CHAPTER

01

演算法概述

電腦技術，特別是電腦程式設計大大改變了人們的工作方式。現代的設計工作大多透過電腦程式設計交給電腦來完成。其中，演算法造成了至關重要的作用。可以毫不誇張地說，演算法是一切程式設計的靈魂和基礎。本章介紹演算法的一些基本概念、發展歷史、演算法表示和應用等。

1.1 什麼是演算法

那麼究竟什麼是演算法（algorithm）呢？從字面意義上也比較好了解，演算法也就是用於計算的方法，透過這種方法可以達到預期的計算結果。

此外，在一般的教科書或者字典上也有關於演算法的專業解釋，例如：演算法是解決實際問題的一種精確描述方法、演算法是對特定問題的求解步驟的一種精確描述方法等。目前，被廣泛認可的演算法的專業定義是，演算法是模型分析的一組可行的、確定的和有限的規則。

其實，通俗地講，演算法可以了解為一個完整的解答步驟，由一些基本運算和規定的運算順序而構成。透過這樣的解答步驟可以解決特定的問題。從電腦程式設計的角度看，演算法由一系列求解問題的指令構成，能夠根據標準的輸入，在有限的時間內獲得有效的輸出結果。演算法代表了用系統的方法來描述解決問題的一種策略機制。

我們可以舉一個例子來看看演算法是如何在現實生活中發揮作用的。最典型的例子就是統籌安排，假設我們要有 3 件事（事件 A、事件 B 和事件 C）要做：

- 做事件 A 需要耗費 5 分鐘；
- 做事件 B 需要耗費 5 分鐘但需要 15 分鐘的時間才可以得到結果，例如燒水等待水開的過程；
- 做事件 C 需要耗費 10 分鐘。

那麼我們應該怎麼來做這三件事情呢？一種方法是依次做，如圖 1-1 所示，做完事件 A，再做事情 B，最後做事情 C。這樣，整體耗時是 5+（5+15）+10=35 分鐘，這顯然是浪費時間的一種方法。

在實際生活中比較可取的方法是，先做事件 B，在等待事件 B 完成的過程中做事件 A 和事件 C。這樣，等待事件 B 完成的 15 分鐘正好可以完成事件 A 和事件 C。此時，整體耗時是 5+15=20 分鐘，效率明顯提昇，如圖 1-2 所示。

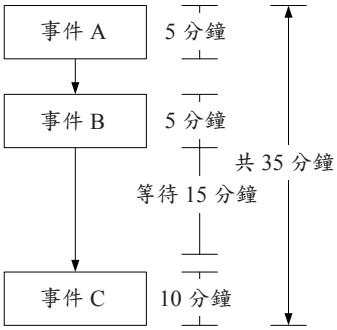

圖 1-1 方法一

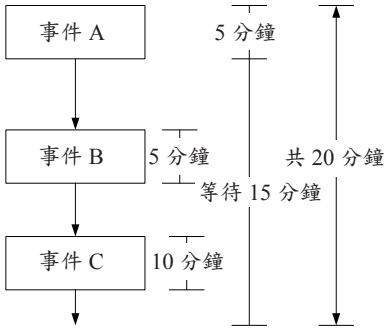

圖 1-2 方法二

在上面的這個例子中，提到了兩種方法，也就可以看做兩種演算法。第一種演算法效率低，第二種演算法效率高，但都達到了做完事情的目的。從這個例子可以看出，演算法也是有好壞區別的，好的演算法可以提供工作和生活的效率。演算法的基本工作就是對於一個實際的問題，找到一個高效率的處理方法，進一步獲得最佳的結果。

一個典型的演算法一般都可以從其中抽象出 5 個特徵：有限性、確切性、輸入、輸出和可行性。下面結合上面的例子來分析這 5 個特徵。

有限性：

演算法的指令或者步驟的執行次數是有限的，執行時間也是有限的。例如在上面的例子中，透過短短的幾步就可以完成工作，而且執行時間都是有限的。

確切性：

演算法的每一個指令或者步驟都必須有明確的定義和描述。例如在上面的例子中，為了完成做完三件事情的工作，每一步做什麼事情都有明確的規定。

輸入：

一個演算法應該有對應的輸入條件，用來刻畫運算物件的初始情況。例如在上面的例子中，有三個待完成的事件（事件 A、事件 B 和事件 C）便是輸入。

輸出：

一個演算法應該有明確的結果輸出。這是容易了解的，沒有得到結果的演算法是毫無意義的。例如在上面的例子中，輸出結果便是三件事情全部做完了。

可行性：

演算法的執行步驟必須是可行的，且可以在有限時間內完成。例如在上面的例子中，每一個步驟都切實可行。其實，無法執行的步驟也是毫無意義的，解決不了任何實際問題。

目前，演算法的應用非常廣泛，常用的演算法包括遞推 (譯註：原文為 recurrence，繁體中文同樣翻為「遞迴」，為和本書其它之「遞迴」分別，延用簡體「遞推」)、遞迴 (recursion)、窮舉、貪婪、分治、動態規劃和反覆運算等多種。本書將逐步向讀者展示各種演算法的原理和應用。

1.2 演算法的發展歷史

關於演算法的起源，可以追溯到中國古代西元前 1 世紀的《周髀算經》，它是算經的十書之一，原名《周髀》，主要闡述古代中國的蓋天說和四分曆法。在唐朝的時候，被定為國子監明算科的教材之一，並改名為《周髀算經》。《周髀算經》中記載了畢氏定理、開平方問題、等差級數問題等，裡面用到了相當複雜的分數演算法和開平方演算法等。在隨後的發展中，出現了割圓術、秦九韶演算法和剩餘定理等一些經典演算法。

在西方，西元 9 世紀波斯數學家 al-Khwarizmi 提出了演算法的概念。“演算法”最初寫為“algorism”，意思是採用阿拉伯數字的運算法則。到了 18 世紀，演算法正式命名為現在的“algorithm”。由於中文字計算的不方便，導致中國古代演算法的發展比較緩慢，而採用阿拉伯數字的西方國家則在隨後的時間中發展迅速。例如著名的阿基米德演算法（又稱輾轉相除法）就是典型的演算法。

在歷史上，大多數人都認可 Ada Byron 為第一個程式設計師。她在 1842 年撰寫巴貝奇分析機上的白努利方程的求解演算法程式，雖然未能執行，但奠定了電腦演算法程式設計的基礎。

後來，隨著電腦的發展，在電腦上實現各種演算法已成為可能。演算法在電腦程式設計領域又得到了再次的重要發展。目前，幾乎所有的程式設計師無論採用何種程式語言，都需要與演算法進行處理。

1.3 演算法的分類

演算法是一門古老且龐大的科學，隨著歷史的發展，演化出多種多樣的演算法。按照不同的應用和特性，可以分為不同的類別。

1．按照應用來分類

按照演算法的應用領域，也就是解決的問題，演算法可以分為基本演算法、資料結構相關的演算法、幾何演算法、圖論演算法、規劃演算法、數值分析演算法、加密 / 解密演算法、排序演算法、尋找演算法、平行算法和數論演算法等。

2．按照確定性來分類

按照演算法結果的確定性來分類，可以分為確定性演算法和非確定性演算法。

- 確定性演算法：這種演算法在有限的時間內完成計算，得到的結果是唯一的，且經常取決於輸入值。
- 非確定性演算法：這種演算法在有限的時間內完成計算，但是得到的結果往往不是唯一的，也就是存在多值性。

3．按照演算法的想法來分類

按照演算法的想法來分類，演算法可以分為遞推演算法、遞迴演算法、窮舉演算法、貪婪演算法、分治演算法、動態規劃演算法和反覆運算演算法等多種。

1.4 演算法相關概念的區別

演算法其實是一個很抽象的概念，往往需要依靠於實際的實現方法才能表現其價值，例如在電腦程式設計中的演算法、數值計算中的演算法等。本書所重點講解的便是演算法在電腦程式設計中的應用。正是由於演算法的抽象性，導致讀者很容易產生混淆，這裡有必要說明一些基本的概念。

1.4.1 演算法與公式的關係

從前面所談到的演算法，讓我們很容易聯想到數學中的公式。公式也是解決某類別問題，有特定的輸入和結果輸出，能在有限時間內完成，並且公式都是完全可以操作計算的。其實公式確實是提供了一種演算法，但演算法絕不完全等於公式。

公式是一種高度精簡的計算方法，可以認為就是一種演算法，其是人類智慧的結晶。而演算法並不一定是公式，演算法的形式可以比公式更複雜，解決的問題更加廣泛。

1.4.2 演算法與程式的關係

正如前面所述，演算法是依靠於實際的實現方式的。雖然我們一提到演算法，就聯想到電腦程式設計，但演算法並非僅用於此。例如，在傳統的筆算中，透過紙和筆按照