淺談「計算機演算法」

許介彥
大葉大學　通訊與計算機工程學系

前言

對剛接觸電腦理論部分的人來說，「計算機演算法」（computer algorithm）可能是一個常常看到但意思卻不甚清楚的名詞，由於幾乎可以算是電腦的專有名詞，因此即使查了一般的字典（甚至電腦字典）也不容易對它有一個清楚的認識；本文將對此名詞作一概略性的介紹。

什麼是「演算法」？

作曲家透過音符來傳達樂思，畫家透過畫筆勾勒出心中的影像，作家透過文字描述所見所感，而對某些電腦科學家來說，「演算法」正是他們表達思想的媒介。

電腦是執行程式的機器，程式設計者透過撰寫程式來指揮電腦，來與電腦溝通；如同世界上人與人的溝通可藉由各種不同的語言（中文、英文、法文...等），人跟電腦溝通所用的「程式語言」（programming language）也有很多的選擇，如較常見的 C、PASCAL、FORTRAN、JAVA 等，至於要選擇什麼語言則依個人的喜好，同時還要考量所用的電腦系統的環境、程式所要完成的工作性質等因素而定；就如同意思相同的一句話在世界上不同地方的不同語言裡有不同的講法，同樣一件工作也常可藉由不同的程式語言所寫的程式來完成。

針對某個程式語言所寫的程式必須符合該語言對語法（syntax）及語意（symantics）的要求，一般的程式語言對語法都有很嚴格的規定，即使在程式裡犯了很小的語法錯誤（如：漏掉一個標點符號）都可能讓程式無法執行或是得到意料之外的結果；因此當程式設計者想利用電腦來解決一個問題時，除了先要想好用電腦解決該問題的步驟以外，在將所需的步驟實際寫成程式時，須特別注意所選擇的程式語言對語法的要求。

計算機演算法可簡單定義爲「透過電腦的計算以完成某項工作的步驟」，也就是一個程式設計者想利用電腦解決某個問題，在實際寫程式前，心中對如何以電腦解決該問題所存的想法；當有需要要將該想法表達出來讓別人了解時，除了真的寫成程式外，也可以訴諸文字描述，或借助圖形，或用其他的表達方式，只要能將心中的想法清楚而完整地表達出來即可。程式溝通的對象是電腦，程式是讓電腦執行的，而演算法溝通的對象是人，演

算法是讓人看的，因此演算法的表達沒有特定程式語言對語法的諸多限制，只要能讓別人了解就行了。

雖然與程式語言的語法相比，演算法的表達方式較自由，但畢竟演算法是人類爲電腦設身處地所想出來的一些動作與步驟，最終很可能是要將演算法轉換成程式由電腦來執行，因此雖然表達方式較不受限制，還是有許多人喜歡用較接近程式的表達方式來敘述演算法，以便將來由演算法轉換成實際的程式的工作做起來可以較順利與直接；因此，許多常見的程式語言裡提供的敘述（statement）都可以用來表達演算法，只是使用上不須太在意語法的細節；所以一般語言裡常見的迴圈方面的敘述，如：

for ... to ...

repeat ... until ...

while ...

等，或條件判斷方面的敘述，如：

if ... then ...

if ... then ... else ...

switch ...

等，都常被用來表達演算法；另外，當然少不了副程式、輸出輸入、設值（assignment）方面的敘述，甚至也可以使用目前大部份較流行的程式語言都有提供的遞迴函數（recursive function）的呼叫；由於沒有統一的語法規定，因此可看成每個人是用一種自己「虛擬」的程式語言來說明其演算法；畢竟人比電腦聰明，人與人的溝通要比人跟電腦的溝通在方式上有更大的彈性。

實際的例子

以下，讓我們用一個例子來說明演算法可能的表達方式：

```
WALLPAPER(side)
    for i ← 1 to 300
        for j ← 1 to 300
            x ← i × side / 100
            y ← j × side / 100
            c ← ⌊ x² + y² ⌋
            if c is odd
                plot(i, j)
```

第一列將此演算法的名稱（就如同一般程式語言中副程式的名稱）定爲 WALLPAPER，它接受一個參數 *side*，內部則是利用兩層 **for** 迴圈進行一些計算並在螢幕上一個大小爲 300×300 的區域內以虛擬的 *plot* 函數描出一些點，此處假設 *plot*(i, j) 可在螢幕上座標爲 (i, j) 的位置畫出一個點。此演算法如果實際被轉換成程式，將可依參數 *side* 的不同在螢幕上畫出許多亂中有序類似壁紙圖案的美妙花紋。

本文的主旨是在說明程式設計者可經由演算法清楚地表達出心中的想法，至於想法從何而來則不是本文的重點；因此，我們在這裡不打算討論爲什麼經由以上迴圈中簡單的算術運算可以得到如此的圖案。

在上面演算法的寫法中，有幾點與一般程式語言有較大的差異：

1.一般程式語言常見的設值敘述（如 C 語言的 '=' 及 PASCAL 的 ':='）在此處是以 '←' 來將箭頭右邊的值放入左邊的變數中，這是許多電腦科學家在寫演算法時喜歡的方式。

2.數學上的次方（如 x^2）、開根號（如 $\sqrt{x}$）、floor 函數（如 $\lfloor x \rfloor$）、對數函數（如 $\log_5 x$）等，都可以直接寫在演算法中，不須考慮特定程式語言中這些數學運算的寫法。

3.一般程式語言對變數在使用前須先宣告（declare）的要求可忽略。

4.可將一個條件判斷（如 **if**）或迴圈（如 **for**）敘述的內容部分（body）純粹由其與 **if**、**for** 等關鍵字的相對位置來表明，也就是說，將內容部分由關鍵字往內縮，以省略掉一般程式語言用來標示內容部分開始與結束的特殊字元或關鍵字（如 C 語言的 '{' 及 '}'，PASCAL 語言的 **begin** 及 **end** 等）。

5.如果碰到某些觀念或運算可以用文字簡短而清楚地表達出來，在演算法中即可直接用文字敘述來表達，如上例中要判斷變數 *c* 是否爲奇數，可直接用 **if** *c* is odd ...。

如前所述，演算法注重的是觀念的表達，它忽略掉一般程式語言的繁文縟節，而能讓人單純地瞭解解決問題的重點與步驟；若想要將演算法實際轉換成程式，只要演算法提供了足夠的細節，通常是件非常容易的事。以上面的 WALLPAPER 演算法爲例，以下爲其轉換成 C 語言程式後的結果，隨後並附上一些由不同的參數 *side* 的值所得的輸出圖形（此程式是在個人電腦上以 Turbo C++ 3.0 編譯），有興趣的讀者可自行試驗其他的值，必能因輕易創造出複雜的圖案而獲得莫大的樂趣。

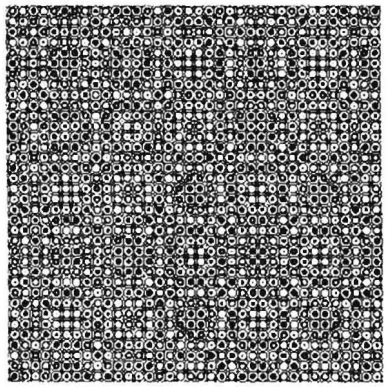
---------- 程式 ----------

```
#include <stdio.h>
#include <graphics.h>

void wallpaper(float side)
{
   int i, j, c;
```

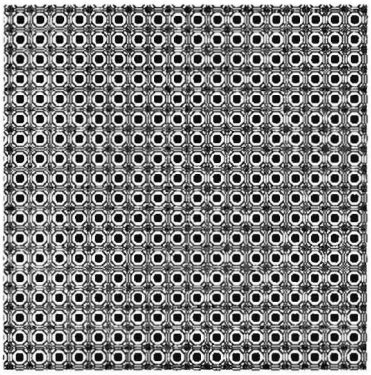
```
   float x, y;

   for (i = 1; i <= 300; i++)
      for (j = 1; j <= 300; j++) {
         x = i * side/100;
         y = j * side/100;
         c = x*x + y*y;
         if (c % 2)                    /* 如果 c 是奇數 */
            putpixel(i, j, WHITE);     /* 在螢幕上畫出一個點 */
   }
}
void main()
{
   float side;

   /* Turbo C 中讓螢幕進入繪圖模式所需的變數 */
   int graphdriver = DETECT, graphmode;

   /* 要求使用者輸入 side 的值 */
   printf("Enter a number between 0 and 100: ");
   scanf("%f", &side);

   /* Turbo C 中讓螢幕進入繪圖模式的固定用法 */
   initgraph(&graphdriver, &graphmode, "c:\\tc\\bgi");

   wallpaper(side);

   getch();          /* 讓螢幕停住以便觀察所畫的圖，按任一鍵繼續 */
   closegraph();     /* Turbo C 中讓螢幕離開繪圖模式的固定用法 */
}
```

---------- 執行結果 ----------

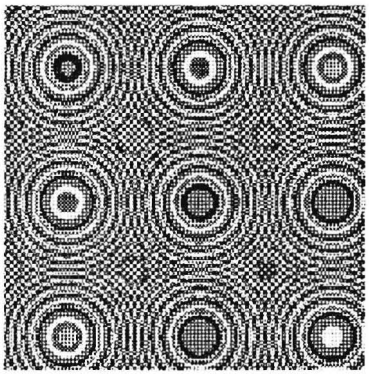
side = 35：



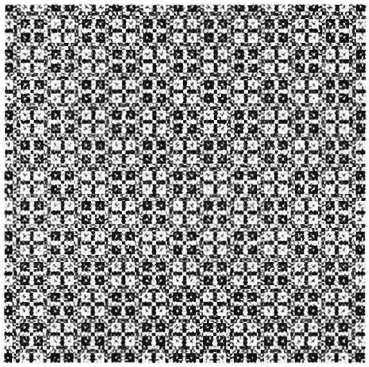
side = 25：



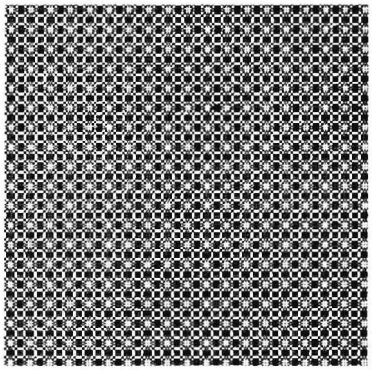
side = 58：



side = 60：



side = 75：



參考資料

1.Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

2.Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms*. McGraw-Hill, 1989.