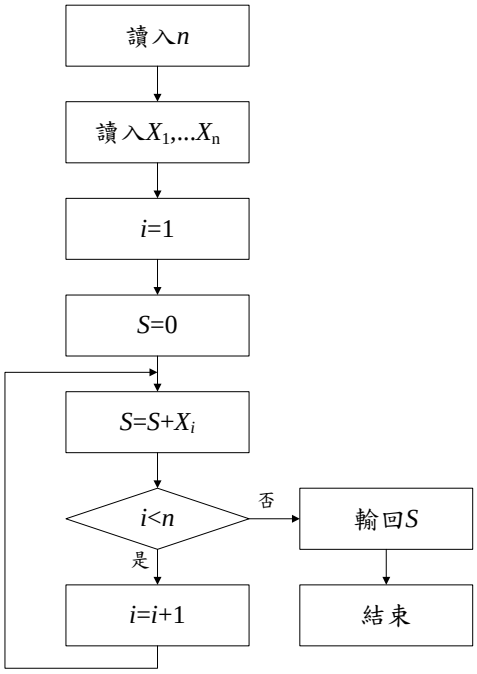
為何要學演算法?

李家同

暨南大學榮譽教授

rctlee@ncnu.edu.tw

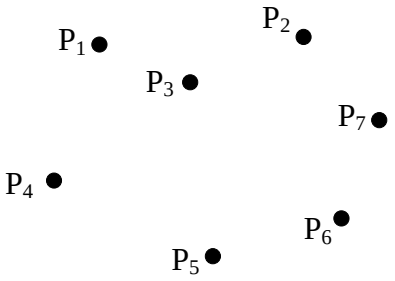
寫一個程式，總要有一個演算法，舉一個簡單的例子，假設我們要將 n 個數字加起來，我們可以用以下的流程圖.



圖一

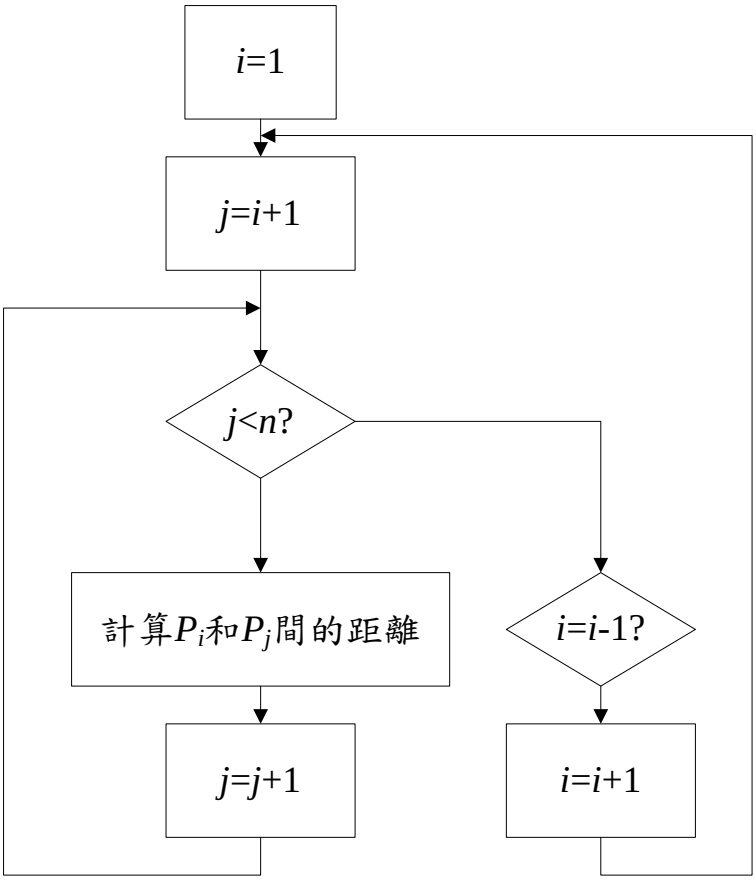
如果要執行這個程式，我們要做 n 個加法，我們說這個程式的複雜度是 O(n)(唸成 Order n)。

假如我們在平面上有 n 個點，以 n=7 為例，我們的點分佈可能如圖二。



圖二

如果要計算這 n 個點之間的距離,我們可以用以下的流程圖:



圖三

假設 n=3,我們的程式執行如下:

(1) i=1,我們計算 P_1 和 P_2,P_3 及 P_4 間的距離。

(2) i=2,我們計算 P_2 和 P_3 及 P_4 間的距離。

(3) i=3, 我們計算 P_3 和 P_4 間的距離。

我們一共要做多少次距離的計算呢? 我們的計算如下:

$$(n-1)+(n-2)+\cdots+1$$

$$=\frac{1}{2}(n-1+1)(n-1)$$

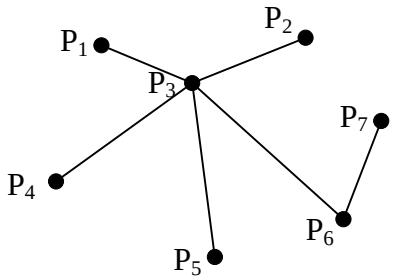
$$=\frac{1}{2}n(n-1)$$

$$=\frac{1}{2}(n^2-n)$$

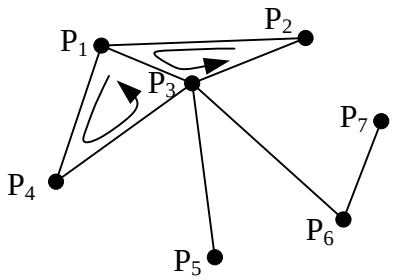
當 n 很大時，我們可以說我們一共計算 n^2 個距離。

我們學演算法的目的就是在節省計算的步驟,假如你的演算法要 n^2 個步驟，而我的只要 n 個步驟，我的程式就會比你的快得多。演算法學得好，很多看上去很難的問題可以用很好的演算法來解決掉。

我現在要舉一個例子，首先我要對"樹"下一個定義，請看圖二中的點，假如我們將這些點連成像圖四中的樣子，所連成的是一個樹。如果我們連成了像圖五的樣子，連成的就不是樣，因為圖五中有二個循環(Cycle)。

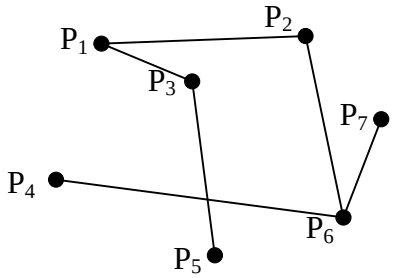


圖四



圖五

我們所要找的樹是要最小的，舉例來說，圖六中的樹就不是最小的。



圖六

如何能找到最小的樹呢？有一個很直接了當的方法，叫做窮舉法。所謂窮舉法，乃是將所有可以達成的樹都找出來。在其中，找出最小的樹出來，但是數學家告訴我們，如果我們有 n 個點，我們可以造成 n^{n-2} 個樹，假如 n=100，我們一共要找 $100^{100-2} \approx 100^{100}$ 個樹。這是絕對不可能的。因為 100^{100} 是天文數字也。

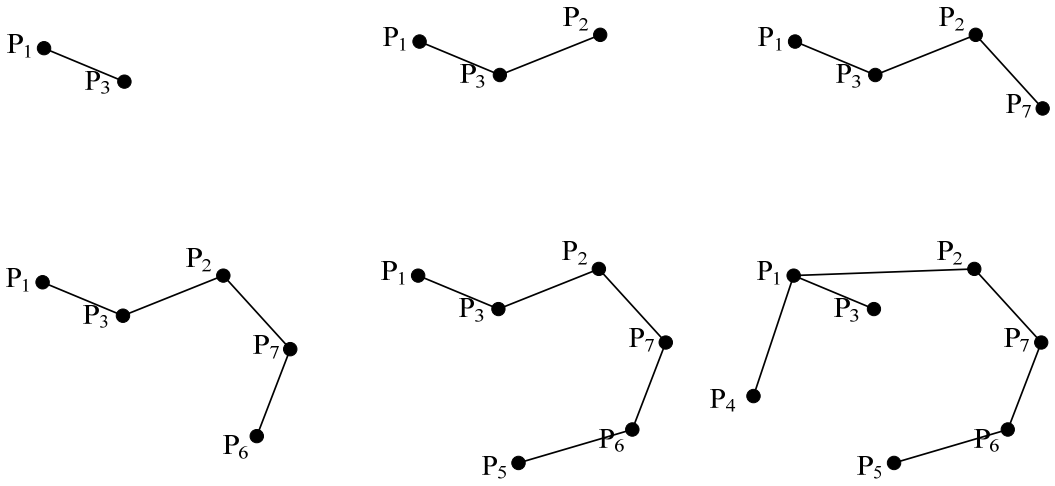
其實，你如果精通演算法，就會知道一個非常簡單的方法。我在此大概地介紹這個方法給各位看。

(1) 任選一點，假如選了 P_1，和 P_1 最近的點是 P_3，我們連接 P_1 和 P_3。

(2) 在 P_1 和 P_3 以外的點中，找一個點和 P_1 和 P_3 最近，我們一定會找到 P_2，而且 P_2 和 P_3 最近，就連 P_2 和 P_3。

(3) 用以上的步驟，我們下一步連接 P_3 和 P_4。

全部過程如圖七所示。



圖七

由以上可以看出唸演算法的重要性，如果我們的演算法沒有學好，會寫出非常不好的程式。

最近，CPU 越來越普及，軟體也越來越重要，而且為了省電，大家喜歡用較慢的 CPU，在這種情況下，我們更需要用好的演算法了。