一分鐘藝廊

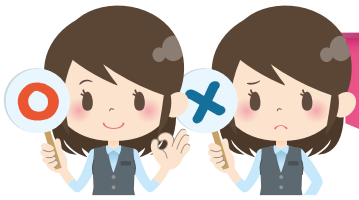
神奈川沖浪裏

Katsushika Hokusai 葛飾 北齋

日本，1760-1849

在十九世紀早期，被稱為柏林或普魯士藍（bero）的顏料變得更加廣泛且價格合理，刺激了印刷設計的蓬勃發展。這幅「神奈川沖浪裏」是日本浮世繪畫家葛飾北齋的著名木版畫，於 1832 年出版，是《富岳三十六景》系列作品之一，與該系列的其它作品一樣，都以富士山為背景。在這幅圖中，描繪的是驚濤巨浪掀卷著神奈川外海（神奈川沖）的漁船，而船工們為了生存努力抗爭的圖像，遠景則是富士山。日本人的閱讀視線習慣從右到左閱讀，波浪的巨大爪子幾乎翻滾到觀眾的臉上，即便是富士山看起來也很脆弱，彷彿即將被無法控制的巨浪所吞沒。這個標誌性的形像已被廣泛用於當代設計，不論是漫畫、廣告、書籍、夾克和唱片封面等等都能見到其蹤跡。這幅作品是北齋最有名的作品，也是世界上最有名的日本美術作品之一。

CH1 演算法



1-0 素養指標

1-1 演算法概念

學習內容	學習表現
1-1.1 演算法與運算思維	能運用「運算思維」解析問題，並設計與描述演算法。
1-1.2 演算法的表示方法	
1-1.3 演算法的效能分析	

已完成：○課文與習題　○自學筆記　　學習成效自評：☆☆☆☆☆

1-2 資料結構

學習內容	學習表現
1-2.1 資料結構－陣列	能舉例說明如何以陣列儲存「樹」與「圖」二種資料結構。
1-2.2 資料結構－堆疊與佇列	
1-2.3 資料結構－樹	
1-2.4 資料結構－圖	

已完成：○課文與習題　○自學筆記　　學習成效自評：☆☆☆☆☆

1-3 搜尋排序演算法

學習內容	學習表現
1-3.1 搜尋演算法	能舉例說明搜尋與排序演算法的原理與過程，並能帶出「遞迴」的概念。
1-3.2 排序演算法	

已完成：○課文與習題　○自學筆記　　學習成效自評：☆☆☆☆☆

1-1 演算法概念

1-1.1 演算法與運算思維

面對每天各式各樣的問題時，我們會針對不同的問題設計相對應的解決方案，其中，能清楚描述步驟或原則的問題解決方案，便稱作演算法（Algorithm）。

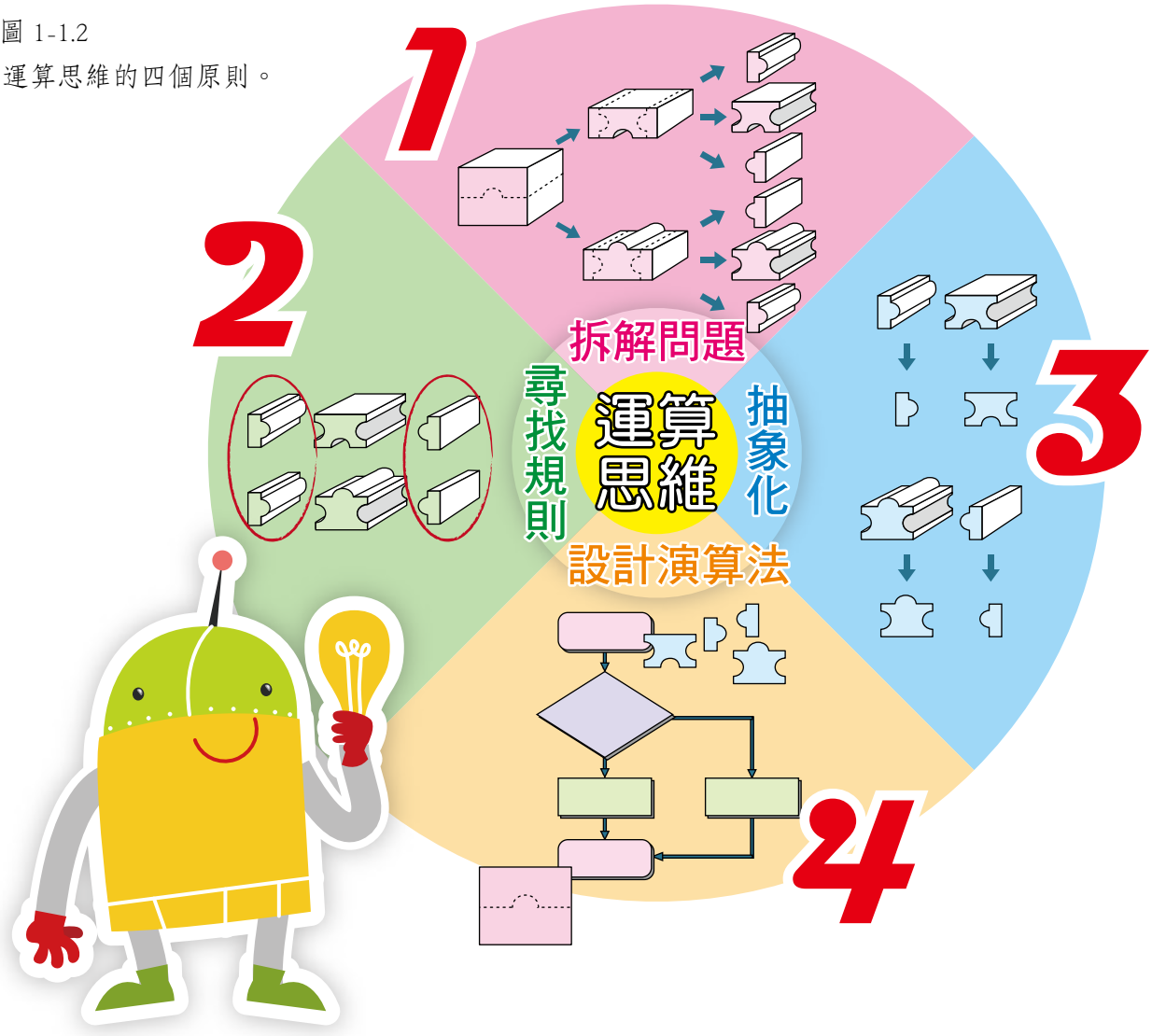
生活中其實充斥著各式各樣解決問題的演算法，只是有時我們並未注意或發現，例如要找出手機通訊錄中的一筆姓名電話時，就用到了搜尋演算法；用 Google 找美食餐廳或產品開箱文時，PageRank 演算法決定了結果網頁的排名次序；當我們使用信用卡在網路上購物交易時，加密演算法則保護卡號等重要資訊不會被盜用外流。

在開始設計演算法之前，最重要的是必須先將問題定義清楚並加以分析，舉例來說，小敏若要替全家安排出遊行程（圖 1-1.1），就要先搞清楚是「開車還是搭車」，是否有「一定想去的景點」，有沒有要「過夜」，「人數」與「預算」是多少等等，若沒有事先將問題定義清楚，小敏就無法決定如何開始安排行程。

圖 1-1.1
沒有什麼比全家出遊更讓人開心了！釐清問題與需求，才能好好安排並享受快樂的行程。

將問題定義清楚後便可以著手開始設計演算法，但如果遇到太過複雜的問題而不知從何下手時該怎麼辦呢？美國卡內基 · 梅隆大學的計算機科學教授周以真（Jeannette M. Wing）在 2006 年提出：以設計電腦運算步驟的思維模式來解決問題，先將問題拆解，而後分析每個子問題，忽略其中不重要的小細節再制訂詳細的步驟，這樣的思維模式稱為運算思維（Computational Thinking），如圖 1-1.2，可歸納成以下四個原則：

圖 1-1.2
運算思維的四個原則。



拆解問題（Decomposition）

若問題太過複雜，可將問題拆解成幾個子問題，直到每個子問題都能被輕易解決為止。例如在準備分組報告時，可將報告的內容分成幾個不同的主題，由組員各自搜集資料、撰寫報告，最後再討論彙整成為一份完整的報告。

尋找規則（Pattern Recognition）

分析問題及子問題中是否存在相似的規則可一併設計解決方案，或是可以將舊有的解決方案應用到某個子問題上。

抽象化（Abstraction）

思考過程中只注重與問題相關的重要細節，忽略不重要的部份，如此一來才不會被模糊焦點。比方說，如圖 1-1.3 的臺北市行政區域圖其實包含了相當豐富的資訊，但對於想搭乘捷運的旅客來說，關鍵重點其實是會經過哪幾站、兩站間需不需要轉乘等，於是捷運路線圖在設計時，便會將地圖上的資訊進行簡化，歸納出共同的呈現規則並只留下最相關的部分，讓旅客可以一目了然找到所需路線，如圖 1-1.4。

圖 1-1.3　臺北市行政區域圖的一部份。

圖 1-1.4　臺北捷運路線圖的一部份。

設計演算法（Algorithmic Thinking）

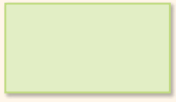
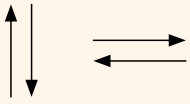
制訂詳細可解決問題的步驟或原則，也就是演算法。例如，家具的組裝說明書會用清楚的步驟教學讓使用者能夠依步驟操作，進而完成組裝，這就是一種組裝家具的演算法。而企業中常說的標準作業程序 SOP（Standard Operating Procedure），也可以說是一種演算法的概念。

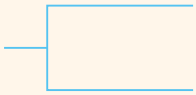
1-1.2 演算法的表示方法

一個演算法通常具有許多步驟或原則，利用適當的描述方式能夠清楚地讓人理解運作過程，常見的表示方法有文字、流程圖及虛擬碼三種。

- ✔ 文字　利用文字直接描述演算法，能夠快速地表達出演算法的流程，例如：電影院的售票員只能將學生票販售給學生，其餘皆販售全票，則售票員執行的演算法可能為：「先確認顧客是不是學生，如果是學生則可售出學生票，不是學生則售出全票」。
- ✔ 流程圖　流程圖（Flowchart）利用特定的幾何圖形符號來描述演算法，藉以說明處理的方法與步驟，不但容易理解及修正處理方法，也利於將演算法轉換為電腦可運行的程式。表 1-1.1 是美國國家標準學會（American National Standards Institue，ANSI）所制訂的流程圖標準圖形中較常用的表示符號與意義。

表1-1.1　流程圖的常用符號。

符號圖示	名稱與說明
開始 結束	開始與結束符號 表示一個流程圖的開始與結束。
	處理符號 表示將執行的動作。
	決策符號 表示一個選擇或一個問題，須以是或否回答。
	重複符號 可設定起始及終止條件，用以重複執行某些動作。
	流程線 表示流程進行的方向，通常流程的方向是由上而下，由左而右。
	輸入／輸出符號 表示從外界輸入資料或把處理結果輸出。
	連接符號 表示連接流程圖的兩部分。

符號圖示	名稱與說明
	預定義過程 表示一組預先定義的動作流程。
	說明符號 表示說明流程的文字

✔ 虛擬碼　虛擬碼（Pseudocode）利用精簡的文字及類似程式碼的敘述（如數學運算符號或邏輯判斷符號等）來描述演算法，卻不像程式碼必須使用固定的程式語法，如圖 1-1.5 售票員執行的演算法可畫成流程圖或寫成虛擬碼。

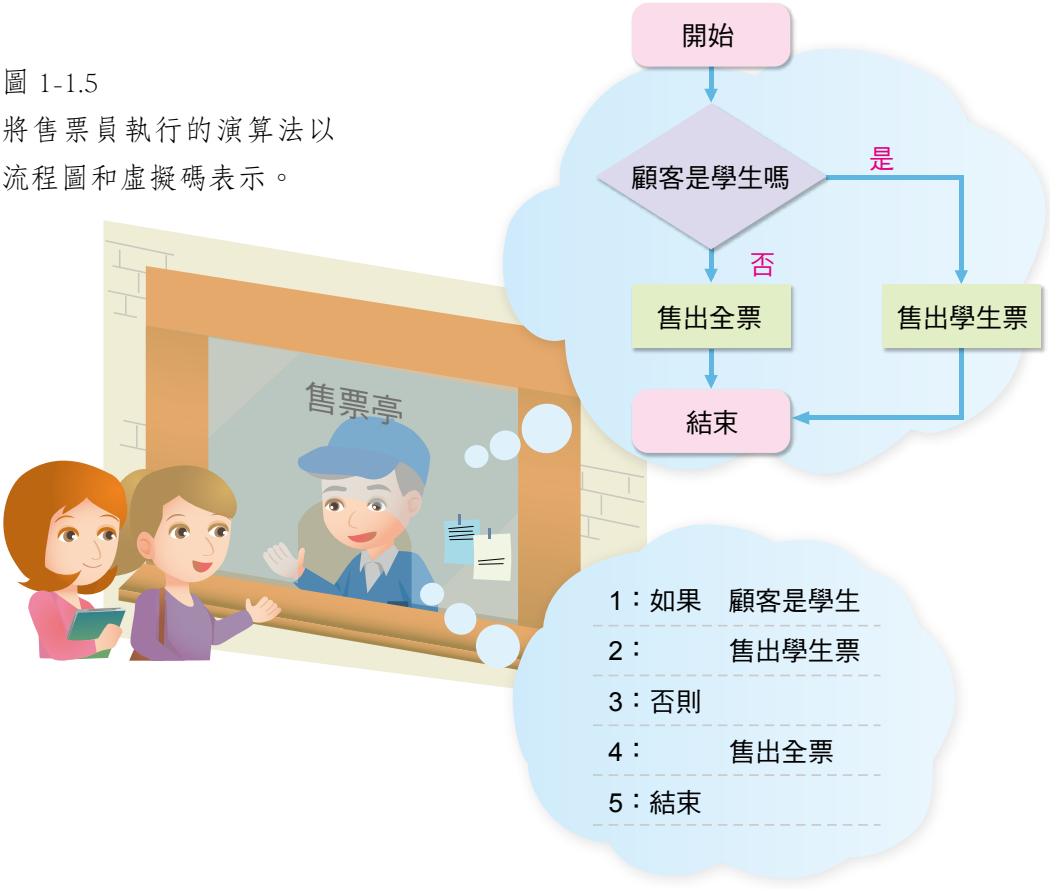


圖 1-1.5
將售票員執行的演算法以流程圖和虛擬碼表示。

立即演練

（　　）1. 在畫流程圖時，如果要表示某個條件判斷（是 / 否），要使用下列哪一個符號？
(A)  　(B)  　(C)  　(D)  。

記憶

討論&表達

＜幫幫新來的工讀生吧 ── 電影售票 SOP＞

在前面的例子中，我們舉了最簡單的情況說明電影售票的演算法（判斷學生票還是全票），實際上當然不可能那麼簡單，可能要先看看還有沒有位置，要不要搭配飲料爆米花等。現在就讓我們來幫幫新來的工讀生吧：設計一個簡單的 SOP，讓任何新手都能知道如何正確完成電影售票流程並達到公司期望。

1）售票過程中，一定要完成的任務是什麼？有哪些希望達成的額外事項？ 理解

2）過程中可能要問顧客哪些問題？發問順序為何？ 分析

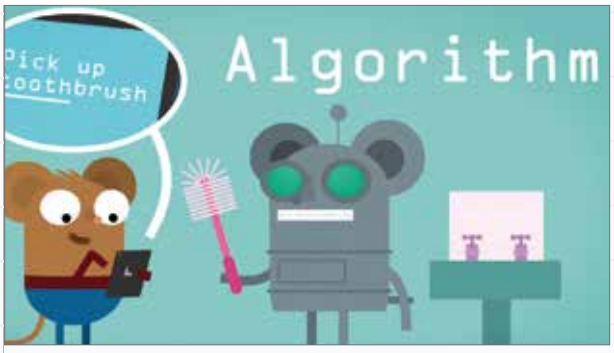
3）顧客回答問題後，如何判斷並做出哪些對應的處理動作？ 分析

4）怎樣才算是完成了一筆交易，並能繼續服務下一個顧客？ 理解

5）你會選擇使用「流程圖」還是「虛擬碼」描述上述過程的演算法，請展示成品並說明原因。 應用 評價

請分成幾組討論，簡單寫下你的觀點與作法，然後上台發表。

自學筆記



http://ln.cfp.com.tw/u/Algorithm
什麼是演算法？（1:07）

http://ln.cfp.com.tw/u/ComputationalThinking
什麼是運算思維？（5:41）

http://ln.cfp.com.tw/u/Algorithm02
人腦也可以執行演算法（4:57）

1-1.3 演算法的效能分析

不同的人針對相同的問題，可能會設計出不同的演算法以解決問題，舉例來說，小麗和小美一起參與猜數字的遊戲，小麗決定從 1 依序猜到 50，小美則打算每次先猜區間裡正中間偏大的整數（例如若中間值為 37.5 就取 38），如圖 1-1.6 所示為兩人各自玩遊戲的過程。

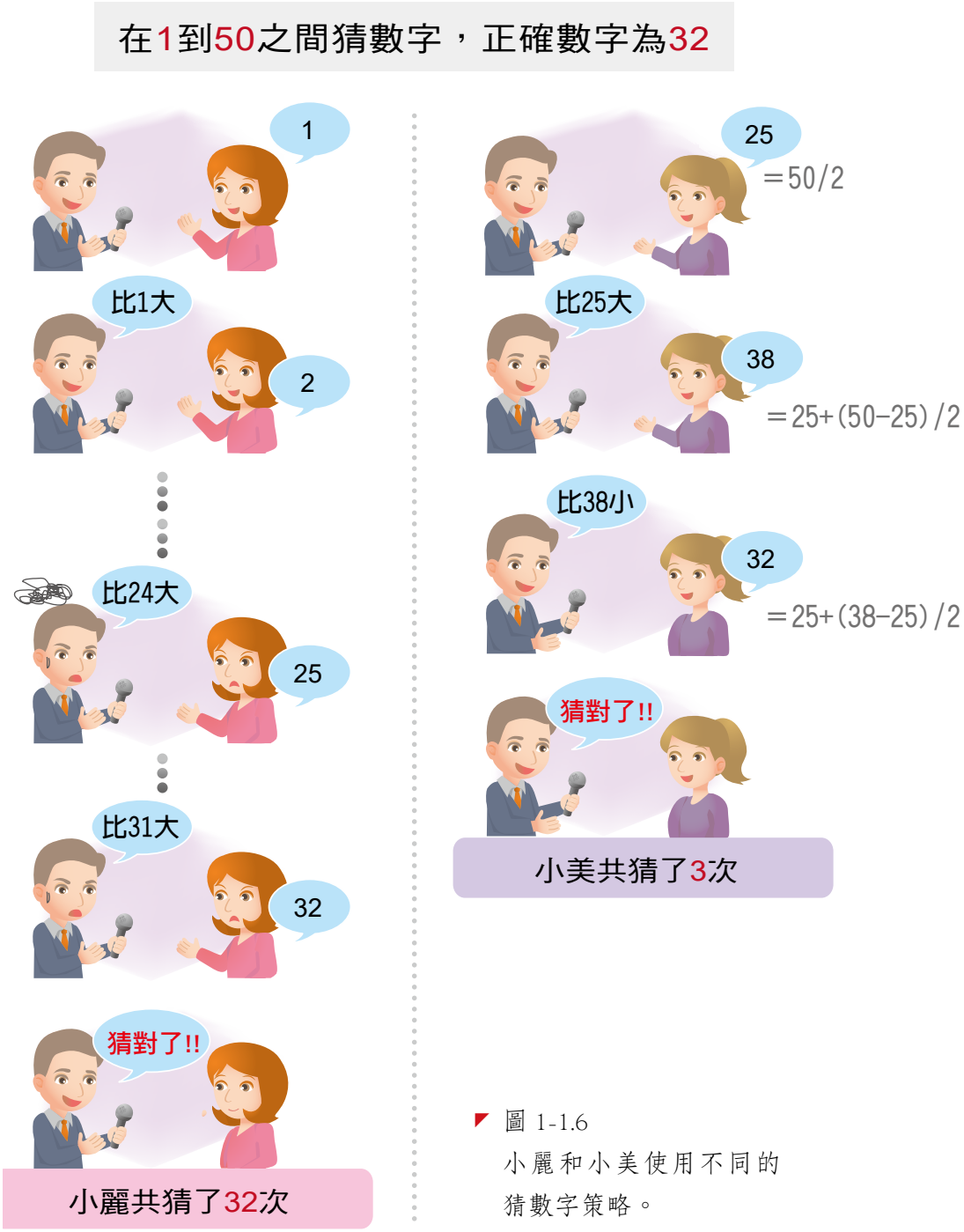


圖 1-1.6
小麗和小美使用不同的猜數字策略。

小麗和小美採用不同的猜數字策略，誰可以比較快猜到正確答案呢？要評斷一個演算法的好壞，最基本的指標就是執行時間和佔用的記憶體空間，然而相同演算法在不同硬體效能的電腦上執行時間會有差異，又由於每個人實作演算法所採用的程式語言、程式設計技巧都不一樣，所以執行時間、記憶體使用量不是一個穩定的評斷標準。因此，一般我們會以演算法執行的步驟數量來做為評斷標準，並習慣用漸進符號：大 O 符號來表述。

在上面的過程中，可觀察到小麗猜的次數比小美多，在最壞的情況下，小麗必須猜 50 次才能猜到正確答案（由 1 開始猜，正確數字為 50）；而小美由於每猜一次就將範圍縮小一半，因此最情況下也只要猜 6 次就能猜到（依序猜 25、38、44、47、49、50）。一般我們會用 O(N) 和 O(logN) 來分別表示小麗和小美演算法的時間複雜度，也就是當數字規模擴展到 N 時，二種演算法在時間複雜度上與 N 之間的關係，例如當數字範圍擴展到 1024 個數字時（也就是 N=1024），則小麗和小美的猜法最多分別是 1024 次與 log1024 = 10 次，如此可以看出小麗和小美的猜法在效能上有明顯差異。

值得注意的是，時間複雜度只是一種相對的概念並非絕對精確的數值，是用來比較用途類似的演算法以利決策，所以實務上會取最壞狀況來做評估，並只取最高次方的那一項，忽略其他的係數，用時間複雜度來比較不同用途的演算法並沒有太大意義。之後我們還會陸續提到相關的例子，這裡只要先有一個大概的概念即可。

大O符號

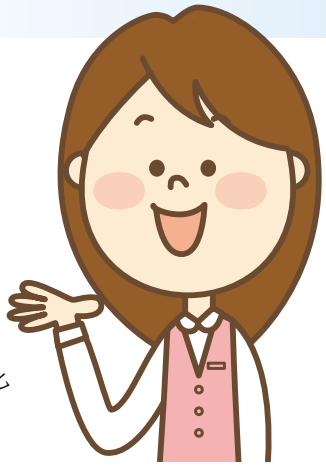
有關大 O 符號更詳細的說明請參閱「自學筆記」中的第一個連結「關於時間複雜度以及幾種漸進符號的介紹」。

logN

還記得什麼是對數嗎？這裡的 logN 指的是以 2 為基底的對數，也就是若 $2^K = N$，則 logN = K。

小麗一定輸嗎？

遊戲例子中如果正確數字是 1 或 2，其實小麗的猜法是明顯獲勝的，想想看，這說明了什麼？



舉手！舉手！多多發表自己的觀點與看法，藉由討論可以讓同學彼此都更加成長進步喔！

討論&表達

＜演算法小遊戲：摺紙還是畫格子？＞

演算法就是解決問題的方法。假設我們要將一張正方形的紙分割成 16 個大小一樣的格子，小麗和小美分別採用 A、B 二種方法實作，如圖 1-1.7，請試著回答後面的問題。

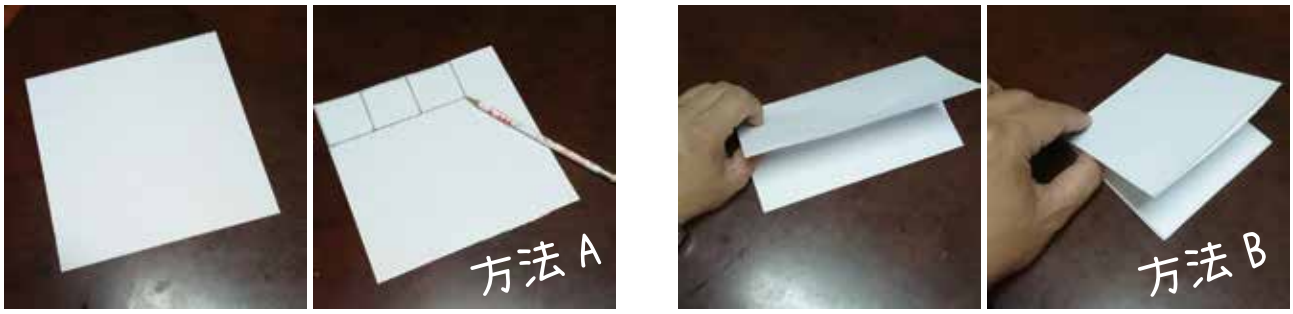


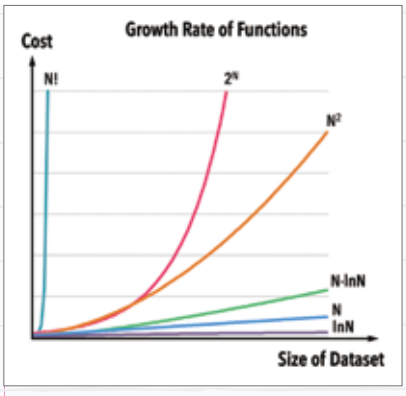
圖 1-1.7　小麗的方法 A：每次在紙上畫出一個方格。小美的方法 B：將紙張對摺再對摺 …。

1）小麗的方法 A 和小美的方法 B 各需要幾次步驟才能完成？　理解

2）你從這二種方法中觀察到了什麼？各有何優缺點？　評價

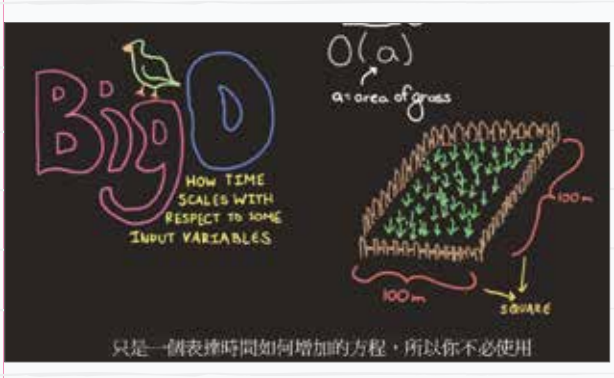
3）還有其他方法也可以完成指定任務嗎？　創造

請分成幾組討論，簡單寫下你的觀點與作法，然後上台發表。



http://ln.cfp.com.tw/u/Complexity
關於時間複雜度以及幾種漸進符號的介紹。

http://ln.cfp.com.tw/u/Intro2Algorithms
漫談演算法與效率（11:43）



http://ln.cfp.com.tw/u/BigO
大 O 符號（8:37）

1-2 資料結構

在實作電腦演算法時，多半也會伴隨資料的輸入、儲存與輸出，而在程式中用以組織與儲存資料的方法則稱為資料結構(Data Structure)。雖然程式中沒有使用資料結構也可以執行，但採用適當的資料結構可以使存取資料、修改資料等操作更有效率，因此，在設計演算法的過程中，也必須同時考量選用何種資料結構，以下簡單列舉幾種常見的資料結構，包括陣列、堆疊、佇列、樹、圖等。

1-2.1 資料結構 — 陣列

陣列（Array）是最常見、使用率最高的資料結構，它是由多個元素組成的集合，並透過指定索引值（Index Value）來存取陣列中的特定元素。我們可以把陣列想像成一個放很多張學生成績單大抽屜，例如圖 1-2.1 所示，這個抽屜分割成很多格，每一格都按學生的學號順序排序（也就是索引值），例如要找大雄（學號 003）的成績單，只要數一下在第幾格就可以找到。

小夫 001
靜香 002
大雄 003
胖虎 004

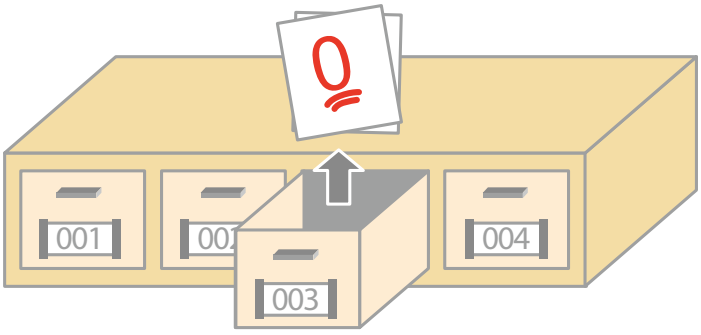
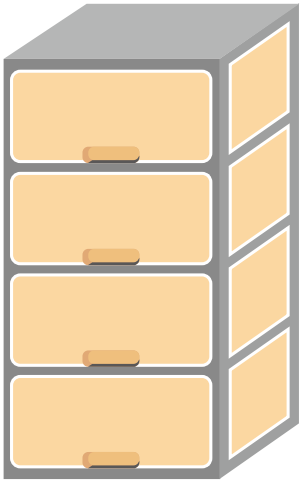


圖 1-2.1
陣列就像是分成很多格的抽屜，要找東西則是透過索引編號去搜尋。

一般常用的陣列有一維陣列和二維陣列兩種類型，如圖 1-2.2，一維陣列是線性的空間，就好比一個具有多層抽屜的收納櫃；而二維陣列的概念就像是由行與列所組成的表格，例如生活中的功課表、成績單等的組成結構，都可視為是由行、列資料組成的二維陣列。

一維陣列　　二維陣列

收納櫃　　功課表　　成績單



高一望 課程表

導師：郭淑金老師

星期 / 時段	一	二	三	四	五
9:00 ~ 9:50	國文	英文	歷史	化學	數學
10:00 ~ 10:50	英文	數學	數學	國文	地理
11:00 ~ 11:50	歷史	國文	國文	英文	電腦
13:40 ~ 14:30	生物	體育	英文	數學	生物
14:40 ~ 15:30	化學	音樂	物理	體育	國文
15:40 ~ 16:30	數學	物理	美術	地理	英文

第一學期成績單

座號	姓名	國文	英文	數學	總分	平	
1	楊小婕	85	90	65	240	80	5
2	李凱西	87	95	70	252	84	2
3	江小倩	80	75	72	227	75.67	8
4	劉阿華	85	90	85	260	86.67	1
5	王小英	80	70	66	216	72	10
6	張小謙	80	75	71	226	75.33	9
7	張　宇	80	80	49	209	69.67	12
8	程阿環	81	80	85	246	82	4
9	胡小雯	75	83	73	231	77	7
10	范特D	88	80	84	252	84	2
11	林杯麵	78	81	79	238	79.33	6
12	張阿中	75	80	60	215	71.67	11

圖 1-2.2
收納櫃和成績單的組成概念，就像是一維陣列和二維陣列的例子。

1-2.2 資料結構 — 堆疊與佇列

堆疊（Stack）是一個串列，它的兩端分別稱為頂端（Top）和底端（Bottom），資料的推入（Push）和彈出（Pop）都在頂端執行，採用後進先出（Last In / First Out，LIFO）的處理方式，也就是最後進來的資料優先處理。堆疊的例子在日常生活中也是隨處可見，像是疊盤子、布丁筒上的光碟片都是堆疊概念的例子，當我們從最上層拿走一片光碟片或一個盤子，這樣的動作就稱為彈出，而放入一片光碟片或一個盤子在最上層的動作就稱為推入（圖 1-2.3）。而在電腦程式中，像是我們操作 Word 文書軟體時的「復原上一步」動作，或是玩棋類遊戲時的「悔棋」動作，也都是堆疊概念的應用實例。

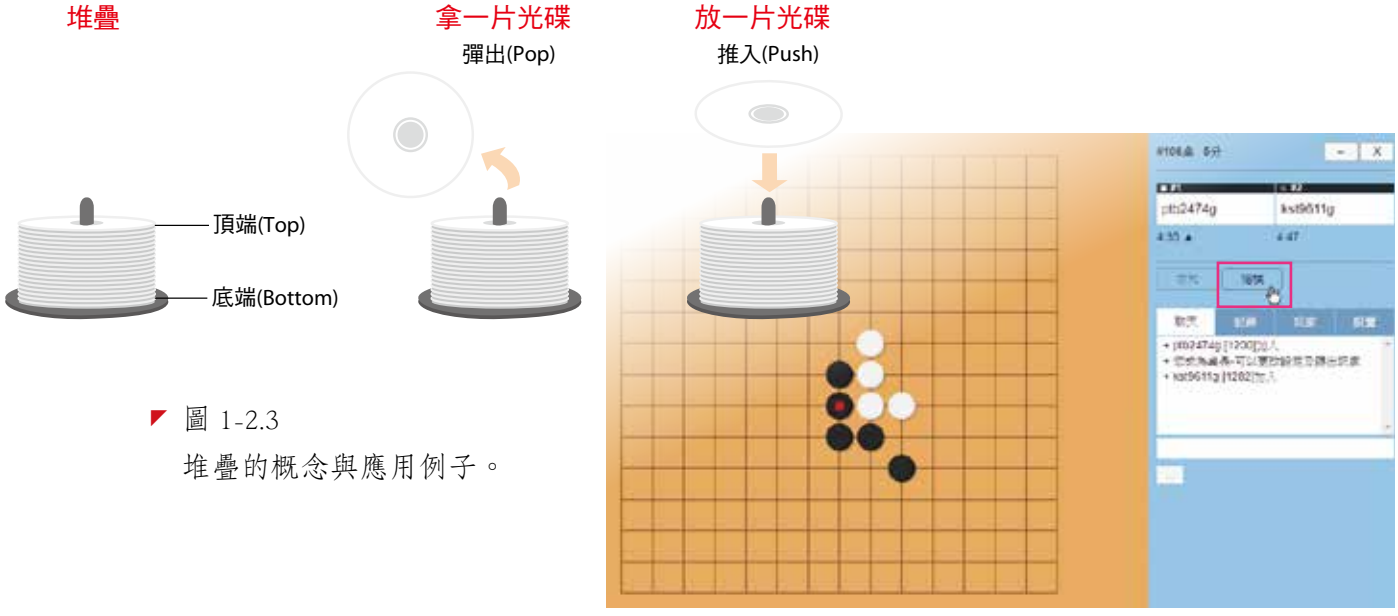


圖 1-2.3
堆疊的概念與應用例子。

佇列（Queue）和堆疊的性質相近，只是佇列加入資料的方式，是由末端（Tail 或 Rear）加入，移除資料時則由前端（Head 或 Front）開始，所以最先進來的資料會最優先處理，屬於先進先出（First In / First Out，FIFO）的處理方式。生活中排隊買票（圖 1-2.4）的例子就是佇列的概念，後到的人必須接在隊伍的最末端，等隊伍最前面的人處理完畢才能往前進。而在電腦程式中，當我們使用印表機列印一系列文件時，印表機會依序處理列印佇列中的文件列印工作，這就是佇列概念的應用實例。

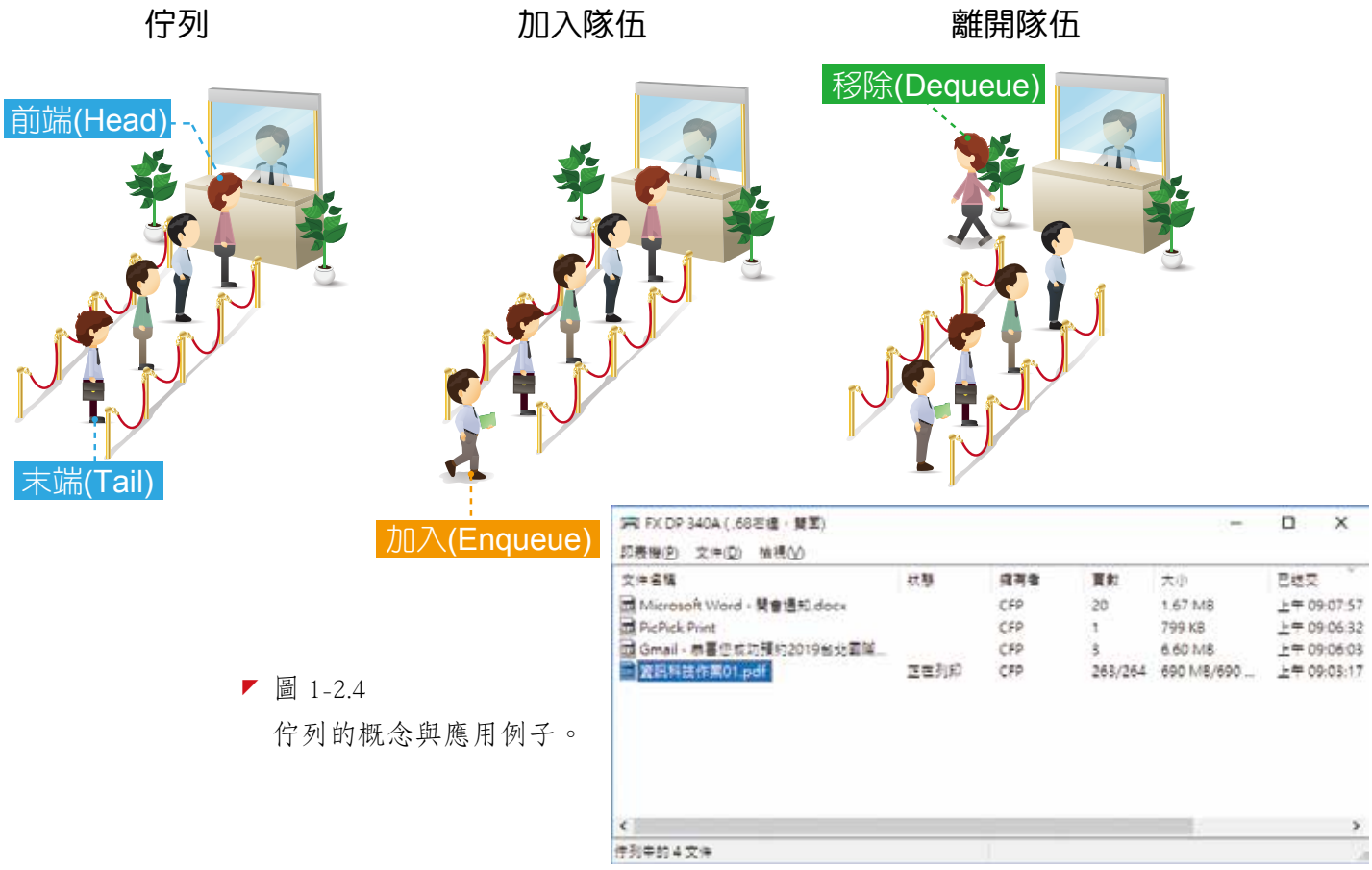


圖 1-2.4
佇列的概念與應用例子。

1-2.3 資料結構 — 樹

樹狀結構（Tree）是一種階層式的資料表示方式，如圖 1-2.5，樹狀結構的最上層是由根（Root）為起點，向下可分成數個節點（Node），節點和上層節點之間連起來的線條稱為分支（Branch），其中任一節點如 C 向下衍生出的節點如 H、I 稱為 C 的子節點（Child），而節點 C 則為 H、I 的父節點（Parent），而位於最下層沒有分支出子節點的節點稱為葉節點（Leaf）。

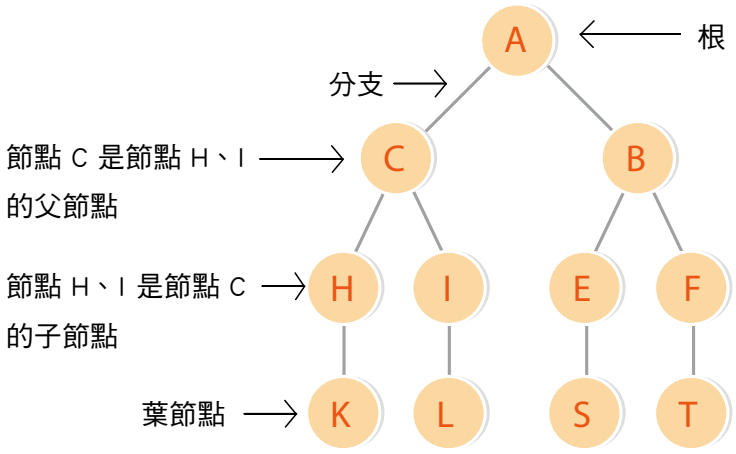
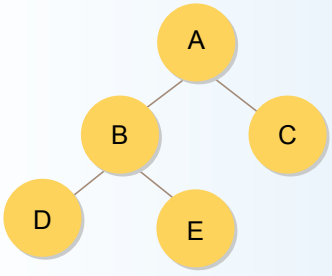


圖 1-2.5　樹狀結構示意圖。

二元樹

在「樹」的資料結構中，如果限制每個節點最多只能有兩個子節點，便稱作二元樹。



樹狀結構適合用來表現具有次序性、階層性、從屬性的資料，因此像是族譜、組織圖、電腦的檔案目錄結構，都很適合用樹狀結構來表示。舉例來說，如圖 1-2.6 是某廣告公司的人事結構圖，呈現了該公司各部門成員間的從屬關係，例如由圖可知企劃部經理為總經理的直屬部下且為企劃部成員的直屬主管。

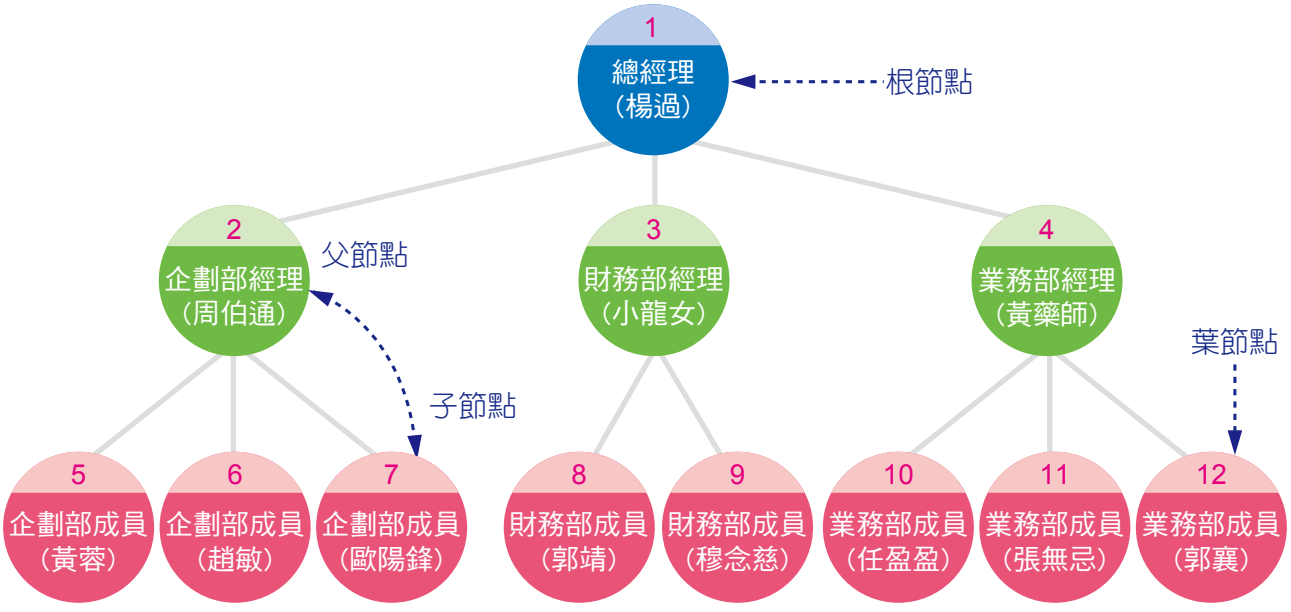


圖 1-2.6　某廣告公司的人事結構圖（圖中的數字代表索引值）。

因為電腦內部沒有直接儲存樹的方法，所以在實作上通常會利用陣列來儲存與實作，例如圖 1-2.6 的人事資料可利用兩個一維陣列加以儲存，如圖 1-2.7，其中一個陣列用來儲存人事資料，另一個陣列則用以儲存各節點的父節點資料索引值，如此即可藉以重建出原始的樹。

註:表示沒有父節點

圖 1-2.7
利用兩個一維陣列儲存圖 1-2.5 的樹狀架構，舉例來說，周伯通和黃蓉為父子節點關係，因此在陣列父節點 [5] 中儲存的數值為 2，代表黃蓉的父節點是陣列人事資料 [2] 中的周伯通。

而透過逐一解析這二個陣列中的從屬關係，我們便能重建出如圖 1-2.5 的樹狀架構。

1-2.4 資料結構 — 圖

除了陣列與樹，圖（Graph）也是生活中常見的一種資料結構，以下便以小麗安排暑假全家出遊的行程做為範例，講解圖的基本定義與概念。

住在臺北的小麗一家人，決定在暑假到離島蘭嶼渡假。為了安排行程，小麗將地圖資訊簡化為圖 1-2.8，只保留與行程有關的部份，例如從本島前往蘭嶼時，只能從臺東或墾丁轉搭交通船前往。圖中，圓形為頂點（Vertex）代表地點、線段為邊（Edge）表示兩地之間有往返的交通方式，像這樣的圖則稱為無向圖（Undirected Graph）。

小麗根據圖 1-2.8 的資訊安排臺北到蘭嶼的路線時，發現有幾種不同的安排方式，例如：< 臺北，臺東，蘭嶼 > 或 < 臺北，墾丁，臺東，蘭嶼 > 是其中的二條路線，這些路線在圖的資料結構中稱為路徑（Path），一條路徑具有起點及終點，且不包含重複的頂點。

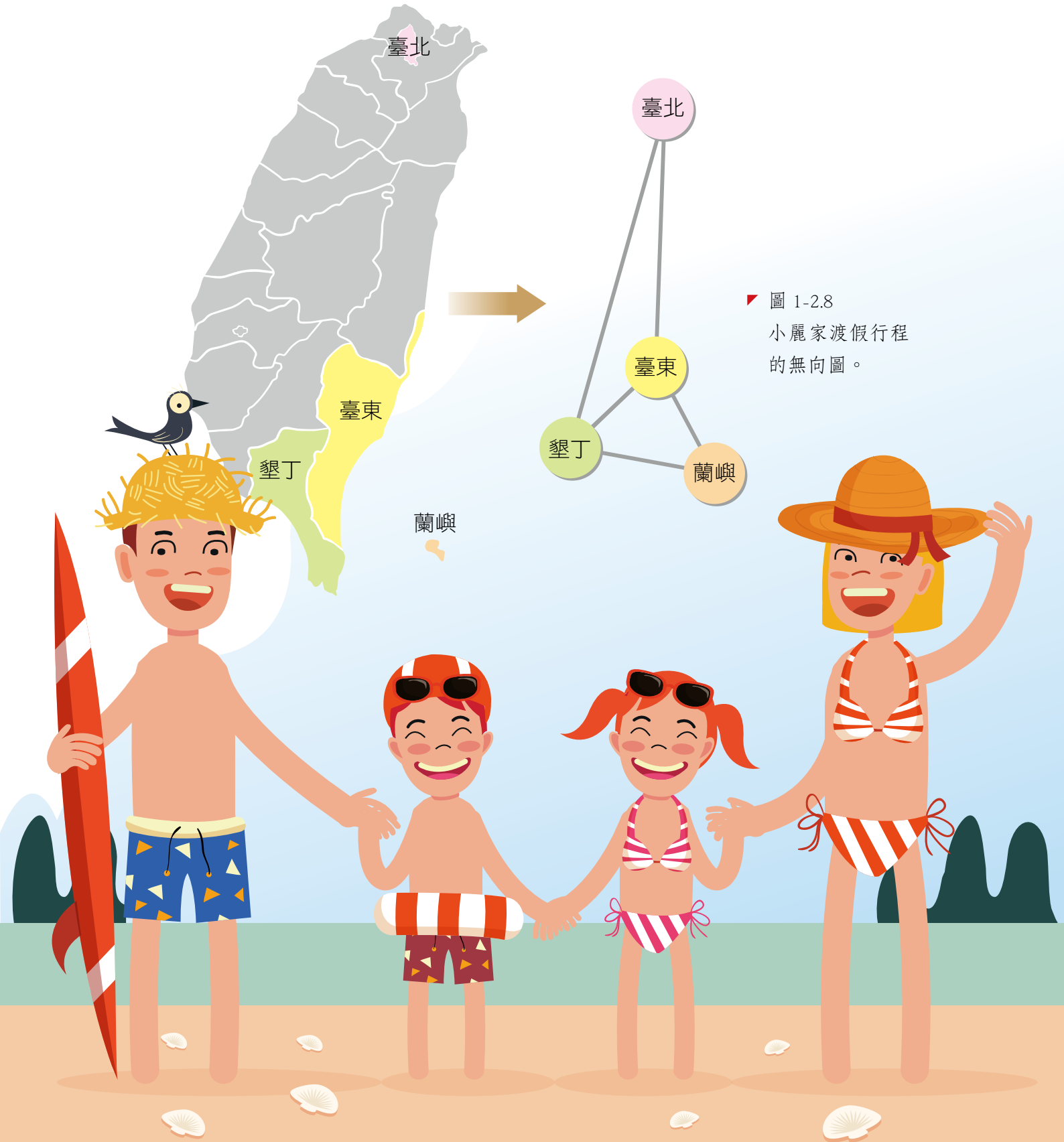


圖 1-2.8
小麗家渡假行程的無向圖。

雖然臺北到蘭嶼的路徑有好幾個選擇，然而小麗在規劃時，希望能加上交通時間的考量，以便能找出最省時的路線；因此，若在圖 1-2.8 的每條邊加上頂點間所需花費的交通時間，如圖 1-2.9，便稱為加權圖（Weighted Graph），其中，每條邊上的交通時間稱為權重（Weight）。

權重可以是頂點之間的交通時間，也可以是距離或是交通花費等等不同的資訊，配合不同的需求選擇不同的權重，才能夠規劃出符合需求的路徑。例如，加上交通時間的權重後，小麗將幾條可能的路徑花費的時間整理成表 1-2.2，比較後發現第 2 條路徑 < 臺北, 墾丁, 蘭嶼 > 所花費的時間最少。

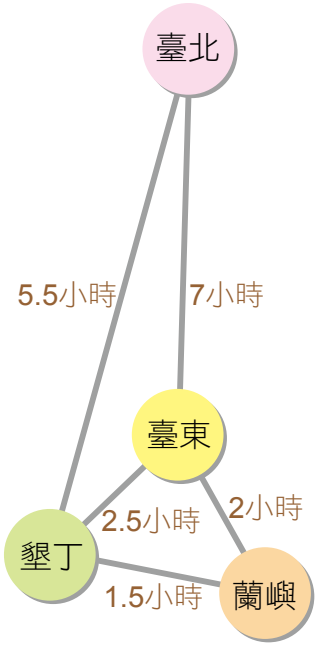


圖 1-2.9
渡假行程的加權圖。

表1-2.2　小麗規劃行程的可能路徑比較表。

	路　徑	花費時間
1	<臺北,臺東,蘭嶼>	7+2=9
2	<臺北,墾丁,蘭嶼>	5.5+1.5=7
3	<臺北,臺東,墾丁,蘭嶼>	7+2.5+1.5=11
4	<臺北,墾丁,臺東,蘭嶼>	5.5+2.5+2=10

如同樹狀結構無法直接儲存在電腦內部，「圖」的資料結構也無法直接儲存在電腦內，圖 1-2.10 則利用一個一維陣列儲存圖 1-2.8 中每個頂點的資料，再利用一個二維陣列儲存頂點間的相對關係（以 0 和 1 代表邊的存在與否）。

同理，儲存加權圖時，將二維陣列中儲存的頂點間的關係改以權重表達，0 表示兩頂點間不存在邊，如此一來，圖 1-2.9 的加權圖資訊就能儲存如圖 1-2.11。

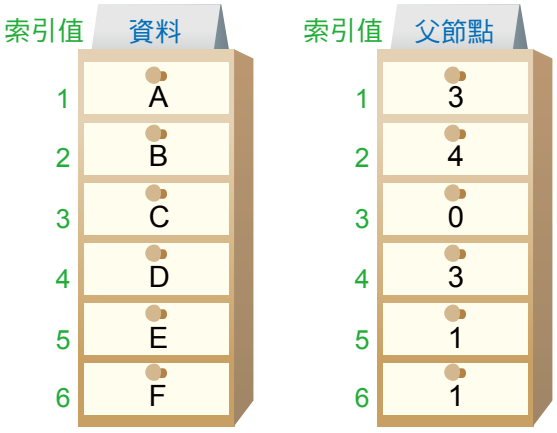
圖 1-2.10　用二個陣列儲存圖 1-2.8 中的無向圖。

圖 1-2.11　用二個陣列儲存圖 1-2.9 中的加權圖。

演練&實作

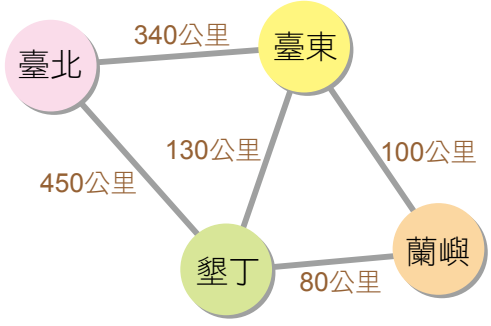
＜資料結構練習＞

1. 試利用圖(一)中的兩個陣列資訊重建出樹狀結構，並找出根節點及資料 D 的父節點及子節點為何。 應用



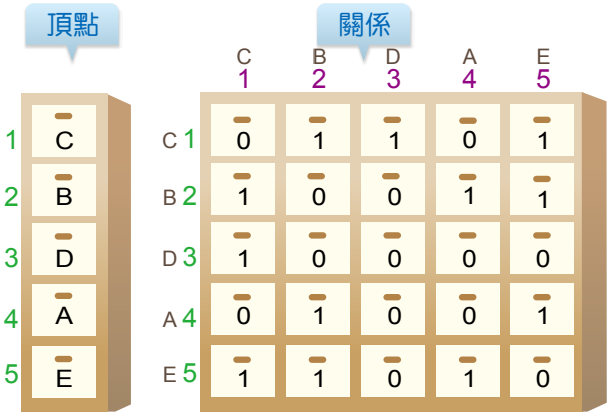
▲圖(一)

2. 請利用圖(二)幫小麗設計一條從臺北到蘭嶼路程距離最短的路徑。 應用



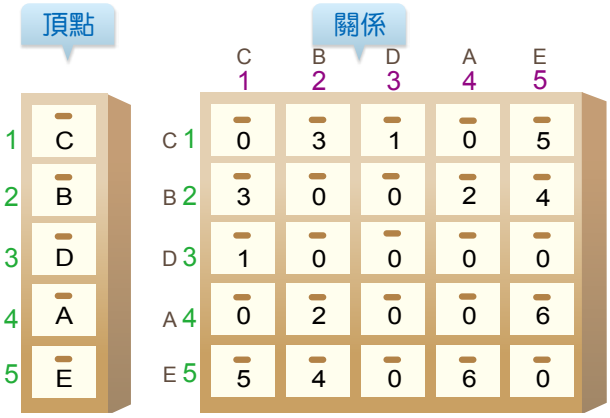
▲圖(二)

3. 試利用圖(三)中兩個陣列的資訊重建出「無向圖」的資料結構。 應用



圖(三)

4. 試利用圖(四)中兩個陣列的資訊重建出「加權圖」的資料結構。 應用



圖(四)

1-3 搜尋排序演算法

1-3.1 搜尋演算法

搜尋演算法是程式設計中使用率很高的演算法之一，以下則利用撲克牌來演示「循序搜尋」與「二元搜尋」二種不同的搜尋策略，幫助同學了解其運作過程與原理。

循序搜尋演算法

如圖 1-3.1，現在有 5 張撲克牌，由左至右一張一張搜尋黑桃 2。因為撲克牌是亂序排列的，所以看了 5 張牌才找到黑桃 2；但若是由右至左搜尋的話，只要看 1 張牌就能找到黑桃 2。

圖 1-3.1
由左至右或由右至左搜尋黑桃 2。

這樣依序搜尋的方法稱為循序搜尋演算法（Sequential Search）。另一種情況是，如果要在這 5 張紙牌中搜尋黑桃 5，雖然黑桃 5 不在搜尋範圍內，但由於事先並不知情，所以還是得在看了 5 張牌之後才能發現沒有黑桃 5。

換句話說，利用循序搜尋演算法在 N 張撲克牌中搜尋時，最多要看 N 張才能確定搜尋範圍內有沒有特定的目標撲克牌，而最少只需要看 1 張。

二元搜尋演算法

要使用二元搜尋演算法（Binary Search）的前提是，撲克牌必須已經從小到大（或從大到小）排列好，接著每次都將搜尋目標與中間的撲克牌比大小，如果是搜尋目標就結束搜尋，否則會依據兩者的大小關係決定下次要搜尋剩餘撲克牌的前半部或後半部，也就是說，每經過一次搜尋動作，要搜尋的範圍紙牌數量就會減少一半（是不是跟 1-1.3 小美的猜數字策略很像呢？）。

圖 1-3.2 則演示如何利用二元搜尋演算法在 1、2、6、7、10 五張紙牌中搜尋紙牌 2 的過程，觀察後可發現搜尋 2 要經過 3 次的比較過程；而若使用同樣的演算法搜尋紙牌 6 則只需要比較 1 次。

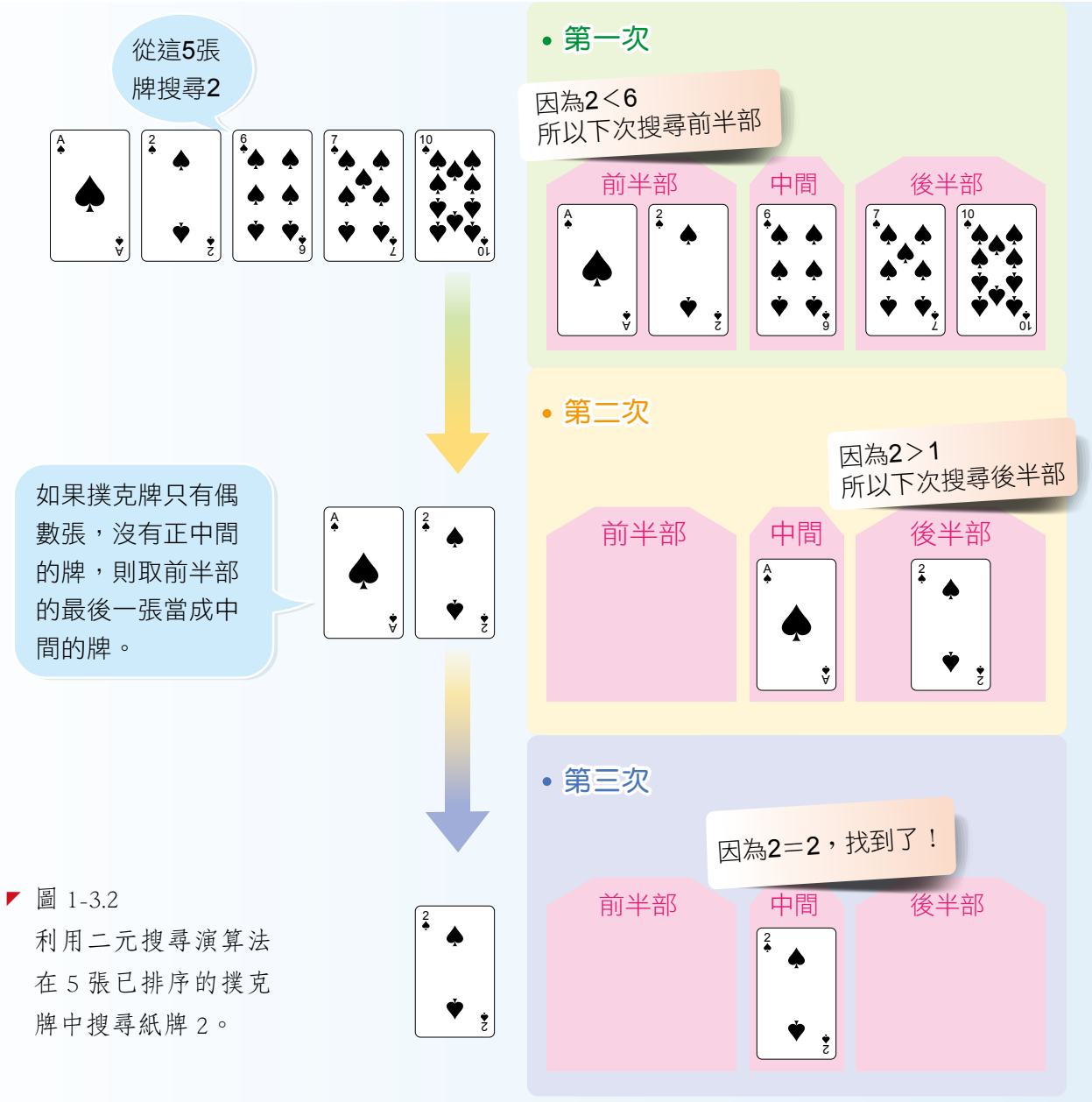


圖 1-3.2
利用二元搜尋演算法在 5 張已排序的撲克牌中搜尋紙牌 2。

二元搜尋演算法只能在已排序好的資料中搜尋，其利用拆解問題的概念將搜尋範圍不斷縮小，每次都會將搜尋的範圍減少一半，所以如果是在 N 張牌中搜尋，最多需要比較 logN 次就能搜尋到特定的牌，最少則需比較 1 次。

遞　迴

想拆開一個俄羅斯套娃時，可以打開最大的娃娃 A、拿出其中次大的娃娃 B、關好娃娃 A，再打開娃娃 B、拿出第三大的娃娃 C、關好娃娃 B，如果重複操作這三個動作（打開、拿出其中的娃娃、關好）直到娃娃不能再被打開為止，排成一列如圖 1-3.3。

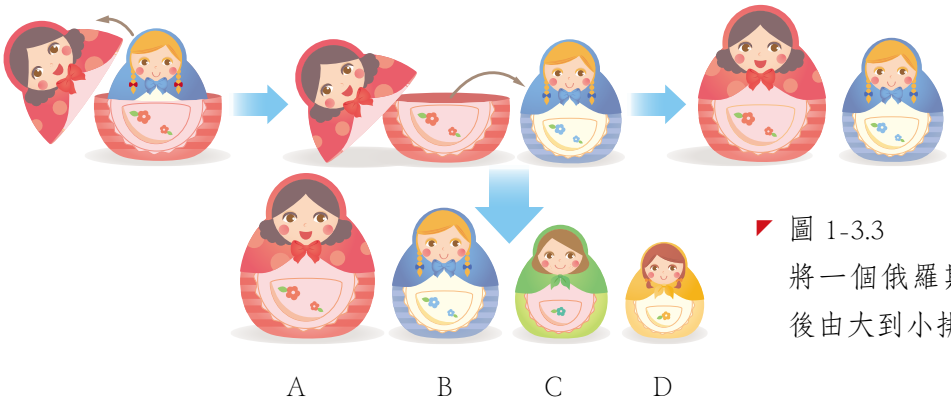


圖 1-3.3
將一個俄羅斯套娃拆開後由大到小排列。

因為都是操作同樣的步驟，所以可以把這些步驟包裝成一個函式，再透過重複呼叫函式的方法完成目標，如圖 1-3.4。像這樣函式中又呼叫自己的方式，就是一種遞迴結構；而二元搜尋演算法中的重複動作，也可以利用遞迴結構完成。

```
1：函式　開啟娃娃（）
2：　　打開娃娃
3：　　拿出娃娃內部的小娃娃
4：　　關好娃娃
5：　　如果　小娃娃可以開啟
6：　　　　開啟娃娃（）
7：　　否則
8：　　　　停止
```

圖 1-3.4
開啟俄羅斯套娃的虛擬碼，其中應用了遞迴結構。

自然界中也存在各種遞迴結構的例子，例如活化石鸚鵡螺具有左右對稱的螺旋形貝殼，其螺線每一圈的直徑與下一圈的比值均固定，大約是黃金比例 1.618，因此，若遞迴式地每次畫四分之一圓，且依此比例縮小其直徑，就可以得到與鸚鵡螺剖面圖螺線相似的圖形，如圖 1-3.5。

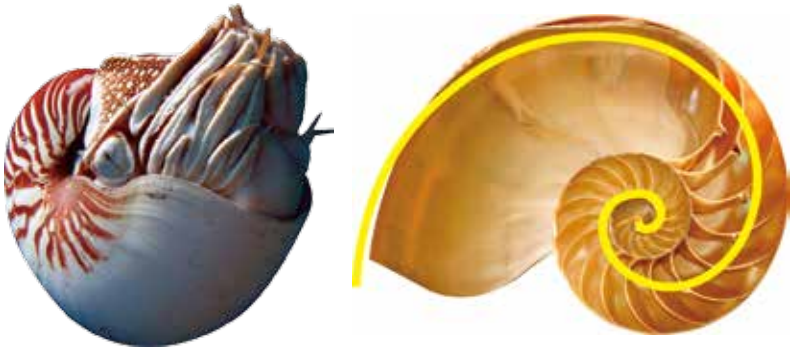


圖 1-3.5　鸚鵡螺及其剖面圖與黃金比例的比較圖。

除了鸚鵡螺具有自然的遞迴結構之外，植物中的羅馬花椰菜也具有遞迴結構，其外觀中的每一個小錐形都是大錐形的相似形，而像這樣的結構又稱為碎形（Fractal）。

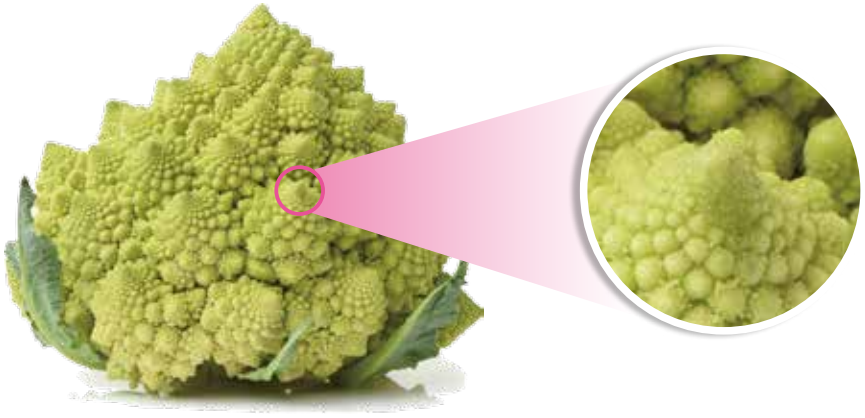


圖 1-3.6　羅馬花椰菜也是自然界中的一種遞迴例子。

數學中有名的費氏數列也是遞迴的例子，其定義為在一串數字中，每一項都是前兩項的和，也就是 0、1、1、2、3、5、8、13、21、34……，若以數學式表示如下：

$$F(n) = F(n-1) + F(n-2),\ F(0) = 1,\ F(1) = 1$$

有趣的是，當 n 非常大時，費氏數列 F(n) / F(n-1) 的值也剛好趨近黃金比例。

演練&實作

＜搜尋演算法練習＞

1. 在這一小節中，我們介紹了「二元搜尋演算法」，也提到其實該演算法可利用「遞迴結構」來實作，請依此條件寫出其虛擬碼。 應用

【二元搜尋法的虛擬碼 - 使用遞迴結構】

1：

2：

3：

4：

5：

真的想不出來的話，就幾個同學討論一下吧！

自學筆記

http://ln.cfp.com.tw/u/RussianDoll

蘇聯時代流行玩具 俄羅斯娃娃風光一時（8:16）

1-3.2 排序演算法

排序演算法也是程式設計中使用率很高的演算法，例如前面介紹過的二元搜尋演算法，就必須是已排序的資料才能使用，本小節就以 5 張撲克牌為例，演示「插入排序」、「氣泡排序」與「合併排序」三種不同的排序策略，幫助同學了解其運作過程與原理。

插入排序演算法

這個演算法是模仿人類整理手牌的方式，由未排序的紙牌堆中由左至右一次拿一張牌，然後把拿到的牌依大小插入已排序的撲克牌中，如圖 1-3.7 所示。

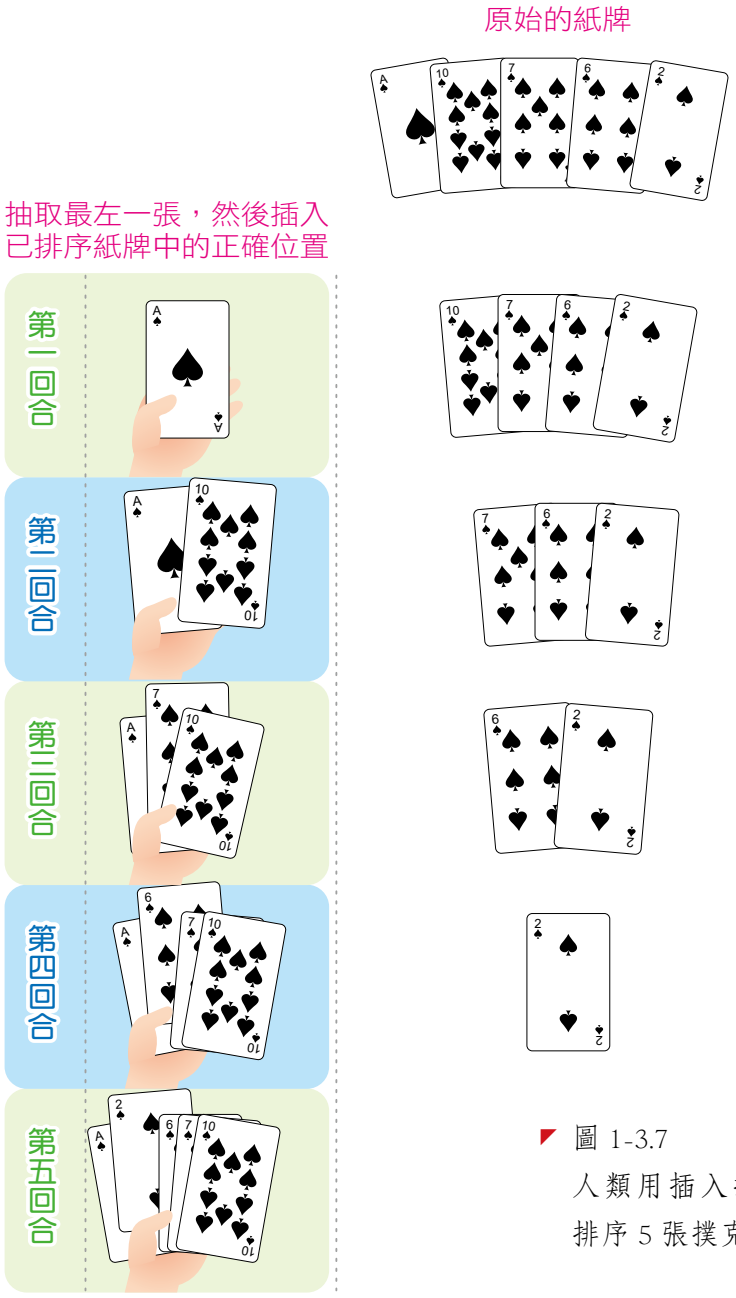


圖 1-3.7
人類用插入排序演算法排序 5 張撲克牌的過程。

人類要將拿到的牌插入已排序的紙牌時，可以直接判斷這張牌的正確位置，但若要設計能讓電腦運作的演算法，則需考慮到電腦一次只能比較並交換兩張撲克牌的位置，因此需要經過多次的比較才能將牌交換至正確位置，圖 1-3.8 演示了電腦在第四回合時實際上如何將紙牌 6 插入撲克牌 1、7、10 中。

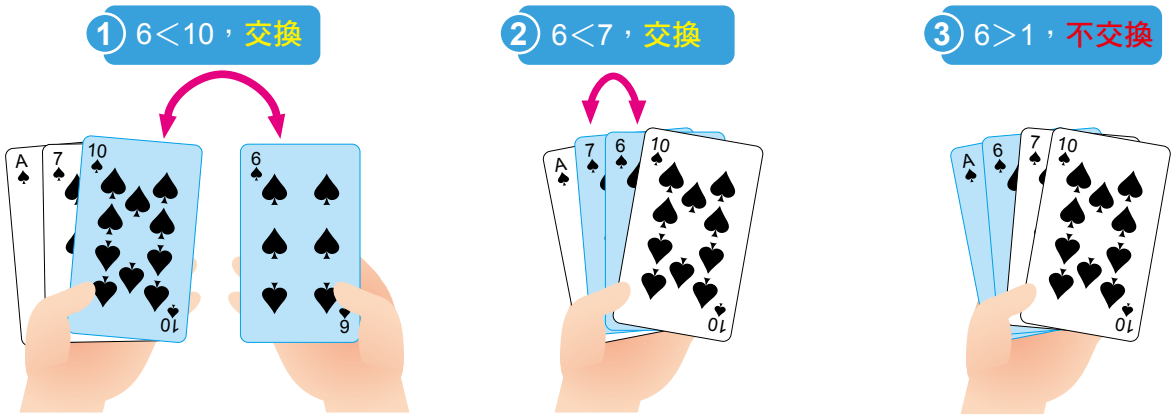


圖 1-3.8　第四回合時，將 6 插入 1、7、10 中的過程。

從圖 1-3.8 中可以發現電腦是往前一張一張的比較大小，直到拿到的牌在正確的順序時才停止交換。繼續分析可以發現，第四回合裡，電腦插入撲克牌的過程中比較了 3 次，若將五個回合中電腦需要使用的比較次數整理成表 1-3.2，則可以得到這個範例中電腦需要比較 10 次才能將 5 張撲克牌排序完成。

表1-3.2　五個回合中電腦所使用的比較次數。

	第一回合	第二回合	第三回合	第四回合	第五回合
比較次數	0	1	2	3	4

因為每次會將拿到的牌「插入」已排序的紙牌中，所以這個演算法被稱為插入排序演算法（Insertion Sort）。如果讓電腦利用插入排序演算法將 N 張亂序的牌由小到大排列，則電腦最多需要比較 $0+1+2+\cdots+(N-1) = N(N-1)/2$ 次才能完成排序，也就是時間複雜度為 $O(N^2)$，當然，這是最壞的情況，不過即使是最佳情況下（也就是紙牌原本就已經由小到大排序完成），插入排序仍需比較 $0+1+1+\cdots+1 = N-1$ 次才能完成排序的動作。

氣泡排序演算法

接下來我們要介紹另一種排序方法。由於電腦一次只能比較 2 張撲克牌，為了不要重複比較，可以由左至右檢查相鄰的兩張撲克牌，如果是右邊的撲克牌號碼較小時交換兩張牌的位置，反之則不換。圖 1-3.9 為第一回合的操作過程。

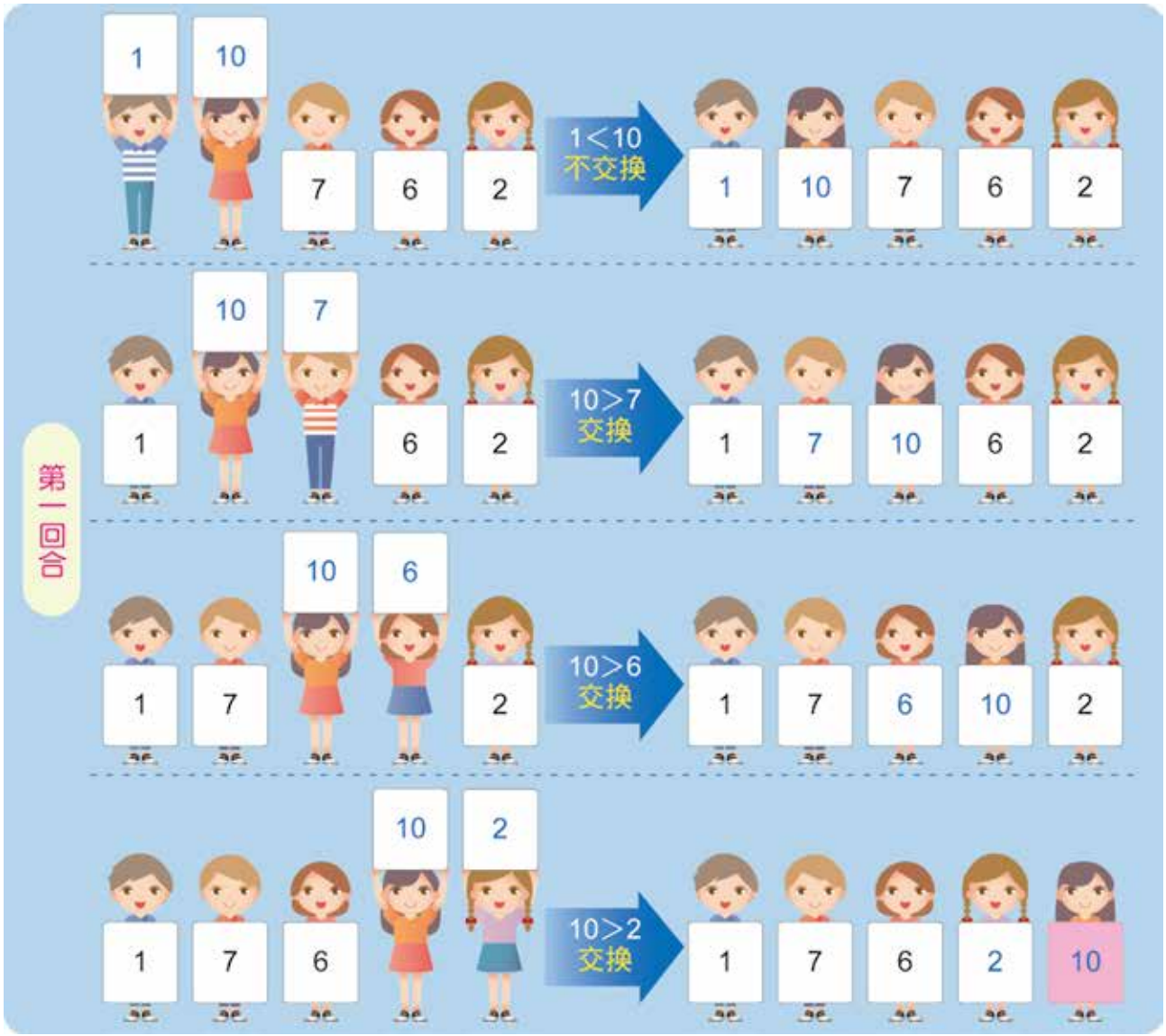


圖 1-3.9　用氣泡排序演算法排序 5 張撲克牌的第一回合。

觀察後可以發現最大的號碼10被交換到最右邊，已經是10號撲克牌的正確的位置。所以下一回合時，一樣由左至右檢查相鄰的兩張撲克牌，但不會再跟10號撲克牌比較大小，如圖1-3.10。

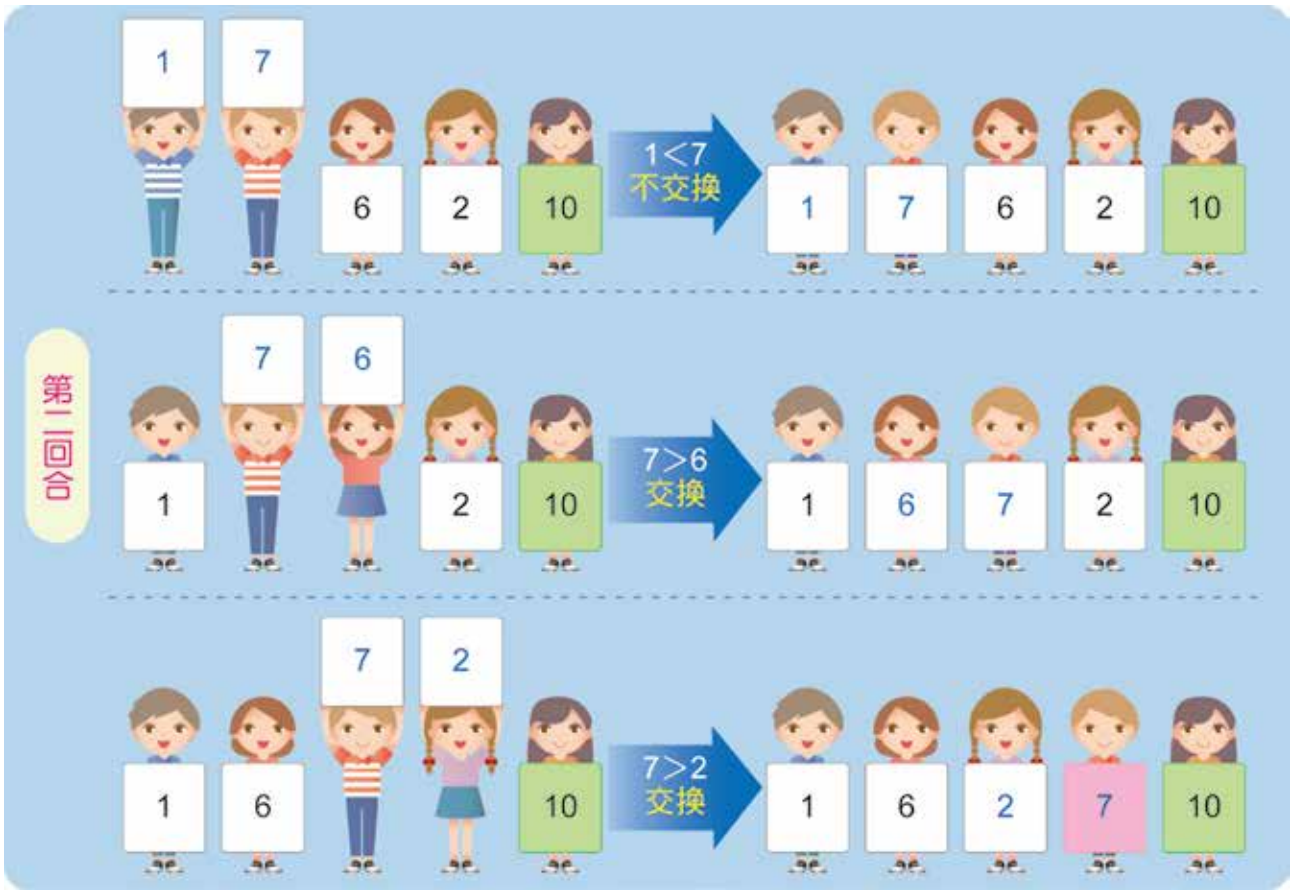


圖 1-3.10　用氣泡排序演算法排序 5 張撲克牌的第二回合。

第二回合後，第二大的號碼7也被交換到正確的位置。從這兩個回合的操作過程可以推論出：

(1) 第 K 回合時會將號碼第 K 大的紙牌交換到正確的位置。

(2) 經過 N-1 個回合後，所有的紙牌都在正確的位置，也就是排序完成。

所以在經過第三、四回合的過程後，完成5張撲克牌的排序動作，如圖1-3.11。

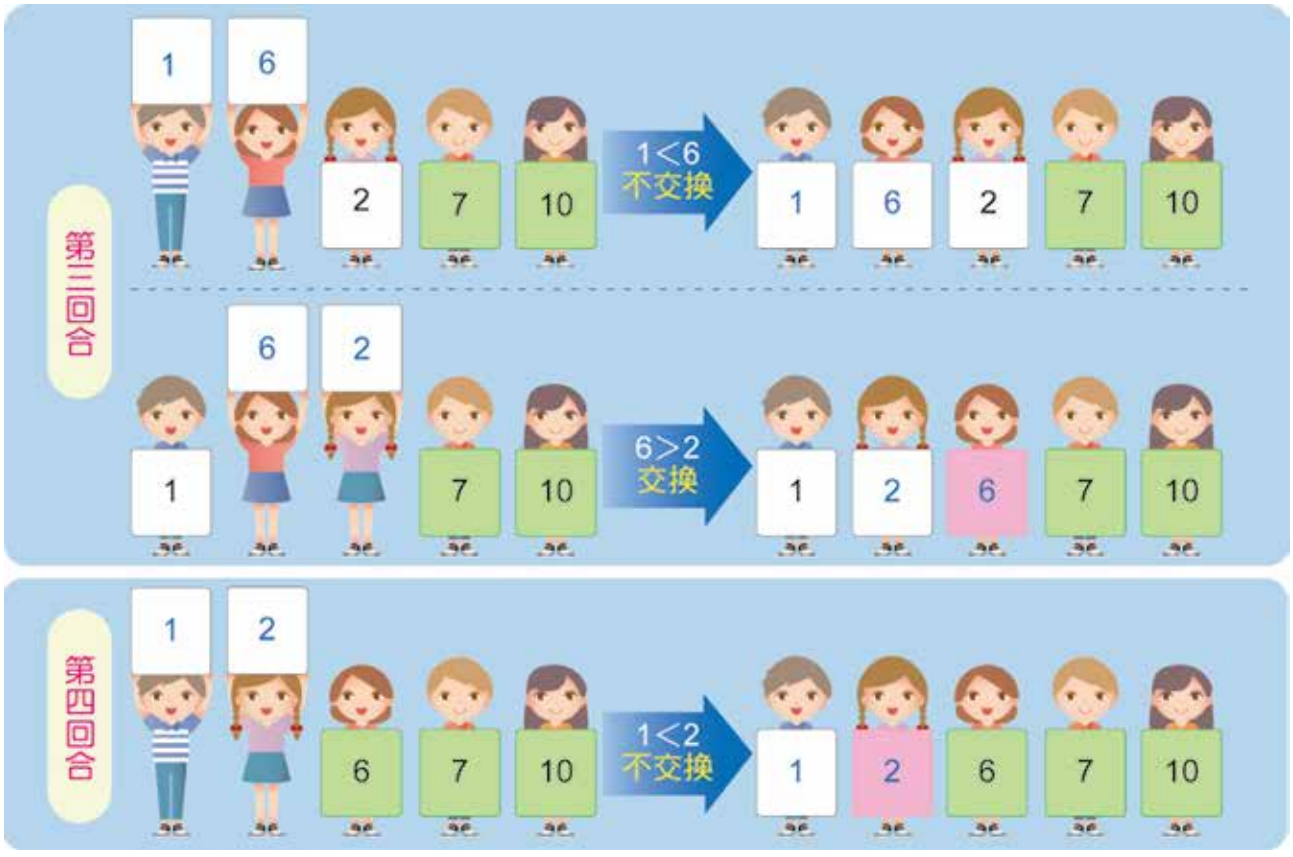


圖 1-3.11　用氣泡排序演算法排序 5 張撲克牌的第三、四回合。

整個排序過程中，每回合會有一個目前最大的號碼最先被交換到定位，就好像水裡面大的氣泡會先浮到水面上，因此這個演算法也被稱為氣泡排序演算法（Bubble Sort）。整理以上的推論可以設計氣泡排序演算法的虛擬碼如圖 1-3.12。

```
1：令 N ＝ 撲克牌總數
2：重複操作 N – 1 回合
3：      第 i 回合中
4：      對於　從第 1 張到第 N – i + 1 張中的相鄰兩張牌 A，B
5：            如果 A ＞ B
6：                  將 A 與 B 交換位置
```

圖 1-3.12　氣泡排序演算法的虛擬碼。

氣泡排序演算法中，只要被排序的資料總數一樣，則需要操作的回合次數及每回合的比較次數都是固定的，不會因為資料的原始順序而改變。因此不論原本的紙牌順序為何，排序 5 張紙牌都需要 10 次的比較才能完成，如表 1-3.3。也就是說，若讓電腦利用氣泡排序演算法將 N 張亂序的紙牌由小到大排列，則需要比較比較 $0+1+2+\cdots+(N-1) = N(N-1)/2$ 次才能完成排序，也就是時間複雜度為 $O(N^2)$。

表1-3.3　以氣泡排序演算法排序5張紙牌所需的比較次數。

	第一回合	第二回合	第三回合	第四回合
比較次數	4	3	2	1

立即演練

（　　）1. 如果電腦用插入排序演算法將 20 張撲克牌由小到大排列，則最少需要比較幾次才能完成排序的動作？　(A)400 次　(B)190 次　(C)19 次　(D)10 次。　記憶

（　　）2. 如果用氣泡排序演算法排序 6 張紙牌，則總共需要比較幾次？
(A)36 次　(B)15 次　(C)6 次　(D)5 次。　記憶

演練&實作

＜排序演算法練習＞

1. 我們剛剛已經介紹了「插入排序演算法」的原理與過程，也提到電腦程式在實作時一次只能比較並交換兩張撲克牌位置的問題，但課本中並未寫出其虛擬碼，請試著將之完成。 應用

【插入排序法的虛擬碼】

1：

2：

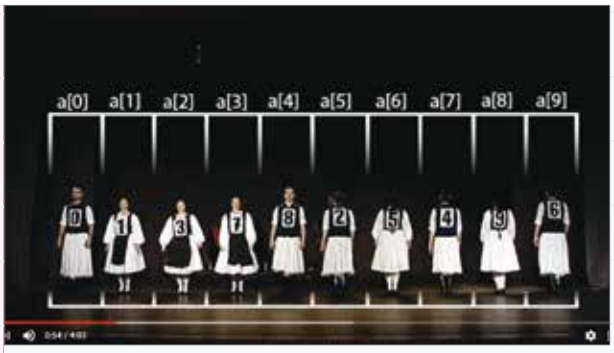
3：

4：

5：

真的想不出來的話，就幾個同學討論一下吧！

自學筆記

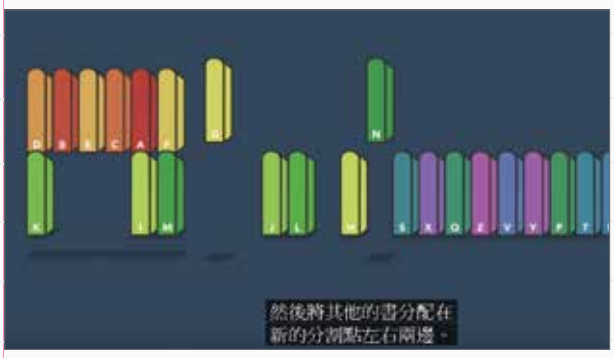


http://ln.cfp.com.tw/u/InsertSort

用羅馬尼亞民俗舞蹈演示「插入排序法」（4:03）

http://ln.cfp.com.tw/u/BubbleSort

用匈牙利民俗舞蹈演示「氣泡排序法」（5:15）



http://ln.cfp.com.tw/u/bookshelves

如何最有效率地整理書架？（4:38）

合併排序演算法

演算法策略中有所謂「分治法」，它的意思是「分而治之」，也就是把問題切分成更小的多個問題，並使用同一套方法應用在這些小問題之上，最終就能完成解答。利用分治法先拆解再合併的技巧，可以設計出合併排序演算法（Merge Sort）如下：每次將紙牌分成差不多數量的兩堆，重複分到一堆只有一張紙牌時，開始合併相鄰的兩堆紙牌，合併時會將兩堆紙牌依號碼大小排序成一列紙牌，並重複直到只剩一列紙牌為止。圖 1-3.13 演示如何利用合併排序演算法排序。

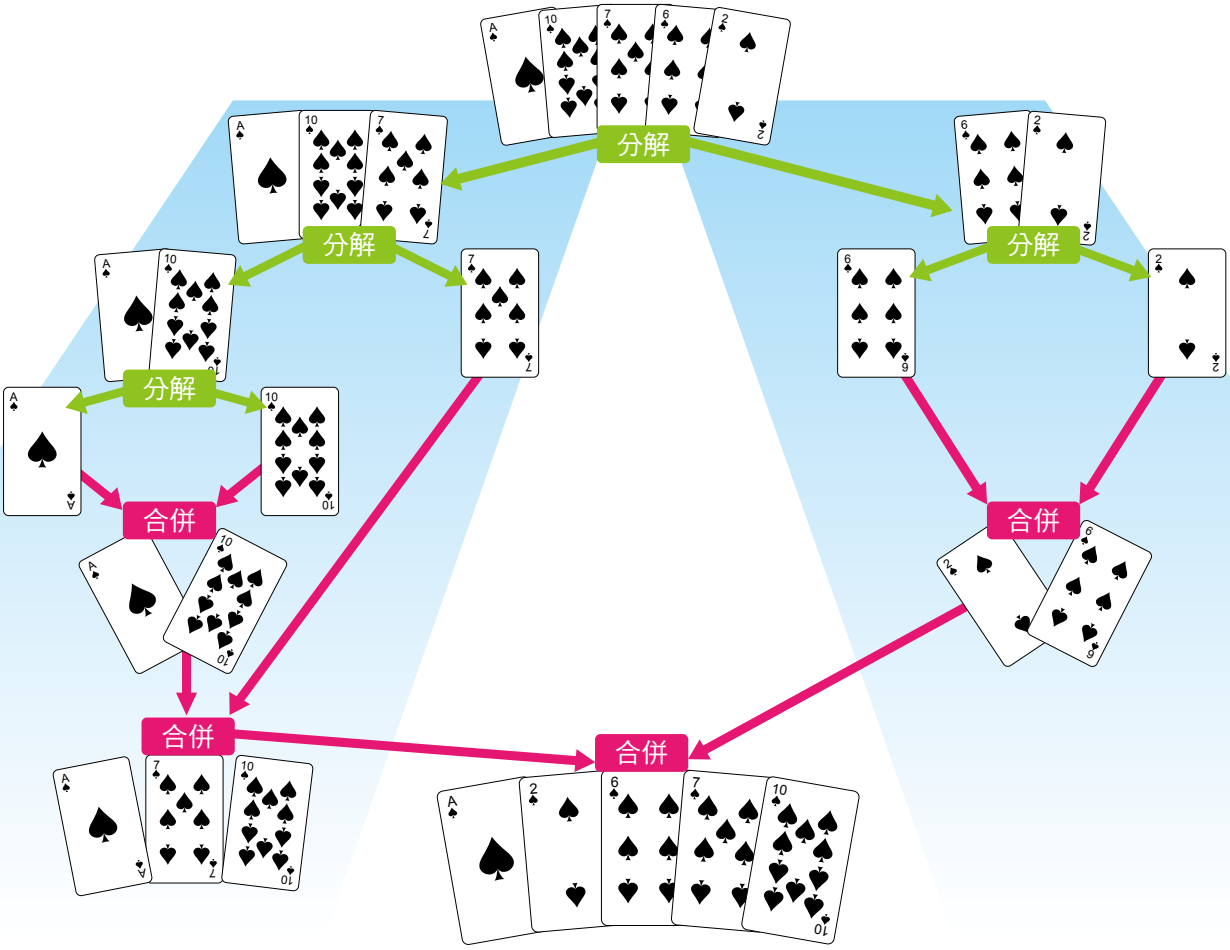


圖 1-3.13　合併排序演算法的排序過程。

從示意圖可知「合併」在這個演算法中是一個很重要的過程，而且一直被重複使用，因此可以將合併包裝成一個函式方便撰寫虛擬碼，但這裡先不解釋如何將兩列已排序紙牌合併成一列已排序紙牌，只需要先知道有這個函式可以呼叫使用。利用遞迴結構的方式，合併排序演算法的虛擬碼可以撰寫如圖 1-3.14：

```
1：函式　合併排序（未排序的紙牌列 A）
2：　　如果　A 只有一張紙牌
3：　　　　回傳結果　紙牌 A
4：　　否則　將 A 分成差不多的兩列紙牌 B 和 C
5：　　　　已排序紙牌列 D = 合併排序（未排序的紙牌列 B）
6：　　　　已排序紙牌列 E = 合併排序（未排序的紙牌列 C）
7：　　　　合併（已排序紙牌列 D，已排序紙牌列 E）
```

圖 1-3.14　合併排序演算法的虛擬碼。

因推導過程繁複我們便不在此探討，然經數學推導後可得，分解的動作需進行 N−1 次，假設 $2^{K-1} \le N < 2^K$ 時，合併排序的動作則需進行 N×K 次的比較，所以利用合併排序演算法排序 N 張牌最多需要比較 $N \times \log N + (N-1)$ 次。

三種排序演算法之比較

表 1-3.4 整理了三種排序演算法在排序 N 張牌時的最多比較次數與時間複雜度，可以發現當資料量 (N) 較少時，三種排序的演算法的最多比較次數相差不大，但當資料量 (N) 增多時，插入排序演算法與氣泡排序演算法的效率相同，但合併排序演算法的效率則明顯優於前二者。

表1-3.4　各種排序演算法在N張牌中的最多比較次數。

N	插入排序演算法	氣泡排序演算法	合併排序演算法
2 (2^1)	1	1	1
32 (2^5)	496	496	191
1024 (2^{10})	523776	523776	11263
時間複雜度	$O(N^2)$	$O(N^2)$	$O(N \log N)$

討論&表達

＜排序大擂台 ── 做得到更要說得出＞

請以下列 5 張撲克牌為例，用自己的話說明如何透過排序演算法將之由大至小排序（即 10、7、6、2、1）。

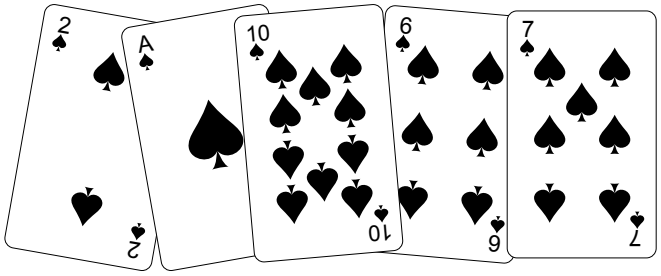


圖 1-3.15
尚未排序的 5 張撲克牌。

1）你打算使用哪一種排序演算法（不限於課本），為什麼？ 理解

2）由大至小排序和課本介紹的由小至大排序，在操作上會有什麼不同？ 分析

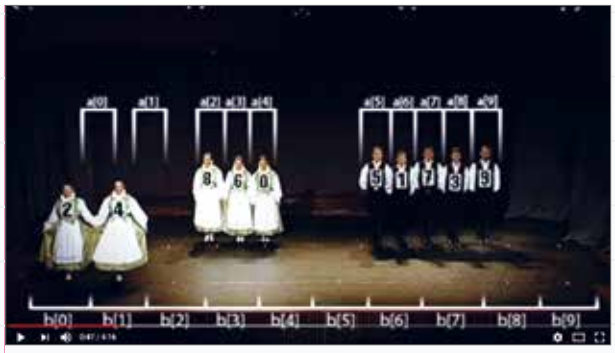
3）請根據你選擇的演算法，說明你的排序過程。 應用

4）請嘗試說明排序過程中有關「遞迴」的觀念。 理解

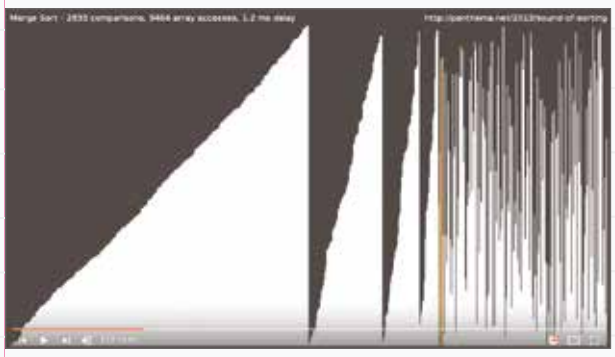
5）請嘗試說明排序過程中有關「分治法」的觀念。 理解

6）請指出其他組的說法中，你覺得值得學習的部份。 評價

請分成幾組討論，完成後每一子題各派一個代表上台說明。



- http://ln.cfp.com.tw/u/MergeSort
  用撒克遜民俗舞蹈演示「合併排序法」（4:16）



- http://ln.cfp.com.tw/u/15SortingAlgorithms
  15 種排序演算法的排序過程展示（5:49）

CH1 結束了，你學會了哪些演算法呢？下一章就要開始寫程式了喔！