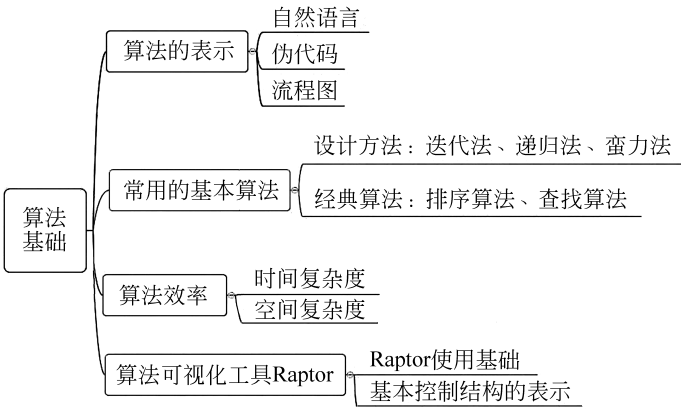
第5章

算法基础

算法这个词对于非计算机专业人士来说，或许有些晦涩或者神秘。其实，广义的算法在人们日常生活中随处可见。例如，一道菜品的烹饪菜谱里，精确地描述了所需配料及步骤（不是诸如“盐少许”这样一些模糊的字眼）；要完成一道四则运算的算术题，应该按照先乘除后加减，有括号从里到外要优先。所以通俗地说，算法就是解决问题的方法和步骤。显然，方法不同，求解问题的步骤也就不同。

本章首先介绍计算机算法的概念，讨论算法的表示方法；其次介绍一些常用的基本算法，对算法效率进行基础分析；最后介绍一种快速算法可视化工具Raptor的使用。



5.1 算法的概念

5.1.1 算法概述

计算机科学家David Harel在他著的《算法学：计算精髓》一书中说道：“算法不仅是计算机科学的一个分支，它更是计算机科学的核心”。在信息技术高速发展的今天，无论是云计算、大数据还是人工智能等现代计算机新技术，要想完成任何实质性的工作，归根到底都是算法与数据结构的比拼。

现实生活中，如 GPS，可以在几秒钟的时间内，从无数条可能线路中找到抵达目的地的最快捷路径；用户在网上购物，同时要解决防止他人窃取信用卡账号问题。这些涉及计算机上所运行的最短路径算法及加密算法等，可以说算法无处不在。

在计算领域，问题求解是一个系统过程，这个过程包括分析问题、设计算法、编程调试和使用维护 4 个阶段。算法是程序的灵魂，是编程思想的核心。程序是算法用某种程序设计语言的具体实现，同一个算法可以用任意一种计算机语言来表达。程序依赖于程序设计语言，甚至依赖于计算机结构。算法质量的优劣很大程度上决定了一个程序的效率。

计算思维的核心之一就是算法思维，算法是一种求解问题的思维方式。深入学习和分析计算机解决各种问题的算法，可以拓展解决一般问题的思路，锻炼和培养计算思维的能力。学习算法的思想，可以使思维更加有条理、更加具有逻辑性，对日后无论是提升工作效率，还是未来的智能生活都会产生深远影响。

5.1.2 算法的定义及特性

3.1 节中介绍了十进制数和二进制数之间的转换方法，了解到多媒体数据涉及压缩和解压缩算法。现实世界中的问题千奇百怪，在计算机科学领域中算法当然也就千变万化。算法究竟是什么？这里给出如下定义。

算法(Algorithm)是指解题方案准确而完整的描述，是一系列解决问题的清晰指令。算法代表着用系统的方法描述解决问题的策略机制。也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。

一个算法应该具有以下 5 个重要的特性。

(1) **有穷性**(Finiteness)：一个算法必须能在执行有限个步骤后终止。也就是说，任何算法必须在执行有限指令后结束，不能出现“死循环”。

(2) **明确性**(Definiteness)：算法的每个步骤必须有确切的含义。也就是说，算法的描述必须无二义性，执行时不会模棱两可，含糊不清。

(3) **可行性**(Effectiveness，也称有效性)：算法中执行的任何计算步骤都可以分解成可执行的基本操作步骤，即算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现。

(4) **零个或多个输入**(Input)：大多数算法开始执行时都必须输入初始数据或初始条件。零个输入是指算法本身已给出了初始条件，或者较为简单的算法，如计算 1+2 的值，不需要任何输入参数。

(5) **一个或多个输出**(Output)：输出项反映对输入数据加工后的结果，没有输出的算法是毫无意义的。

凡是算法都必须满足以上 5 个特性，否则不能称为算法，只能称为计算过程。由此可以说，算法是一个详细、精确、不模糊的“菜谱”。算法是在有限的时间内用有限的数据解决问题的明确指令集合。

5.1.3 算法要素

一个算法由一系列基本操作组成，这些操作又是按照一定的控制结构所规定的次序执行的，所以算法由基本操作和控制结构两个要素组成。

1. 基本操作

算法中的每步都必须能分解成计算机的基本操作，否则算法是不可行的。计算机可以执行的基本操作有如下4类。

(1) **算术运算**：加、减、乘、除等。

(2) **关系运算**：大于、小于、等于、不等于等。

(3) **逻辑运算**：与、或、非等。

(4) **数据传输**：输入、输出、赋值等。

2. 控制结构

一个算法的功能不仅取决于所选用的操作，还与各操作之间的执行顺序相关。算法的控制结构就是指各操作之间的某种执行顺序。理论和实践证明，无论多复杂的算法都可以通过顺序结构、选择结构和循环结构这3种基本控制结构组合构造出来。

(1) **顺序结构**。顺序结构是指算法按照自上而下的先后顺序一步一步地执行，这是最简单的结构。

(2) **选择结构**。选择结构也称分支结构，是算法根据不同情况选择性地执行。满足条件，执行一组操作；否则不执行或执行另外一组的操作。

(3) **循环结构**。循环结构是根据条件对某一部分的操作重复执行多次，直到条件不满足为止。利用循环结构可以实现有规律的重复计算。

5.2 算法的表示

任何算法都需要将其明确地描述出来。有了算法的描述，才能更好交流和研究，知道如何去解决问题，这个描述过程就是算法的表示。算法的表示方法可以用自然语言、伪代码、流程图来描述。当然，算法最终要用计算机语言编写程序，上机实现。

5.2.1 自然语言

自然语言是指人们日常生活使用的语言，可以是汉语、英语或者其他语言。用自然语言表示算法简单、通俗易懂。但不足之处：一是文字比较冗长，不易清楚地表达算法的逻辑流程；二是不够严谨，易产生歧义性。因此，自然语言用于描述一些简单的问题步骤，或是对算法大致步骤做粗略描述时使用。

下面给出两个采用自然语言描述算法的例子。

例5.1 写出这个问题的自然语言算法描述：计算机输入3个数，求这3个数中的最大数。

算法描述如下。

V5.1 算法表示示例

(1) 输入 x、y、z 的值。

(2) 比较前两个数 x 和 y。如果 $x > y$,则将 x 存入 max;否则,将 y 存入 max。

(3) 再将 z 和 max 比较,如果 $z >$ max,则将 z 存入 max。

(4) 输出 max,即为 3 个数中的最大数。

例 5.2 写出欧几里得算法的自然语言描述。

说明:欧几里得算法又称辗转相除法,是人类历史上最早记载的算法。由古希腊数学家欧几里得在其著作 *The Elements* 中描述的,用于求两个正整数的最大公约数的算法。

最大公约数是指两个或多个整数共有约数中最大的一个。如 24 和 60 的最大公约数是 12。

24 的约数有 1、2、3、4、6、8、12、24。

60 的约数有 1、2、3、4、5、6、10、12、15、20、30、60。

共有的约数有 1、2、3、4、6、12,所以 12 是 24 和 60 的最大公约数。

欧几里得算法求解步骤如下。

(1) 输入两个数 a 和 b。

(2) 如果 $a<b$,则交换 a 和 b。

(3) 求出 a、b 两数相除的余数 r。

(4) 如果余数 r 为 0,则较小数 b 为最大公约数,算法结束;否则执行步骤(5)。

(5) 将 b 替换 a,r 替换 b。

(6) 重复执行步骤(3)。

5.2.2 伪代码

伪代码是用介于自然语言和计算机语言之间的文字和符号(包括数字符号)来描述算法。它借助于计算机语言的控制结构,但不拘泥固定的语法和格式,结合部分自然语言混合设计。伪代码具有书写简洁、结构清晰、可读性好等优点,便于向程序过渡,所以一般专业人员习惯用伪代码进行算法描述。当然不足之处是不够直观,错误不易排查。

使用伪代码描述算法,不用拘泥于具体的实现。伪代码描述形式上并不是非常严格,通常主要操作、相关符号和关键字如下。

算术运算符:+、-、*、/、mod(取余)、^(乘方)

关系运算符:=、≠、<、>、<=、>=

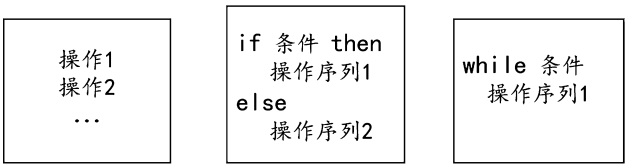
逻辑运算符:and(与)、or(或)、not(非)

输入输出:input、output

赋值:用←或者=表示。如:n=1;x←x+1。表示将赋值号右边的值赋值给左边的

变量。

伪代码表示的3种控制结构如图5.1所示。



(a) 顺序结构　(b) 选择结构　(c) 循环结构

图5.1 伪代码表示的3种控制结构

下面给出两个采用伪代码描述算法的例子。

例5.3 根据给定的成绩,写出判定成绩等级的算法伪代码,条件:成绩60分及以上为及格,否则为不及格。

问题分析:①输入:成绩。输入的成绩用变量score表示。②处理:判定等级。利用选择结构实现,判定等级用变量grade表示。③输出:等级grade。

算法的伪代码如下:

```
begin
    input score
    If score >=60 then
        grade←"及格"
    else
        grade←"不及格"
    output grade
end
```

例5.4 用伪代码描述,求解1+2+3+…+n累加和的算法。

问题分析:面对这样的式子1+2+3+…+n,最简单的计算思想就是从1开始,逐项依次累加,最终得到累加后的结果。系列数相加可以利用循环来实现。

①输入:累加的起始和终止值。系列要加的数据用变量i表示,i为1～n,n输入给定。②处理:利用循环依次累加。累加和用变量sum表示,其初始值设为零(通常这个变量也称为累加器)。③输出:累加后的最终结果sum。

算法的伪代码如下:

```
begin
    input n
    i =1
    sum =0
    while i <=n
        sum =sum +i
        i =i +1
```

```
    output sum
end
```

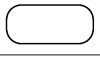
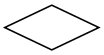
5.2.3 流程图

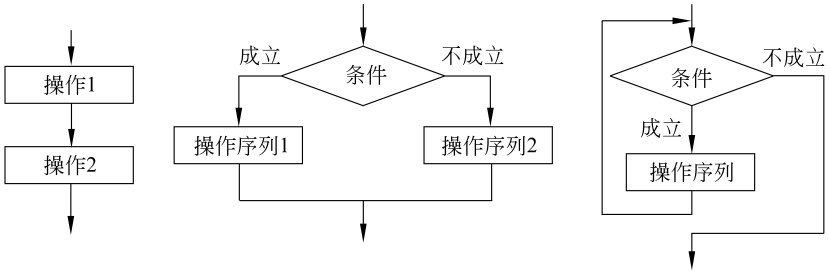
流程图是使用一些图形框和带箭头的流程线描述各种不同性质的操作和执行走向，形象直观，易于理解。这种方法在程序语言发展的早期广泛应用，美国国家标准研究所(ANSI)还规定了一些常用的流程图符号。但是流程箭头可以随意表达操作步骤的次序和转移，对于大型、复杂问题用流程图绘制非常不方便，所以适合描述简单算法。

流程图在表明结构，或者用于理顺和优化业务过程的其他领域还是很有帮助的。如企业使用流程图说明生产线上的工艺流程，或者描述业务走向、数据信息流向等。

常用的流程图符号如表 5.1 所示，3 种控制结构的流程图表示如图 5.2 所示。

表 5.1 常用的流程图符号

符号	符号名称	功 能 说 明
	起止框	表示算法的开始和结束
	处理框	表示执行一个步骤
	判断框	表示要根据条件选择执行路线
	输入输出框	表示需要用户输入或由计算机自动输出的信息
	流程线	指示流程的方向



(a) 顺序结构　(b) 选择结构　(c) 循环结构

图 5.2 3 种控制结构的流程图表示

下面给出两个采用流程图描述算法的例子。

例 5.5 根据给定的成绩，使用流程图表示判定成绩等级的算法，条件：成绩 85 分及以上为优秀，60 分及以上为及格，否则为不及格。

算法流程图表示如图 5.3 所示。

例 5.6 使用流程图表示例 5.2 欧几里得算法求最大公约数，如图 5.4 所示。

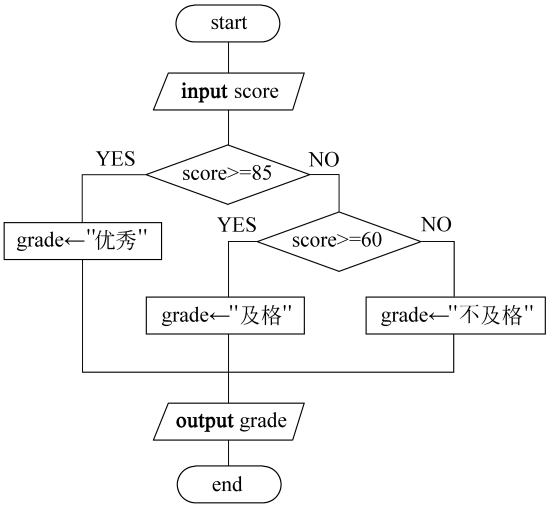


图 5.3 成绩判定流程图

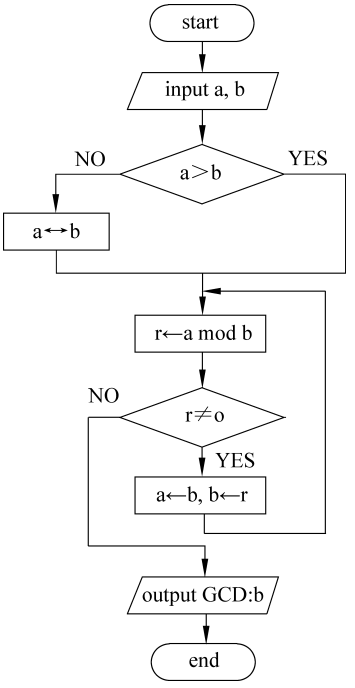


图 5.4 欧几里得算法流程图

5.3 常用的基本算法

人类希望利用计算机解决的问题千差万别，而且同一个问题可以用不同的算法解决。在计算机科学中，针对不同问题，算法设计的基本思想和方法有很多，大致可以分为基本算法（诸如迭代法、递归法、蛮力法、排序和查找算法等），数据结构算法、分治算法、贪心算法、回溯算法、动态规划算法、加密算法、图论算法等一些经典算法。有些算法充满智慧但难度较大，作为基础教材，这里只介绍几个最基本的常用算法，以此让读者理解算法设计的基本方法，明白计算机世界如何解决问题。

5.3.1 迭代法

迭代（Iteration）算法也称辗转法，是一种不断用变量的旧值递推出新值的过程。

迭代算法是用计算机解决问题的一种常用算法，它利用计算机运算速度快、适合做重复性操作的特点，从已知的变量值出发，根据递推公式，让计算机对一组操作进行重复执行，每次执行这组操作，都从变量的原值推出它的一个新值，把一个复杂的、庞大的计算过程转换为简单过程的多次重复。

5.2 节例题中求最大公约数、累加和，都是迭代算法的基础应用。

利用迭代算法解决问题，主要有 3 方面的设计工作。

（1）**确定迭代变量**。在用迭代法解决的问题中，至少存在一个直接或间接地不断由

旧值递推出新值的变量，这个变量就是迭代变量。

(2) **建立迭代关系式**。迭代关系式是指如何从变量的前一个值推出其下一个值的公式(或关系)。迭代关系式的建立是解决迭代问题的关键，可以用顺推或者倒推的方式来完成。

(3) **对迭代过程进行控制**。在什么情况结束迭代过程，不能让迭代过程无休止地重复执行下去。这是迭代算法必须考虑的问题。

例如，在求最大公约数的算法设计时：循环不变式第一次是求 a 和 b 相除的余数 r，经 $a=b$，$b=r$ 操作，就实现了第二次还是求 a 和 b 相除的余数，这就是迭代关系式。不断由旧值递推新值的迭代变量是 a 和 b，在余数 r 为 0 时，结束循环迭代过程。

为进一步理解迭代算法，下面再给出两个实例。

例 5.7 写出求解 $P = n!$ 的算法($n! = 1\times2\times3\times\cdots\times n$)。

问题分析：迭代变量为 p，设置 p 初始值为 1，累乘变量 i 初始值为 2。$p = p\times i$ 为迭代关系式，每循环一次，累乘迭代变量 p 完成一次与变量 i 的乘积。i 也为循环控制变量，迭代过程在 i 超过 n 时结束循环，完成累乘的计算。

算法的伪代码如下：

```
begin
    input  n
    p =1
    i =2
    while  i <=n
        p =p * i
        i =i +1
    output  p
end
```

例 5.8 计算斐波那契序列(Fibonacci Sequence)的第 n 项值。

说明：斐波那契序列指的是数列的第 1 项和第 2 项为 1，从第 3 项开始，每项均等于它前两项之和。

问题分析：前两项 $F_1=1$，$F_2=1$，在 $n>2$，F_n 总可以由 $F_{n-1}+F_{n-2}$ 得到。由旧值递推出新值，这是一个典型的迭代关系。假设前一项用变量 a 表示，后一项用变量 b 表示，要求的项用变量 c 表示。

算法的伪代码如下：

```
begin
    input  n
    a =1
    b =1
    i =3
    while  i <=n
        c =a +b
        a =b
```

```
        b =c
        i =i +1
    output  c
end
```

5.3.2 递归法

递归(Recursion)在数学与计算机科学中是指在函数的定义中使用函数自身的方法。递归算法是一种直接或间接地调用自身算法的过程。

递归算法解决很多的计算机科学问题是十分有效的。从直观上说,递归是将一个大问题分解为同类的小问题,从待求解的问题出发,一直分解到已知答案的最小问题为止,再逐级返回,从而得到大问题的解。递归算法只需少量的代码就可描述出解题过程所需要的多次重复计算,大大地减少程序的代码量。

实现递归定义的两个要素。

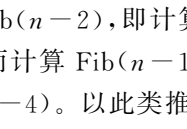
(1) **递归出口**。递归过程必须有一个明确的递归结束条件、结束值。

(2) **递归公式**。过程或函数自身调用的等价关系式,且能向结束条件发展。

例 5.9 使用递归算法,计算斐波那契序列的第 n 项函数值 Fib(n)。

问题分析:依据例 5.8 对斐波那契序列的分析,斐波纳契序列 Fib(1) =1,Fib(2) =1,Fib(n) = Fib(n−1) + Fib(n−2)。

求解 Fib(n),把它推到求解 Fib(n−1)和 Fib(n−2),即计算 Fib(n),必须先计算 Fib(n−1)和 Fib(n−2)。而计算 Fib(n−1)和 Fib (n−2),又必须先计算 Fib(n−3)和 Fib(n−4)。以此类推,直至计算 Fib(2)和 Fib(1),而 Fib(2)和 Fib(1)的值就是已知条件(结束条件)1。也就是说,当 n 为 1 或 2 的情况下可以结束递归。这个递归过程如图 5.5 所示,假设 n=6。



V5.2 斐波那契递归实现

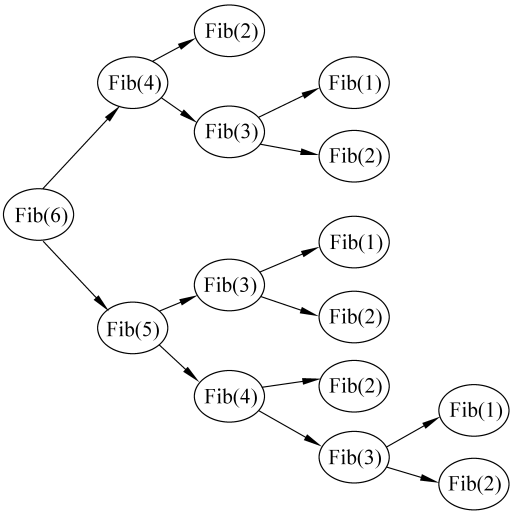


图 5.5 计算斐波那契递归过程

以递归的方法定义关系式如下：

$$\begin{cases} \mathrm{Fib}(n)=1 & (n=1,n=2) \quad \text{（递归出口）} \\ \mathrm{Fib}(n)=\mathrm{Fib}(n-1)+\mathrm{Fib}(n-2) & (n>2) \quad \text{（递归公式）} \end{cases}$$

伪代码表示这个递归算法如下：

```
def fib(n)
if n =1 or n =2
    return 1
else
    return fib(n-1) +fib(n-2)
```

同样，在计算阶乘($n!$)问题上也可以通过递归方式定义函数实现。因为当 $n=0$ 时，$0!=1$；当 $n>0$ 时，$n!=n\times(n-1)\times(n-2)\times(n-3)\times\cdots\times3\times2\times1=n\times(n-1)!$。例如：$5!=5\times4\times3\times2\times1=5\times4!$。所以计算阶乘，以递归的方法定义关系式如下：

$$\begin{cases} \mathrm{Fac}(n)=1 & (n=0) \quad \text{（递归出口）} \\ \mathrm{Fac}(n)=n\times\mathrm{Fac}(n-1) & (n>0) \quad \text{（递归公式）} \end{cases}$$

5.3.3 蛮力法

蛮力(Brute-force)法是一种简单而直接地解决问题的方法。它直接基于问题的描述，从有限集合中逐一列举集合中的所有元素，对每个元素逐一判断和处理，从中找出问题的解。

枚举法是蛮力法的一种表现形式，它依据给定的条件，遍历所有可能的情况，从中找出满足条件的正确答案。有时一一列举出的情况数目很大，如果超过了所能忍受的范围，则需要进一步考虑排除一些明显不合理的情况，尽可能减少问题可能解的列举数目。

用枚举法解决问题，通常从两方面进行算法设计。

(1) **找出枚举范围**：分析问题所涉及的各种情况。

(2) **找出约束条件**：分析问题的解需要满足的条件，并用逻辑表达式表示。

例 5.10 百鸡问题。百鸡问题是一个著名的数学问题，出自中国古代 5—6 世纪成书的《张邱建算经》。该问题描述：今有鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问翁、母、雏各几何？

题意：公鸡 5 元 1 只，母鸡 3 元 1 只，小鸡 3 只 1 元，花 100 元买 100 只鸡，可买公鸡、母鸡和小鸡各多少只？

问题分析：先假设每种鸡至少买一只，公鸡、母鸡、小鸡可买只数分别为 x、y、z，则买公鸡的钱数为 $5x$，买母鸡的钱数为 $3y$，买小鸡的钱数为 $z/3$。根据题意可列出代数方程：

$$x+y+z=100 \text{（只）}$$
$$5x+3y+(1/3)z=100 \text{（元）}$$

有两个方程式，三个未知量，是典型的不定方程组，应该有多种解。这类问题用蛮力