第 1 章　概　　论

“数据结构”是计算机专业的核心课程，本章介绍它研究的内容、使用的术语，以及算法的度量标准。

1.1　引言

1.1.1　什么是数据结构

数据结构包含两方面的内容：一是构成集合的数据元素，二是数据元素之间存在的关系。数据结构也就是带有结构的数据元素的集合，结构指的是数据元素之间的相互关系，即数据的组织形式。由此可见，计算机所处理的数据并不是数据的杂乱堆积，而是具有内在联系的数据集合。如表 1-1 所示的成绩表就是一个数据结构（线性表），其中每一行表示一个数据元素，表中的所有行构成了一个数据元素集合，数据元素之间具有线性结构关系（详见第 2 章）。如图 1-1 所示是一软件公司员工职务组织结构图，其中每一个方框表示一位员工的信息（如职工号、姓名、性别、年龄和电话等），即一个数据元素，全部员工的信息构成了一个数据元素集合，数据元素之间具有树形结构关系（详见第 5 章）。如图 1-2 所示是城市交通示意图，其中每一个顶点表示一座城市，即一个数据元素，全部顶点构成了一个数据元素集合，每一条边表示两座城市间的交通线路，之间的距离用边上的值表示，数据元素之间具有图状结构关系（详见第 6 章）。

表 1-1　成绩表

学　　号	姓　　名	性　　别	数　　学	数据结构	英　　语	计算机组成原理
0504028	秦占军	男	88	76	78	86
0504029	武晓云	女	90	82	84	79
0504030	关国江	男	92	90	86	88
…						

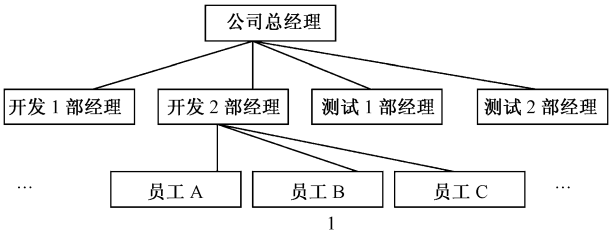

图 1-1　公司员工职务组织结构图

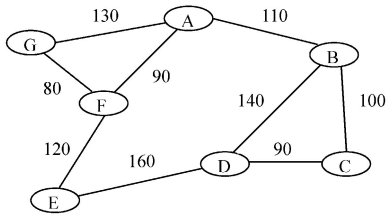

图 1-2　城市交通示意图

1.1.2　数据结构研究什么

数据结构可定义为一个二元组：Data_Structure=(D,R)

其中 D 是数据元素的有限集合，R 是 D 上关系的有限集合。

数据结构具体应包括 3 个方面：数据的逻辑结构、数据的存储结构和数据的运算集合。

1．逻辑结构

数据的逻辑结构是指数据元素之间逻辑关系的描述。

根据数据元素之间关系的不同特性，有下列 4 种基本的逻辑结构，如图 1-3 所示。

（1）集合结构。结构中的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。结构中的数据元素之间存在着一对一的线性关系。

（3）树形结构。结构中的数据元素之间存在着一对多的层次关系。

（4）图状结构或网状结构。结构中的数据元素之间存在着多对多的任意关系。

由于集合的关系非常松散，因此可以用其他的结构代替。本书所讨论数据的逻辑结构如图 1-4 所示。

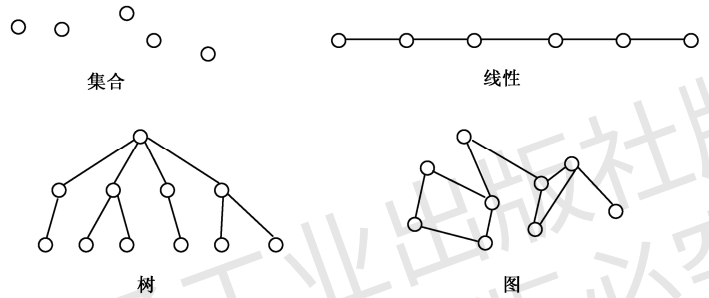

图 1-3　4 种基本逻辑结构

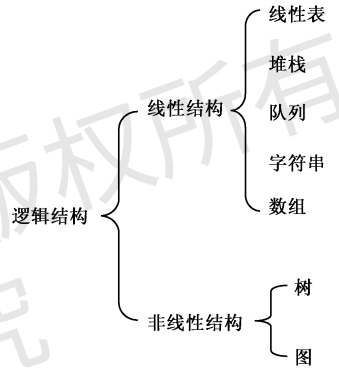

图 1-4　本书所讨论数据的逻辑结构

2．存储结构

存储结构（又称物理结构）是逻辑结构在计算机中的存储映像，是逻辑结构在计算机中的实现（或存储表示），它包括数据元素的表示和关系的表示。有数据结构 Data_Structure=(D,R)，对于 D 中的每一个数据元素都对应有存储空间中的一个单元，D 中全部元素对应的存储空间必须明显或隐含地体现关系 R。逻辑结构与存储结构的关系是，存储结构是逻辑结构的映像与元素本身的映像。逻辑结构是抽象，存储结构是实现，两者综合起来建立了数据元素之间的结构关系。

数据元素之间的关系在计算机中有两种不同的表示方法，即顺序映像（顺序存储结构）与非顺序映像（非顺序存储结构）。数据结构在计算机中的映像，包括数据元素映像和关系映像。关系映像在计算机中可用顺序存储结构或非顺序存储结构两种不同方式来表示。

3．运算集合

讨论数据结构的目的是为了在计算机中实现所需的操作，施加于数据元素之上的一组操作构成了数据的运算集合，因此在结构上的运算集合是数据结构很重要的组成部分。

以表 1-1 所示的成绩表为例，该表的数据元素之间是一种简单的线性关系，所以逻辑结构采用线性表。存储结构既可采用顺序存储结构，也可采用非顺序存储结构。对于成绩表，

当学生退学或转出时要删除相应的数据元素，转入学生时要增加数据元素，发现成绩输入错误时要修改。这里的增加、删除和修改就构成了数据的操作集合。

综上所述，数据结构的内容可归纳为 3 个部分：逻辑结构、存储结构和运算集合。按某种逻辑关系组织起来的一批数据，依一定的映像方式把它存放在计算机存储器中，并在这些数据上定义一个运算的集合，就构成了一个数据结构。

“数据结构”是一门主要研究怎样合理地组织数据，建立合适的数据结构，提高计算机执行程序所用的时、空效率的学科。数据结构课程不仅讲授数据信息在计算机中的组织和表示方法，同时也训练用高效的设计算法解决复杂问题的能力。“数据结构”属于计算机专业的核心专业基础课，对于程序设计者来说掌握该课程的知识是非常必要的，数据结构贯穿程序设计的始末，缺乏数据结构功底的人，很难设计出高水平的应用程序。

1.2 数据结构的基本概念

1．数据

数据是描述客观事物的数值、字符，以及所有其他能输入到计算机中，且能被计算机处理的各种符号的集合。简言之，数据就是计算机化的信息（或存储在计算机中的信息）。

2．数据元素与数据项

数据元素是组成数据的基本单位，是数据集合的个体，在计算机中通常作为一个整体进行考虑和处理。数据元素可由一个或多个数据项组成，数据项是具有独立含义的数据的最小单位（不可再分割）。在如表 1-1 所示的成绩表中，每一个学生的信息（每一行）是一个数据元素，每一个数据元素包含学号、姓名等数据项。

3．数据对象

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。例如，整数数据对象是集合 N={0,±1,±2,…}，字母字符数据对象是集合 C={'A','B',…,'Z'}。表 1-1 中的成绩表也可看做一个数据对象，由此可以看出，不论数据元素集合是无限集（如整数集）、有限集（如字符集），还是由多个数据项组成的复合数据元素（如成绩表），只要性质相同，都是一个数据对象。

4．数据类型

数据类型是一组性质相同的值集合，以及定义在这个值集合上的一组操作的总称。数据类型中定义了两个集合，值集合确定了该类型的取值范围，操作集合确定了该类型中允许使用的一组运算。在高级语言中有很多数据类型，数据类型是高级语言中允许的变量种类，是程序语言中已经实现的数据结构（程序中允许出现的数据形式）。例如，在高级语言中的整型类型，它可能的取值范围是−32 768～+32 767，可进行的运算为加、减、乘、除、乘方和取模。

按“值”的不同特性，高级程序语言中的数据类型可分为两大类。第一类是非结构的原子类型，它的值是不可分解的，如 C 语言中的标准类型（整型、实型和字符型）及指针。第二类是结构类型，它的值是由若干成分按某种结构组成的，是可以分解的，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。

1.3 算法和算法的分析

1.3.1 算法及算法的描述

算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或多个操作。算法具有下列 5 个重要特性。

（1）有穷性。一个算法必须总是（对任何合法的输入值）在执行有穷步之后结束，且每一步都可在有穷时间内完成。

（2）确定性。算法中每一条指令必须有确切的含义，不能产生二义性。在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

（3）可行性。一个算法是可行的，即算法中描述的操作可通过已经实现的基本运算执行有限次来实现。

（4）输入。一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。

（5）输出。一个算法有一个或多个输出。这些输出是与输入有着一定关系的量。

“数据结构”中所讨论的算法，可用不同的方式进行描述，常用的有类 Pascal、类 C、类 C++、类 Java 程序设计语言，本教材以类 C 语言为描述工具。每一算法可用一个或多个简化了的 C 语言（类 C）函数进行描述，简化是针对高级语言的语法细节所做的，如对函数内部的局部变量可不做声明而直接使用，交换两个变量 x、y 的值，不使用 3 条赋值语句，而仅简记为一条语句 x←→y；再如对结构体变量可以整体赋值，等等。需要注意的是，“数据结构”中的算法，因为用类 C 描述，所以不等同于 C 语言程序。若要上机运行某一算法，则必须将其完善为 C 语言程序，即增加 main()函数，在 main()函数中增添实现算法的函数调用语句，在实现算法的函数中补充、完善语法细节。

1.3.2 算法设计的要求

1．正确性

正确性的含义是算法对于一切合法的输入数据都能够得出满足规格说明要求的结果，事实上要验证算法的正确性是极为困难的，因为通常情况下合法的输入数据量太大，用穷举法逐一验证是不现实的。所谓的算法正确性是指算法达到了测试要求。

2．可读性

可读性是指人对算法阅读理解的难易程度，可读性高的算法便于人与人之间的交流，有利于算法的调试与修改。

3．健壮性

对于非法的输入数据，算法能给出相应的响应，而不是产生不可预料的后果。

4．效率与低存储量需求

效率指的是算法的执行时间。对于解决同一问题的多个算法，执行时间短的算法效率

高。存储量需求指算法执行过程中所需要的最大存储空间。效率与低存储量需求两者都与问题的规模有关，求 100 个人的平均分与求 1 000 个人的平均分显然不同。

1.3.3 算法的分析

1．算法效率的度量

算法执行的时间是其对应的程序在计算机上运行所消耗的时间。程序在计算机上运行所需时间与下列因素有关：

（1）算法本身选用的策略。

（2）书写程序的语言。

（3）编译产生的机器代码质量。

（4）机器执行指令的速度。

（5）问题的规模。

度量一个算法的效率应抛开具体机器的软、硬件环境，而书写程序的语言、编译产生的机器代码质量、机器执行指令的速度都属于软、硬件环境。对于一个特定算法只考虑算法本身的效率，而算法自身的执行效率是问题规模的函数。对同一个问题，选用不同的策略就对应不同的算法，不同的算法对应有各自的问题规模函数，根据这些函数就可以比较（解决同一个问题的）不同算法的优劣。

2．算法的时间复杂度

一个算法的执行时间大致上等于其所有语句执行时间的总和，语句的执行时间是指该条语句的执行次数和执行一次所需时间的乘积。语句执行一次实际所需的具体时间是与机器的软、硬件环境（机器速度、编译程序质量和输入数据等）密切相关的，与算法设计的好坏无关。所以，可以用算法中语句的执行次数来度量一个算法的效率。

首先定义算法中一条语句的语句频度。语句频度是指语句在一个算法中重复执行的次数。以下给出了两个 n×n 阶矩阵相乘算法中的各条语句，以及每条语句的语句频度。

语句	语句频度
for (i=0; i<n; i++)	n+1
for (j=0; j<n; j++)	n(n+1)
{ c[i][j]=0;	n^2
for (k=0;k<n; k++)	$n^2(n+1)$
c[i][j]=c[i][j]+a[i][k]*b[k][j];	n^3
}	

算法中所有语句的总执行次数为 $T_n=2n^3+3n^2+2n+1$，从中可以看出，语句总的执行次数是问题的规模（矩阵的阶）n 的函数 f(n)（$T_n= f(n)$）。进一步简化，可用 T_n 表达式中 n 的最高次幂，即最高次幂项忽略其系数来度量算法执行时间的数量级，称为算法的时间复杂度，记做：

$$T(n)=O(f(n))$$

以上算法的时间复杂度为 $T(n)=O(n^3)$。

算法中所有语句的总执行次数 T_n 是问题规模 n 的函数，即 $T_n= f(n)$，其中 n 的最高次幂项与算法中称为原操作的语句的频度对应，原操作是实现算法基本运算的操作，上面算

法中的原操作是 c[i][j]=c[i][j]+a[i][k]*b[k][j]。一般情况下，原操作由最深层循环内的语句实现。

T(n)随 n 的增大而增大，增长得越慢，其算法的时间复杂度越低。下列 3 个程序段中分别给出了原操作 count++的 3 个不同数量级的时间复杂度。

（1）count++;

其时间复杂度为 O(1)，称为常数阶时间复杂度。

（2）for (i=1; i<= n; i++)

```
    count++;
```

其时间复杂度为 O(n)，是线性阶时间复杂度。

（3）for (i=1; i<= n; i++)

```
    for (j=1;j<= n; j++)
        count++;
```

其时间复杂度为 $O(n^2)$，是平方阶时间复杂度。

又如以下两个算法，它们所呈现的时间复杂度分别是 $O(\log_2 n)$和 $O(n\times m)$。

（1）i=1；

```
while (i<n)
    i=2*i;
```

（2）for (i=0; i<n; i++)

```
for (j=0; j<m; j++)
    a[i][j]=0;
```

3．最坏时间复杂度

算法中基本操作重复执行的次数还随问题的输入数据集的不同而不同，例如下面的冒泡排序算法：

```
void Bubble(int a[], int n)
/*对整数数组 a 中的 n 个元素从小到大排序*/
{   int i=0, j;
    int change;
    do
    {   change=0;
        for (j=0; j<n−i−1; j++)
        if( a[j]>a[j+1])
            {
                a[j] ←→ a[j+1]; /*交换序列中相邻的两个整数*/
                change=1;
            }
        i=i+1;
    }while (i<n−1 && change )
}
```

在这个算法中，“交换序列中相邻的两个整数”（a[j]←→a[j+1]）为原操作。当 a 中初始序列为自小到大有序时，原操作的执行次数为 0；当初始序列为自大到小有序时，原操作的

执行次数为 n(n−1)/2。对于这类算法时间复杂度的分析，一种解决的方法是计算它的平均值，即考虑它对所有可能输入数据集的期望值，此时相应的时间复杂度为算法的平均时间复杂度。然而在很多情况下，算法的平均时间复杂度是难以确定的，通常的做法是讨论算法在最坏情况下的时间复杂度。例如，冒泡排序在最坏情况下（初始序列为自大到小有序时）的时间复杂度就为 $T(n)=O(n^2)$。本教材中，如不进行特殊说明，所讨论的各算法的时间复杂度均指最坏情况下的时间复杂度。

4．常见的时间复杂度

常见的时间复杂度有：O(1)常数阶、O(n)线性阶、$O(n^2)$平方阶、$O(n^3)$立方阶、$O(2^n)$指数阶、$O(\log_2 n)$对数阶和线性对数阶 $O(n\log_2 n)$。常用的时间复杂度（从小到大排列）的比较如表 1-2 所示。

表 1-2　常用的时间复杂度的比较

$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	5 096	65 536
5	32	160	1 024	32 768	2 147 483 648

5．算法的空间复杂度

关于算法的存储空间需求，类似于算法的时间复杂度，采用空间复杂度作为算法所需存储空间的量度，记为

$$S(n)=O(f(n))$$

其中 n 为问题的规模。一般情况下，一个程序在机器上执行时，除了需要寄存程序本身所用的指令、常数、变量和输入数据以外，还需要一些对数据进行操作的辅助存储空间。其中对于输入数据所占的具体存储量只取决于问题本身，与算法无关，这样只需要分析该算法在实现时所需要的辅助空间单元数就可以了。若算法执行时所需要的辅助空间相对于输入数据量而言是个常数，则称这个算法为原地工作，辅助空间为 O(1)。如果所占辅助空间量依赖于特定的输入，则除特别指明外，均按最坏情况来分析。

算法的执行时间和存储空间的耗费是一对矛盾体，即算法执行的高效通常是以增加存储空间为代价的，反之亦然。不过，就一般情况而言，常常以算法执行时间作为算法优劣的主要衡量指标。本教材对算法的空间复杂度不做进一步讨论。

1.4　总结与提高

1．数据的逻辑结构与存储结构

数据结构包括数据的逻辑结构、存储结构以及对数据施加的一组操作（运算集合）。逻辑结构定义了数据间的逻辑关系，数据的存储结构（物理结构）则是这种关系在计算机内部的表示（实现）。同一种逻辑关系可以有不同的存储表示，比如一个班的成绩表，既可以用

数组（顺序结构）存放，也可以用链表（非顺序结构）存放。对于不同的存储结构自然对应不同的算法实现，比如在班级成绩表中查找“关国江”的数据结构课程成绩，就有数组结构上的查找算法与链表结构上的查找算法。

2．算法的时间复杂度

对于一个特定算法只考虑算法本身的时间效率，可用算法中所有语句总的执行次数来度量，进一步可用其中一条执行次数最多的语句的执行次数（语句频度）来度量，它是问题规模（比如矩阵的阶）n 的函数。比如输入一维数组 a[n]，其对应的时间复杂度是问题规模 n 的线性级 O（n），输入二维数组 a[m][n] 的时间复杂度是 O（m×n），求矩阵 a[m][n]与矩阵 b[n][k]积的时间复杂度是 O（m×n×k）。

如果同一个算法，对于不同的输入数据集对应不同的时间复杂度，在此种情况下常用算法的平均时间复杂度与最坏时间复杂度来度量，除特别声明，本教材的时间复杂度均指最坏时间复杂度。

习题

1．填空题

（1）数据结构所研究的内容包括数据的逻辑结构、数据的存储结构和数据的运算集合。存储结构（又称物理结构）是逻辑结构在_____，它包括_____和_____的表示。施加于_____之上的一组操作构成了数据的运算集合。

（2）数据类型是高级语言中_____的数据结构。

（3）算法的设计要求包括：正确性、可读性、健壮性和_____，可读性的含义是_____，健壮性是指_____。

（4）算法效率的度量应抛开具体机器的_____，对于一个特定算法只考虑算法本身的效率，而算法自身的执行效率是_____函数。

（5）一个算法的时间复杂度随问题的输入数据集的不同而不同，通常讨论_____情况下的时间复杂度。

（6）一个算法的语句频度表达式是 $5n*n*\log_2 n+2n*n*n*\log_2 n+1\ 000*n*n*n*n$，这个算法的时间复杂度是_____，另一个算法的语句频度表达式是 $40*n*n+2\log_2 n+1\ 000$，这个算法的时间复杂度是_____。

（7）算法的时间复杂度与空间复杂度相比，通常以_____作为主要度量指标。

（8）在下面程序段中，s=s+p 语句的频度为_____，p*=j 语句的频度为_____，该程序段的时间复杂度为_____。

```
i=0,
s=0;
while (++i<=n)
{    p=1;
     for(j=1; j<=i; j++)
          p*=j;
     s= s+p;
}
```

2．判断题

（1）数据元素是数据的最小单位。

（2）算法可以用不同的语言描述，如果用类 C 语言或类 Pascal 语言等高级语言来描述，算法实际上就是程序了。

（3）存储结构既要存储数据元素本身，又要表示数据元素之间的逻辑关系。

（4）数据结构是带有结构的数据元素的集合。

（5）数据的逻辑结构是指各数据元素之间的逻辑关系，是用户根据需要建立的。

（6）数据结构在计算机中的映像（或表示）称为存储结构。

（7）算法的可读性的含义是：对于非法的输入数据，算法能给出相应的响应，而不是产生不可预料的后果。

（8）数据的物理结构是指数据在计算机内实际的存储形式。

（9）算法的时间复杂度是算法执行时间的绝对度量。

（10）算法的正确性是指算法不存在错误。

3．简答题

（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、存储结构。

（2）设 n 为正整数，给出下列算法中原操作语句的语句频度及程序段的时间复杂度。

（a）
```
i=1;
k=0;
while (i<=n−1)
{
    k=k+2*i;
    i++;
}
```

（b）
```
i=1; k=0;
do
{
    k=k+2*i;
    i++;
}
while (i!=n)
```

（c）
```
x=91;
n=100;
while (n>0)
    if   (x>100 )
    {
        x=x−10;
        n=n−1;
    }
    else
        x++;
```

（d）
```
x=n;   /* n>1 */
y=0;
while (x>=(y+1)*( y+1))
      y++;
```
（e）
```
for(i=1;i<=n;i++)
      for(j=i;j<=i;j++)
            for(k=j;k<=j;k++)
                  x++;
```