Unit

01

演算法

學習內容

- ★ Chapter01 資料結構－堆疊、佇列、樹、圖
- ★ Chapter02 程式設計的概念
- ★ Chapter03 陣列資料結構
- ★ Chapter04 排序演算法－程式設計實作
- ★ Chapter05 搜尋演算法－程式設計實作
- ★ Chapter06 分而治之演算法與遞迴結構

與程式設計

專題學習活動 玩「河內塔」思考解決問題的方法

（活動時間：15分鐘）

活動摘要

傳說在越南河內一座古老的寺廟中有3根銀柱，由下到上套著由大到小的64個金圈圈，如果能夠依照某種規則（以下會說明）將所有的圈圈從1根銀柱移到另1根銀柱（假設每1秒移動1個圈圈），當這些圈圈移動完成時，世界就會產生巨大的變化[註]。這種有3根柱子與N個圈圈的移動遊戲稱之為「河內塔」遊戲，我們將透過以下的活動，帶領同學完成移動3個圈圈的河內塔遊戲！

學習目標

1. 能應用運算思維評估問題解決的方法。
2. 能使用多元的觀點進行思辨。
3. 能具備與人溝通、協調、合作的能力。

活動進行

1. 將全班同學分為數組。
2. 各組連上提供河內塔遊戲的網站，如：樂和遊戲（https://www.novelgames.com/zh-HK/tower/），並依照下列規則進行遊戲，將3個圈圈從A柱移動到C柱，即完成河內塔遊戲。
   (1) 每次只能移動1個圈圈。
   (2) 大圈圈不能疊在小圈圈上。
   (3) 在移動過程中可利用B柱作為暫時疊放的柱子。

3. 比較各組的移動次數，看哪一組的步數最少。

想想看：有什麼方法能以最少的步數完成河內塔遊戲？圈圈的移動是不是有規律可遵循呢？

同學在探討河內塔遊戲的問題及思考解決方法時，與本單元要介紹的「演算法」及如何設計程式解決問題的概念十分類似。本單元將介紹演算法與程式設計的基礎概念，並帶領同學撰寫出能以最少步數完成河內塔遊戲的程式

註 若依照規則每1秒移動1個圈圈，則完成64個圈圈的河內塔遊戲需要5,845億年。

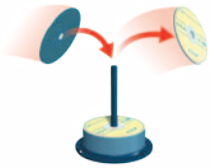
Chapter

01 資料結構－堆疊、佇列、樹、圖

同學在國中階段已學過基本的程式設計，也了解到「程式」的主要功能，就是要運算及處理電腦「資料」。本章將進一步介紹，若能規劃、組織好資料的「結構」，將會使程式的運作更有效率。

1-1 基本資料結構

電腦程式的運算與運作，其效率是否完善，最主要取決於下列兩點，一、程式碼的設計是否能讓運算單元在處理時做最有效的執行，採用正確適當的「演算法」（將於後續章節介紹）將有助於達到此目的。二、存放在記憶單元中的資料是否兼顧到空間節省及存取便捷，此即本章「資料結構」探討的重點。本章除了國中課程教過的「陣列」資料結構外，將介紹基本「堆疊」與「佇列」，以及進階的「樹」、「圖」資料結構。



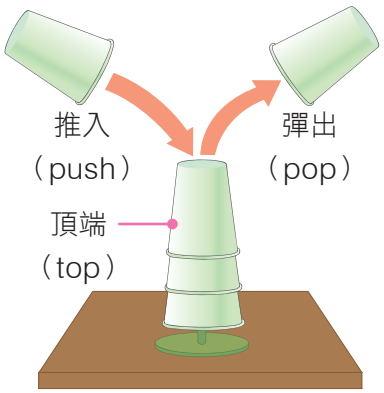
1-1.1 堆疊

要從盛裝光碟片的桶子（俗稱布丁桶）取出光碟片時，通常是從放置在最上面的光碟片開始拿取，放在桶子最底下的光碟片會最後才被取出。而放入光碟片時，該片光碟會放在整疊光碟片的最上方。這種取出與放入光碟片的方式，與本小節要介紹的**堆疊**結構有類似之處，以下就來介紹堆疊的概念、應用及實作。

堆疊的介紹
https://www.youtube.com/watch?v=gb1j7qhDPqY&t=49s

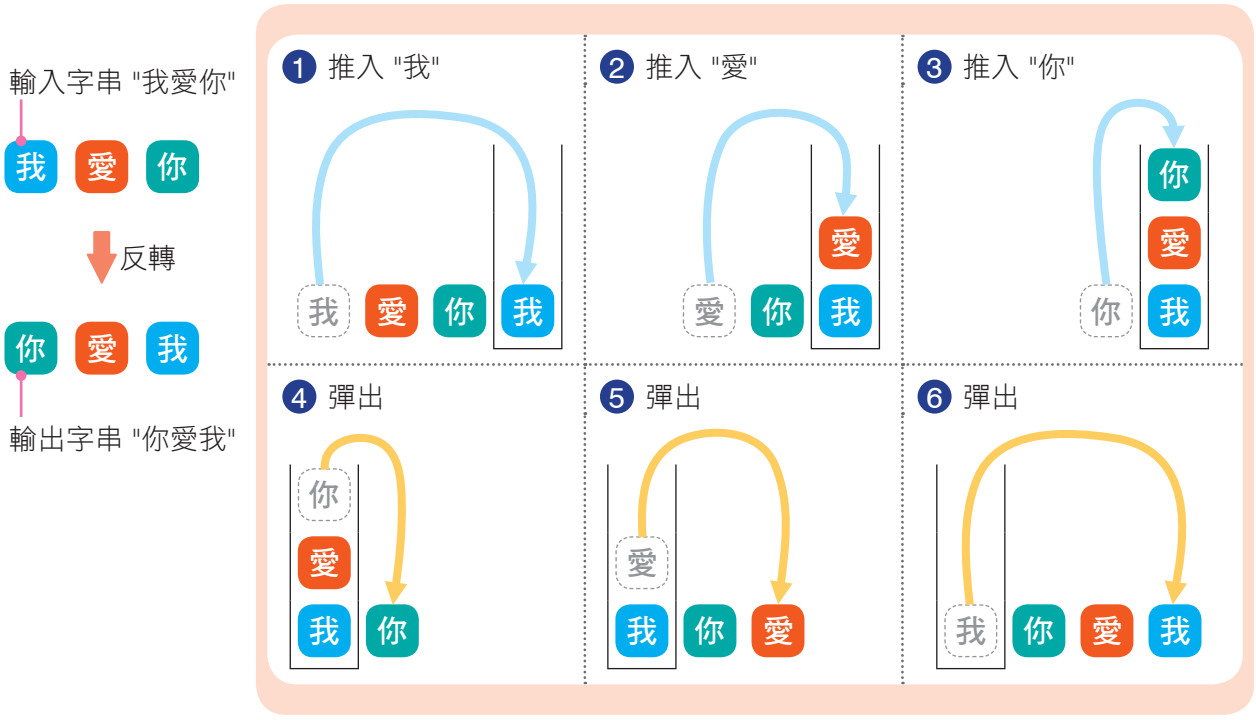
堆疊的概念

堆疊（stack）是一種有特定存取規則的線性儲存結構，它的資料存取規則是：資料項目加入時，只能加到堆疊的**頂端**（top）；取出資料項目時，必須由頂端將資料取出（圖1-1）。在堆疊中加入資料的動作稱為**推入**（push），取出資料的動作則稱為**彈出**（pop），它是一種具有**後進先出**（Last In First Out, LIFO）特性的資料結構，也就是最後放進去的資料會最先被取出來。



▲ 圖1-1 堆疊示意圖與存取方式

我們該如何應用資料結構的特性來將一個字串 "我愛你" 反轉成 "你愛我" 呢？只要利用堆疊結構的「後進先出」特性，就可以將字串反轉，圖1-2示範在堆疊中「推入」與「彈出」，將字串反轉的過程。



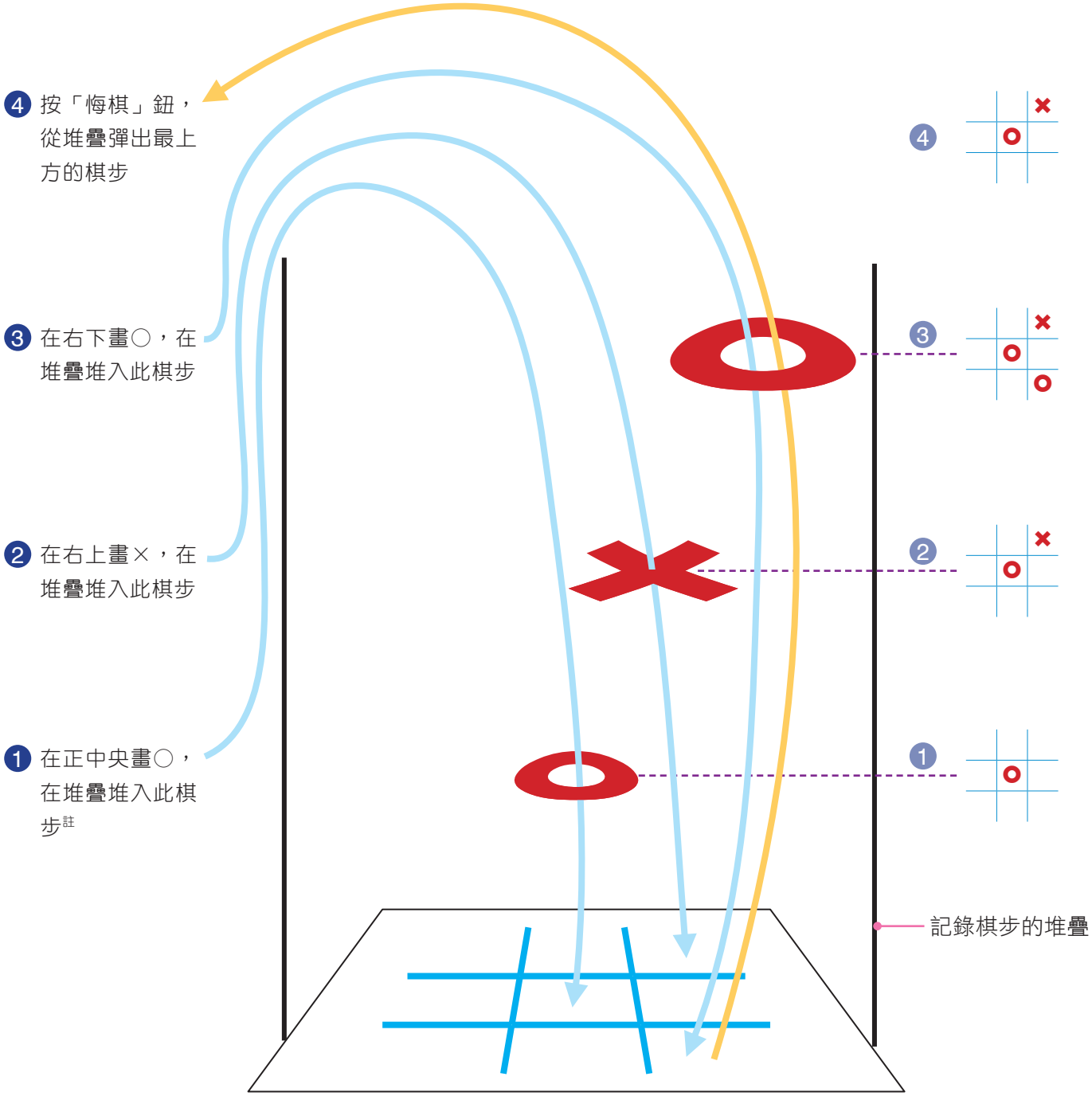
▲ 圖1-2　在堆疊中推入與彈出將字串反轉的過程

堆疊的應用

堆疊的應用相當廣泛，例如一些棋藝遊戲軟體有「悔棋」功能（圖1-3），可讓使用者按「悔棋」鈕回復上一步棋步，這是因為棋藝遊戲軟體中就可使用一個堆疊，來依序存放（推入）使用者下的棋步，每按一次「悔棋」鈕，就會從堆疊中消除（彈出）最上方的棋步，使棋盤上的棋子排列回復到前一步的狀態。

▲ 圖1-3　棋藝遊戲的悔棋功能

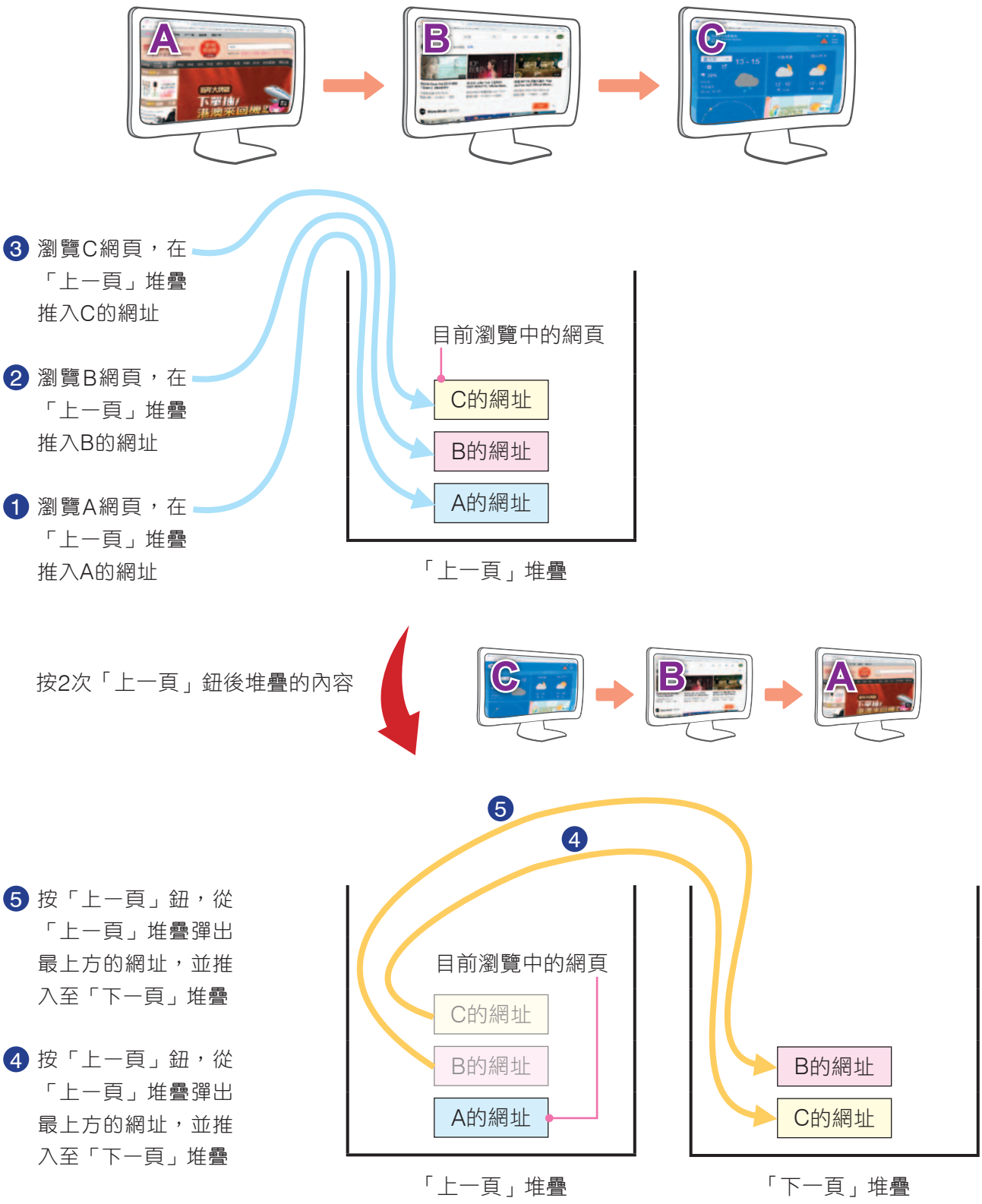
例如用最簡單的「井字棋」（也稱為井字遊戲）為例（圖1-4），就可很容易地了解如何透過堆疊達成悔棋功能。



▲ 圖1-4 堆疊應用的示意圖（棋藝遊戲的悔棋功能）

註 圖1-4僅是推入堆疊的示意圖，程式實際撰寫時，會使用其他方式來記錄，例如將九宮格分為1～9格，左上（第1格）畫○可記錄為「1○」、右下（第9格）畫×可記錄為「9×」。

除了棋藝遊戲之外，瀏覽器也可利用堆疊結構，來設計上一頁／下一頁功能（圖1-5）。



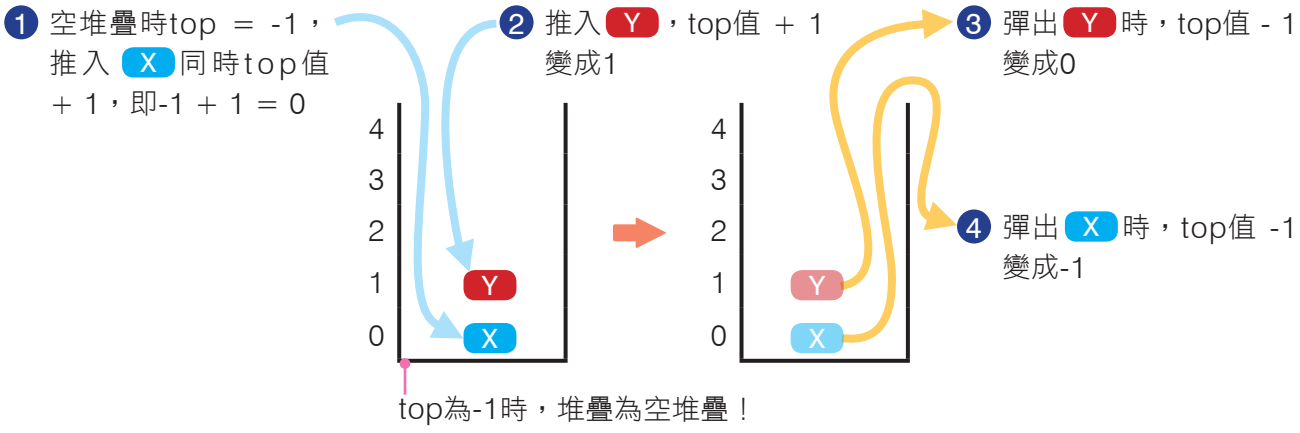
▲ 圖1-5 堆疊應用的示意圖（瀏覽器的上一頁／下一頁功能）

補充資料：上述僅是堆疊應用示意圖，瀏覽器的上一頁/下一頁功能可依程式設計師設計而有所不同，例如設計步驟2在瀏覽B網頁時，才將「A的網址」放入「上一頁」堆疊中，依此類推。

堆疊的實作

堆疊資料結構常用「陣列」來實作。基本作法是先建立一個空陣列準備存放資料，並用一個變數「top」來記錄何處是最上層的資料。

設top初始值為-1，表示堆疊一開始是空的。每次推入一筆資料，就將top值+1，每次彈出一筆資料，就將top值-1。圖1-6示範如何用陣列來實作堆疊的資料存取。

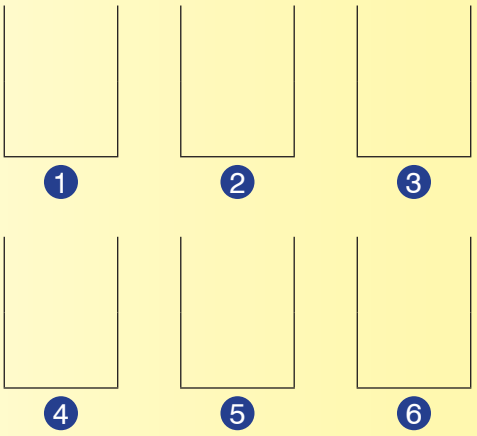


▲ 圖1-6 利用陣列實作堆疊的資料存取示意圖

馬上練習

1. 如果要將一個原始字串 "tab"（標籤）中的每一個字元利用堆疊的運作方式，分別推入及彈出堆疊來改變字元的順序，使結果字串成為 "bat"（球棒），該如何推入及彈出呢？請將過程填在下圖及下表中。

原始字串： t a b



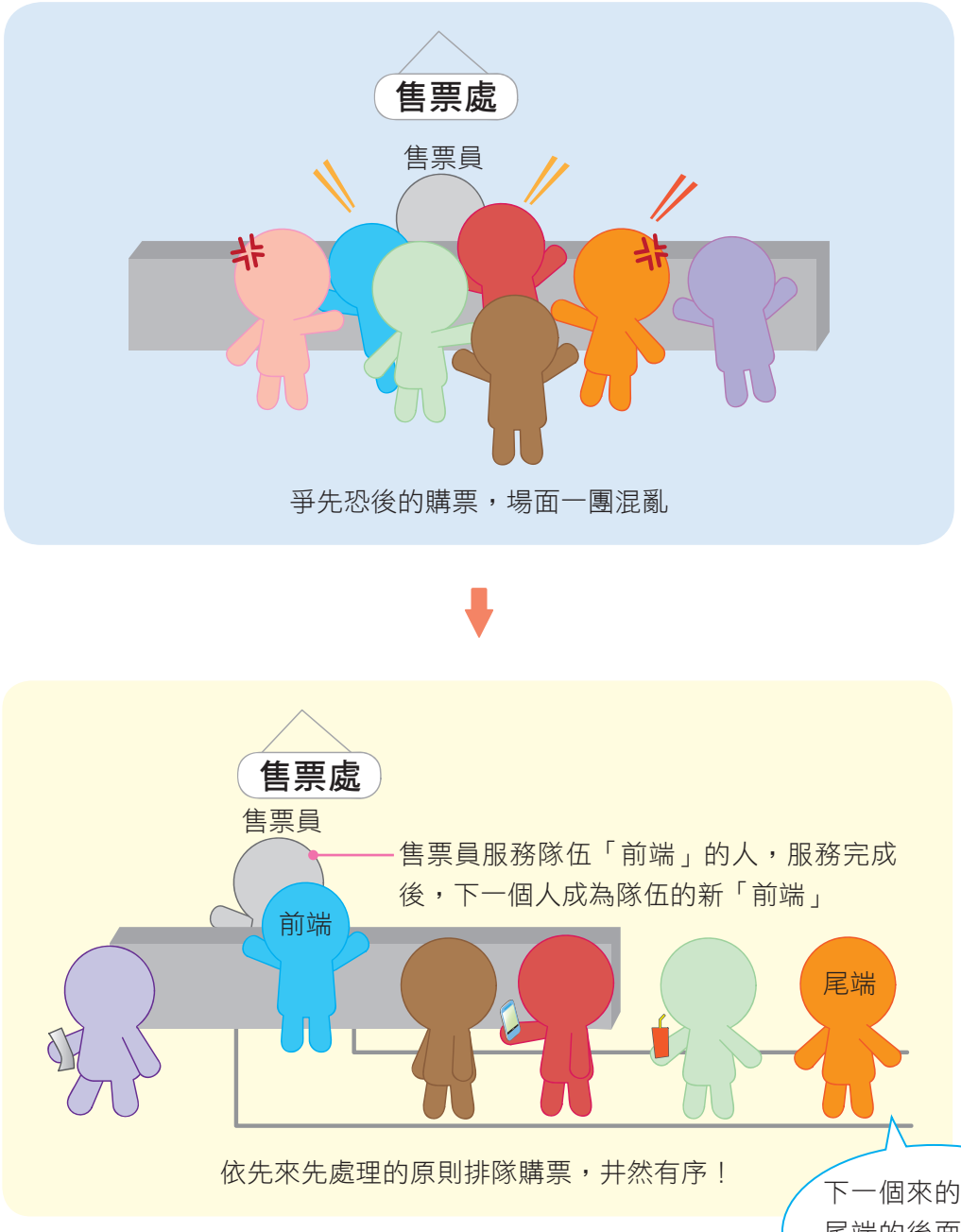
步驟	動作	
1	推入	t
2	?	?
3	?	?
4	彈出	b
5	?	?
6	?	?

結果字串： b a t

佇列的介紹
https://www.youtube.com/watch?v=r-3qNhCdUQw&t=44s

1-1.2　佇列

在車站購票時，如果一群人一湧而上，購票作業會很難進行，但若在開始售票前，請民眾依照先後順序排隊，第1個到的人排第1位（隊伍前端），第2個到的人排第2位，……，最晚到的人排最後（隊伍尾端），售票作業開始時，售票人員即可從隊伍前端開始依序售票。這種排隊購票的方式（圖1-7），與本小節將介紹的**佇列**結構有類似之處，以下就來介紹佇列的概念、應用及實作。

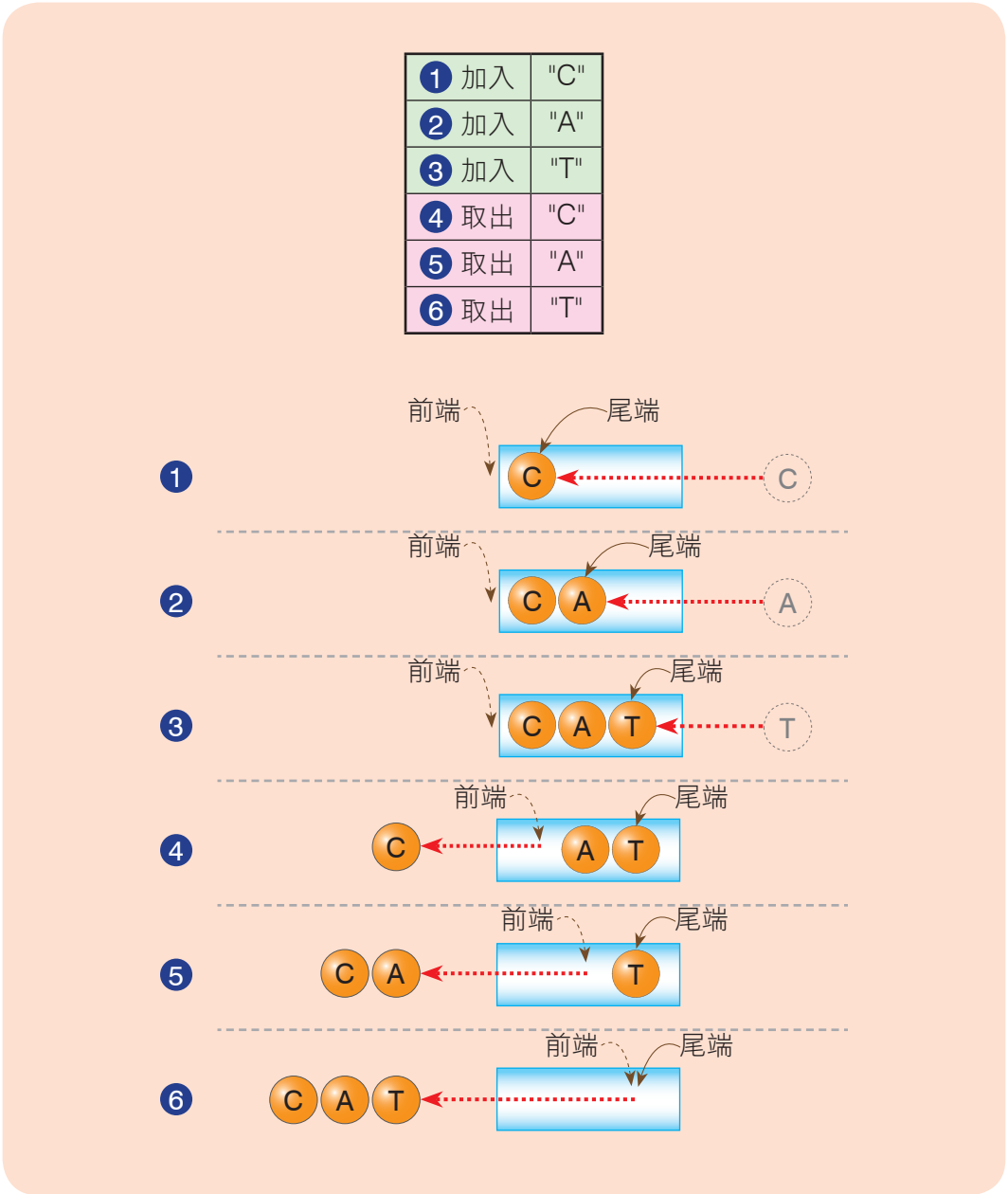


▲ 圖1-7　處理購票的方法

佇列的概念

佇列（queue）也是一種有特定存取規則的線性儲存結構，它的資料存取規則是：加入資料項目時，只能從佇列的**尾端**（rear）加入；取出資料項目時，則只能從佇列的**前端**（front）取出，是一種具有**先進先出**（First In First Out, FIFO）特性的資料結構。

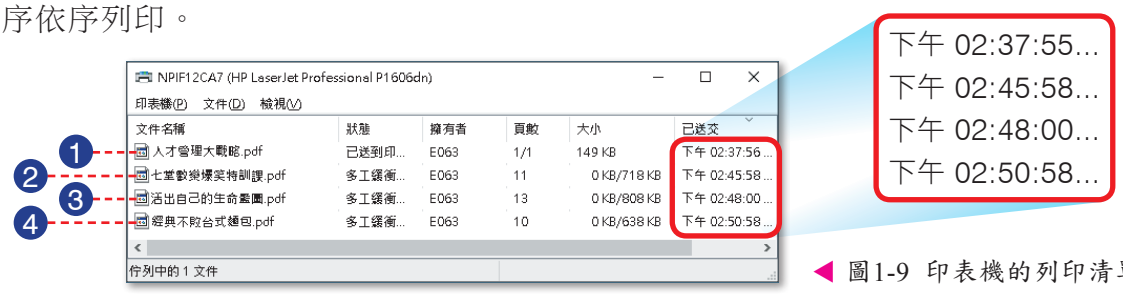
圖1-8示範在佇列中「加入」與「取出」資料的過程。



▲ 圖1-8 在佇列中「加入」與「取出」資料的過程

佇列的應用

瀏覽器中的下載清單、作業系統的工作排程、印表機的列印清單都屬於佇列的應用。以印表機的列印清單為例說明，每次提出一項列印要求，電腦即會在清單最後方新增一個列印任務，而印表機會依據列印任務的先後加入的順序，將最早加入列印清單中的項目最早列印出來。圖1-9中共有4項等待列印的工作，會依❶（前端）→ ❷ → ❸ → ❹（尾端）的順序依序列印。



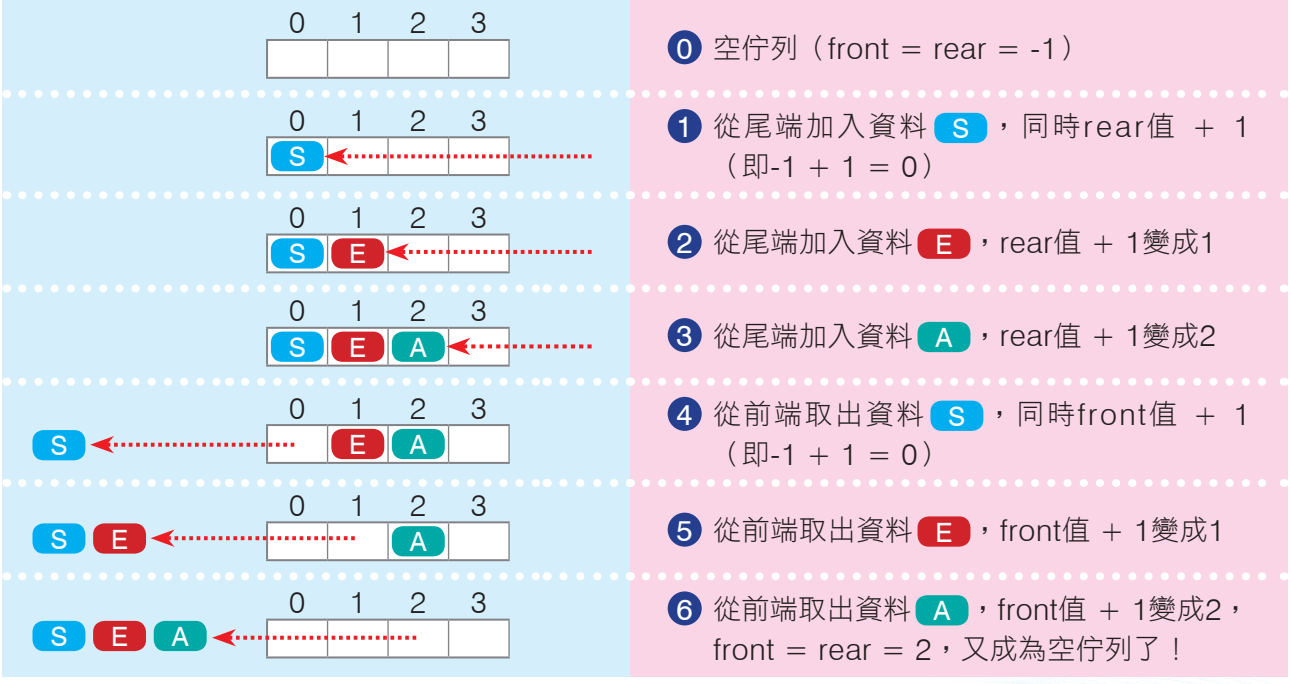
◀ 圖1-9　印表機的列印清單

佇列的實作

佇列資料結構也常用「陣列」來實作。首先要建立一個空陣列準備存放資料，並用變數「front」（前端）及「rear」（尾端）來記錄資料的前端及尾端。

初始時預設front及rear值為-1，表示佇列一開始是空的。每次要加入一筆資料，就將rear值+1，然後陣列註標為rear值的元素中便會加入該筆資料；每次要取出一筆資料，就將front值+1，然後從陣列註標為front值的元素中取出資料。

圖1-10示範如何用陣列來實作佇列的資料存取，在空佇列中依序加入3個資料 S 、 E 、 A ，然後再從佇列中分次取出資料，在過程中，觀察front、rear、及佇列內容的變化。

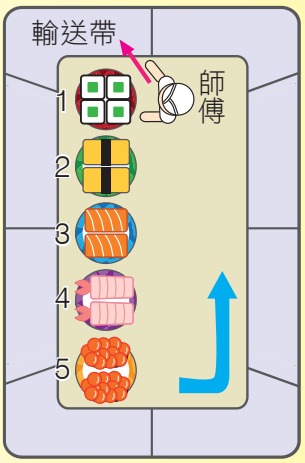


▲ 圖1-10　利用陣列實作佇列的資料存取示意圖

堆疊與佇列的介紹
https://www.youtube.com/watch?v=Pp8EkF_6WyE

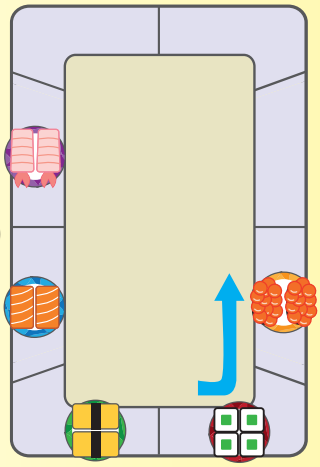
馬上練習

1. 阿強是迴轉壽司師傅，他每次捏好一個壽司，都要放到輸送帶上，請問他放完5個壽司後，輸送帶上的壽司位置關係會是下列哪一個選項呢？

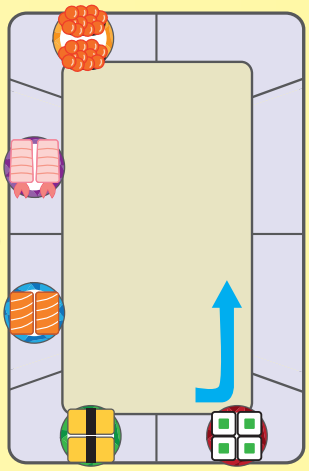


答案：（？）。

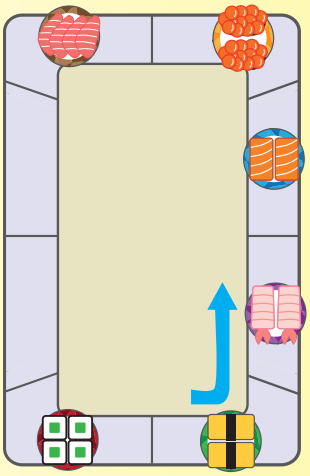
(A)



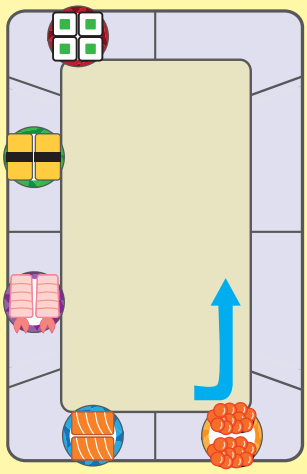
(B)



(C)

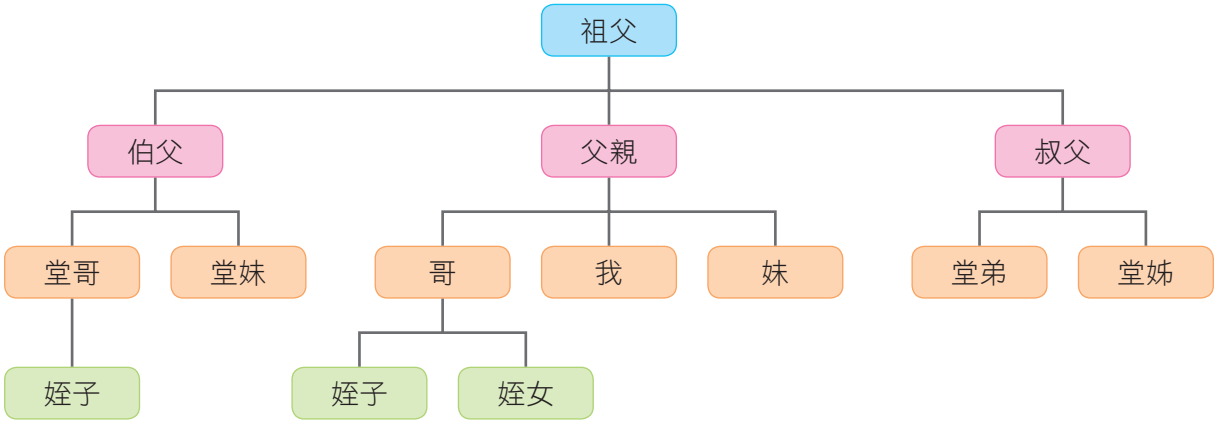


(D)



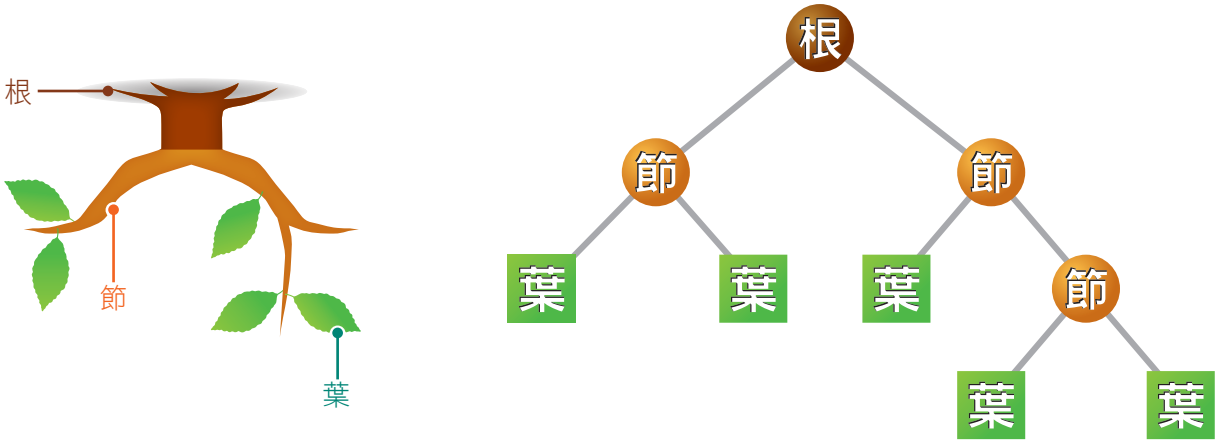
1-2　樹

在日常生活中，對於具有上下從屬特性的相關資料，我們常用階層式的結構來記錄，例如記錄「族譜」（圖1-11）時，即是從某個最上層的祖先開始，然後延伸出他的子輩、孫輩……，最後組成一個家族的族譜。



▲ 圖1-11　族譜示意圖

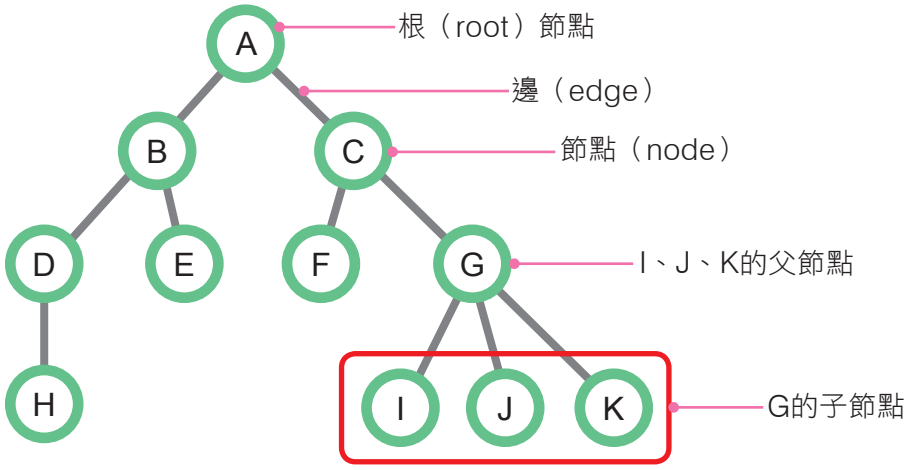
上述所舉的族譜圖例，是由位階最高的始祖開始，往下分支出子輩、孫輩等，就像一個「**倒立的樹狀結構**」（圖1-12），這種結構與本節將介紹的「**樹**（tree）」頗為類似，以下將介紹「樹」的概念及應用。



▲ 圖1-12　倒立的樹狀結構

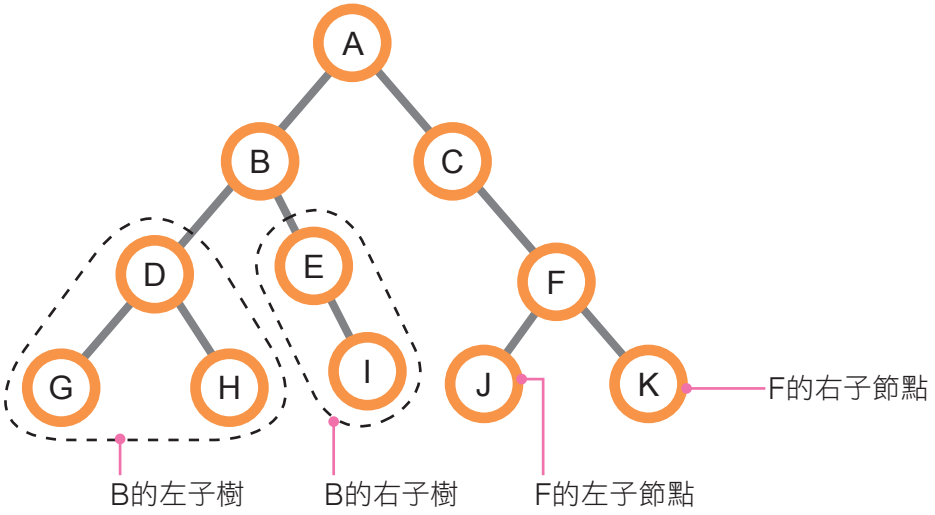
1-2.1 樹的基本概念

樹是由一串相連的**節點**（node）所構成，所有的節點都是由一個**根**（root）節點衍生出來，節點與節點間是透過**邊**（edge）相連接，任一節點可以衍生出若干個節點，由G衍生出來的節點，稱為G的**子節點**，而G為這些節點的**父節點**。圖1-13為一個樹資料結構。



▲ 圖1-13 樹資料結構

如果一棵樹的節點最多只有2個子節點，這種樹稱為「**二元樹**（binary tree）」。二元樹是一種常見的樹狀結構，常用在資料的搜尋、排序等工作上。**二元樹**的特色是每個節點下方只能接0～2個節點，若有2個節點，分別稱為「左子節點」及「右子節點」，而左子節點後方的樹稱為「左子樹」，右子節點後方的樹稱為「右子樹」。圖1-14中  都是二元樹的基本結構。

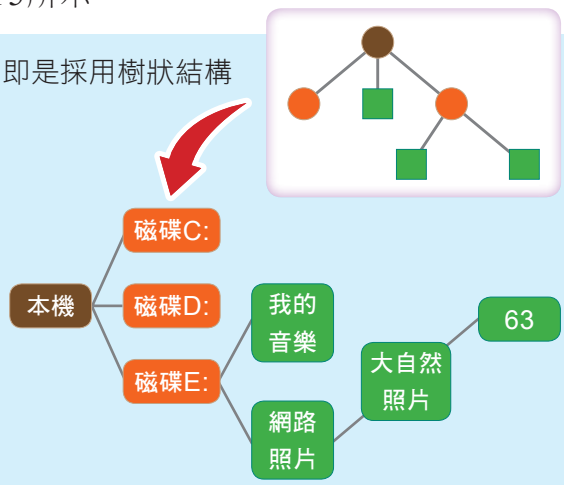


▲ 圖1-14 二元樹

在電腦系統中，也有許多樹的應用，如圖1-15所示。

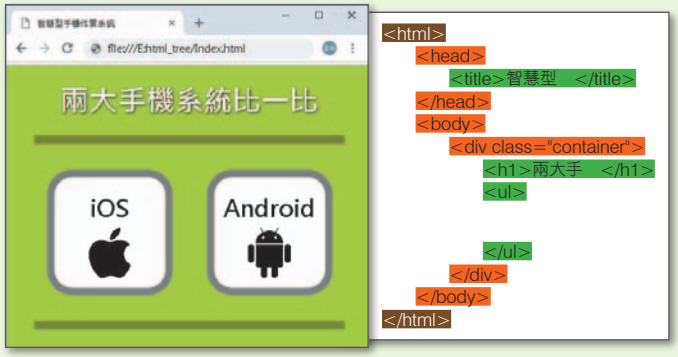
Windows檔案結構

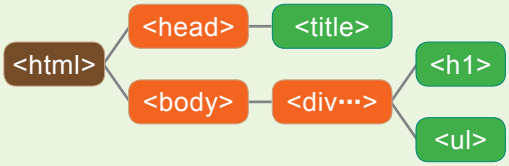
Windows作業系統的檔案總管，即是採用樹狀結構來管理眾多的檔案



HTML網頁語言結構

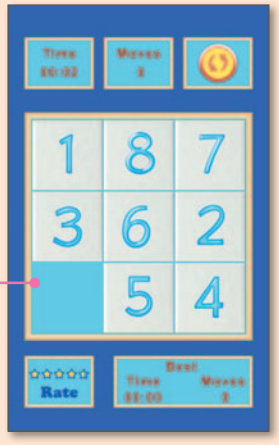
<html>標籤就是根節點，下層可再包含多個標籤節點（如<head>、<body>等），每個標籤可再延伸出下層標籤（如<body>可延伸許多個 標籤），這種結構即是樹狀結構的應用



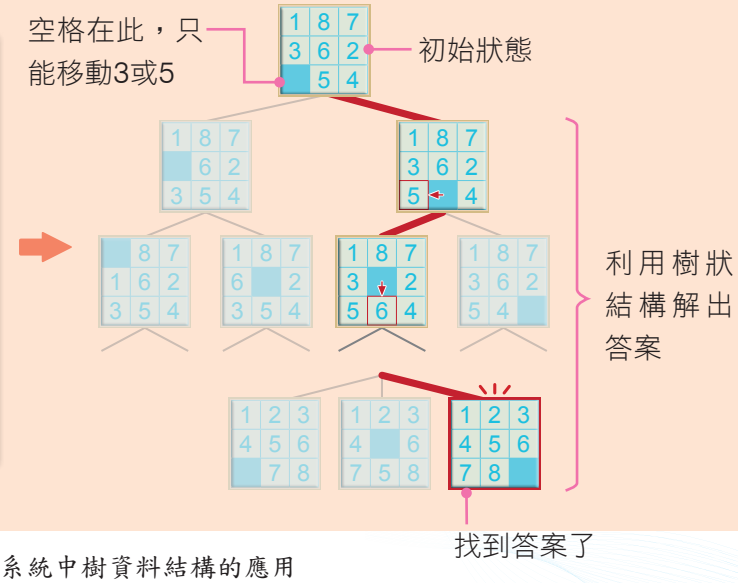


數字推盤遊戲

數字推盤遊戲在尋找解題方法時，可將推盤的移動過程利用樹狀結構一一排列出來，其中有一個（或多個）節點所呈現的就會是完成遊戲的解答，從該節點回溯至根節點，即可找出遊戲的解法（推動數字移動的順序及方向）



只有1個空格，推動方塊讓數字依序排列即可過關



▲ 圖1-15　電腦系統中樹資料結構的應用

二元搜尋樹的介紹
https://www.youtube.com/watch?v=-InrkWn0ek0

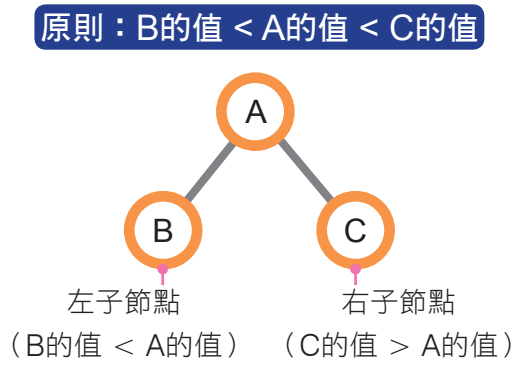
1-2.2 樹常見的應用

「樹」可以應用在解決許多不同的問題上，以下介紹「**二元搜尋樹**」、「**決策樹**」、「**字典樹**」等3種常見的應用。

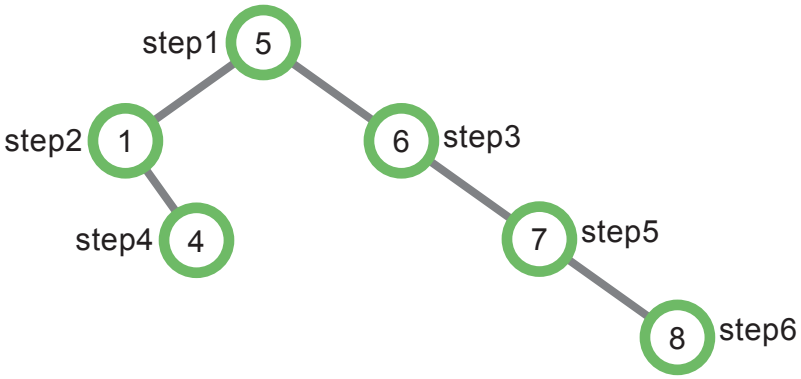
二元搜尋樹

二元搜尋樹（binary search tree）是一種應用二元樹來做資料搜尋的技巧，要使用二元搜尋樹前必須將資料儲存至二元樹中，儲存資料的原則如圖1-16。

例 若輸入資料為：5、1、6、4、7、8，請建立二元搜尋樹。



▲ 圖1-16 二元搜尋樹儲存資料原則



建立方法：依5、1、6、4、7、8的順序建立二元搜尋樹。

step1：5做為根節點

step2：1 < 5，為5的左子節點

step3：6 > 5，為5的右子節點

step4：4 < 5，再於5的左子樹判斷，
4 > 1，為1的右子節點

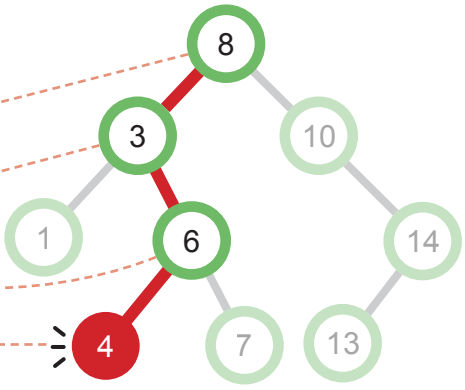
step5：7 > 5，再於5的右子樹判斷；
7 > 6，為6的右子節點

step6：8 > 5，再於5的右子樹判斷；
8 > 6，再於6的右子樹判斷；
8 > 7，為7的右子節點

在二元搜尋樹中搜尋數值的方法是：由根節點開始搜尋，如果要搜尋的數值小於根節點的數值，就往左下搜尋，否則就往右下搜尋。接著再比對下一層節點，直到找到數值。若已搜尋到樹的最下層仍未找到要搜尋的數值，就代表此數值不在此樹中。

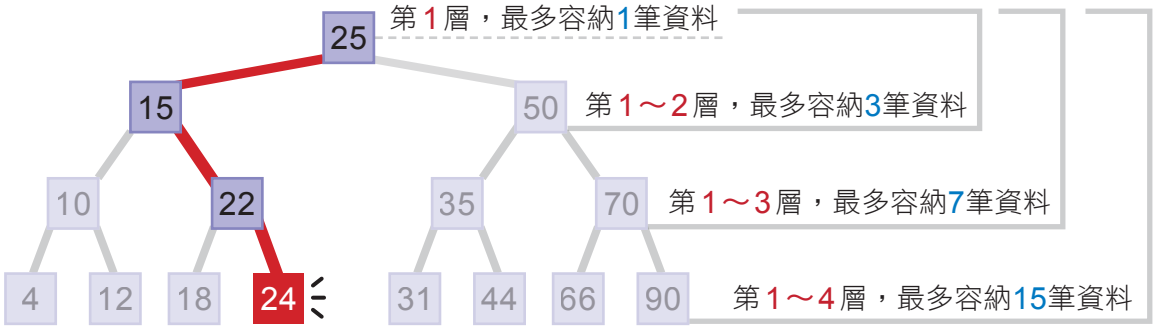
圖1-17為一個二元搜尋樹，要在此樹中搜尋數字4，看它是否存在此樹中，步驟如下：

	比較點	下一個搜尋路徑
a	4 < 8	往左下
b	4 > 3	往右下
c	4 < 6	往左下
d	4 = 4	找到了



▲ 圖1-17 二元搜尋樹

從上圖可看出，二元搜尋樹愈多層，所需的最多搜尋次數就愈多，當二元搜尋樹有N層，最多搜尋次數即是N次，例如4層的二元搜尋樹，最多需搜尋的次數就是4次。當二元搜尋樹的每一層都剛好容納2個子節點時，二元搜尋樹能容納最多的資料數量如圖1-18所示。



▲ 圖1-18　容納最多資料的二元樹

馬上練習

1. 若輸入資料為：4、5、8、1、7、6，請建立二元搜尋樹。

在此畫出二元搜尋樹

(1) 二元搜尋樹共有幾層？＿＿＿＿＿＿。

(2) 在此二元搜尋樹中搜尋存在的資料，最多需搜尋幾次？＿＿＿＿＿＿。

2. 一個5層的二元搜尋樹，最多可以容納＿＿＿＿＿＿筆資料。

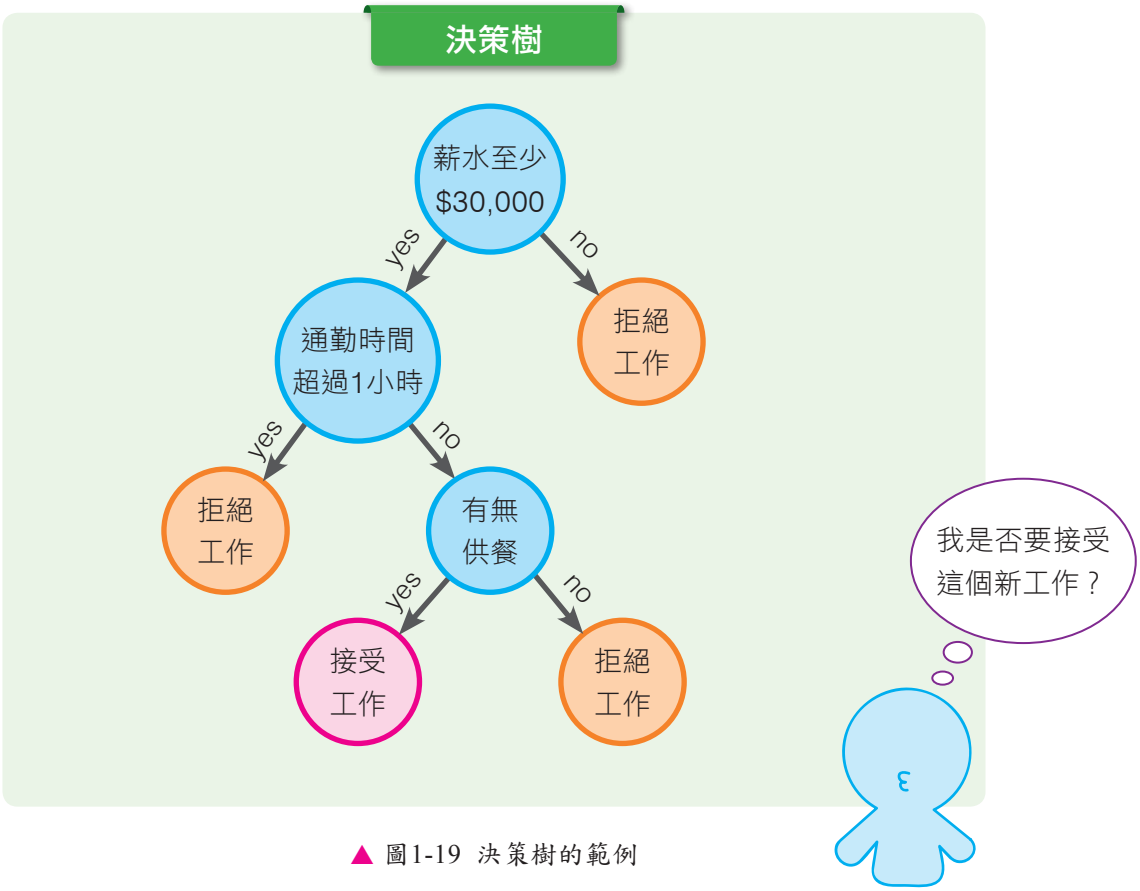
第1題輸入的資料（4、5、8、1、7、6）與前一頁例輸入的資料（5、1、6、4、7、8）數值相同，僅順序不同，但畫出來的二元搜尋樹會長得不一樣！

etb 決策樹的介紹
https://www.youtube.com/watch?v=jkJ4dbbpVCQ

決策樹

決策樹（decision tree）是由上至下，將每一層決策可能引發分歧的選項用「樹」的結構記錄下來，並在最底層歸類整理出決策結果。決策樹建立之後，當遇到問題時，即可依此決策樹的脈絡來找到決策方案。

圖1-19為一個用來抉擇「我是否要接受這個新工作？」的簡單決策樹範例，若決策過程較複雜，樹的分支即可能更多。



▲ 圖1-19 決策樹的範例

善用決策樹，可以幫助我們分析複雜問題，並在未來遇到相同問題時，使用決策樹來加快判斷速度。例如醫療專家系統，會根據患者提供的多項症狀，用決策樹來判斷可能的病因。

馬上練習

1. 「兔兔農場」中有2種兔子，分別為「甲類兔子」與「乙類兔子」。有一天2種兔子意外混雜在一起，農場主人要聘請工讀生依照兔子的特徵將兔子分類，他將2種兔子的特徵書寫如下：

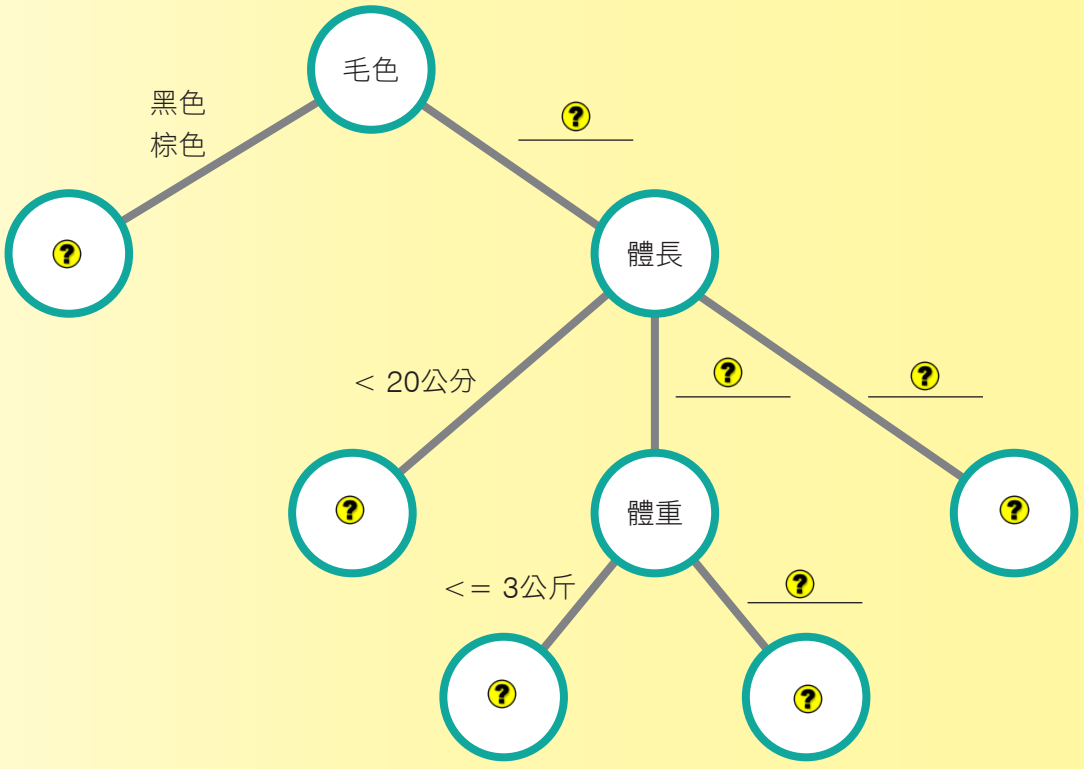
   (1) 甲類兔子的毛色可能是白色、黑色、棕色，而乙類兔子只有白色。

   (2) 甲類兔子的體長為30公分以內（含），而乙類兔子則為20公分（含）以上。

   (3) 甲類兔子的體重小於3公斤（含），而乙類兔子的體重超過3公斤。

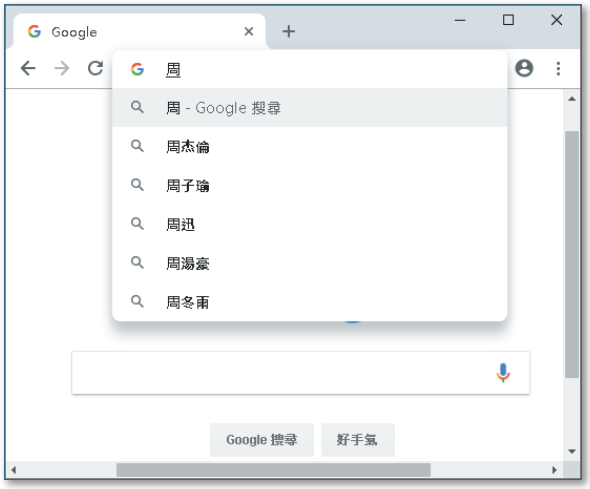
   請將上述兔子分類的原則繪製成「決策樹」，供工讀生依決策樹進行分類。

請在空白圓圈與底線處填入答案，以完成決策樹



字典樹

字典樹又稱為前綴樹（prefix tree），我們在搜尋引擎搜尋資料時，只要輸入一部分文字，網頁就會列出你輸入文字開頭的相關資訊供你選擇（圖1-20），這就是字典樹的應用。

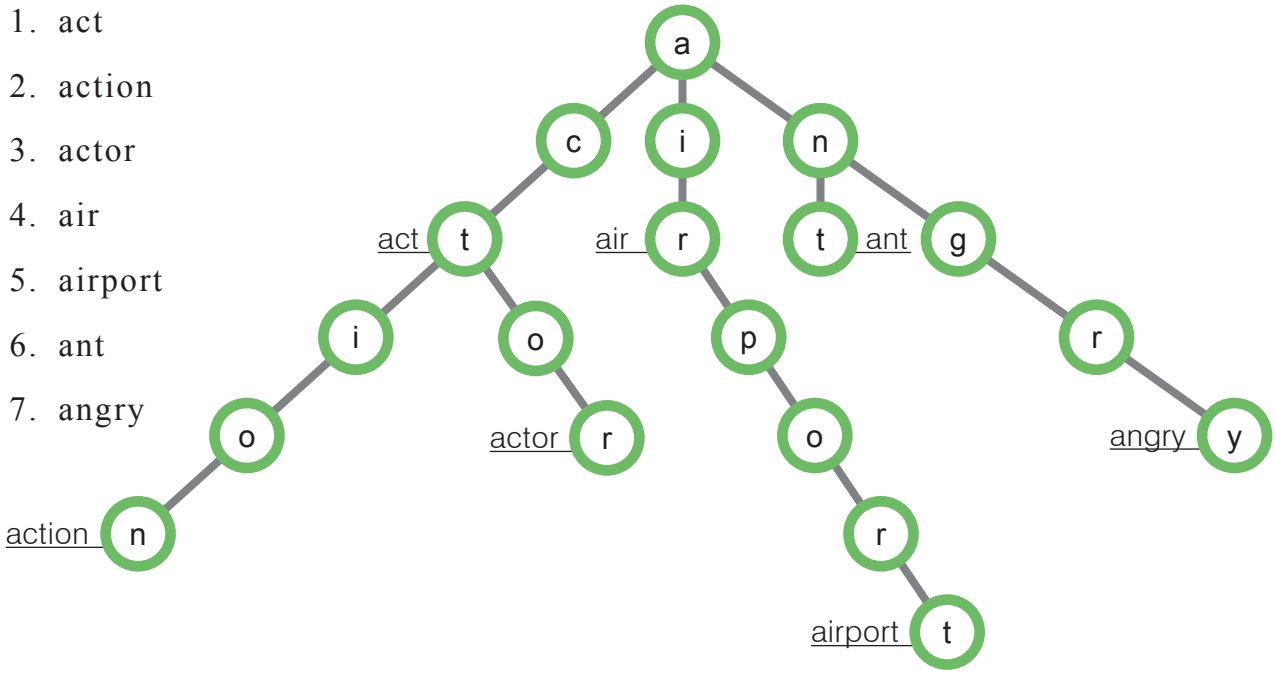


（https://www.google.com.tw/）

◀ 圖1-20 搜尋引擎提供的關鍵字推薦

字典樹的結構如圖1-21所示，它會將相關字串共同的字首部分儲存在父節點，而其他相異的部分，則存在衍生的子節點。

1. act
2. action
3. actor
4. air
5. airport
6. ant
7. angry



▲ 圖1-21 字典樹的範例

建立字典樹以後，即可依循樹中儲存的節點，來列出相同字源的單字。以上圖的字典樹為例，輸入「an」，即會列出「ant」與「angry」。

使用字典樹來儲存資料，具有下列2項優點：

- **可節省儲存空間**：由於字典樹儲存詞彙時，共同的字首部分不必重複儲存，因此比一筆一筆詞彙分別儲存的方法更節省儲存空間。
- **可加快搜尋時間**：由於在字典樹中搜尋詞彙時，每找一個字母，都可以縮小搜尋範圍，因此搜尋時間會比逐詞尋找特定詞彙快。

馬上練習

請將下列10個中文電影名稱依序加入至字典樹中，並繪製在下方空白處。要注意，新節點加入的順序為由上至下、由左至右。繪製字典樹後，請回答下列問題。

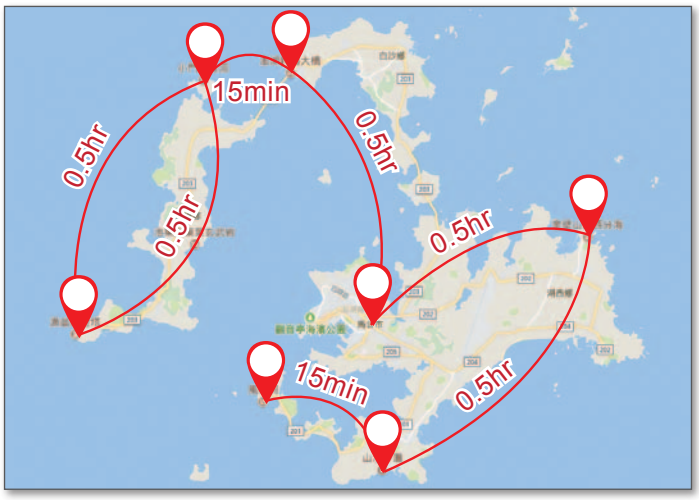
(1) 神力女超人
(2) 神鬼奇航
(3) 神偷軍團
(4) 神鬼交鋒
(5) 神隱少女
(6) 神鬼獵人
(7) 神偷奶爸
(8) 神鬼交易
(9) 神力女超人的秘密
(10) 神隱任務

1. 開頭為 "神鬼" 的電影共有幾部？＿＿＿＿＿＿＿＿＿＿。
2. 若「樹根」算第1層，此字典樹的第2層有幾個節點？＿＿＿＿＿＿＿＿＿＿。

在此畫出字典樹

1-3 圖

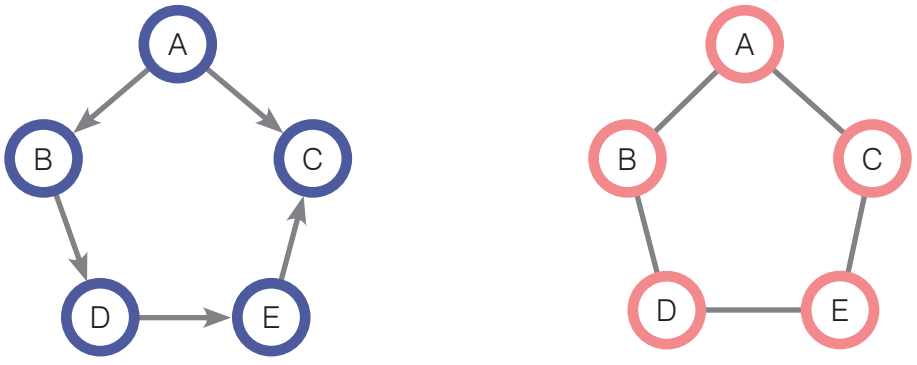
規劃旅遊行程時，常會將要去的景點標記出來，並用線條與數字標註出行程路線以及景點之間的距離（圖1-22）。這種以圖示來記錄事物與事物之間關係的方式，與本節將介紹的**圖**結構有類似之處，以下就來介紹圖的概念及應用。



▲ 圖1-22 用圖表示景點位置關係

1-3.1 圖的基本概念

圖（graph）常用來表示多個事物之間的關聯，它和樹一樣都由節點與邊所構成，圖的邊可以具有方向性，這種圖稱為「有向圖」，而沒有方向性的圖則稱為「無向圖」。圖的每個邊可以賦予一個相關的數值，這個數值稱為**權重**（weight）。圖1-23為有向圖、無向圖的示意圖。

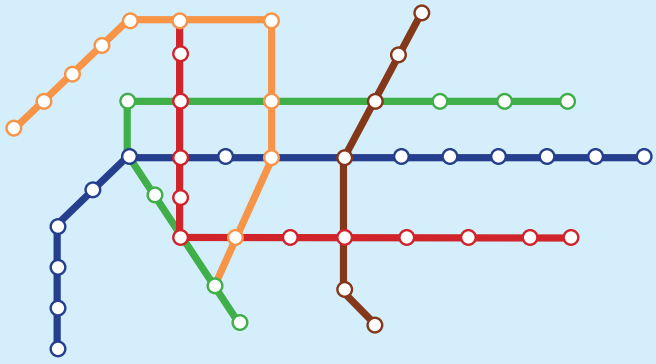


有向圖　　無向圖

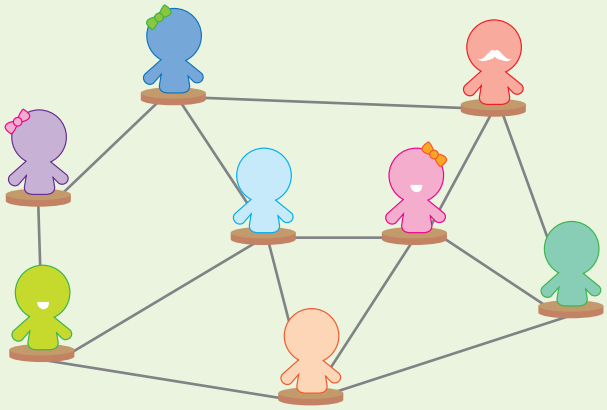
▲ 圖1-23 有向圖與無向圖

在日常生活中，有許多圖的應用，如圖1-24所示。藉由這些圖，我們可以更簡單明瞭地了解事物間的關係。

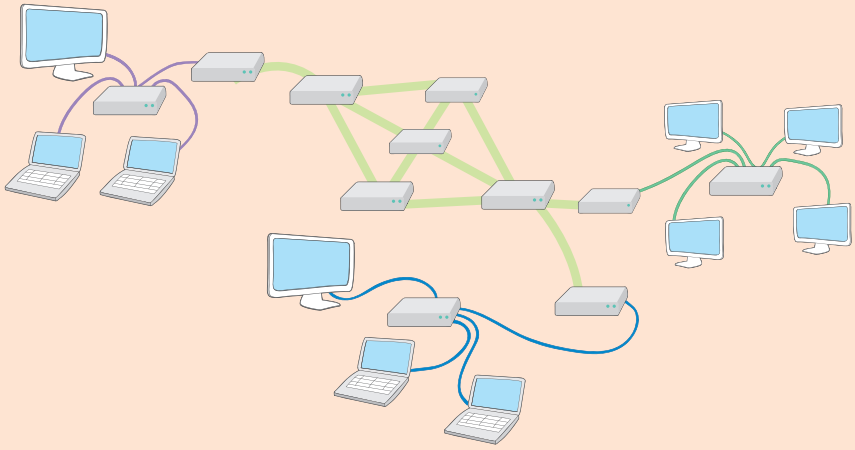
捷運路線圖　圖中的節點代表捷運站，邊代表捷運路線，有捷運路線連結代表站與站之間可以交通往來



人際關係圖　圖中的節點代表人，邊代表人際關係，有人際關係連結代表人與人之間互相認識



網路佈線圖　圖中的節點代表電腦與網路設備，邊代表網路線，有網路線連接代表電腦與設備之間可以透過網路傳輸資料



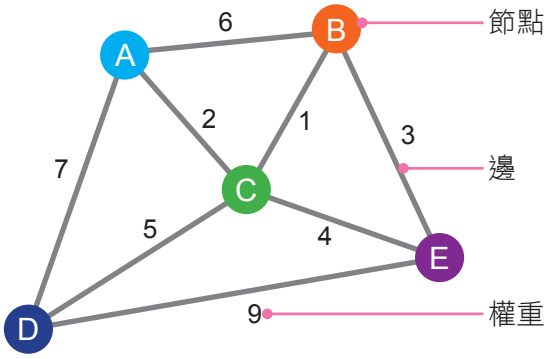
▲ 圖1-24　生活中「圖」的應用

1-3.2 圖常見的應用

「圖」結構的應用有很多，以下將介紹「建立最小生成樹」與「尋找不重複路徑」等2種常見的應用。

建立最小生成樹

我們常用「圖」來表示地圖上點與點之間的方位與距離關係，例如要規劃建設城市中的下水道管渠工程，我們可以像圖1-25一樣，將這幾個重要建築標示為「節點」，建築之間連接的管渠為「邊」，建設管渠所要花費的成本則為「權重」，來構成一個表示城市下水道管渠工程建設模型的「圖」。



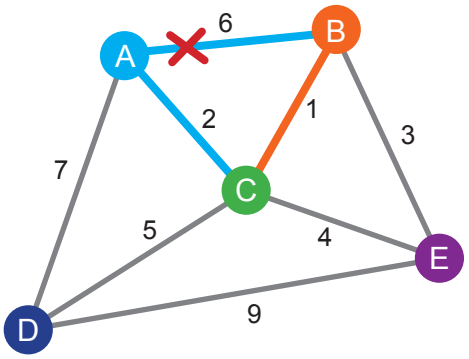
▲ 圖1-25 下水道管渠工程建設模型圖

繪製城市下水道管渠工程建設模型的「圖」後，如果想用最少的成本來建設連結全部建築的管渠，要如何規劃呢？**生成樹**是一棵包含圖上所有節點的樹，**最小生成樹**（minimum spanning tree）則是指含括圖中所有節點且權重值最小的生成樹，因此找出最小生成樹，即可找出用最少的成本連結全部建築的管渠線路組合。

產生最小生成樹常用的方法是每次都以目前最小的成本或最大的效益為原則來找出答案，這種方法稱為**貪心法**（greedy method）。

用貪心法選取連結所有建築的管渠線路組合時，必須遵循以下原則：

1. 在未選過的線路中，選取成本最低的線路。
2. 檢查此線路是否可連接到未連接上管渠的建築，若可則選取此線路，否則就放棄此線路。
3. 重複步驟1～2，直到所有建築皆連接上管渠，並且每個建築可透過管渠彼此連通。



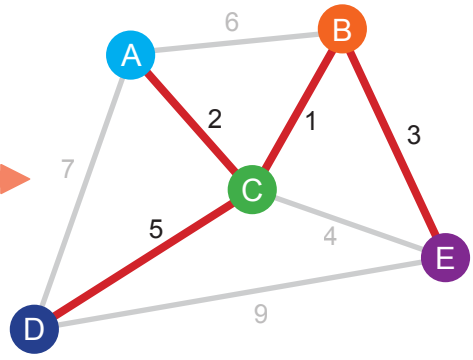
▲ 圖1-26 選取線路的原則

要注意，用貪心法選出的線路不會有只增加成本，但不會增加建築連結的線路。例如選取線路$\overline{BC}$、$\overline{AC}$之後，建築A、B、C已可連結在一起，在此情況下如果還選取線路$\overline{AB}$，就稱為「徒增成本」，所以貪心法在此情況下不會再選線路$\overline{AB}$（圖1-26）。

表1-1列出一種連結所有建築的管渠線路組合。

▼ 表1-1 找出連結所有景點的路段組合

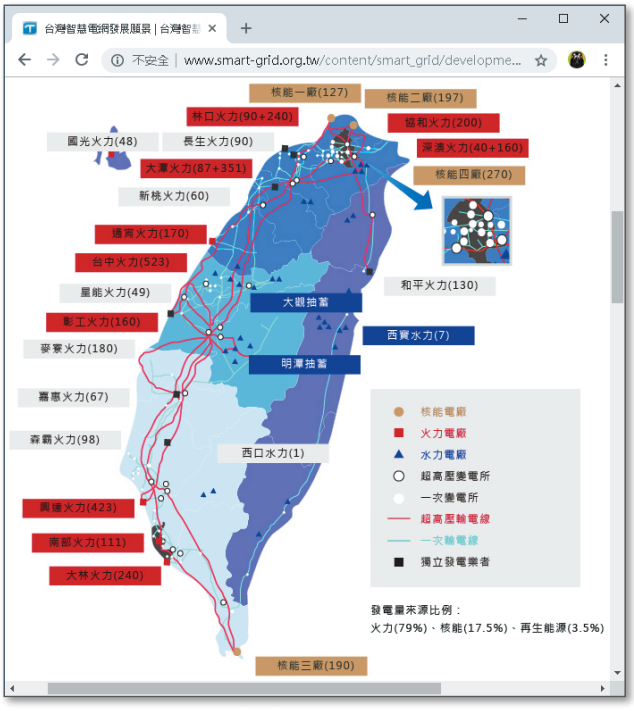
選取線路	成本	徒增成本	管渠線路組合
$\overline{BC}$	1	否 → 選取	{$\overline{BC}$}
$\overline{AC}$	2	否 → 選取	{$\overline{BC}$，$\overline{AC}$}
$\overline{BE}$	3	否 → 選取	{$\overline{BC}$，$\overline{AC}$，$\overline{BE}$}
$\overline{CE}$	4	是 → 不選	{$\overline{BC}$，$\overline{AC}$，$\overline{BE}$}
$\overline{CD}$	5	否 → 選取	{$\overline{BC}$，$\overline{AC}$，$\overline{BE}$，$\overline{CD}$}
$\overline{AB}$	6	是 → 不選	{$\overline{BC}$，$\overline{AC}$，$\overline{BE}$，$\overline{CD}$}
$\overline{AD}$	7	是 → 不選	{$\overline{BC}$，$\overline{AC}$，$\overline{BE}$，$\overline{CD}$}
$\overline{DE}$	9	是 → 不選	{$\overline{BC}$，$\overline{AC}$，$\overline{BE}$，$\overline{CD}$}



到此所有建築皆連接上管渠，並且每個建築可透過管渠彼此連通，可不必再往下判斷

日常生活中常見的GPS導航系統及Google Maps的路線規劃（圖1-27），或是電力公司的電力輸送線路（圖1-28），即會應用到「圖」的觀念。

▲ 圖1-27 路線規劃



（http://www.smart-grid.org.tw/）

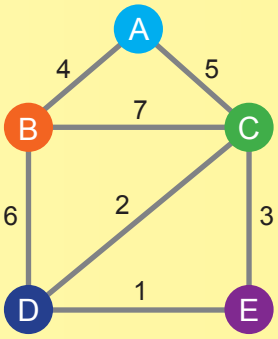
▲ 圖1-28 電力公司的電力輸送圖

馬上練習

1. 甲農場有A～E五塊田地，若要開通灌溉渠道使這五塊田地都可以引水入田，請選出成本最低的渠道路段組合。

   路段組合：_____(?)_____。

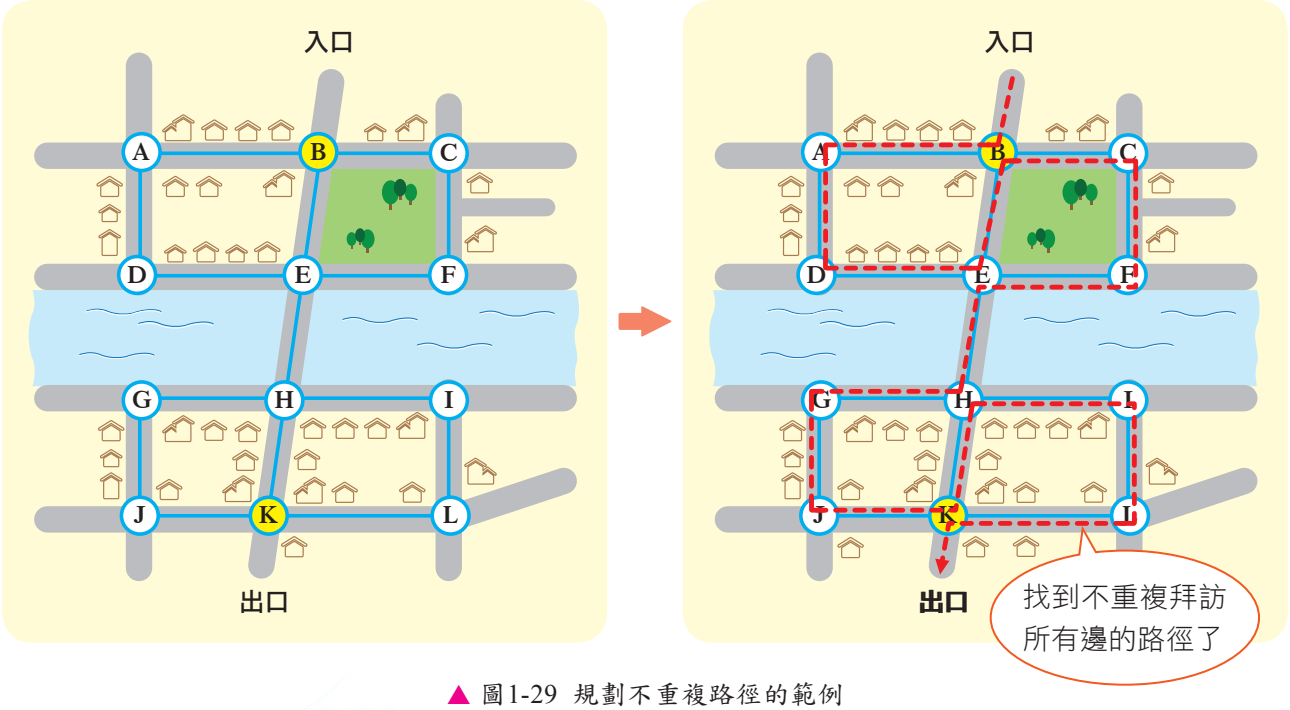
   距離：_____(?)_____。



尋找不重複路徑

生活中有許多要尋找不重複路徑的情況，例如郵差要沿街投遞郵件、遊客要逛遍整個市集街道、消費者在賣場要採購不同的商品等，如果能事先規劃出不重複的路線，以避免反覆行經相同道路，就可早一點達成目標。

我們可以先畫出「圖」，以便找出不重複拜訪所有邊的路徑。首先要將路線畫成「邊」與「節點」，例如圖1-29為2個市集的地圖，我們可用筆從入口開始畫出不重複拜訪所有邊的路徑，直到抵達出口為止。



▲ 圖1-29 規劃不重複路徑的範例

課外閱讀

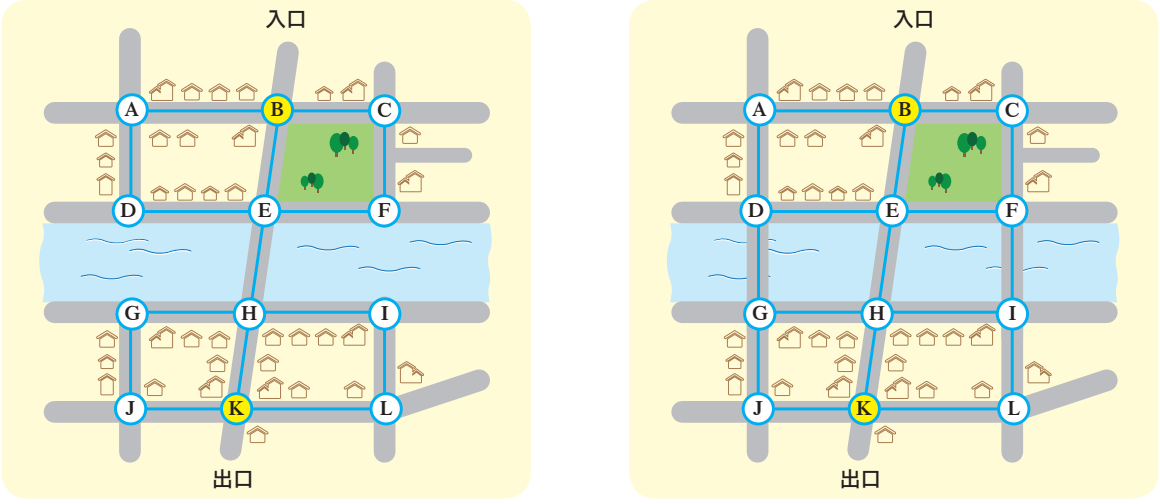
歐拉圖（Euler graph）

不是在每一個「圖」中都可以找出不重複的路徑，18世紀數學家李昂哈德・歐拉（Leonhard Euler）歸納出一種規則，可以快速找出「圖」中是否存在不重複拜訪所有邊的路徑：

1. 當每個節點皆有偶數個「邊」時，就可找出由原節點進出的不重複路徑。若在「圖」中找出由原節點進出的不重複路徑，則每個節點必皆有偶數個邊。
2. 若所有節點中有2個節點的「邊」為奇數個，其餘節點的「邊」為偶數個時，就可找出由不同節點進出的不重複路徑。若在「圖」中找出由不同節點進出的不重複路徑，則必有2個節點的「邊」為奇數個，其餘節點的「邊」為偶數個。

符合上述任一項條件的圖就可以找出不重複路徑，這種「圖」稱為「歐拉圖」（Euler graph）。以圖1-30(a)為例，此圖的A～L共12個節點中，只有B、K的邊為奇數（3）個，其餘皆為偶數個，符合條件2，所以此圖可找出不重複路徑。

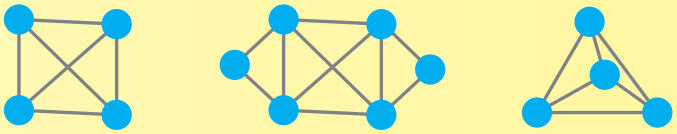
下圖1-30(b)中，A～L共12個節點，其中B、D、F、G、I、K共6個節點的邊為奇數個，不符合歐拉圖的2項條件，所以此圖沒有不重複路徑。



▲ 圖1-30　範例圖

馬上練習

1. 下列有三張圖，請判斷是否可找出不重複的路徑，並在空格處填入答案。



有偶數個邊的節點數量	？ 個	？ 個	？ 個
有奇數個邊的節點數量	？ 個	？ 個	？ 個
有無不重複路徑	？有　？無	？有　？無	？有　？無

本章習題

選擇題

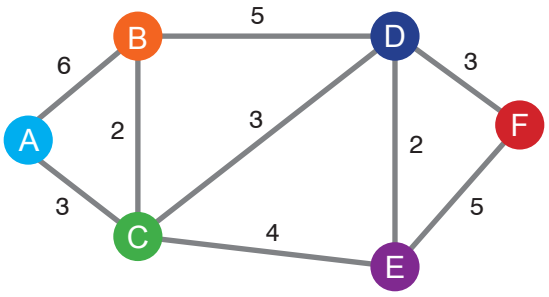
(?) 1. 下列何種資料結構的資料元素存取方式是先進先出（First In First Out）？
(A)樹 (B)圖 (C)堆疊 (D)佇列。

(?) 2. 在資料結構中，將每一層決策可能引發分歧的選項由上至下用「樹」的結構記錄下來，並在最底層歸類整理出決策結果，這種樹稱為
(A)決策樹 (B)二元樹 (C)多元樹 (D)字典樹。

(?) 3. 一個有31筆資料的二元搜尋樹，若每一層都剛好容納2個子節點，請問在此樹中最多要搜尋幾次可找到資料？ (A)5 (B)9 (C)30 (D)31。

(?) 4. 每一次做選擇時，都以目前最小的成本或最大的效益為原則，這種方法稱為
(A)貪心法 (B)暴力法 (C)黑箱測試法 (D)各個擊破法。

(?) 5. 請問下圖中可以連結所有節點的最低成本為？ (A)8 (B)11 (C)13 (D)18。



多元練習題

1. 山上有個村莊住了6戶人家（A～F），假設要拉電線讓這6戶人家都有電可用，每戶之間拉電線所需花用的興建成本如右圖所示，請同學找出花費成本最低的連接方式及其興建成本。

連接方式：_____(?)_____。

成本：_____(?)_____。