JISUANJI SUANFA SHEJI
YU FENXI

计算机算法设计与分析

寇 伟 申国霞 王文霞 编著

中国水利水电出版社
www.waterpub.com.cn

JISUANJI SUANFA SHEJI YU FENXI

计算机算法设计与分析

寇 伟 申国霞 王文霞 编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书深入浅出地介绍了计算机算法的基本理论和方法，主要内容包括算法导引、图的周游与最小支撑树算法分析、递归与分治策略分析、动态规划法的设计与分析、贪心算法的分析与优化、回溯法问题分析、分支限界法问题分析、NP 完全性分析、随机算法分析、近似算法的设计与分析、智能优化算法研究等。

图书在版编目（CIP）数据

计算机算法设计与分析 / 寇伟，申国霞，王文霞编著. -- 北京 : 中国水利水电出版社，2014.9
ISBN 978-7-5170-2461-3

Ⅰ. ①计… Ⅱ. ①寇… ②申… ③王… Ⅲ. ①电子计算机－算法设计②电子计算机－算法分析 Ⅳ. ①TP301.6

中国版本图书馆CIP数据核字(2014)第207482号

策划编辑：杨庆川　责任编辑：杨元泓　封面设计：崔　蕾

书　名	计算机算法设计与分析
作　者	寇　伟　申国霞　王文霞　编　著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座 100038） 网址：www. waterpub. com. cn E-mail：mchannel@ 263. net（万水） sales@ waterpub. com. cn 电话：（010）68367658（发行部）、82562819（万水）
经　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　版	北京鑫海胜蓝数码科技有限公司
印　刷	三河市天润建兴印务有限公司
规　格	184mm × 260mm　16 开本　19.75 印张　480 千字
版　次	2015 年 4 月第 1 版　2015 年 4 月第 1 次印刷
印　数	0001—3000 册
定　价	69.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

计算机的普及极大地改变了人们的生活。目前，各行业、各领域都广泛采用了计算机信息技术，并由此产生出开发各种应用软件的需求。为了以最小的成本、最快的速度、最好的质量开发出适合各种应用需求的软件，必须遵循软件工程的原则。设计一个高效的程序不仅需要编程小技巧，更需要合理的数据组织和清晰高效的算法，这正是计算机科学领域数据结构与算法设计所研究的主要内容。

算法设计与分析是一门理论性与实践性相结合的学科，是计算机科学与计算机应用专业的核心。学习算法设计可以在分析解决问题的过程中，培养学习者抽象思维和缜密概括的能力，提高学习者的软件开发设计能力。

本书在编撰方面力求突出以下特点：

1. 力求做到难易适当、深入浅出，融会贯通；
2. 讲解理论重点、层次分明、通俗易懂；
3. 案例丰富，有利于读者掌握所学理论知识；
4. 体系完整、内容先进；
5. 取舍合理。

在学习算法设计的过程中，由于现实中的实际问题往往较复杂，需要具备丰富的领域知识、算法设计方法和技巧规范及软件工程的开发规范等综合技能。因此，需要通过一些简单、抽象的例子，对基础的算法策略进行讲解。

随着信息化时代的到来，计算机开发平台日新月异，软件应用拓展到了各个领域，各类算法和技巧层出不穷，本书只能是管中窥豹。若能达到本书的初衷——使读者能掌握到算法设计的基本方法和技巧，打好软件开发的基础，就深感满意了。

本书由寇伟、申国霞、王文霞撰写，具体分工如下：

第 5 章 ~ 第 7 章、第 10 章：寇伟（兰州职业技术学院）；

第 1 章 ~ 第 4 章：申国霞（兰州职业技术学院）；

第 8 章、第 9 章、第 11 章：王文霞（运城学院）。

本书在编撰过程中得到了许多同行专家的支持和帮助，在此表示衷心的感谢；编撰时参考了大量的相关著作和文献资料，选用了其中的部分内容，在此向有关作者表示由衷的感谢。

由于作者水平有限，书中错误或不当之处在所难免，热忱欢迎同行和广大读者朋友批评指正。

作　者

2014 年 6 月

目　录

第1章　算法导引

1.1　算法研究的初衷

算法是计算机科学的重要基础，同时也是计算机科学研究中一项永恒主题。

1.1.1　算法在问题求解中的地位

Donald Knuth 曾经说过：程序就是蓝色的诗。若按照这个说法的话，那么，算法就是这首诗的灵魂。在各种计算机软件系统的实现中，算法设计往往处于核心地位。因为计算机不能分析问题并产生问题的解决方案，必须由人来分析问题，确定问题的解决方案，采用计算机能够理解的指令描述问题的求解步骤，然后让计算机执行程序最终获得问题的解。用计算机求解问题的一般过程如图 1-1 所示。

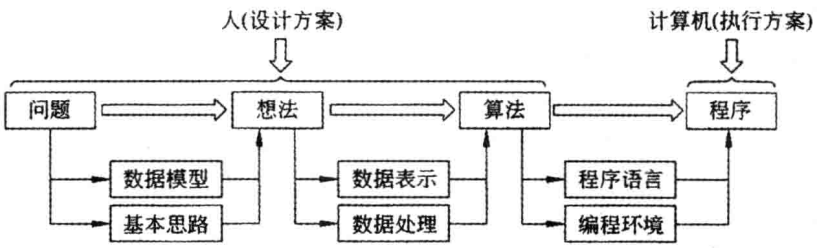

图 1-1　用计算机求解问题的一般过程

由问题到想法需要分析问题，将具体的数据模型抽象出来，形成问题求解的基本思路；由想法到算法需要完成数据表示（将数据模型存储到计算机的内存中）和数据处理（将问题求解的基本思路形成算法）；由算法到程序需要将算法的操作步骤转换为某种程序设计语言对应的语句。

算法用来描述问题的解决方案，是形式化的、机械化的操作步骤。将人的想法描述成算法可以说是利用计算机解决问题的最关键的一步，也就是从计算机的角度设想计算机是如何一步一步完成这个任务的，告诉计算机需要做哪些事，按什么步骤去做。一般来说，对不同解决方案的抽象描述产生了相应的算法，而不同的算法将设计出相应的程序，这些程序的解题思路不同，复杂程度不同，解题效率也不相同。下面请看一个例子。

【例 1-1】　求两个自然数的最大公约数。

想法一：可以用短除法找出这两个自然数的所有公因子，将这些公因子相乘，结果就是这两个数的最大公约数。例如，48 和 36 的公因子有 2、2 和 3，则 48 和 36 的最大公约数是

$2 \times 2 \times 3 = 12$。

算法一：目前，只能用蛮力法逐个尝试找出两个数的公因子，可以用 $2 \sim \min\{m,n\}$ 进行枚举尝试。短除法求最大公约数的算法用伪代码描述如下。

输入：两个自然数 m 和 n

输出：m 和 n 的最大公约数

factor = 1；

循环变量 i 从 2 ~ rain{m,n}，执行下述操作；

如果 i 是 m 和 n 的公因子，则执行下述操作；

    factor = factor * i；

    m = m/i；n = n/i；

如果 i 不是 m 和 n 的公因子，则 i = i + 1；

输出 factor

算法实现一：程序由两层嵌套的循环组成，外层循环枚举所有可能的公因子，内层循环尝试 i 是否为 m 和 n 的公因子，重复的公因子可能是无法避免的，因此，内层循环不能用 if 语句。算法用 C ++ 语言描述如下：

```
int CommFactor1(int m,int n)
{
  int i,factor = 1;
  for(i = 2;i <= m && i <= n;i ++)
  {
    while(m% i ==0 && n% i ==0)                    //此处不能用 if 语句
    {
      factor = factor * i;
      m = m/1;n = n/1;
    }
  }
return factor;
}
```

想法二：一种效率更高的方法是欧几里得算法，其基本思想是将两个数辗转相除直到余数为 0。

算法二实现一：欧几里得算法用伪代码描述如下。

输入：两个自然数 m 和 n

输出：m 和 n 的最大公约数

m% n；

循环直到 r 等于 0

  m = n；

  n = r；

  r = m% n；

输出 n;

算法二实现二:欧几里得算法用 C ++ 语言描述如下:

```
int CommFactor2(int m,int n)
{
  int r = m% n;
    while(r! =0)
    {
      m = n;
      n = r;
      r = m% n;
    }
    return n;
}
```

两种算法的比较:算法一的时间主要耗费在求余操作(m% i ==0 &&n% i ==0)上,算法二的时间主要耗费在求余操作(r = m% n)上,以 48 和 36 为例,算法一要进行 10 次求余操作,而算法二只需进行两次求余操作,也就是说,算法二比算法一的时间效率高。

需要强调的是,有些问题比较简单,其问题的解决方案非常容易得到,如果问题比较复杂,就需要更多的思考才能得到问题的解决方案。对于许多实际问题,写出一个可以正确运行的算法还不够,如果这个算法在规模较大的数据集上运行,那么运行效率就成为一个首要的问题。

1.1.2 算法训练能够提高计算思维能力

冯·诺依曼计算机是按存储程序方式进行工作的,计算机的工作过程可以看成是运行程序的过程。但是计算机不能分析问题并产生问题的解决方案,必须由人来分析问题,确定并描述问题的解决方案,因此,用计算机求解问题是建立在高度抽象的级别上,表现为采用形式化的方式描述问题,问题的求解过程是建立符号系统并对其实施变换的过程,并且变换过程是一个机械化、自动化的过程。在描述问题和求解问题的过程中,抽象思维和逻辑思维是主要的方式手段,如图 1-2 所示。

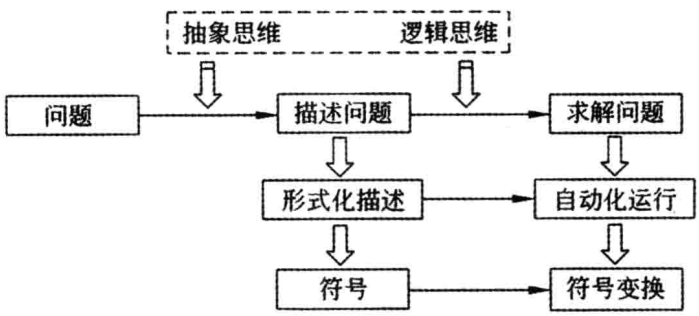

图 1-2 计算机学科的符号化特征

在 ACM/IEEE-CS 提交的 CC2005 中，将计算机专业的基本学科能力归纳为计算思维能力、算法设计与分析能力、程序设计与实现能力和系统能力，其中形式化、模型化、抽象思维与逻辑思维共同组成了计算思维。如前所述，在用计算机求解问题的过程中，最重要的环节就是将人的想法抽象为算法。算法不是问题的答案，而是经过精确定义的、用来获得答案的求解过程。算法设计过程是计算思维的具体运用，因此，算法训练就像一种思维体操，能够锻炼我们的思维，使思维变得更清晰、更有逻辑。

1.1.3 算法研究是推动计算机技术发展的关键

时间（速度）问题就是算法研究的核心。人们可能有这样的疑问：既然计算机硬件技术的发展使得计算机的性能不断提高，算法的研究还有必要吗？

计算机的功能越强大，人们就越想去尝试更复杂的问题，与此相应的，计算量也会相应地扩大。现代计算技术在计算能力和存储容量上的革命仅仅提供了计算更复杂问题的有效工具，无论硬件性能如何提高，算法研究始终是推动计算机技术发展的关键。实际上，我们不仅需要算法，而且需要“好”算法。可以肯定的是，发明（或发现）算法是一个非常有创造性和值得付出的过程。下面看几个例子。

1. 检索技术

20 世纪 50—60 年代，检索仅仅是在规模比较小的数据集合中发生。例如，编译系统中的标识符表，表中的记录个数一般在几十至数百这样的数量级。

20 世纪 70—80 年代，数据管理采用数据库技术，数据库的规模在 K 级或 M 级，检索算法的研究在这个时期取得了巨大的进展。

20 世纪 90 年代以来，Internet 引起计算机应用的急速发展，研究的热点也转向了海量数据的处理技术，而且数据驻留的存储介质、数据的存储方法以及数据的传输技术也发生了许多变化，这些变化使得检索算法的研究更为复杂也更为重要了。

近年来，智能检索技术成为基于 Web 信息检索的研究热点。使用搜索引擎进行 Web 信息检索时，一些搜索引擎前 50 个搜索结果中几乎有一半来自同一个站点的不同页面，这是检索系统缺乏智能化的一种表现。另外，在传统的 Web 信息检索服务中，信息的传输是按 Pull 模式进行的，即用户找信息。而采用 Push 方式，是信息找用户，用户想要获得自己感兴趣的信息无需进行任何信息检索，这就是智能信息推送技术。这些新技术的每一项重要进步都与算法研究的突破有关。

2. 压缩与解压缩

随着多媒体技术的发展，计算机的处理对象由原来的字符发展到图像、图形、音频、视频等多媒体数字化信息，这些信息数字化后，其特点就是数据量非常庞大。例如，音乐 CD 的采样频率是 44kHz，假定它是双声道，每声道占用 2 字节存储采样值，则 1 秒钟的音乐就需要 $44000 \times 2 \times 2 \approx 160\text{KB}$，存储一首 4 分钟长的歌曲，总计需要 $4 \times 60 \times 160 \approx 36\text{MB}$。而且，计算机总线无法承受处理多媒体数据所需的高速传输速度。因此，对多媒体数据的存储和传输都要求对数据进行压缩。MP3 压缩技术就是一个成功的压缩/解压缩算法，一个播放 3～4 分钟歌曲的 MP3 文件通常只需 3 MB 左右的磁盘空间。

3. 信息安全与数据加密

在计算机应用迅猛发展的同时,也面临着各种各样的威胁。一位酒店经理曾经描述了这样一种可能性:"如果我能破坏网络的安全性,想想你在网络上预订酒店房间所提供的信息吧! 我可以得到你的名字、地址、电话号码和信用卡号码,我知道你现在的位置,将要去哪儿,何时去,我也知道你支付了多少钱,我已经得到足够的信息来盗用你的信用卡!"这种情景的确非常可怕。所以,在电子商务中,信息安全是最关键的问题,保证信息安全的一个方法就是对需要保密的数据进行加密。在这个领域,数据加密算法的研究是绝对必需的,其必要性与计算机性能的提高无关。

1.2 算法与程序

对于计算机科学来说,算法(algorithm)的概念至关重要。通俗地讲,算法是指解决问题的方法或过程。严格地讲,下述性质的指令序列是算法需要满足的。

①输入:有零个或多个外部量作为算法的输入。

②输出:算法产生至少一个量作为输出。

③确定性:组成算法的每条指令是清晰的、无歧义的。

④有限性:算法中每条指令的执行次数有限,执行每条指令的时间也有限。

程序(program)与算法有一定的差别。程序是算法用某种程序设计语言的具体实现。对于算法的性质,即有限性程序是无法满足的。例如操作系统,它是在无限循环中执行的程序,因而不是算法。然而可把操作系统的各种任务看成一些单独的问题,每一个问题由操作系统中的一个子程序通过特定的算法实现。该子程序得到输出结果后便终止。

1.3 算法的描述

要使计算机能够顺利完成人们预定的工作,设计一个算法是工作的第一步,然后再根据算法编写程序。

可以设计不同的算法来求解一个问题,同一个算法可以采用不同的形式来表述。

算法是问题的程序化解决方案。描述算法可以有多种方式,如自然语言方式、流程图方式、伪代码方式、计算机语言表示方式与表格方式等。

当一个算法使用计算机程序设计语言描述时,就是程序。本书采用C语言与自然语言相结合来描述算法。之所以采用C语言来描述算法,因为C/C++语言功能丰富、表达能力强、使用灵活方便、应用面广,对于算法所处理的数据结构、计算过程都能够进行有效的描述,是目前计算机程序设计的首选语言。

为方便算法描述与程序设计,下面把C语言的基本要点作简要概括。

1. 标识符

可由字母、数字和下划线组成,标识符必须以字母或下划线开头,大小写的字母分别认为是两个不同的字符。

2. 常量

整型常量:十进制常数、八进制常数(以 o 开头的数字序列)、十六进制常数(以 ox 开头的数字序列)。

长整型常数(在数字后加字符 L)。

实型常量(浮点型常量):小数形式与指数形式。

字符常量:用单引号(撇号)括起来的一个字符,转义字符在此处也可以使用。

字符串常量:用双引号括起来的字符序列。

3. 表达式

(1)算术表达式

整型表达式:参加运算的运算量是整型量,结果也是整型数。

实型表达式:参加运算的运算量是实型量,运算过程中先转换成 double 型,结果为 double 型。

(2)逻辑表达式

用逻辑运算符连接的整型量,结果为一个整数(0 或 1),逻辑表达式可以认为是整型表达式的一种特殊形式。

(3)字位表达式

用位运算符连接的整型量,结果为整数。字位表达式可以看成是整型表达式的一种特殊形式。

(4)强制类型转换表达式

用"(类型)"运算符使表达式的类型进行强制转换,如(float)a。

(5)逗号表达式(顺序表达式)

形式为:表达式 1,表达式 2,…,表达式 n

顺序求出表达式 1,表达式 2,…,表达式 n 的值,结果为表达式 n 的值。

(6)赋值表达式

将赋值号"="右侧表达式的值赋给赋值号左边的变量,赋值表达式的值为执行赋值后被赋值的变量的值。

(7)条件表达式

形式为:逻辑表达式? 表达式 1: 表达式 2

逻辑表达式的值若为非 0(真),则条件表达式的值等于表达式 1 的值;若逻辑表达式的值为 0(假),则条件表达式的值等于表达式 2 的值。

(8)指针表达式

对指针类型的数据进行运算。例如 p-2、p1-p2、&a 等(其中 p、p1、p2 均已定义为指针变量),结果为指针类型。

以上各种表达式可以包含有关的运算符,也可以是不包含任何运算符的初等量。例如,常数是算术表达式的最简单的形式。

表达式后加":",即为表达式语句。

4. 数据定义

对程序中用到的所有变量都需要进行定义,对数据要定义其数据类型,其存储类别也

需要指定。

(1)数据类型标识符

int(整型),short(短整型),long(长整型),unsigned(无符号型),char(字符型),float(单精度实型),double(双精度实型),struct(结构体名),union(共用体名)。

(2)存储类别

auto(自动得),static(静态的),register(寄存器的),extern(外部的)。

变量定义形式:存储类别 数据类型 变量表列

如:static float x,y

5. 函数定义

存储类别 数据类型 <函数名>(形参表列)

{函数体}

6. 分支结构

(1)单分支

if(表达式)<语句1>[else <语句2>]

功能:如果表达式的值为非0(真),则执行语句1;否则(为0,即假),执行语句2。所列语句可以是单个语句,也可以是用{}界定的若干个语句。多分支的实现可通过if嵌套的应用来表现。

(2)多分支

switch(表达式)

{case 常量表达式1:<语句1>

case 常量表达式2:<语句2>

…

case 常量表达式n:<语句n>

default:<语句n+1>

}

功能:取表达式1时,执行语句1;取表达式2时,执行语句2;…,其他所有情形,执行语句n+1。

case 常量表达式的值必须互不相同。

7. 循环结构

(1)while 循环

while(表达式)语句

功能:表达式的值为非0(条件为真),执行指定语句(可以是复合语句)。直到表达式的值为0(假)时,脱离循环。

特点:先判断,后执行。

(2)do-while 循环

do 语句

while(表达式)

功能:执行指定语句,判断表达式的值非0(真),再执行语句;直到表达式的值为0(假)

时,循环才会结束。

特点:先执行,后判断。

(3)for 循环

for(表达式 1;表达式 2;表达式 3)语句

功能:解表达式 1;求表达 2 的值:若非 0(真),则执行语句;求表达式 3;再求表达式 2 的值;……;直至表达式 2 的值为 0(假)时,脱离循环。

以上三种循环,若执行到 break 语句,提前终止循环。若执行到 continue,结束本次循环,跳转下一次循环判定。

需要注意的是,在不致引起误解的前提下,有时对描述的 C 语句进行适当简写或配合汉字标注,用以简化算法框架描述。

例如,从键盘输入整数 n,按 C 语言的键盘输入函数应写为:

```
scanf("%d",&n);
```

可简写为:scanf(n);

或简写为:输入整数 n;

要输出整数量 $a(1),a(2),\cdots,a(n)$,按 C 语言的输出函数应写为:

```
for(k=1;k<=n;k++)
    printf("%d",a[k]);
```

可简写为:printf(a(1)-a(n));

或简写为:输出 a(1-n);

1.4 算法设计的一般过程

算法是问题的解决方案,这个解决方案本身并不是问题的答案,而是能获得答案的指令序列。不言而喻,由于实际问题千奇百怪,问题求解的方法也就各不相同,所以,算法的设计过程是一个充满智慧的灵活过程,它要求设计人员根据实际情况具体问题具体分析。在设计算法时,遵循图 1-3 所示的一般过程可以在一定程度上指导算法的设计。

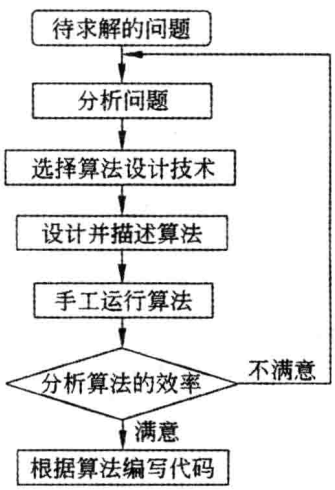

图 1-3 算法设计的一般过程

1. 理解问题

对于待求解的问题,首先搞清楚求解的目标是什么,已经给出了哪些已知信息、显式条件或隐含条件,计算结果的表达应当使用哪种形式的数据来对其进行表达。准确地理解算法的输入是什么,明确要求算法做的是什么,即明确算法的入口和出口,这是设计算法的切入点。如果没有全面、准确和认真地分析问题,结果往往是事倍功半,造成不必要的反复,甚至留下严重隐患。

2. 选择算法设计技术

算法设计技术是本书讨论的主题。算法设计技术(algorithm design technique,也称算法设计策略)是设计算法的一般性方法,

不同计算领域的多种问题都可以用其进行解决。本书讨论的算法设计技术是已经被证明对算法设计非常有用的通用技术,这些算法设计技术构成了一组强有力的工具,在为新问题(即没有令人满意的已知算法可以解决的问题)设计算法时,可以运用这些技术设计出新的算法。

3. 设计并描述算法

在构思和设计了一个算法之后,将所设计的求解步骤记录下来必须进行清晰准确的记录,即描述算法。常用的描述算法的方法有自然语言、流程图、程序设计语言和伪代码等,其中伪代码是比较合适的描述算法的方法。本书采用C++语言和自然语言相结合的伪代码来描述算法,并且几乎所有算法都采用程序设计语言给出了算法实现。

4. 手工运行算法

由计算机无法检测出逻辑错误,因为计算机只会执行程序,而不会理解动机。经验和研究都表明,发现算法(或程序)中的逻辑错误的重要方法就是手工运行算法,即跟踪算法。跟踪者要像计算机一样,用一个具体的输入实例手工执行算法,并且这个输入实例要最大可能地暴露算法中的错误。即使有几十年经验的高级软件工程师,也经常利用此方法查找算法中的逻辑错误。

5. 分析算法的效率

算法效率体现在两个方面:时间效率和空间效率,时间效率显示了算法运行得有多快,空间效率则显示了算法需要多少额外的存储空间,相比而言,算法的时间效率是我们关注的重点。事实上,计算机的所有应用问题,包括计算机自身的发展,都是围绕着"时间——速度"这样一个中心进行的。一般来说,一个好的算法首先应该是比同类算法的时间效率高,算法的时间效率用时间复杂性来度量。

6. 实现算法

现代计算机技术还不能将伪代码形式的算法直接"输入"进计算机中,而需要把算法转变为特定程序设计语言编写的程序。在把算法转变为程序的过程中,虽然现代编译器提供了代码优化功能,然而一些技巧还是会被用到的,例如,在循环之外计算循环中的不变式、合并公共子表达式、用开销低的操作代替开销高的操作等。一般来说,这样的优化对算法速度的影响是一个常数因子,程序可能会提高10% ~50%的速度。

需要强调的是,一个好算法是反复努力和重新修正才会得到的,即使得到了一个貌似完美的算法,它也还是有改进的空间的,换言之,需要不断重复上述问题求解的一般过程,直到算法满足预定的目标要求。

1.5 算法的复杂性分析

算法复杂性的是由运行该算法所需计算机资源的多少来体现的。算法的复杂性越高,所需的计算机资源越多;反之,算法的复杂性越低,所需的计算机资源越少。

计算机资源,最重要的是时间资源与空间资源。因此,算法的复杂性有时间复杂性与空间复杂性之分。需要计算机时间资源的量称为时间复杂度,需要计算机空间资源的量称为空间复杂度。算法的效率是由时间复杂度和空间复杂度来体现的。

算法分析是指对算法的执行时间与所需空间的估算,定量给出运行算法所需的时间数量

级与空间数量级。

1.5.1 时间复杂度

算法是解决问题的方法，一个问题可以有多种解决方法，不同的算法有不同的优缺点。如何对算法进行比较呢？算法可以比较的方面很多，如易读性、健壮性、可维护性、可扩展性等，但这些都不是最关键的方面，算法的核心和灵魂是效率，也就是解决问题的速度。试想，一个需要运行很多年才能给出正确结果的算法，就是其他方面的性能再好，也是一个不实用的算法。

算法的时间复杂性(time complexity)分析是一种事前分析估算的方法，它是对算法所消耗资源的一种渐进分析方法。忽略具体机器、编程语言和编译器的影响就是所谓的渐进分析，渐进分析只关注在输入规模增大时算法运行时间的增长趋势。渐进分析的好处是大大降低了算法分析的难度，是从数量级的角度评价算法的效率。

1. 输入规模与基本语句

刨除和计算机软硬件有关的因素，输入规模可以说是影响算法时间代价的最主要因素。输入规模(input scale)是指输入量的多少，一般来说，它可以从问题描述中得到。例如，找出100以内的所有素数。输入规模是100；对一个具有 n 个整数的数组进行排序，输入规模是 n。一个显而易见的事实是：几乎所有的算法，对于规模更大的输入都需要运行更长的时间。例如，需要更多时间来对更大的数组排序，更大的矩阵转置需要更长的时间。所以运行算法所需要的时间 T 是输入规模 n 的函数，记作 $T(n)$。

通常情况下，要精确地表示算法的运行时间函数很困难，即使能够给出，也可能是个相当复杂的函数，函数的求解本身也是相当复杂的。考虑到算法分析的主要目的在于比较求解同一个问题的不同算法的效率，为了客观地反映一个算法的运行时间，可以用算法中基本语句的执行次数来度量算法的工作量。基本语句(basic statement)是执行次数与整个算法的执行次数成正比的语句，基本语句对算法运行时间的贡献最大，是算法中最重要的操作。

【例1-2】 对如下顺序查找算法，请找出输入规模和基本语句。

```
inc seqsearch(int A[],int n,int k)          //在数组 A[n]中查找值为 k 的记录
{
    for(int i=0;i<n;i++)
      if(A[i]==k)break;
    if (i==n)return 0;                      //查找失败，返回失败的标志 0
    else return(i+1);                       //查找成功，返回记录的序号
}
```

解：算法运行时间是循环语句的主要消耗，循环的执行次数取决于待查找记录个数 n 和待查值 k 在数组中的位置，每执行一次 for 循环，都要执行一次元素比较操作。因此，输入规模是待查找的记录个数 n，基本语句是比较操作(A[i]==k)。

【例1-3】 对如下起泡排序算法，请找 m 输入规模和基本语句。

```
void BubbleSort(int r[],int n)
{
```

```
    int bound,exchange = n - 1;                          //第一趟起泡排序的区间是[0,n - 1]
    while( exchange! - 0)                                //当上一趟排序有记录交换时
    {
    bound = exchange;exchange = 0;
    for( int j = 0;j < bound;j ++ )                      //一趟起泡排序区间是[0,bound]
      if( r[j] > r[j + 1])
      {
        int temp = r[j];r[j] = r[j + 1];r[j + 1] = temp; //交换记录
        exchange = j;                                    //记载每一次记录交换的位置
      }
    }
  }
```

解:算法由两层嵌套的循环组成,每一趟待排序区间的长度决定了内层循环的执行次数,也就是待排序记录个数,外层循环的终止条件是在一趟排序过程中没有交换记录的操作,是否有交换记录的操作取决于相邻两个元素的比较结果,也就是说,每执行一次 for 循环,都要执行一次比较操作,而交换记录的操作却不一定执行。因此,输入规模是待排序的记录个数 n,基本语句是比较操作(r[j] > r[j + 1])。

【例 1-4】 下列算法实现将两个升序序列合并成一个升序序列,请找出输入规模和基本语句。

```
  void Union( int A[ ],int n,int B[ ],int m,int C[ ])   //合并 A[n]和 B[m]
  {
    int i = 0,j = 0;k = 0;
    while( i < n && j < m)
    {
      if( A[i] <= B[j]) c[k ++ ] = A[i ++ ];             //A[i]与 B[j]中较小者存入 c[k]
      else c[k ++ ] = B[j ++ ];
    }
  while( i < n) c[k ++ ]A[i ++ ];                        //收尾处理,序列 A 中还有剩余记录
  while( j < m) c[k ++ ] = B[j ++ ];                     //收尾处理,序列 B 中还有剩余记录
  }
```

解:算法由三个并列的循环组成,三个循环将序列 A 和 B 扫描一遍,因此,两个序列的长度 n 和 m 就是输入规模。第 1 个循环根据比较结果决定执行两个赋值语句中的一个,因此,可以将比较操作(A[i] <= B[j])作为基本语句,第 2 个循环的基本语句是赋值操作(C[k ++] = A[i ++]),第 3 个循环的基本语句是赋值操作(C[k ++] = B[j ++])。

2. *算法的渐进分析*

算法的渐进分析不是从时间量上度量算法的运行效率,而是度量算法运行时间的增长趋势。也就是说,只考察当输入规模充分大时,算法中基本语句的执行次数在渐近意义下的阶,通常使用大 O(读做大欧)符号表示。