問題的計算複雜度 – P 與 NP

吳邦一, 2006/8

目錄

一、 前言

我們如何表達一個問題是容易計算或是難以計算的呢？也就是訂定一個衡量的指標來定義問題在計算上的難度，計算機理論學家爲了去表達這樣一個概念做出了努力，發展出了一些理論與相關的結果。這些結果是一個計算機研究者最基本的知識，現在也可以找到很多的教科書，對這個部分做了很好的整理也有很詳細的有系統的介紹。隨著計算機的廣泛運用在生活中，一般的人對於計算機也越來越不陌生，很多非計算機領域的技術人員或者是一些未來可能加入計算機科學研究的人，應該可以有更簡單的方式對這個部分有些粗淺的認識。

本文以不太專業的方式來介紹這個部分，理論計算機科學的領域是很數學的，數學的東西講求嚴謹，要以不專業的方式來介紹實在是個難題，爲了達到簡單容易了解的目的，因此省卻了很多的定義與術語，有些東西也就因此而不嚴謹，而目的是希望介紹給初入行的學生看。

二、 爲何要定義問題的難度

計算機科學家的主要目標之一就是找出好的方法來運用計算機解決問題，爲此我們必須評估方法的好壞，計算方法的好壞通常以此方法所需要的計算時間，做爲評估的標準，但是要評估一個方法的優缺不能單以其所花的時間做標準。這好比我們說一個人參加賽車跑出來的成績是兩個小時，你無法判斷他跑的

是快還是慢，如果賽程的距離是 300 公里就很快；但是如果是一百公里的比賽，可就算是慢了，也就是說，我們要看速度而非時間來判斷。

在計算方法的評估上也是一樣，我們要看計算時間與所處理的資料量的一個相對關係，在計算理論上我們稱之爲一個方法的計算複雜度（computational complexity）。計算複雜度是一個輸入資料量的函數，他評估的是當資料量增加時，計算時間的增加情形。

除了要考慮資料量的多少之外，我們還需要考慮問題的難度。以上述賽車的例子來說，時速 100 公里如果在一個筆直的高速公路上跑可能不算快，但是如果是在一個崎嶇的越野山路上跑出的速度，可能就算是非常快了。在計算方法上來說，我們知道一個方法的效率還不夠，我們還需要知道他所解的問題是難還是容易，一個問題的難度我們稱之爲問題的計算複雜度，他定義的是解此問題的最佳效率爲何。

三、 問題以及問題的解

在進一步探討問題的難度之前，我們有必要對於「問題」給予一個定義，我們這裡談的當然是計算問題。一個問題諸如「1+3=？」，他當然可以算是個問題，我們也可以寫一個演算法或程式來解他。但是我們對這樣的問題通常是比較不感興趣的，因爲這樣的一個問題他的輸入已經在問題中固定了，他的解就是個答案，求出來之後也就固定了。相反的，我們對於「輸入兩個整數 x 與 y，x+y=？」這樣的問題更感到興趣，這樣的問題我們稱爲「抽象問題」，而前述如「1+3=？」這種問題就稱爲「具體問題」。

一個抽象問題具有輸入與輸出，如果給定了輸入（稱爲 instance）他就變了具體問題，對於抽象問題我們要尋求的不是某個特定輸入的答案，而是一個計算方法，也就是描述如何運用計算機的基本運算來求得答案的程序與步驟。

爲了簡化起見，在探討問題的難度時，我們往往只考慮「是與否」的問題，也就是所謂的是非題，這樣的問題的答案只有「是」與「否」兩種可能的結果，例如「輸入兩個整數 x 與 y，x+y 的值是否大於 10？」。

四、 易處理與不易處理的問題

如果我說「這個問題很難」，這句話可能有兩種不同的意思：一種是說，我們很難想出一個好的解決方法來解決他，也許目前我已經有一種還不錯的解法，只是想進一步找出更好的方法是件困難的事；另一種意思就是指的這個問題本身是難以計算的。第一種意思指的是對人而言很困難，而第二種意思指的是對計算

機而言很難。在探討問題的難度時我們通常指的是第二種意思，比較正確的講法應該是指一個問題是易處理的（tractable）或是不易處理（intractable）。

我們說過問題的難度，也就是複雜度，是以相對於他的輸入資料量所需增加的計算時間的快慢來表示，具體來說，輸入的資料量是 n，一個問題的複雜度是 n 的一個函數 f(n)。比方說 f(n)=3n 代表了計算時間與資料量成正比；又如 f(n)=n^2 則代表了如果資料量增加了 k 倍時間會增加爲 k 平方倍。對我們來說，這個函數越平緩代表此問題越是易於處理，而如果上升的越陡峭則代表資料資加一點點，解題的計算時間就需要增加很多，所以是越不易於處理。

因此理論上來說，問題的難度可以畫分爲無窮多種等級，但是我們對於特別難處理的問題感到特別的興趣。長久以來我們就碰到一些問題，以目前最好的解法所需要花的時間是隨資料量以指數函數的方式成長，例如說 f(n)=2^n。

> 注意：我們這裡說「以目前好的解法」而不說「此問題的複雜度」，是因爲可能目前還不知道真正最好的解法，我們必須證明一個問題的最佳解法，也就是以後都不可能更好了，才能說是他的複雜度。

指數成長的函數就好像是高利貸的利滾利，他的增加情形是極爲可怕的。比方說解某一個問題的方法的複雜度是 f(n)=2^n，在 100 個資料時，處理的時間是 1 秒鐘，那麼當資料增加 10 倍變成 1000 個資料時，計算時間就會成長爲 1024 秒；而如果資料量增加 100 倍時，時間會增加到 10 的 30 次方秒，相當於超過 10,000,000,000 兆年，這根本不只是難以處理而是根本無法處理。計算機理論上簡單的將問題區分爲易處理與不易處理兩類，就是以他的複雜度函數是多項式函數或指數函數加以區分。

五、 杜林機、P 與 NP

問題是簡單或容易計算，取決於計算機器的能力。好比說，

> 找喬登灌籃是容易的（我當然指的不是年紀已老的喬登）；但是找蔡依林灌籃卻是困難的。

我們如何定義機器的能力呢？廿世紀初的數學家杜林（Turing）給了兩種機器的模型。第一種機器是所謂的「確定性杜林機」，這種機器的特色是他每一個計算步驟都是由輸入以及計算過程中的狀態所決定的，簡單說當他的計算法則（稱爲程式）以及輸入資料確定時，他的每一個計算步驟都也就被確定了。我們現在發展出所有的計算機器，都是屬於確定性杜林機這種類別。

> Remark: 現在有一些遊戲機或者程式中間有運用到所謂的隨機

> 亂數，但這些隨機數的產生也是有一訂的規律，他們會隨所使用的參數（有時候用當時的時間）以及產生的法則而被決定，因此並非真實的隨機亂數。

所謂的計算法則，是由一條一條的計算規則（指令）組成，每一條計算規則指示計算機在什麼狀態時應該去做什麼事情，我們現在談論的是在最原始底層的定義，在這個階段計算規則並無順序性。如果這個機器是確定性杜林機，我們可以看到，在每一個狀態時，必然最多只有一條規則可以適用，否則機器將無法運作，如果再某個時刻，沒有任何一條計算規則可以適用，我們定義此時機器會停下來。

現在我們可以看，如果是一個確定性杜林機所運作的計算規則（稱爲確定性演算法），我們可以將這一群規則依照他所運作的前後順序，排成一個序列，這就是目前大多數的程式語言（例如 C, Java, Basic 等等）所採取的方式。一個問題如果「可以用確定性的杜林機在多項式的時間複雜度求得答案」，我們就稱這個問題是屬於 P 這個類別。

另一種機器的能力"似乎"更爲強大，稱爲「非確定性杜林機」，他與前者的差異在於：在面對一個狀態時，可能有不只一條的計算規則可資運用。爲了探究這樣的機器的最大計算能力，我們假設在任何一個時刻，如果有一條或一條以上的規則可以適用而且其中有一條可以得到正確解答，那麼這個機器會使用正確的那條規則。

這使得事情看起來複雜多了，我們知道，如果在狀態 A 時有 n 條規則可以適用，他可能將機器帶到 n 種不同的狀態，而這些狀態每一個如果又有 m 條規則可以適用，就會可能有 n*m 個不同的狀態。如果步驟數多一點，可能的狀態數馬上就變成了天文數字。似乎不可能存在這樣的機器，的確現在是沒有這樣的機器。我們可以想像成有一個極其厲害的人，他不知怎麼的（連他自己都不知道）就可以對這個問題找到答案。

那麼在這樣的機器能力之下，所有的問題不都是應該是非常容易解的嗎？其實不然，我們再看一下這個定義：

> 「如果有一條或一條以上的規則可以適用而且其中有一條可以得到正確解答，那麼這個機器會使用正確的那條規則」

這其中隱含的另一個意義是說：「如果沒有一條規則可以得到正確的答案，他會隨便使用一條」，因爲機器停下來的條件是「沒有可以適用的規則」。那麼問題就來了，當這個機器找到一個解答時，我們必須有一個計算法則能夠驗證所得到答案是否是正確的。如果是，代表真的找到一個答案；如果不是，依據我們的假設（有正確的必會找到），就可知道並不存在解答。記得嗎？我們定義所有的問題的答案只有「是」與「否」兩種，因此當驗證的結果是不正確的時候，我們

知道他的答案就是「否」。

而一個問題「可以用非確定性的杜林機在多項式的時間複雜度求得答案」，我們就稱這個問題是屬於 NP 這個類別。

六、 演算法的層次

從演算法的角度來定義此問題時，我們省卻計算機器的講法，我們說非確定性演算法具有一個特別的指令稱爲「guess」。在演算法中可以使用這個指令去猜測一個答案的某個部分，在組成整個答案之後，必須驗證此答案。如果是在很少的時間內能夠完成的就稱爲 NP；而一個演算法如果沒有使用 guess 這樣的指令，就稱爲確定性的演算法。一個問題如果可以用確定性的演算法在很少的時間計算出解答就稱爲一個 P 的問題，而如果是以非確定性的演算法在很少的時間計算出解答就稱爲一個 NP 的問題。

因此簡單的來說，所謂一個屬於 P 類別的問題是已知可以有確定步驟快速求得解答的問題；而 NP 的問題則是可以很快的驗證答案的問題。因爲確定性的杜林機也是屬於非確定性的杜林機的一種特例，我們立刻可以得到一個結論

「P 包含於 NP」

這與我們的經驗是相符合的，計算解答與驗證答案的雙向難度通常是不對等的，求解往往是比較難的，舉例來說，求解一個高次方程式的答案並不容易，但是驗證某個值是否爲高次方程式的根卻只需要一些簡單的算術就可以辦得到。我們會很直覺的認爲非確定性杜林機要比確定性杜林機的能力來得強大，也就是有些 NP 的問題並不在 P 中。但是很遺憾的事，經過了許多年的努力，我們至今無法得到證明。「NP 是否等於 P」這個問題目前已被列爲 7 大數學難題之一，如果證明的出來可以得到一百萬美金的獎勵。

七、 寫在學術與專業之外

NP 是否等於 P 這個問題已經懸了幾十年了，現在大多數的科學家傾向於相信答案是否定的。在 2002 年的一項調查中，一百位研究者裡面有 61 位相信 NP 不等於 P，9 位相信 NP=P，22 位不確定，而有 8 位研究者認爲此問題在目前的假設基礎下是無法證明的。

歷史上無法解決的數學問題多的是，根據歌德爾的不完全理論，總有些是無法證明的，但是問題要能夠遺留下來而受到重視，必然是他對人類科學發展的重要性。

我們人類的科學發展到現在，對於人類的知識與智慧如何形成仍有太多的未知之處，有些人對於某些問題就是能夠得到答案，但是他的答案是如何得到的可能連他自己也不知道，這好比電影雨人中的主角能夠知道掉落的火柴有幾根。如果只能知其然而不知其所以然，對於知識的累積與創造當然是個遺憾，但是我們可以寄望未來，有一天能夠對知識的形成有更多的了解的時候，或許有機會將這一類的知識形成得到更好的答案（這或許就是人工智慧學者最大的夢想）。但是科學與信仰不同，一人所知最起碼必須能夠驗證答案，否則就只能是宗教的領域而非科學的範疇。

因此，我們可以很不正式的說，NP 就是以人類最大極限可以有效率求解的問題，而 P 則是以現代計算機科學所能夠有效率解決的問題，探討 NP 是否等於 P 也就是在問：計算機可否實現人類所有可能的計算能力。