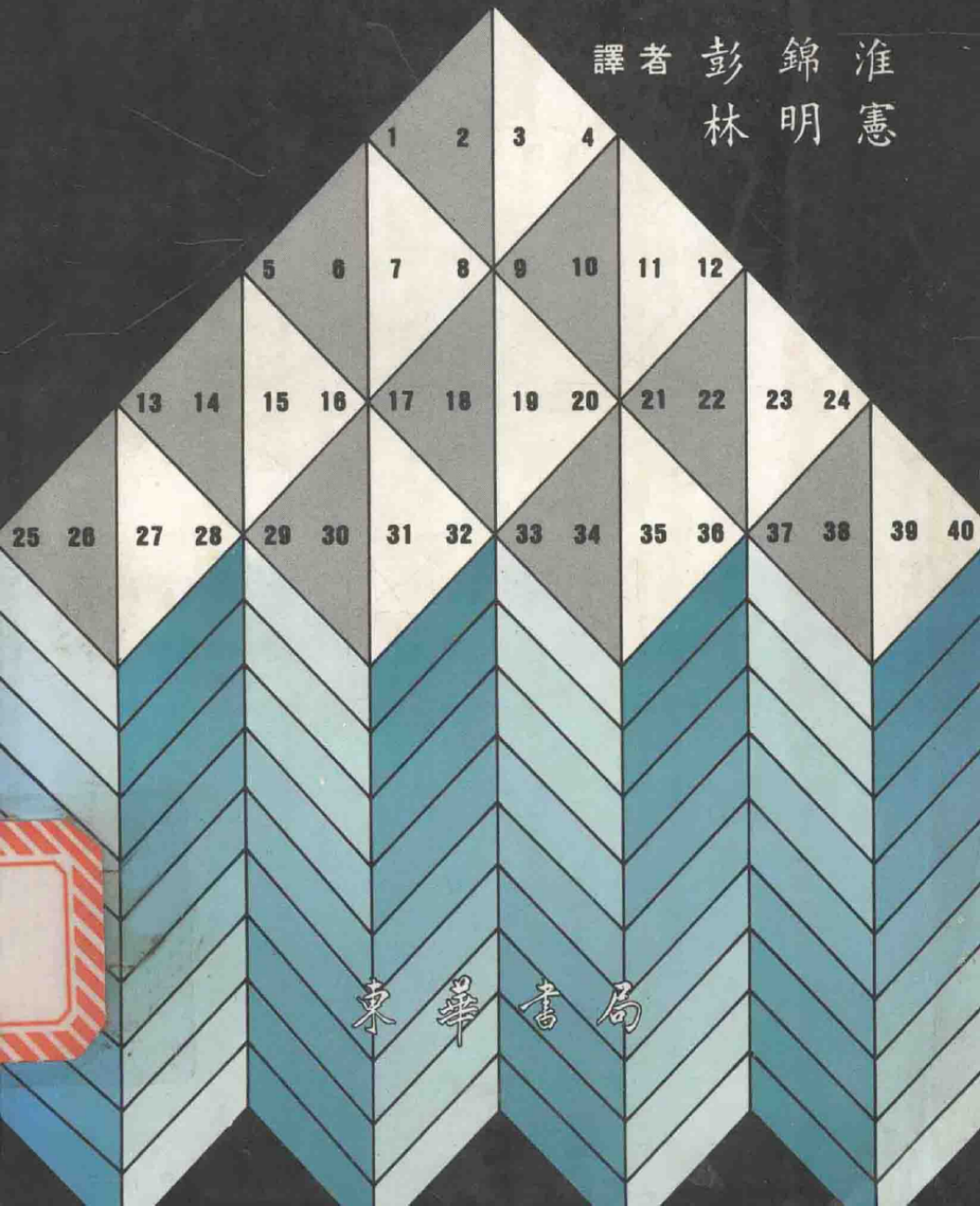
資料結構及運算法則

譯者 彭錦淮
林明憲



東華書局

資料結構及運算法則

(1983)

原 著 者

ALFRED V. AHO

JOHN E. HOPCROFT　JEFFREY D. ULLMAN

譯　　者

彭錦淮　林明憲

東華書局印行

中華民國七十三年十一月 初 版

大專用書 資料結構及運算法則

定價 新臺幣貳佰貳拾元整

（外埠酌加運費滙費）

著 者 彭錦淮 林明憲

發行人 卓鑫淼

出版者 臺灣東華書局股份有限公司
臺北市博愛路一〇五號
郵撥帳號：00064813

印刷者 合興印刷廠

行政院新聞局登記證 局版臺業字第零柒貳伍號
(73030)

序　言

本書指出今日電腦程式用的最多的資料結構及運算法則。在這本書所提的前面六章是以“電腦運算法則”一書做爲基礎，我們擴大它所講的範圍，再加上外部儲存器及記憶器處理的運算法則，所以本書很適合作爲“資料結構及運算法則”的首先課程。唸本書唯一的前提是假定對於高階語言－帕斯卡（ PASCAL ）均已熟悉。

我們盡力的涵蓋在利用電腦解決問題中有關於資料結構及運算法則的範圍。我們利用一些純理論的資料類型來描述及實行運算法則，雖然這些純理論的資料類型僅在一些廣泛使用到的程式語言開頭出現，我們認爲，無論是何種語言，它們均是在程式設計上一種有用的工具。

我們也介紹步階計數（ STEP COUNTING)及時間複雜性的觀念，並視爲在解決過程中一個不可或缺的部份。我們一直深信，隨著電腦速度愈快及程式的加大，對程式設計員此種觀念是將要繼續處理的問題。所以運算法則的時間複雜性甚至將比新產生且可用的硬體線路倍顯重要。

運算法則的表示

我們利用帕斯卡語言來描述運算法則及資料結構，因爲此語言爲衆所週知。起初我們提出純理論及帕斯卡程式的各種運算法則，因爲我們覺得在跑程式解決過程中，盡一切的把問題公式化非常重要。然而我們所提出的運算法則在任何高階程式語言中均可容易地實現。

本書的用途

第一章是介紹一些敍述，包括從問題到程式過程的觀念的說明，以及在過程中純理論的資料類型所扮演的角色。同時也介紹步階計數

和“大-歐”（big-oh）和“大-亞米茄”（big-omega）的觀念。

第二章介紹傳統的串列（LIST），堆疊（STACK）和佇列（QUEUE）的結構，以及當成一種純理論資料類型的數學函數之對映。第三章介紹樹（Tree）和一些使樹有效地應用在各種不同運算之基本的資料結構。

第四、五兩章介紹一些以數學類型爲主的重要純理論之資料類型。同時更深地涵蓋語彙（Dictionaries）及佇列的優先權。對於雜湊表列（Hash Table）、二元搜索（Binary Search）樹、部份次序（partially order）樹、三樹（tries）和2-3樹等這些概念用標準型實現，均在第五章中進一步的探討。

第六、七兩章是講到圖形，其中第六章是講直接圖形，第七章則講間接圖形。這兩章對於本書的貢獻在運算法則上要比資料結構來得多，雖然我們如用圖形的表示方法來討論基本的資料結構較爲適當。很多重要的圖形運算法則表現其中，包括先深後廣搜索（depth-first search），找到最小擴張（spanning）樹、最短路徑（paths）和最大配對（matchings）。

第八章是提出內部分類（sorting）運算的主要法則：快速（quick）分類，累堆（heap）分類，貯藏（bin）分類以及像插入（insert）分類較簡單、效率較差的方法。本章我們也涵蓋了線性時間的運算法則用來求出中間值（medians）和統計次冪。

第九章是討論遞迴過程（recursive procedures）的漸進式分析，當然它包涵了解決這些問題的反復關係及技巧。

第十章描述設計運算法則的重要技巧，包括“劃分及統合（divide-and-conquer）”、動態程式、當地（local）搜尋等運算法則，和其他不同結構形態樹的搜尋。

最後這兩章是談到外部儲存器組織和記憶器的管理，十一章是涵蓋了外部分類和大容量的儲存器組織，包括B樹和註標（index）結構。

第十二章包括了記憶器的處理，可依據資料塊（ block ）大小的設定是否固定還是可變的，以及依據由使用者程式所產生或是由系統程式所處理的餘料收集（ garbage collection ）之資料塊大小是否伸縮自如，而分爲四個區域來討論。

作者在哥倫比亞大學、康乃爾大學、史丹福大學的大學部及研究所，同時以本書作爲“資料結構和運算法則”這門課的教本。比方說，在史丹福，就以本書的初版作爲教從大三到研一各種程度不同的學生，爲期十週“資料結構”課程的教本。這譯本的範圍祇限在第一章到第四章、第九章、第十章、第十二章和第五章到第七章的部份而已。

習題

在每章的末了都有一些難易不同的習題。大多數這些都是直截地測驗本章所提的主要內容。有些題目需要再深思一番，這些就用一個“＊”來表示。有兩個“＊＊”的題目就比較難，這些適合更高級的課程。在每章的最末尾有參考書目供額外閱讀參考之用。

謝銘

我們感謝貝爾實驗室所準備以UNIX™ 爲基底極佳的教材和提供資料傳輸設備，使得分在不同區域的作者們能容易地製作原稿。很多我們的同事在讀完不同部份的原稿後，也給我們有價值的見解。尤其是我們謝謝瓊貝特利，布倫柯尼罕，史蒂芬麥哈利，凱利那摩維雀，保羅南貝基，羅勃派克，羅勃泰利安，和彼得溫伯格他們助益的建議。最後我們要向克萊爾莫滋格太太，對於她在準備原稿打字上熟練的幫忙，致最誠摯的謝意。

阿弗雷得・愛華
約翰・霍柯弗特
傑弗瑞・愛得曼

目　　錄

第一章
運算法則之分析與設計

撰寫一電腦程式以解決一特定的問題時，需牽涉許多步驟。這些步驟包括問題的定型與敍述、設計解答、完成解答、測試與註解，最後尙有解答的評估。本章概述我們對上述步驟所採取的對策。其後各章將討論一般電腦程式所應用之運算法則及資料結構。

§1-1 由問題至程式

如能瞭解正待解決的問題，便已成功了一半。在初期進行時，多數的問題並無簡單、明確的描述。事實上，有些問題，就如同構思一份精緻的菜譜或是維護世界和平般，不易予以有系統的描述，期以電腦作答。即使問題終究得以電腦解答，卻仍存在極多問題的參數待選擇。通常僅能依據實驗才能獲得這些參數的合理數值。

如能將問題的各種狀況代之以合適的模式，對解決問題極有助益。一旦問題類型化後，我們便可依據此一精確模式尋找解答，甚至，可以針對問題的模式，採用現成的程式來解決。即使沒有現成程式可循，至少，我們可以由已知的條件，就模式的特性，建立完整的解答。

幾乎所有數學或科學的分支，都可使用於協助模擬一些問題的領域。本質上有關數值的問題，可藉助於線性聯立方程式（例如：求電路中的電流值；或求樑架結構中所受之應力等），或者以微分方程式來模擬（例如：預估人口的成長率；或是化學反應的速率等）。在處理符號或文字的問題上，則可以文字串及正式的文法來定型。具有此類性質的問題如編輯程式（Compilation）（將程式語言翻譯成機器語言）及資料的存取作業等，如在資料庫中辨認出一特定文字。

運算法則（Algorithms）

一旦問題有了合適的數學模式，我們即可嘗試依據此一模式尋求答案。初期的目標僅在尋得一具運算法則形式的答案；亦即，在有限的時間內，以有限的效力完成一組清晰明確的指令。一個整數指定的陳述如 $x := y + z$ 便是一有限效力下執行的實例。在運算法則中，所有的指令均能無限制重複提及，只要指令本身指出其重複的次數即可。不過，不論輸入的數值爲何，此一運算法則終須結束。因而，只要程式不因各種不同輸入下而致產生一無法終止的迴路，便可稱爲一運算法則。

對於運算法則的定義，還須要加以更明確的說明。我們所謂在運算法則中的指令，皆須有“清晰明確”的意義，以及以“有限效力”執行等。但是，對某甲而言，或極明確，對某乙則未必然；同時，很難證明一指令是否在有限的時間內完成。即使能完全了解指令的意義，也很難證明在何種輸入狀況下，指令序列會終止。不過，藉著相互的討論，一組指令序列是否構成一運算法則，便可獲得定論。證明的困難處在於人們通常自誇其符合運算法則。在1-5節中，我們將討論如何估計一般程式語言結構所須的執行時間，以顯示其確切的有限執行時間。

除了使用PASCAL 程式作運算法則外，通常也使用虛擬語言（pseudo - language）來表示運算法則，此類虛擬語言乃結合程式語言的結構及簡單的語文。雖然，我們採用PASCAL 作爲程式語言，但幾乎任何一般程式語言皆可取而代之，符合我們所討論的運算法則。下面的例題說明我們在構思完成一電腦程式時，所須採行的步驟。

【例題1.1】 在一複雜的交叉路口的交通號誌，可藉助數學模式來設計。我們可構思一程式，將交叉路口中允行的進行方向視若輸入（一連續的路徑便是一進行方向），同時在互不碰撞前提下，作最簡的分劃，以建立號誌的模式。而後，便可將獨立的狀況規劃一組號誌。由尋求最簡組合，可以建立一套最簡單的交通號誌系統。

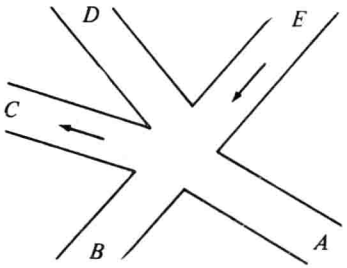


圖 1.1 一個交叉路口。

例如，如圖1.1所示的交叉路口位於普林斯頓大學附近，因一個稱爲 JoJo's 的水窪所造成，此一路口經常造成交通不便，尤其在廻程時，更產生交通阻塞。圖中，C 與 E 路是單行道，其他皆爲雙向道路。在此交叉路口，共有十三種行進路線。其中，某些行進路線可同時並存，如 AB（由 A 向 B）與 EC；而某些行進路線則不可同時實行，如 AD 及 EB，否則將造成壅塞。故而，此一路口的交通號誌決不允許諸如 AD 及 EB 共列的行進方向。然而，AB 及 EC 並行的號誌卻可容許。

對這個問題，我們可用一稱爲圖形（graph）的數學結構來摹擬。在圖形中，包括了一組點，稱作頂點（vertices），及一些連接點的線，稱作邊（edge）。回顧此一交叉路口號誌問題，我們可以繪出一個圖形，其頂點表示了行進方向；其邊則將不能同時並行的路線，所代表的頂點連結起來。由圖 1.1的交叉路口，得到圖 1.2的圖形，以及圖 1.3，另一種表格類型的圖形，其中，在圖 1.2的圖形裏，凡有一邊聯結的兩頂點 i 及 j，便在表格中相對應的 i，j 行列中填入1。

此一圖形可幫助我們解決交通號誌的設計問題。在圖形上著色（coloring），即將每一頂點賦予不同的顏色，期使相接連的每對頂點皆不同色。不難看出，我們的問題便是設法以最少的顏色，在不相容路線所代表的頂點上著色。

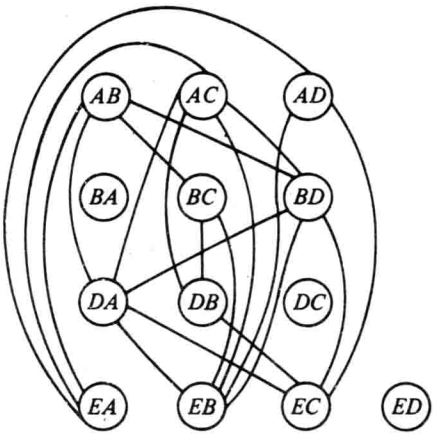


圖 1.2 表示不相容路線的圖形

	AB	AC	AD	BA	BC	BD	DA	DB	DC	EA	EB	EC	ED
AB					1	1	1			1			
AC						1	1	1		1	1		
AD										1	1	1	
BA													
BC	1							1			1		
BD	1	1					1				1	1	
DA	1	1				1					1	1	
DB		1			1							1	
DC													
EA	1	1	1										
EB		1	1		1	1	1						
EC			1			1	1	1					
ED													

圖 1.3 表示不相容路線的表格

著色圖形的問題已經研究了數十年，其運算法則的理論導引了不少有關的問題。很不幸的，以最少的顏色賦予任意的圖形上的問題歸屬於一類稱爲“ NP-完全問題”，基本上，便是“試盡各種可能”的類型。在著色的問題中，“試盡各種可能”便意味著對各頂點嘗試賦予顏色。首先用單一顏色，接下來用兩種顏色，再以三種顏色施色等等，直到一合格的著色圖形出現爲止。小心謹愼的做，或可稍快得到解答，但一般相信，目前尚無更有效率的運算法則來取代此類直截了當的嘗試作法。

面對此類尋求最佳解答或須耗費極鉅的計算作業，我們可從三種方法中選出一個。如果圖形很小，我們可以試盡各種可能，得到一最佳解答。但易以較大圖形，不論我們如何嘗試有效的寫作一程式，仍將導致龐大的耗費。另一種嘗試的方法或許對手中的問題得以提供更多的解答資料。亦即，指出圖形的某些特性，因而在尋求最佳解答上，不須試盡各種可能。第三種嘗試方法是將問題略作改變，並尋求一好卻非最佳的解答。在小圖形上，得到一接近於最少顏色的解答，已足堪安慰了，因爲，多數的交叉路口不似圖 1.1 般複雜。很快的找到一近似最佳解答的好方法，這種運算法則稱爲啓發式法則（heuristic）。

對圖形上色的合理啓發式法則如下所述的“渴望（ greedy ）”運算法則。首先，將第一種顏色塗佈於儘可能最多的頂點數，接著，在未著色的頂點中，將第二種顏色著上，期使其頂點數儘可能最多，繼續下去，直到完成。每著上一新的顏色時，我們採用如下的步驟：

1. 選取某些尚未著色的頂點，並賦上顏色。
2. 掃視所有未著色的頂點。對每個未著色頂點，是否有一邊連結於已著上新的顏色的頂點。若無這類的邊，便可賦予目前新的顏色。

這種方法被稱爲“渴望”，乃因其儘可能賦色於頂點上，而不考慮進行此一方法中潛在的先天缺陷。如果，我們不過於“渴望”，而跳過一些我們本可合法著色的頂點，有些時候，我們可以一種顏色塗

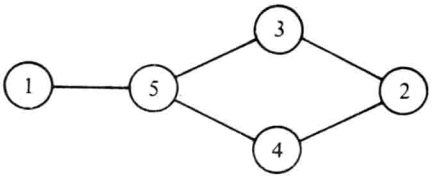


圖 1.4 一種圖形

佈於更多的頂點。例如，考慮如圖 1.4，若已將頂點 1 著上紅色，則只要不先將頂點 2 著色，我們便可將紅色著於頂點 3 及 4 上。如果按數字的順序，依“渴望”運算法則，必將導到頂點 1 及 2 著上紅色。

再回顧圖 1.2的例子，施以“渴望”的方法，假設我們先將 AB 著上藍色。我們可以將 AC，AD 及 BA 著上藍色，因爲該四頂點互相間都沒有邊連結。但我們無法將藍色著於 BC，因爲有一邊聯結了 AB 與 BC。同樣的，我們亦無法將 BD，DA 及 DB 著上藍色，因爲這些頂點都與一個以上已著上藍色的頂點連接著。不過，我們可以將 DC 著上藍色。然後，EA，EB 及 EC 無法著上藍色，而 ED 則可著上藍色。

接下來，開始著第二種顏色，譬如將 BC 著上紅色。BD 可著上紅色，DA 則否，因 BD 及 DA 間存在一邊相連著。同樣的，DB 不得著上紅色，而 DC 已著上藍色，但 EA 可著上紅色。其餘未著色的頂點都已有一邊連結於一紅色的頂點，顯然無法再著上紅色。

尚未著色的頂點分別爲 DA，DB，EB，以及 EC。如果將 DA 著上綠色，則 DB 亦可著上綠色，但是 EB 及 EC 則否。這兩個頂點可著上第四種顏色，譬如黃色。這些顏色擺列於圖 1.5。其中“額外”的路線，乃依據“渴望”推論方式，與已經著色的頂點相容者，就如同其他正式的路線一般。亦即，當號誌允許某一同色路線行進時，額外的行進路線亦可安全進行。

顏 色	路 線	額外的路線
藍色	*AB*, *AC*, *AD*, *BA*, *DC*, *ED*	—
紅色	*BC*, *BD*, *EA*	*BA*, *DC*, *ED*
綠色	*DA*, *DB*	*AD*, *BA*, *DC*, *ED*
黃色	*EB*, *EC*	*BA*, *DC*, *EA*, *ED*

圖 1.5 圖 1.2 的圖形著色表

“渴望”法則並未經常得以最少的顏色來上色。我們可再用運算法則的理論去評估如何獲致最好的解答。在圖形理論中，k-系（k-clique）表一組含有 k 個頂點，其中，任二頂點間，皆聯結了一邊。顯然地，對 k-系著色須使用 k 種顏色，因爲在 k-系中，沒有兩點得用同色來賦予。

在圖 1.2 的圖形中，AC, DA, BD, EB 四個頂點構成了一 4-系。因此，不可能用三或更少的顏色來著色，因而，圖 1.5 的解答，已爲最佳者，因其使用了最少的顏色。就我們的原始題意，即圖 1.1 所示的交叉路口，交通號誌無法以少於四種方式的組合來完成。

故而，考慮以圖 1.5 爲基礎的交通號誌控制系統，每種顏色表示了一種狀況。在一狀況下，相對於表中該顏色所在之列中路線允許通行，而其他路線則禁止通行。這種模式已爲最少狀況者。

虛擬語言及逐級精細化

一旦我們對問題有了適當的數學模式，便可依據此一數學模式建立一運算法則。初期的運算法則通常委婉陳述而不明確，必須再進一步精細成較小，更明確的架構。例如，我們以“挑出未著色的頂點”之類的語句來描述逐步在圖中著色的運算法則。希望這些架構足夠清楚，讀者們得藉此理解我們的構思意圖。然而，將此非正式的運算法則轉換爲程式，我們仍須經過一些程式化的步驟（稱爲“逐級精細化”）以達到程式語意合乎正式語言所合格定義的程度。