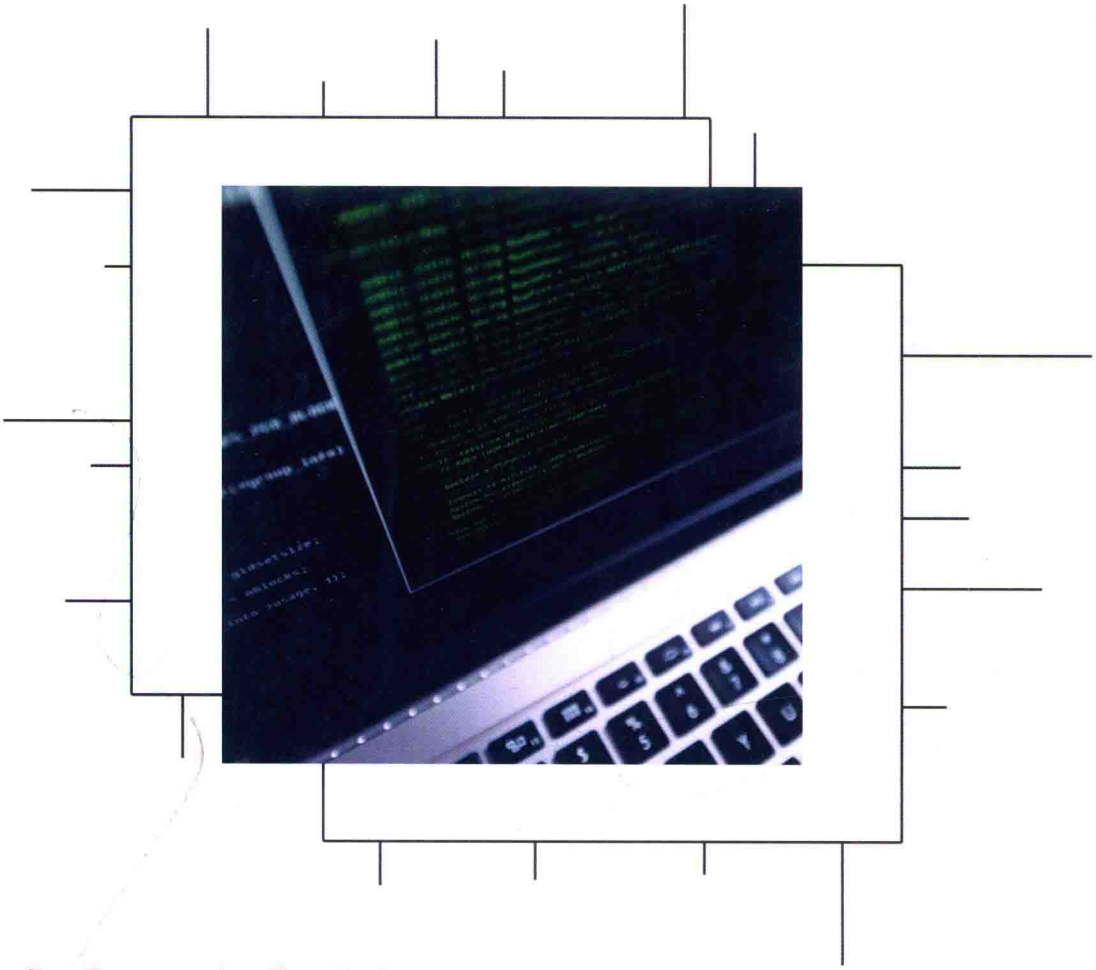
国家新闻出版改革发展项目库入库项目
普通高等教育“十三五”规划教材
全国高等院校计算机基础教育研究会重点立项项目



数据结构（C语言版）

袁和金　刘　军　牛为华
王　妤　王翠茹　编著

“互联网+”创新型教材
北邮智信

北京邮电大学出版社
www.buptpress.com

国家新闻出版改革发展项目库入库项目
普通高等教育“十三五”规划教材
全国高等院校计算机基础教育研究会重点立项项目

数据结构（C语言版）

袁和金　刘　军　牛为华
王　妤　王翠茹　编著

“互联网+”创新型教材
北邮智信

北京邮电大学出版社
www.buptpress.com

内 容 简 介

本书从抽象数据类型的观点出发，系统全面地介绍了“数据结构”课程中的基本理论、方法及技巧。本书共9章，包括绪论、顺序表、链表、数组和广义表、字符串、树、图、查找表、内排序，介绍了各种数据结构的定义和性质，详细分析和讨论了这些结构的逻辑特点、存储表示以及在这些结构上定义的一些运算的实现方法及其复杂性。在每章的末尾配备了足够的习题，附录对实验步骤和内容作了较详细的介绍。

本书适合作为计算机科学与技术、软件工程、网络工程、信息安全以及电子、信息相关专业的教材，也可供从事相关工作的科技与工程人员参考。

图书在版编目（CIP）数据

数据结构：C语言版 / 袁和金等编著. -- 北京：北京邮电大学出版社，2019.8
ISBN 978-7-5635-5797-4

Ⅰ. ①数… Ⅱ. ①袁… Ⅲ. ①数据结构②C语言－程序设计 Ⅳ. ①TP311.12②TP312.8

中国版本图书馆CIP数据核字（2019）第161441号

书　　名：数据结构(C语言版)
作　　者：袁和金　刘　军　牛为华　王　妤　王翠茹
责任编辑：刘春棠
出版发行：北京邮电大学出版社
社　　址：北京市海淀区西土城路10号(邮编:100876)
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：保定市中画美凯印刷有限公司
开　　本：787 mm×1 092 mm　1/16
印　　张：17
字　　数：442千字
版　　次：2019年8月第1版　2019年8月第1次印刷

ISBN 978-7-5635-5797-4　　定　价：39.00元

前　言

“数据结构”是计算机类专业的重要专业基础课，是算法设计与分析、操作系统、软件工程、数据库原理、编译技术、计算机图形学、人工智能等专业基础课和专业课的先修课。它所讨论的知识内容和技术方法，对进一步学习计算机科学领域的相关知识和大型信息系统的开发，都有着重要作用。

本书从抽象数据类型的观点出发，系统全面地介绍了“数据结构”课程中的基本理论、方法及技巧。本书共9章，包括绪论、顺序表、链表、数组和广义表、字符串、树、图、查找表、内排序，介绍了各种数据结构的定义和性质，详细分析和讨论了这些结构的逻辑特点、存储表示以及在这些结构上定义的一些运算的实现方法及其复杂性。附录对实验步骤和内容作了较详细的介绍。

本书是作者在多年来讲授“数据结构”“算法设计与分析”等课程的基础上，结合自身的教学体会和学生的建议编著而成的，适合作为计算机科学与技术、软件工程、网络工程、信息安全以及电子、信息相关专业的教材，也可供从事相关工作的科技与工程人员参考。

本书由袁和金副教授组织并统稿。其中，第1～3章由牛为华编写，第4、5章由王妤编写，第6～8章由袁和金编写，第9章及附录部分由刘军编写，王翠茹教授认真审阅了全部书稿并提出了许多宝贵的修改意见。

由于作者水平和编写时间有限，书中必定存在错误和缺点，恳请读者批评指正。

读者可以扫描封底二维码下载安装“北邮智信”App，加载配套资源。

作　者

“北邮智信”App使用说明

目　录

1.2 基本概念

本节介绍本书中常用的名词和术语的含义。

1. 数据

随着计算机科学的发展和计算机应用的普及,计算机加工处理的对象已从早期的数值、布尔值等扩展到字符串、表格、图像、声音等。因此,凡是能被计算机存储、加工的对象通称为数据,它是计算机程序加工的“原料”。

2. 数据元素和数据项

数据元素是数据的基本单位,在程序中作为一个整体加以考虑和处理。有时,一个数据元素可由若干个数据项组成,数据项是数据不可分割的最小单位。

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 $D=\{0, \pm1, \pm2, \cdots\}$,字母字符数据对象是集合 $C=\{\text{'A'}, \text{'B'}, \cdots, \text{'Z'}\}$。

4. 数据结构

数据结构不同于数据类型,也不同于数据对象,它不仅涉及数据类型和数据对象,而且要描述数据对象各元素之间的相互关系。这种数据元素之间的相互关系就称为结构。根据数据元素之间关系的不同,通常有如图 1.1 所示的四类基本的结构形式。

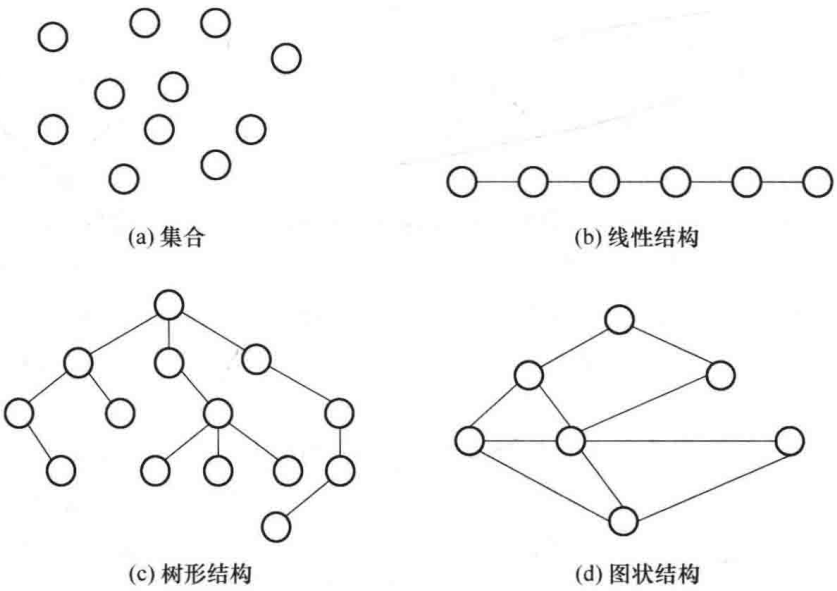


图 1.1 四种基本结构关系图

① 集合:集合中的任何两个元素之间都没有逻辑关系,组织形式松散。

② 线性结构:结构中元素之间存在一对一的关系,依次排列形成一条“锁链”。

③ 树形结构:结构中元素之间存在一对多的关系,具有分支、层次特性。

④ 图状结构:结构中元素间存在多对多的关系,元素间互相缠绕,任何两个元素都可以邻接。

数据结构可以用二元组的形式来进行数学意义上的形式定义:

Data_structure=(D,S)

式中，D 是数据元素的有限集，S 是 D 上关系的有限集。它是从操作对象中抽象出来的数学模型，仅仅描述了数据元素之间的某种逻辑关系。为了在计算机中实现对数据元素的操作，还需要考虑数据关系在计算机中的存储。因此数据结构应包括三方面的内容。

① 数据的逻辑结构：数据元素之间的逻辑关系。

② 数据的存储结构：数据元素及其关系在计算机存储器中的表示。

③ 数据的运算：对数据对象施加的操作。

(1) 数据的逻辑结构

从逻辑关系上描述数据，它与数据的存储无关，是独立于计算机的。因此，数据的逻辑结构可以看作是从具体问题上抽象出来的数学模型，通常把数据元素之间的关联方式(邻接关系)称为数据元素间的逻辑关系。数据元素之间逻辑关系的整体称为逻辑结构。

(2) 数据的存储结构

数据结构在计算机中的表示称为数据的存储结构。它包括数据元素的表示和关系的表示。在计算机中表示信息的最小单位是二进制数的一位，称作位(bit)。在计算机中，我们可以用一个若干位组合起来形成的一个位串表示一个数据元素，通常称这个位串为元素或结点(node)。当数据元素由若干数据项组成时，位串中对应于各个数据项的子位串称为数据域(data field)。元素间的关系在计算机内的表示方法通常有四种。

① 顺序存储方法：该方法是指每个存储结点只含有一个数据元素，所有存储结点相继存放在一个连续的存储区里。用存储结点间的位置关系表示数据元素之间的逻辑关系。

② 链式存储方法：该方法是指每个存储结点不仅含有一个数据元素，还包含一组指针。每个指针指向一个与本结点有逻辑关系的结点。

③ 索引存储方法：该方法是指在存储结点信息的同时，还建立索引表，索引表的每一项称为索引项，索引项的一般形式是：(关键字，地址)。关键字是能唯一标识一个结点的那些数据项，地址表示该关键字所在结点的起始存储位置。

④ 散列存储方法：该方法是指根据结点的关键字值，采用某种方法，直接计算出该结点的存储地址。

5. 数据类型

数据类型是一个值的集合和定义在这个值集上的一组操作的总称。例如 C 语言中的整数类型，其值集为[－Maxint，Maxint]上的整数，定义在其上的一组操作为加、减、乘、整除和取模等。按“值”的不同特性，数据类型可分为两类：非结构的原子类型和结构类型。原子类型的值是不可分解的，如 C 语言中的标准类型(整型、实型、字符型、布尔型)、枚举类型、子界类型和指针类型。结构类型的值是由若干成分按某种结构组成的，因此其值是可分解的，并且它的成分可以是非结构的，也可以是结构的。例如，一个记录类型的元素的值由若干个分量组成，每个分量可以是整数或数组，也可以是指针类型等。引入“数据类型”的目的，仅从使用数据类型的角度来说，实现了信息的隐蔽，将一切不必了解的细节都封装在类型中。例如，用户在使用“整数”类型时，既不需要了解“整数”在机内的表示，也不需要知道其操作是如何实现的。如“两整数求和”程序设计者注重的仅仅是其“数学上求和”的抽象特性，而不是其硬件的“位”操作如何进行。

6. 抽象数据类型

抽象数据类型和数据类型实质上是一个概念，是指一个数学模型及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在机内的表示和实现无关，

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。抽象数据类型与数据类型相比范畴更广些,它不再局限于前述程序语言中已定义并实现的数据类型(固有的数据类型),还包括用户在设计软件系统时自己定义的数据类型。

假设要设计一个圆(circle)为抽象数据类型,它提供的操作除了按给定半径构造一个圆外,还要包括计算面积(area)和周长(circumference)的操作。下面是 circle 的一个抽象数据类型定义,关于这个类型的具体实现(包括数据结构的表示和完成操作的代码)请读者自己完成。

```
ADT circle is
    data
        非负实数表示圆的半径
    operations
        constructor
            输入的初值:非负实数
            处理:构造一个圆
        area
            输入:无
            输出:圆的面积
            处理:计算圆的面积
        circumference
            输入:无
            输出:圆的周长
            处理:计算圆的周长
end ADT circle
```

和数据结构的形式定义相对应,抽象数据类型可以用以下三元组表示:

$$\text{ADT}=(D,\ S,\ P)$$

式中,D 是数据对象,用结点的有限集合表示;S 是 D 上的关系集,用结点间序偶的集合表示;P 是对 D 的基本操作集。其操作的定义格式为:

```
基本操作名(参数表)
```

初始条件:

```
<初始条件描述>
```

操作结果:

```
<操作结果描述>
```

【例 1.1】 构造复数。

```
ADT Complex
{
    数据对象:D = {e1, e2|e1, e2 ∈Realset}
    数据关系:R1 = {< e1, e2 >|e1 是复数的实数部分,e2 是复数的虚数部分}
    基本操作:
    InitComplex(e1, e2);                    //初始化:以 e1 和 e2 为实部和虚部构造一个复数
    DestroyComplex(Z);                      //销毁复数 Z
```

```
    GetReal(Z, realPart);                    //得到复数 Z 的实部
    GetImag(Z, ImagPart);                    //得到复数 Z 的虚部
    Add(Z1, Z2, Sum);                        //复数 Z1 和 Z2 相加,结果保存到 Sum 中
    Subtract(Z1, Z2, Difference);            //复数 Z1 和 Z2 相减,结果保存到 Difference 中
    Multiply(Z1, Z2, Product);               //复数 Z1 和 Z2 相乘,结果保存到 Product 中
}ADT Complex
```

至于上述抽象类型具体如何实现,读者可以根据所学过的知识来思考一下,此处不再详细介绍。

7. 基本运算

基本运算只描述处理功能,不包括处理步骤和方法,是在逻辑结构上的操作。

8. 运算实现

运算实现的核心是处理步骤的规定,即算法设计。对一个复杂的运算仍需按照逐步求精的方法构造它的实现。

9. 算法

算法是指解决某一特定类型问题的有限运算序列。一个算法应该具有下列特性。

① 有穷性:一个算法必须总是在执行有穷步之后结束。

② 确定性:算法的每一步必须有确切的定义,读者理解时不会产生二义性。

③ 输入:一个算法有零个或多个输入,这些输入取自于特定的对象集合。

④ 输出:一个算法有一个或多个输出,它们是同输入有某种特定关系的量。

⑤ 可行性:算法应该是可行的,即算法中所有待实现的运算都是能够正确地执行的。

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着十分密切的关系。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,数据结构的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,同时涉及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并实现,还需考虑数据结构及其实现的评价与选择。因此,数据结构课程的内容包括三个层次的五个“要素”,其内容体系如表 1.1 所示。

表 1.1 “数据结构”课程的内容体系

方面 / 层次	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	不同结构的比较及算法分析	

1.4 类C语言和算法评价

1.4.1 类C语言

本书算法的描述采用类C语言,它精选了C语言的一个核心子集,同时做了若干扩充和修改,增强了语言的描述功能。以下对其进行简要说明。

(1) 数据结构的表示(存储类型)用类型定义(typedef)描述。数据元素类型约定为ElemType,由用户在使用该数据类型时自行定义。

(2) 基本操作的算法都用以下形式的函数描述:

```
函数类型  函数名(函数参数表)
{  //算法说明
    语句序列;
}  //函数名
```

除了函数的参数需要说明类型外,算法中使用的辅助变量可以不做变量说明,必要时对其作用给出注释。一般而言,a、b、c、d、e 等用作数据元素名,i、j、k、l、m、n 等用作整型变量名,p、q、r 等用作指针变量名。

为了便于描述,在函数表中除了值调用方式外,增添了C++语言的引用调用的参数传递方式。在形参表中,以 & 打头的参数即为引用参数。引用参数能被函数本身更新参数值,可以此作为输出数据的管道。参数表中的某个参数允许预先用表达式的形式赋值,作为默认值使用,以简化参数表。

(3) 内存的动态分配与释放过程为:使用 new 和 delete 动态分配和释放内存空间。

```
分配空间      指针变量 = new 数据类型;
释放空间      delete 指针变量;
```

(4) 赋值语句有:

```
简单赋值      变量名 = 表达式;
串联赋值      变量名 1 = 变量名 2 = … = 变量名 k = 表达式;
成组赋值      (变量名 1, …, 变量名 k) = (表达式 1, …, 表达式 k);
              结构名 = 结构名;
              结构名 = (值 1, 值 2, …, 值 k);
              变量名[ ] = 表达式;
              变量名[起始下标..终止下标] = 变量名[起始下标..终止下标];
交换赋值      变量名←→变量名;
条件赋值      变量名 = 条件表达式? 表达式 T: 表达式 F;
```

(5) 选择语句有:

```
条件语句 1   if (表达式)
             {
                 语句序列;
             }
条件语句 2   if (表达式)
```

```
{
    语句序列;
}
else
{
    语句序列;
}
```

开关语句 1

```
switch(表达式)
{
    case  值 1:{语句序列 1; break;}
    case  值 2:{语句序列 2; break;}
    …
    case  值 n:{语句序列 n; break;}
    default:    {语句序列 n+1;}
}
```

开关语句 2

```
switch
{
    case  条件 1:{语句序列 1; break;}
    case  条件 2:{语句序列 2; break;}
    …
    case  条件 n:{语句序列 n; break;}
    default:      {语句序列 n+1;}
}
```

(6) 循环语句有：

for 语句

```
for(赋值表达式序列; 条件; 修改表达式序列)
{
    语句序列;
}
```

while 语句

```
while(条件)
{
    语句序列;
}
```

do-while 语句

```
do
{
    语句序列;
} while(条件)
```

(7) 结束语句有：

函数结束语句

```
return 表达式;
return;
```

异常结束语句

```
exit(异常代码);
```

(8) 输入和输出语句有：

输入语句

```
scanf([格式串], 变量 1, …, 变量 n);
```

输出语句

```
printf([格式串], 表达式 1, … , 表达式 n);
```

(9) 注释为:

单行注释　　//文字序列;

(10) 基本函数有:

求最大值　max(表达式1, …, 表达式n);

求最小值　min(表达式1, …, 表达式n);

求绝对值　abs(表达式);

求进位整数值　ceil(表达式);

求不足整数值　floor(表达式);

判定文件结束　eof(文件变量) 或 eof;

判定行结束　　eoln(文件变量) 或 eoln;

(11) 逻辑运算约定有:

与运算 &&　　对于 $A\&\&B$,当 A 的值为 0 时,不再对 B 求值;

或运算||　　对于 $A||B$,当 A 的值为非 0 时,不再对 B 求值。

1.4.2 算法评价

解决同一问题,常常可以设计出不同的算法,如何比较这些算法的优劣是一个有意义的问题。本书第 9 章对排序问题就介绍了多种不同的方法,这些方法的排序速度和占用内存的多少存在着很大的差异。评价算法既是为了从解决同一问题的不同算法中选出较为适用的一种,同时也有助于考虑对现有的算法如何进行改进或设计出新的算法。

1. 评价算法的一般原则

一个好的算法通常应具有如下特点。

(1) 正确性:指算法在允许的输入数据范围内,能在有穷时间内得出正确的结果。关于算法正确性的证明需要使用有关的数学理论(如集合论、代数学的定理及离散数学等知识),严格地说它属于程序设计方法学课程所研究的内容。通常,对于较复杂的问题,可以将算法分解成一些局部的模块来分析,只有每个模块都是正确的,才能保证整个算法的正确性。对于本书中所讨论的算法,有些正确性是显而易见的,有些则对其正确性做了简单证明,还有些则将正确性证明过程省略了,因为这部分内容不属于本书所要讨论的范围。

在实践中,对于算法的正确性,一般采用测试的方法来验证。算法的正确性可以分为以下四个层次:①程序不含语法错误;②程序对于几组输入数据能够得出满足规格说明要求的结果;③程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得出满足规格说明要求的结果;④程序对于一切合法的输入数据都能产生满足规格说明要求的结果。显然,达到第④层意义下的正确是极为困难的,所有不同输入数据的数量相当大,逐一验证的方法是不现实的。对于大型软件需要进行专业测试,而一般情况下,通常以第③层意义的正确性作为衡量一个程序是否合格的标准。

(2) 算法执行速度快:依据算法编程后在计算机上运行时所消耗的时间越少越好。

(3) 算法所占用的空间省:依据算法编程后在计算机上运行时所占的存储量较少。其中主要考虑程序运行时所需辅助存储量的多少。

(4) 算法结构简单,易写,易读,易于转换成可运行的程序且易于调试和修改。这里应当指出,结构简单、易写、易读的算法常常不一定是运行效率最高的方法,其运算工作量可能较大(如递归算法)。

2. 算法的复杂性

复杂性是指实现或运行某一算法所需资源的多少，包括时间复杂度(time complexity)、空间复杂度(space complexity)和人工复杂度(manual complexity，指编程及改错等所需人工)。从主观上讲，人们希望选用一个不占用很多存储空间、运行时间短，其他性能指标好的算法。然而实际上往往不可能做到十全十美，一个看上去很简单的程序，其运行时间可能要比一个形式上复杂的程序慢得多，而一个运行时间较短的程序常常要占用较多的附加存储空间。因此，应针对不同情况选用不同的算法。当前由于计算机硬件的迅猛发展，扩展内存或外存的容量已经没有什么困难了，而高级语言的出现使编程、改错等也较以前容易多了，所以本书中研究算法的复杂性，一般主要是指时间复杂度，偶尔也讨论某些算法的空间复杂度。

显然，同一个算法用不同的语言实现，或者用不同的编译程序进行编译，或者在不同的计算机上运行时，效率均不相同。这表明使用绝对的时间单位衡量算法的效率是不合适的。撇开这些与计算机硬件、软件有关的因素，可以认为一个特定算法“运行工作量”的大小，只依赖于问题的规模(通常用整数量 n 表示)，或者说，它是问题规模的函数。

例如，在如下所示的两个 n 阶矩阵相乘的算法中，“乘法”运算是“矩阵相乘问题”的基本操作，整个算法的执行时间与该基本操作(乘法)重复执行的次数(n^3)成正比。

```
for (i = 1; i <= n; i++)
    for(j = 1; j <= n; j++)
    {
        c[i][j] = 0;
        for(k = 1; k <= n; k++)
            c[i][j] + = a[i][k] * b[k][j];
    }
```

一般情况下，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 $f(n)$，算法的时间量度记作 $T(n)=O(f(n))$，它表示随问题规模 n 的增大，算法执行时间的增长率和 $f(n)$ 的增长率相同，称为算法的渐近时间复杂度(asymptotic time complexity)，简称时间复杂度。

显然，某一算法运行所需时间(不考虑它所处的软、硬件环境)主要与算法中各语句重复执行的次数有关，同时与所解决问题的规模大小也有关，当解决的问题规模固定时，常常把语句重复执行的次数作为算法的时间耗用量度。为此，需引入一个语句频度的概念，所谓语句的频度是指语句重复执行的次数。

由于语句执行频度与算法要解决问题的规模有密切关系(如上面例子中矩阵的阶 n 决定了语句执行频度)，因此，一般情况下，若用 n 表示问题规模的量(例如，排序问题中 n 为需排序的元素个数；图的问题中 n 是图中的顶点数或边数；矩阵运算中 n 为矩阵的阶数等)，则一个算法的时间复杂度为 n 的函数，记为 $T(n)$，且 $T(n)=O(f(n))$。它表示了时间复杂度的一个数量级的概念，即如果 $f(n)$ 是正整数 n 的一个函数，则上式表示存在一个正的常数 M，使得当 $n\geqslant n_0$ 时，都满足 $|T(n)|\leqslant M|f(n)|$。对于这里的 M 和 n_0 不必也无法说出它的确切数量。

对解决同一问题的不同算法，若对应的 $f(n)$ 不同，其算法的时间复杂度当然也不同。

例如，对下列三个简单的程序段：

```
(1) k=k+1;
(2) for(i = 1; i <= n; i++)
        k = k+1;
```