1.1 演算法概念

使用電腦的目的是要解決問題，為了解決問題，電腦要能依照給定的步驟執行，這些執行的步驟就是演算法 (algorithm) 的概念。演算法是資訊科學很重要的主題，各個領域幾乎都會用到。

甚麼是演算法？演算法是解決問題的方法，是指完成一件任務的步驟和方法，例如：決定上學的路線；交通號誌的使用規則；依照天氣狀況穿衣服；參考食譜準備餐點 (圖 1-1)；整理房間；依照樂譜演奏樂器等，都是執行演算法的一種。

圖 1-1 參考食譜製作餐點是演算法的一種

使用電腦就一定會用到演算法，例如：將音樂轉成 MP3 檔案，會用到聲音格式轉換的演算法；上網搜尋資料，會用到資料搜尋的演算法；壓縮檔案，會用到資料壓縮的演算法；收發電子郵件，會用到許多與網路相關的演算法。

在資訊科學，**演算法**是**電腦解題**的思考流程。電腦解題時，可以先設計好解題的演算法，再將演算法轉換成相應的程式。演算法是由許多步驟所組成，這些步驟具有以下特性：

1. **輸入** (input)
   要有零個或零個以上的輸入值。
2. **輸出** (output)
   經過演算法處理後，至少會有一個輸出結果。
3. **明確性** (definiteness)
   每一步驟都必須明確，不可含糊混淆，不同人解讀，都會得到相同的結果。
4. **有限性** (finiteness)
   必須在有限的步驟或時間內結束。
5. **有效性** (effectiveness)
   步驟必須是有效且具體可行的。

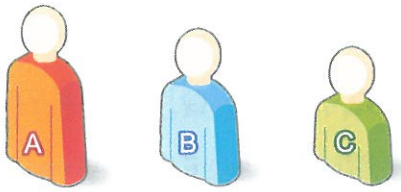
因此我們可以將**演算法**定義成：

一個演算法是明確、有效、最終會結束的可執行步驟。

解決問題的方法可能會有多種不同的方法，設計出來的演算法，也可能不同，所以同一個問題的解題方法，可能有多種不同的演算法。例如：按身高排隊，可以使用以下兩種方法，這兩種方法不同，但可以得到相同的結果 (圖 1-2)。

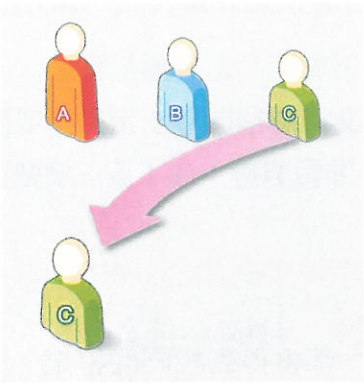
方法 1：每次從尚未排好隊的人中，選出最高或最矮的出來排。

方法 2：相鄰的兩個人比較高矮，如果高的在前面，則兩人交換位置。反複此動作，直到按高矮次序排好為止。

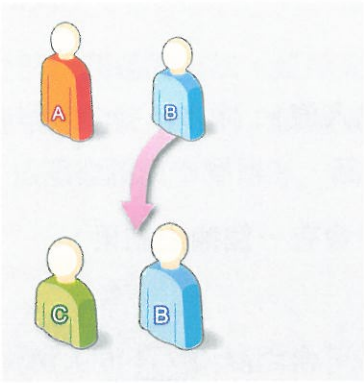


將 A, B, C 三位，由矮至高排列。

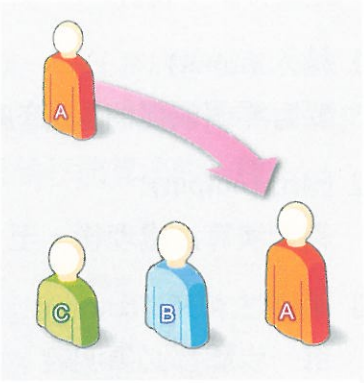
3人排高矮



1 挑選 A, B, C 中最矮的 C。

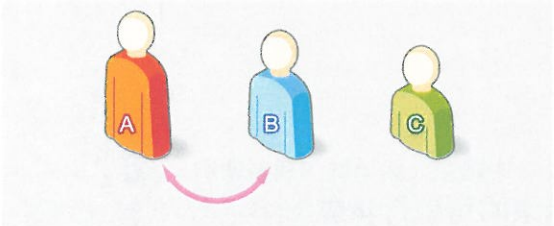


2 挑選剩餘的 A, B 中最矮的 B。

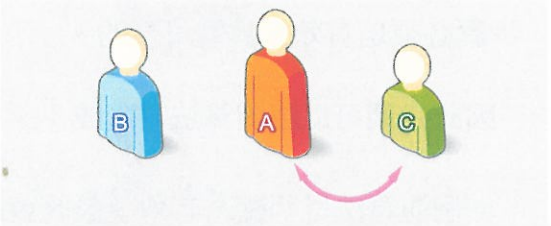


3 將最後剩下的 A 加入。

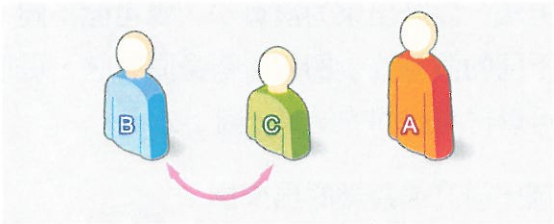
(A) 排序方法 1



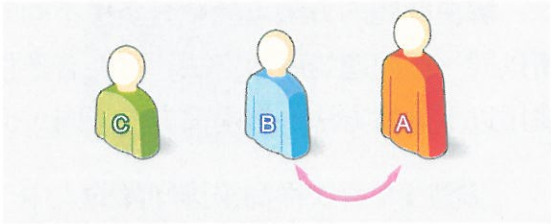
1 比較 A, B，A 比 B 高，所以 A, B 位置交換。



2 比較 A, C，A 比 C 高，所以 A, C 位置交換。



3 比較 B, C，B 比 C 高，所以 B, C 位置交換。



4 比較 B, A，B 比 A 矮，所以次序維持不變，排序完成。

(B) 排序方法 2

圖 1-2 同一問題兩種不同的演算法

1.2 演算法表示方法

演算法的表示方法是指透過文字或符號，將電腦解題的步驟、邏輯判斷或流程表示出來。演算法的表示並沒有一定的方法，但要能清楚的表達解題的流程。就像傳達故事一樣，可以用說的，或用文字及圖畫表達，方式雖然不同，但故事的內容都是相同。演算法的表示方法可使用文字、流程圖 (flowchart)、虛擬碼 (pseudo code) 其中一種。

一、文字

以文字敘述的方式來表示演算法，例如：攝氏溫度 C 與華氏溫度 F 轉換的演算法，可使用以下任一種方式表示：

F = (9 / 5) C + 32	華氏溫度是將攝氏溫度乘以 9 / 5，再加上 32。

因為使用接近自然語言的文字來表示演算法，所以和人類的使用習慣相近，但若對文字認知不同，容易造成誤解。自然語言種類繁多，不同使用者常會發生認知差異，因此使用文字表示演算法，會比較不精確。

二、流程圖

所謂「一圖勝過千言萬語」，某些情況下，使用圖形表示會比文字敘述更容易讓人理解。流程圖是將演算法的步驟，使用特定的圖形表示出來。為了方便流通與閱讀，流程圖的繪製有一定的規範，例如：美國國家標準協會就規定有流程圖的符號和繪製方法 (圖 1-3)。

流程圖符號	名　稱	意　義	範　例
	開始 結束	表示流程的開始 或結束	開始 結束
	流向	表示流程進行 的方向	
	輸入 輸出	表示資料的輸入 或結果的輸出	輸出總和
	處理程序	表示執行或處理 某一項工作	a+1
	決策判斷	針對某一條件進行 判斷	A > B
	迴圈	表示迴圈控制變數 的初始值或終值	i 由 1 到 n i
	副程式	用以表示一群已經 定義流程的組合	查詢密碼
	報表	列印出的報表文件	印出成績單

圖 1-3 流程圖的符號

設計程式最好使用結構化程式設計，也就是只使用循序、選擇、重複三種基本結構。以下是這三種結構的流程圖。

一、循序結構

按照敘述出現的順序，一步一步循序地執行。例如：攝氏溫度 C 與華氏溫度 F 轉換的流程圖可繪製如下 (圖 1-4)：

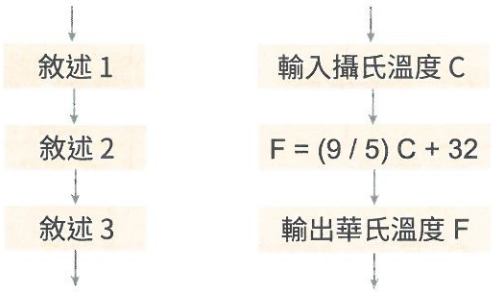


圖 1-4 循序結構的流程圖

二、選擇結構

選擇結構會根據條件式判斷的結果，選擇執行不同的程式碼。選擇結構一般可分為以下幾類：

1. 單一選擇結構

單一選擇結構是如果條件式成立 (true)，就執行敘述，否則 (false) 不執行，所以單一選擇結構可以當成是「如果 … 就 …」。例如：「如果成績低於 60 分時，就顯示不及格。」的流程圖如下 (圖 1-5)。

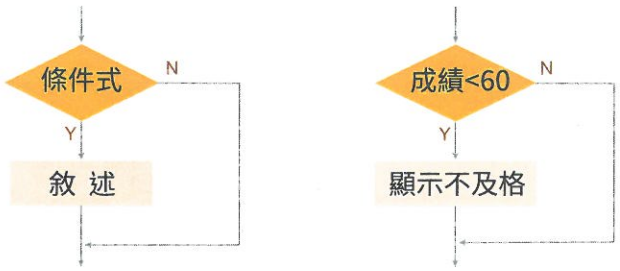


圖 1-5 單一選擇結構的流程圖

2. 雙重選擇結構

雙重選擇結構是如果條件式成立 (true)，就執行敘述 1；否則 (false) 執行敘述 2，所以雙重選擇結構可以當成是「如果 …，就 …，否則 …」。例如：「如果下雨，就在家玩遊戲，否則出去騎車。」的流程圖如下 (圖 1-6)：

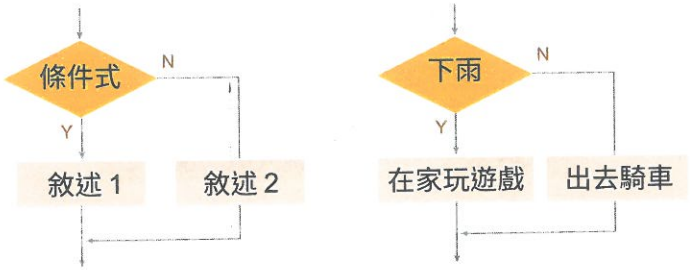


圖 1-6 雙重選擇結構的流程圖

3. 多重選擇結構

多重選擇結構是依序判斷多個條件式，那個條件式成立，就執行此條件式內的敘述，最後若都不成立，就執行最後的敘述。例如：購物打折問題「如果購物金額滿 3000，打 8 折；滿 2000，打 9 折；滿 1000，打 95 折；否則不打折。」的流程圖如下 (圖 1-7)。

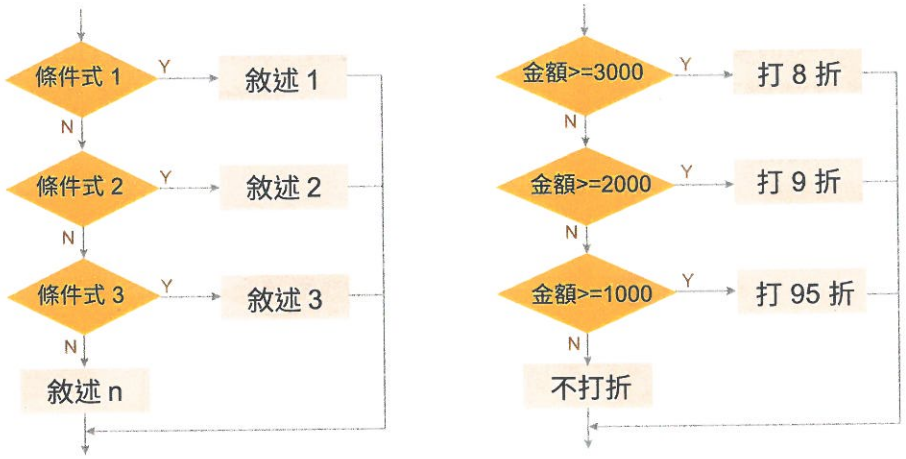


圖 1-7 多重選擇結構的流程圖

三、重複結構

重複結構是指重複執行某些敘述，直到滿足特定條件為止。重複結構先檢查條件式，如果條件式為真，就執行敘述，執行完後，再回去檢查條件式，如此反複執行；若不滿足條件式，就會跳到重複結構外的下一個敘述，繼續執行。例如：「購買車票時，如果還有車票，就可以反複賣票，直到沒有車票為止。」的流程圖如右 (圖 1-8)。

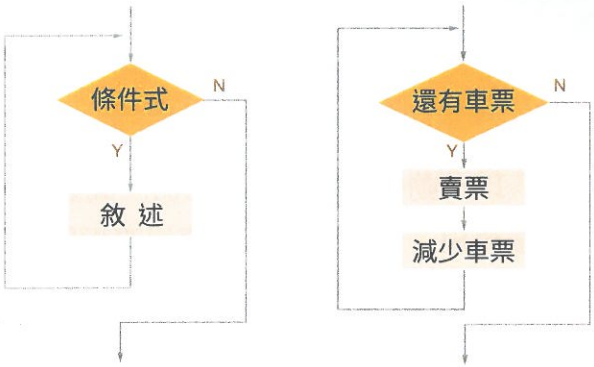


圖 1-8 重複結構的流程圖

日常生活許多演算法都可以使用流程圖表示，例如：大富翁遊戲的簡易規則，就可使用流程圖表示如下 (圖 1-9)：

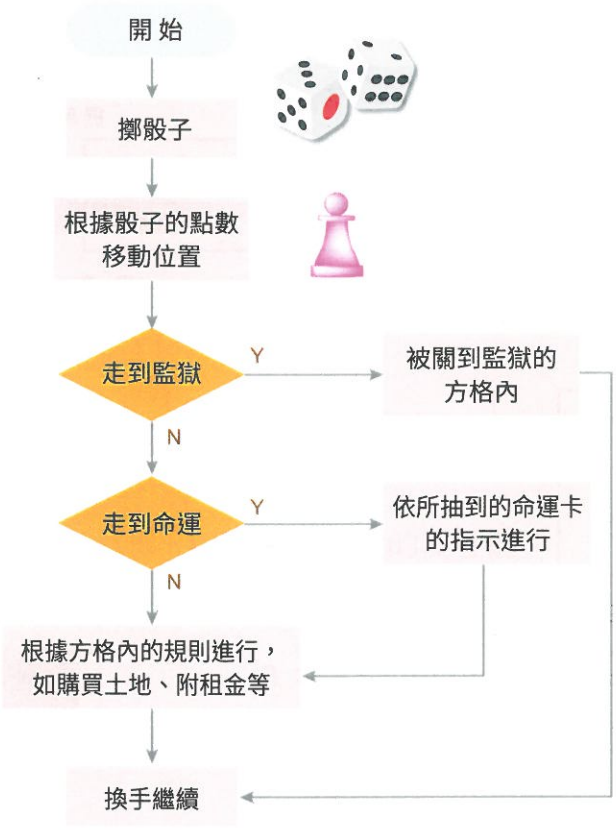


圖 1-9 大富翁遊戲及其演算法

較複雜的演算法可切成幾個模組來表示，先用較大模組的流程圖來表示演算法，再將各大模組細分成多個小模組，每個小模組再使用各別的流程圖描述。就像一幢大樓的設計圖，不會只用一張藍圖來表示，而是會將大樓分成外觀、樓層、甚至各房間的藍圖來呈現。

以自動櫃員機 ATM 為例，簡易流程圖如下，其中「更改密碼」與「提款」兩個模組，可分別另外行使用流程圖說明 (圖 1-10)。

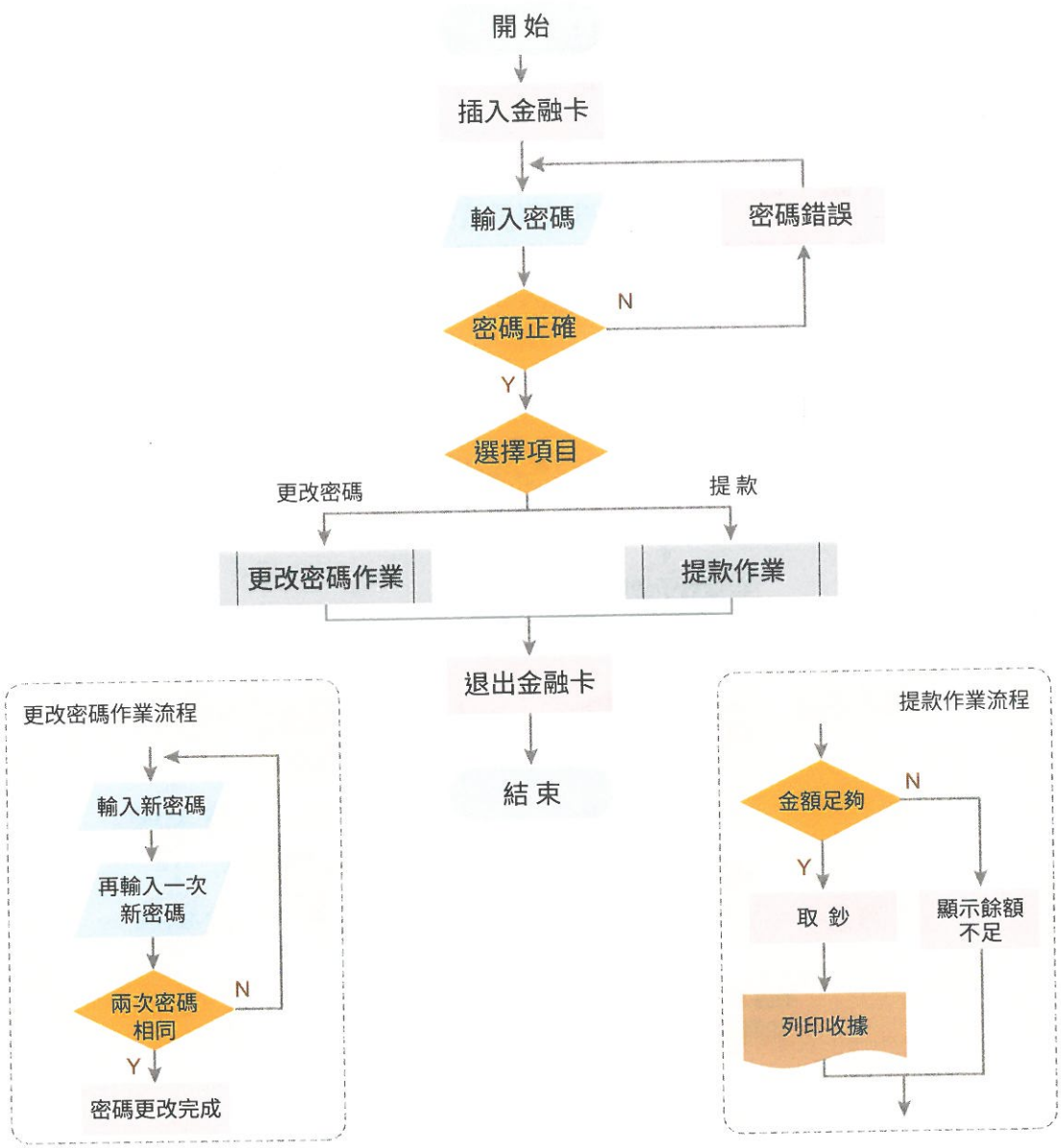


圖 1-10 ATM 的簡易流程圖

流程圖具有許多優點，例如：方便他人了解流程，有利分工合作；有助於修改與維護等，但如果流程圖太複雜，代表流向的箭頭符號太多交錯時，反而會使所表示的演算法不易閱讀。

繪製流程圖可使用文書處理或簡報軟體內建的繪圖功能，也可選用專業的繪製軟體，例如：自由軟體 Dia 或其他線上軟體，如 https://www.draw.io 等 (圖 1-11)。

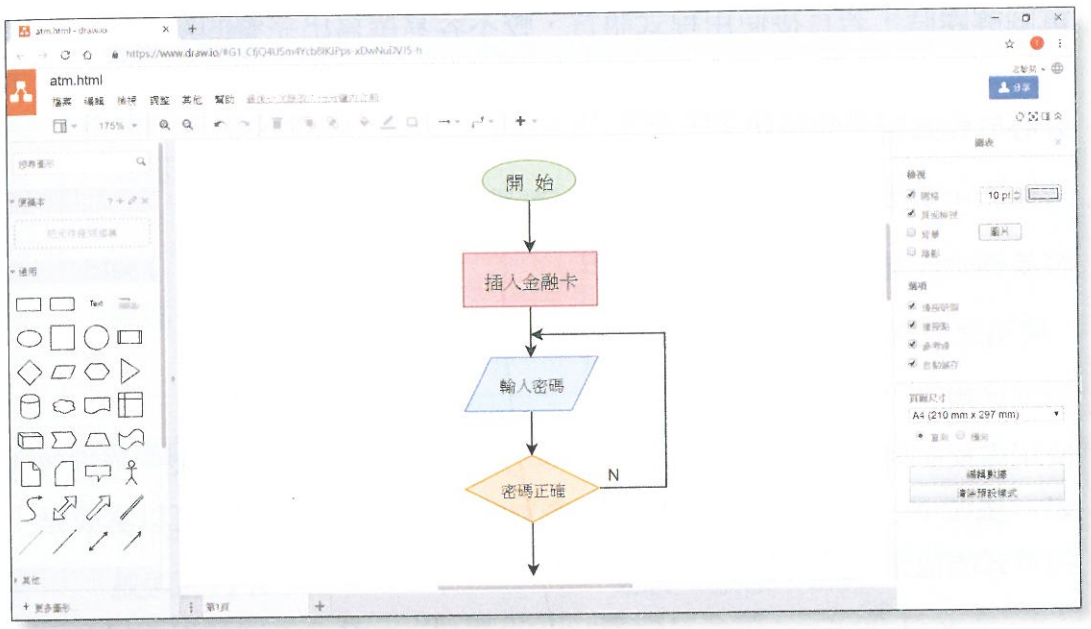


圖 1-11 線上繪製流程圖的軟體

課堂練習

1. 請畫出一個能判斷整數 n 是奇數或偶數的流程圖。
2. 請將學生成績五等第畫分的演算法，使用流程圖的方式表示。

   A：90 分以上　　B：80 分以上未滿 90 分

   C：70 分以上未滿 80 分　　D：60 分以上未滿 70 分

   E：未滿 60 分

三、虛擬碼

虛擬碼並不是程式語言，也無法直接在電腦上執行，它是介於自然語言與程式語言的敘述，使用非正式語法來描述演算法，讓設計者能把演算法設計成程式。

虛擬碼沒有一定的語法標準，通常是由使用者以特定的關鍵字、敘述或運算式來表示。在電腦解題時，可以在提出解題概念後，將解題的演算法，使用虛擬碼記錄下來，再依內容發展成電腦程式。使用虛擬碼的目的包含：

1. 電腦解題時，若直接使用程式語言，較不容易撰寫出完整的解題架構，且較缺乏直覺性。使用虛擬碼，可以讓設計者專注於問題解決的處理過程，不用分心於程式語言的語法。
2. 方便表示演算法，敘述清晰，容易了解整體解題架構。
3. 容易轉化成電腦程式。

使用虛擬碼表示循序、選擇、重複三種基本結構的方式如下：

1. 循序結構

使用虛擬碼時，常會使用指定符號「＝」，將運算結果儲存起來，指定敘述的表示法如下，其中變數是用來表示資料的名稱，以下表示式可以當成是「將運算式的值指定給名稱」。

變數 ＝ 運算式

循序結構的虛擬碼表示法與實例如下：

敘述 1 敘述 2 敘述 3	a = 5 a = a + 1

(1) a = 5

表示將值 5 指定給變數 a。

(2) a = a + 1

表示將 a 原來的值加 1 後，再指定給變數 a。

原來 a = 5，執行 a = a + 1 時，會先執行 5 + 1 = 6，然後再執行 a = 6，所以執行後 a 的值就會是 6。

2. 選擇結構

單一選擇結構、雙重選擇結構、多重選擇結構的虛擬碼表示法與實例如下：

(1) 單一選擇結構

If 條件式 Then (　敘述)	If 成績 < 60 Then (　顯示不及格)

(2) 雙重選擇結構

If 條件式 Then (　敘述 1) Else (　敘述 2)	If 下雨 Then (　在家玩遊戲) Else (　出門騎車)

(3) 多重選擇結構

If 條件式 1 Then (　敘述區段 A) Else If 條件式 2 Then (　敘述區段 B) Else If 條件式 3 Then (　敘述區段 C) Else (　敘述區段 Z)	If 金額 >= 3000 Then (　金額 = 金額 * 0.8) Else If 金額 >= 2000 Then (　金額 = 金額 * 0.9) Else If 金額 >= 1000 Then (　金額 = 金額 * 0.95) Else (　金額 = 金額 * 1)

3. 重複結構

重複結構的虛擬碼表示法與實例如下：

While 條件式 Do (　敘述)	While 還有車票 Do (　賣車票 　原來的車票數 - 賣出的車票數)

1.3 演算法效能分析

一個問題可能會有多種解題的演算法，要如何選擇其中一種？除了正確性，也可以分析演算法的效能後，採用效能較高者。

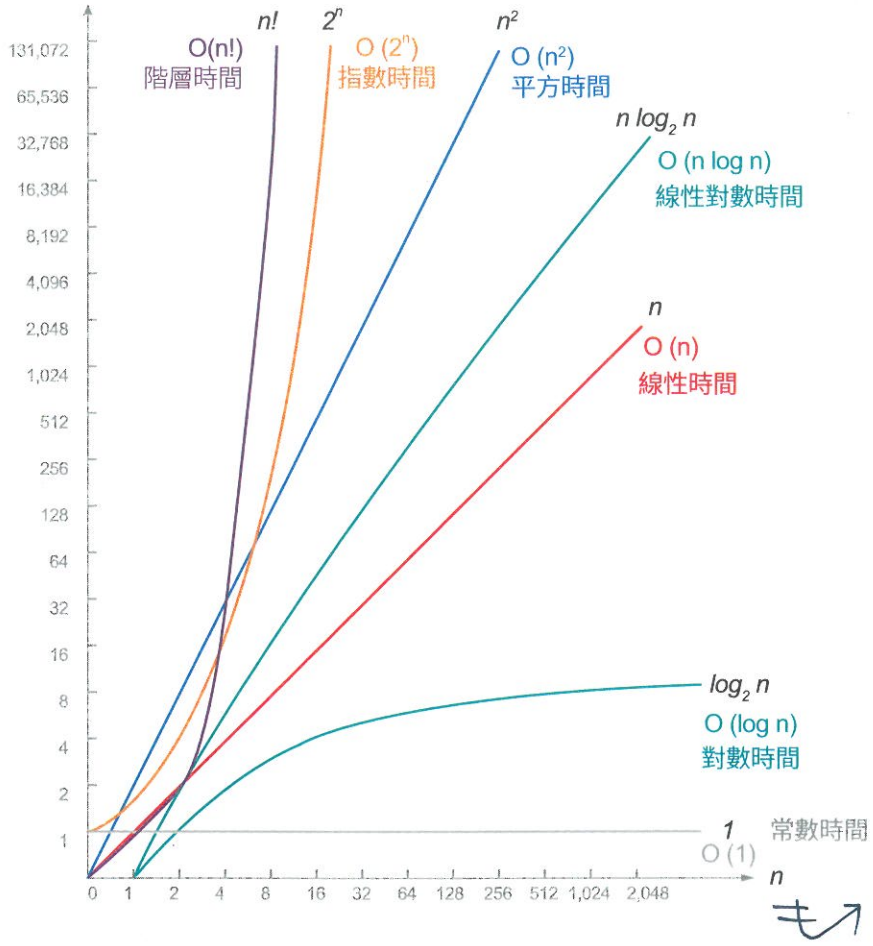
演算法的優劣和問題解決的品質息息相關，例如：若搜尋引擎使用高效能、且更精準的資料搜尋演算法，就能更快速、更正確的搜尋到資料；使用更嚴密的加密演算法，更能保護資料在網路傳輸的安全性；更有效率的排程演算法，能為貨運公司設計出最省錢的交通路徑。

演算法分析最重要的分析因素就是執行所需的時間，也就是在不同的輸入大小時，計算該演算法指令執行的次數，當輸入資料的情形不同時，演算法的執行次數可能也會不同。通常可分成以下三種情形分析：

1. 最差情形：演算法執行基本運算次數的最大值。
2. 平均情況：演算法執行基本運算次數的平均值。
3. 最佳情況：演算法執行基本運算次數的最小值。

分析演算法需要一個衡量各種方法的指標，通常會使用 **Big O** 來表示演算法指令執行的次數，作為演算法時間複雜度或效能的依據。Big O 取執行次數中最高次方或最大指數即可。常用的 Big O 如以下幾種 (圖 1-12)：

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$$



n	log n	n	n log n	n^2	2^n	n!
8	3	8	24	64	256	40320
16	4	16	64	256	65536	$> 10^{13}$
32	5	32	160	1024	$> 10^{9}$	$> 10^{35}$
64	6	64	384	4096	$> 10^{19}$	$> 10^{89}$

圖 1-12 演算法的時間複雜度

演算法的時間複雜度中，O(1) 是指演算法的指令執行次數固定，和資料量無關，例如：溫度單位轉換的演算法等；O(log n) 是和資料量 n 的對數成正比，例如：資料量 n = 32 (2^5) 時，O(log n) 指令執行次數為 5 次，原因如下：

$$\log_2 n = \log_2 32 = \log_2 2^5 = 5$$

O(n) 演算法的指令執行次數和資料量 n 成正比；O(n^2) 則是和資料量平方成正比。

演算法的時間複雜度中，觀察 O(2^n) 和 O(n!) 可以發現，即使 n 值不大，例如：n = 64 時，若電腦每秒可運算千兆 (10^{15}) 次，指令執行所需的時間如下：

$$O(n!) > 10^{89}，需\ 10^{89} / 10^{15} = 10^{74}\ 秒 > 10^{66}\ 年$$

宇宙的年齡約 137 億 (1.37×10^{10}) 年，所需的指令執行次數非常巨大，碰到這類問題，需研究如何提升演算法的效能。

1.3.1 活動　演算法效能分析 — 質數問題

質數是大於 1，且除了 1 和本身外，沒有其他因數的正整數。若要檢查某一整數 n (n > 2) 是否為質數，請寫出檢查的演算法，分析其效能後，思考有無提升效能的方法。

問題解析

1. 檢查某整數 n (n > 2) 是否為質數，最直覺的方法是檢查 2 ~ n-1 的所有整數是否可整除 n，例如：要檢查 23 是否是質數，可以檢查 2 ~ 22 的所有整數是否有其中一數可以整除 23。

2. 檢查某數 n 是否為質數的演算法可設計如下：

演算法

1. 檢查 2 ~ n-1 的所有整數是否可整除 n，
2. 若其中一個數可以整除 n，則 n 不是質數，
3. 若全部都不能整除 n，則 n 是質數。

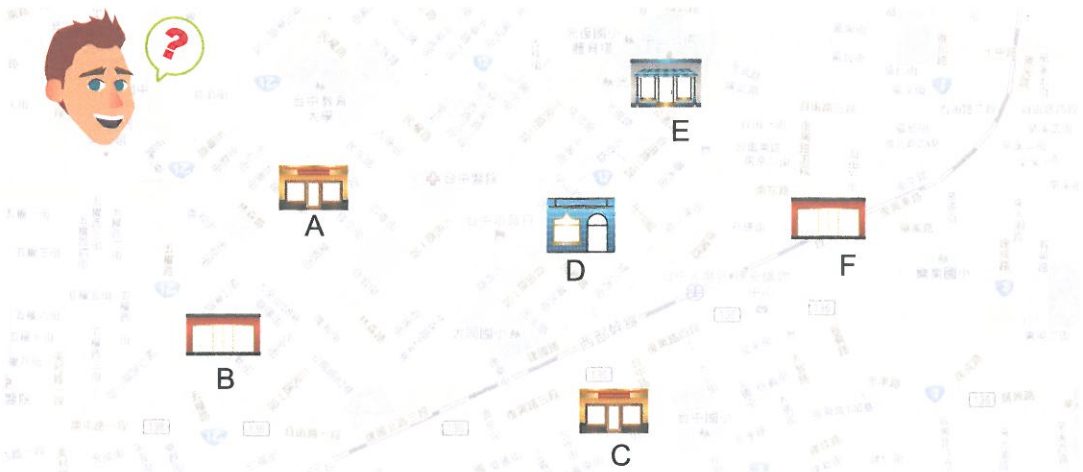
3. 檢查次數和 n 值的大小成正比，所以時間複雜度為 O(n)。
4. 某數的因數會成對出現，例如：18 的因數有 (2, 9)、(3, 6)、(6, 3)、(9, 2)，檢查一個因數，就等同檢查另一個因數，如上例，檢查 2 等同檢查 9；檢查 3 等同檢查 6。因此成對的因數中，一個 $\leq \sqrt{n}$ ，另一個 $\geq \sqrt{n}$ ，所以只要檢查 $2 \sim \sqrt{n}$ 的所有整數，是否是 n 的因數即可。
5. 此方法檢查次數和 $\sqrt{n}$ 成正比，所以時間複雜度為 O(log n)。

問題與討論

請將以上兩個演算法使用流程圖表示。

1.3.2 活動 演算法效能分析 — 行程問題

有一遊戲的規則是，誰能經過 8 個不同的點，且每一點剛好經過一次，最後最早回到起點者優勝。請寫出找到一條最短路徑的演算法，並分析此演算法的時間複雜度。



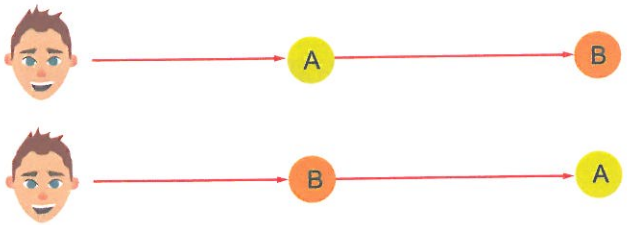
問題解析

1. 要找出一條「經過每一點一次，最後回到起點」的演算法，可以一一列出所有可能的路徑，然後算出每一條路徑長，最後再找出其中最短者。

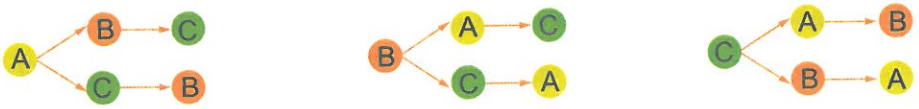
2. 可先考慮簡單的路徑，再逐步考慮到複雜的情形：

(1) 1 個點：1 種路線。

(2) 2 個點：有 2 種路線。



(3) 3 個點：有 3 × 2 = 6 種路線。



(4) 8 個點：有 8 × 7 × 6 × 5 × 4 × 3 × 2 × 1 = 40320 種路線。

3. 依此類推，n 個點共有 n× (n–1)× (n–2)× ·····× 3× 2 = n! 種路線。

所以此演算法的時間複雜度為 O(n!)

問題與討論

請和同學討論，有什麼方法可以提升上述問題之演算法的效能。

學習挑戰

一、單選題

( ) 1. 表達解決問題的方法中，下列何者是以圖示符號或文字敘述來表達各步驟執行的先後順序？
(A) 函數 (B) 副程式 (C) 程式 (D) 演算法

( ) 2. 下列何者不是演算法應具備的特性？
(A) 至少要有一個輸入 (B) 至少要有一個輸出
(C) 每一個處理動作都必須明確 (D) 處理動作必須是有效且具體可行

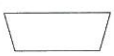
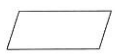
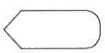
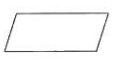
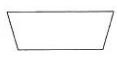
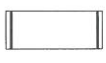
( ) 3. 下列問題何者具有正確的演算法？
(A) 找出最大的整數
(B) 找出 3 的平方根至完全正確
(C) 風速如果超過 8 級，就不要出門
(D) 如果天氣熱，就吹冷氣

( ) 4. 下列何者不是演算法的基本結構？
(A) 循序 (B) 跳躍 (C) 選擇 (D) 重複

( ) 5. 演算法的表達方法中，下列何者是使用一種介於自然語言與程式語言之間的表示法？
(A) 二進碼 (B) 對照碼 (C) 機器碼 (D) 虛擬碼

( ) 6. 下列何者不屬於常用的演算法表示法？
(A) 數學表示法 (B) 文字 (C) 虛擬碼 (D) 流程圖

( ) 7. 依據美國國家標準協會編製的標準化流程圖符號，下列何者名稱與其符號不相符？
(A) 處理 
(B) 判斷 (決策) 
(C) 列印 
(D) 副程式 

( ) 8. 在流程圖中，「資料輸入或輸出」與「報表輸出」分別以何種圖示表示？
(A)  與 
(B)  與 
(C)  與 
(D)  與 

學習挑戰

( ) 9. 下列那一個符號常被作為演算法的指定敘述使用？
(A) == (B) = (C) & (D) <<

( ) 10. 使用虛擬碼表示演算法中，敘述 a = 5，a = a - 1 執行後，a 的值為？
(A) 4 (B) 5 (C) 6 (D)0

( ) 11. 有一虛擬碼如右，初始金額為 500，執行完演算法後，金額會變為多少？

(A) 400
(B) 450
(C) 475
(D) 490

```
If 金額 >= 3000 Then (
  金額 = 金額 * 0.8
) Else If 金額 >= 2000 Then (
  金額 = 金額 * 0.9
) Else If 金額 >= 1000 Then (
  金額 = 金額 * 0.95
) Else (
  金額 = 金額 * 0.98
)
```

( ) 12. 有一虛擬碼如右，若晴天且溫度為 30 度時，會執行那一項動作？

(A) 游泳
(B) 打籃球
(C) 看電影
(D) 甚麼都不做

```
If 下雨 Then (
  If 功課寫完 Then (
    看電影
  )
  Else (
    繼續寫功課
  )
) Else (
  If 溫度 >28 Then (
    游泳
  )
  Else (
    打籃球
  )
)
```

( ) 13. 計算演算法執行的基本運算次數時，通常不包含那種情形？
(A) 最差情形 (B) 最佳情況 (C) 平均情況 (D) 隨機情況

( ) 14. 關於下面虛擬碼的敘述，何者正確？

```
count = 0
While count ≠ 3  Do
    count = count + 2
```

(A) 這是一個演算　(B) 不會停止執行

(C) 執行效率很好　(D) 步驟的描述不明確

( ) 15. 若 n > 8，下列演算法的時間處雜度中，何者需要的時間最久？

(A) O (n log n)　(B) O (n^2)　(C) O (2^n)　(D) O (n!)

( ) 16. 要檢查 101 是否為質數，需要檢查 2 ~ n 是否能整除 101。n 的最小值為下列那一個數？

(A) 100　(B) 101　(C) 10　(D) 11

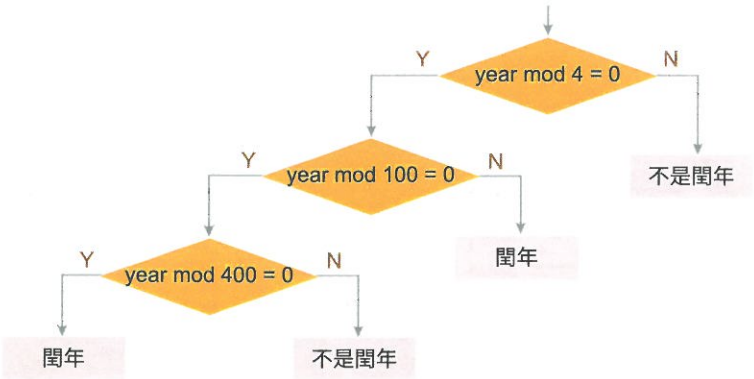
二、應用題

1. 下圖為判斷閏年的流程圖，其中 a Mod b 為 a 除以 b 的餘數，year 為西元年，請完成以下任務：

(1) 標示下面條件下的執行路徑：

a. 1996 年　b. 2020 年　c. 2000 年

(2) 將此流程圖使用虛擬碼表示。



2. 請和同學合作，畫出提款機操作的流程圖。