 60 年代,对算法的有效性没有一个令人满意的客观标准。一个研究者可能会在杂志上发表一个算法及对于少数实例运行并得到一个执行时间,而几年后,第二个研究者又会给出一个改进了的算法以及对于同样实例运行并得到另一个执行时间。新的算法肯定更快,因为在相隔这几年里,计算机性能和程序设计语言都得到改善和提高。算法在不同的计算机上用程序设计语言编程运行,编程语言也可能不同,因此这种比较不能令人满意。一方面,要弄清计算机性能提高与编程实现技术对算法执行时间的影响是很困难的;另一方面,有可能第二个研究者无意中将算法搞得对这些实例运行起来特别有效。可以想象如果对于另外的实例再运行这两个算法,则可能第一个算法的运行速度更快。

因此,对算法的分析必须脱离具体的计算机和程序设计语言来进行。

比较两个算法的好坏,首先是看其所需的运行时间,时间的长短是由算法所需的运算次数来决定的。算法的运算次数又称为工作量或者称为算法的时间复杂度。任何一个算法都可能有多种运算(操作),但不可能对所有运算的量都作统计。因此,只能抓住其中影响算法运行时间最长的一种或几种运算作为基本运算。例如,在一个用户名字和电话号码的线性表中,对于查找用户 X 相应电话号码的问题,是将 X 和表中元素(用户名字)的比较运算作为算法的基本运算。又如在两个实数矩阵相乘的问题中,则是把两个实数相乘的运算作为算法的基本运算。但是,即使是同一个问题,对于不同的输入,其基本运算的次数也可能不同,它与输入元素的个数有关。因此,我们引入问题规模的概念。例如,在一个用户名字和电话号码的线性表中,对于查找用户 X 相应电话号码的问题,问题的规模用表中数据(用户名字,电话号码)的个数来表示。对于两个实数矩阵相乘的问题,问题规模是用矩阵的阶来表示。

这样,一个算法的基本运算的次数就可以用问题大小的函数 $f(n)$ 来表示。$f(n)$ 是自然数 $N\to\infty$ 的一个函数。

对于算法的好坏可以进行最坏情形和平均情形分析。

① 算法的平均情形复杂度。

设 D_n 是问题大小为 n 的输入集合,I 是 D_n 的一个元素,$P(I)$ 是 I 出现的概率,而 $T(I)$ 是算法在输入 I 时所需的执行次数。则算法的平均情形复杂度定义为:

$$A(n)=\sum_{I\in D_n}P(I)T(I)$$

② 算法的最坏情形复杂度。

设 D_n 是问题大小为 n 的输入数据的集合，I 是 D_n 的一个元素，$T(I)$ 是算法输入 I 时所需的执行次数。则算法的最坏情形复杂度定义为：

$$W(n) = \max\{T(I): I \in D_n\}$$

$W(n)$ 给出了算法时间复杂度的上界，用它来估算执行一个算法的时间上界，这在实际应用中特别重要。因此，对于本书中给出的大多数算法将进行最坏情形复杂性分析，当说到一个算法的时间复杂度时，除非特别说明，否则均指最坏情形时间复杂度。

【例 1.1】设 L 是一个包含 n 个整数元素的表，对某一指定元素 z，如果 z 在表中，则输出其下标，如果 z 不在表中，则输出零。用 C 语言描述的算法如下。

```
int Serial_Search(L,n,z)
  int L[ ];
  int n;
  {
   int j;
   j=0;
   while((j<=n-1)&&(L[j]!=z))
      j=j+1;
   if(j>n-1)
     j=0;
   return(j);
}
```

上述算法所用到的运算有赋值、加法和比较。因此，其基本运算为比较操作。

显然，在最坏的情形下，z 或者不出现在表中或者为表中最后一个元素。此时，算法所作的比较次数是 n 次，即 $f(n)=n$，即算法的时间复杂度为 n 的线性函数。

【例 1.2】考虑计算 $f=1+2!+3!+\cdots+n!$ 的问题。用 C 语言描述的算法如下。

```
long int Factor Sum(n)
int n;
{
  int i;
  long int f,w;
  f=0
  for(i=1;i<=n;i++)
  {
      w=1;
      for(j=1;j<=i;j++)
          w=w*j;
      f=f+w;
  }
  return(f);
}
```

上述算法所用到的运算有乘法、加法、赋值和比较，其基本运算为乘法操作。

在上述算法的执行过程中，对外循环变量 i 的每次取值，内循环变量 j 循环 i 次。因为内循环每执行一次，内循环体语句 $w=w*j$ 只作一次乘法操作，即当内循环变量 j 循环 i 次时，内循环语句 $w=w*j$ 作 i 次乘法。所以，整个算法所作的乘法操作总数是：

$$f(n)=1+2+3+\cdots+n=n(n+1)/2$$

衡量两个算法的好坏主要是比较它们工作量的大小，但当 n 较小时难以看出谁优谁劣。例如，对于某个问题 P，其第一个算法的工作量是 $w_1=3n^2$，第二个算法得工作量是 $w_2=25n$。当 $n=8$ 时，$w_1(8)=192$，$w_2(8)=200$，显然，$w_1(8)<w_2(8)$，即 w_1 的运算次数比 w_2 的运算次数少。但是，不能据此立即断定第一个算法就一定比第二个算法好。这是因为当 $n=9$ 时，$w_1(9)=243$，$w_2(9)=225$，而且当 $n>9$ 时都有 $w_1(n)>w_2(n)$，即第一个算法的工作量比第二个算法的大。

因此，衡量两个算法的好坏，应当是在 n 足够大的情形下对算法的工作量进行比较，即对算法进行渐近性态分析。

设 $f(n)$ 和 $g(n)$ 是定义域为自然数集 **N** 的非负函数，如果存在正整数 c 和 n_0，使得当 $n\geqslant n_0$ 时，有 $f(n)\leqslant cg(n)$，则称函数 $f(n)$ 的阶低于或等于函数 $g(n)$ 的阶。记为“$f(n)$ 是 $O(g(n))$”或“$f(n)\in O(g(n))$”，读作 $f(n)$ 是 $g(n)$ 的大 O。

【例 1.3】设 $f(n)=3n^3+2n^2$，$g(n)=5n^3$，取 $c=5$，$n_0=0$，则当 $n\geqslant n_0$ 时，有 $3n^3+2n^2\leqslant 5n^3$，所以 $f(n)\in O(n^3)$。

【例 1.4】设 $f(n)=3^n$，$g(n)=2^n$，则 $f(n)$ 不是 $O(2^n)$。因为若假设存在常数 c 和 n_0，使得当 $n\geqslant n_0$ 时，有 $3^n\leqslant c2^n$，则有 $c\geqslant(3/2)^n$ 对于一切的 $n\geqslant n_0$ 成立。但是，实际上随着 n 的增大，$(3/2)^n$ 可以大于任意预先指定的常数 c，这说明所假设的常数 c 并不存在。

用定义判断两个算法时间复杂度函数 $f(n)$ 和 $g(n)$ 的阶比较繁琐，可以采用求极限的方法来比较：

若 $\lim\limits_{n\to\infty} f(n)/g(n)=c$，

则

① 当 $c\neq 0$ 时，表示 $f(n)$ 和 $g(n)$ 同阶，记为 $f(n)=\Theta(g(n))$。

② 当 $c=0$ 时，表示 $f(n)$ 比 $g(n)$ 低阶，记为 $f(n)\in O(g(n))$。

③ 当 $c=\infty$ 时，表示 $f(n)$ 比 $g(n)$ 高阶，记为 $f(n)=\Omega(g(n))$。

只要求出极限，就可以方便地判断任意两个时间复杂度函数阶之间的关系。

【例 1.5】极限 $\lim\limits_{n\to\infty}(n^2/2)/(307n^2)=1/614$，所以 $f(n)=n^2/2$ 与 $g(n)=307n^2$ 同阶。

【例 1.6】极限 $\lim\limits_{n\to\infty}\log_2 n/n=\lim\limits_{n\to\infty}\ln n/(n\ln 2)=(1/\ln 2)\lim\limits_{n\to\infty}((1/n)*\ln n)=0$，所以 $f(n)=\log_2 n$ 比 $g(n)=n$ 低阶。

从上述分析可以看出，比较两个算法的优劣，并不是要精确统计其各种运算次数，而是要比较其工作量的阶。因此，把算法工作量 $f(n)$ 表示为 $O(g(n))$，其中大写字母 O 为英文单词 Order（即数量级）的第一个字母。

当 $f(n)$ 数量级与对数函数、幂函数或它们的乘积同阶时，算法的运行时间是可以接受的，称这些算法为有效算法。当 $f(n)$ 为指数函数或阶乘函数时，算法的运行时间随 n 的增大而变得不可接受，这类算法不是有效算法。

随着 n 的增大，对数函数的增长速度最慢，线性函数次之，其余类推。当 n 足够大后，各种不同数量级的 $f(n)$ 函数存在下列关系：

$$O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < \cdots < O(2^n) < O(n!)$$

算法时间复杂度除了与问题的规模 n 有关外，还与输入的具体数据和数据的输入次序即与数据的分布有关。

分析算法的目的是为了改进算法。对上述例 1.2 的算法，在算法的第三步，每次都做 i 次乘法，这是不必要的。只要把 $(i-1)!$ 保存起来，下一次循环时只需做一次乘法就可以了。因此，改进的算法如下。

```
long int Update_Factor_Sum(n)
  int n;
    {
      int i;
      long int f,w;
      f=0;
      w=1;
      for (i=1;i<=n;i++)
          {
            w=w*i;
            f=f+w;
          }
      return(f);
    }
```

这样，上述算法所做的乘法次数只有 n 次，即 $f(n)=n$，$f(n)\in O(n)$，而例 1.2 中未改进的算法的时间复杂度为 $f(n)=n(n+1)/2\in O(n^2)$，它们相差整整一个数量级。

1.3 本章小结

本章从程序设计的角度出发，着重讨论数据结构的定义及其在程序设计中的作用。一般来说，用计算机来解决一个实际问题，其步骤主要是：首先从实际问题抽象出一个适当的数学模型，然后设计一个解此数学模型的算法，最后编出程序并调试直至得到最终的答案。这里讲的数学模型就是数据结构。数据结构是一门研究非数值计算的程序设计问题中计算机的操作对象以及它们之间关系和操作的学科。本书将在后续的相关章节中讨论线性、树型和图状这三种常见的数据结构。同一问题有各种不同的算法，我们总是希望选择一个好算法，通过算法分析可以判断一个算法的好坏，一般以最坏、平均或渐进情形复杂性作为评判依据。

习题1

1. 简述下列概念：数据、数据元素、数据类型、数据结构、逻辑结构和物理结构。

2. 试举一个数据结构的例子，叙述其逻辑结构、物理结构和操作三个方面的内容。

3. 什么是算法？请简述评判一个算法优劣的标准。

4. 设有两个算法在同一机器上运行，其执行时间分别为 $100n^2$ 和 2^n，要使前者快于后者，n 至少要多大？

5. 请按增长率由小至大的顺序排列下列各函数：

$$(2/3)^n,(3/2)^n,n^n,n!,2^n,n\lg n,n^3/2,n^2。$$

6. 试设计一个算法，按自大到小依次输出顺序读入的三个整数 X，Y 和 Z 的值。

7. 从阶的意义上分析下列程序段的时间复杂度：

(1)
```
i=1;
k=0
while(i<n)
  {
    k=k+10*i;
    i++;
  }
```

(2)
```
i=0;
k=0;
do {
    k=k+10*i;
    i++;
  }
while(i<n);
```

(3)
```
i=1;
j=0;
while(i+j<=n)
{
  if (i<j)
    j++;
  else
   i++;
}
```