問題的難與易──介紹NP 完備的概念

江南波

在數學的領域裡, 數學的結構、性質的探討, 自然是一件賞心悅目的事; 可是, 問題的提出及解的追求卻又是另一件更原始、更能引人入勝的事。在本期的數學傳播裡, 傅恆霖教授將圖論中幾個重要的問題歸納成圖上的數字問題提出; 黃國卿教授則對演算法 ── 解問題的一種方法 ── 加以討論。問題一旦提出, 解法一旦求出, 我們常會想到的是: 這問題還有沒有更好的解法? 要回答這一疑問, 應該要先了解問題本身的難易度及解法的好壞程度才能加以判斷。有一種很有意思的觀念 ── NP 完備性 ── 可用來描述問題的難易程度。本文就是想以較鬆散的方式將此觀念加以介紹。

1. 演算法的複雜度

如何判斷一個演算法的好壞呢? 一般我們以此演算法在最壞的情況下在計算機中所需要使用的記憶空間的多少 (稱爲此演算法的空間複雜度) 及此演算法在最壞的情況下所需要完成的運算 (包括加、減、乘、除及比較等) 的多少 (稱爲此演算法的時間複雜度) 二方面去衡量。

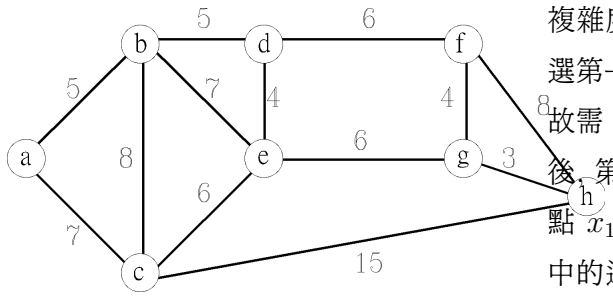
問題: 給定任一個網路 (G, w), 其中 G 爲一個圖而 w 爲給G的邊填上數字的重量函數, 如何求出 G 中邊的重量和爲最小之生成樹。

對於這個問題, 有一個 Prim 的演算法是這樣求解的:

演算法: 設 G 有 n 個點。按下列步驟去求

1. 求出 G 中重量最小的邊 e 放入空集合 T 中。
2. 若 T 中邊的個數未達 $n-1$ 時, 繼續從 G 中一端點在 T 中而另一端點不在 T 中的邊裡, 求出重量最小的邊加入 T 中, 直到 T 中的邊數達到 $n-1$ 時停止。
3. 當此演算法停止時所得到的 T 即爲所欲求的G中重量和爲最小的生成樹。

圖 1(a) 表給定的一個網路 (G, w), 圖 1(b) 表用 Prim 的演算法, 求出 G 中重量和最小的生成樹 T 的過程。



圖一 (a)

(1) $T = \{gh\}$
(2) $T = \{gh, gf\}$
(3) $T = \{gh, gf, eg\}$
(4) $T = \{gh, gf, eg, de\}$
(5) $T = \{gh, gf, eg, de, bd\}$
(6) $T = \{gh, gf, eg, de, bd, ab\}$
(7) $T = \{gh, gf, eg, de, bd, ab, ce\}$

並停止。

圖一 (b)

讓我們算算看此演算法的空間複雜度$\varphi(n)$ 及時間複雜度 $\psi(n)$。由於此演算法求解過程須記憶下前後求出的 $n-1$ 個邊, 故其空間複雜度 $\varphi(n) = n-1$。至於時間的複雜度 $\psi(n)$, 我們粗略的估計其上界如下: 選第一邊, 是由 e 邊中選出重量最小的一邊, 故需 $e-1$ 個比較; 若已經選了 k 邊加入 T 後, 第 $k+1$ 邊的選取只需在與 T 中 $k+1$ 點 $x_1, x_2, \ldots, x_{k+1}$ 相連且另一端點不在 T 中的邊中選出重量最小的一邊即可, 故頂多只需 $d(x_1) + d(x_2) + \cdots + d(x_{k+1}) \leq 2e$ 個比較即可, 故 $\psi(n) \leq e + 2e + \cdots + 2e = e + 2e(n-2) \leq e(2n-3) \leq \frac{n(n-1)(2n-3)}{2}$。

到底怎樣的複雜度才算是好的演算法呢? 爲了區分方便, 我們先討論"大 O"的符號。我們說 "$f(n) = O(g(n))$" 乃表示存在二常數 c 及 N 使得 "$n \geq N \Rightarrow f(n) \leq cg(n)$"。如上面所提的二函數則有 $\varphi(n) = O(n)$; 而 $\psi(n) = O(n^3)$。一般而言, 若對一演算法之時間複雜度 $\psi(n)$ 存在有一多項式 $p(n)$ 使得 $\psi(n) = O(p(n))$, 則此演算法稱爲多項式型演算法, 並被歸類爲好的演算法。而對一個問題若存在有多項式型的演算法來解它, 則此問題 q 稱爲易處理的問題, 並記爲 $q \in P$, 其中 P 表由所有的易處理的問題所構成的問題類。不過, 對於易處理的問題一般希望能有時間複雜度爲 $O(n^2)$ 之演算法才滿意。當問題參數有多個時複雜度亦可同樣的定義。

時間複雜度	8000 運算/秒 所需時間	400,000 運算/秒 所需時間	1,500,000 運算/秒 所需時間
n	0.125	0.0025	0.00067
n^2	125	2.5	0.67
n^3	125000	2500	670
1.1^n	10^{38}	2×10^{36}	5.3×10^{35}

表一

表一表示在不同速率的計算機上模擬不同時間複雜度的演算法, 當問題的參數 $n = 1000$ 時所需要的時間。從此表可看出時間複雜度爲 1.1^n 之演算法, 即使在每秒鐘能處理 1,500,000 個運算的計算機上解參數 $n = 1000$ 的個案時, 需要時間約 5.3×10^{35} 秒 $\approx 6.1 \times 10^{30}$ 日 $\approx 1.7 \times 10^{28}$ 年, 根本就是一件不可能辦到的事。事實上從第一個人類出現在這世上到現在也沒有這麼長的時間。而時間複雜度爲 n^3 的演算法即使在每秒僅能處理 8000 運算的計算機上解參數 $n = 1000$ 的個案, 只需約 1.5 日即可完成。

2. 判斷性的問題

在進一步討論集合 P 之前, 我們將所有的問題限定成判斷性的問題 —— 即答案僅爲是或否的問題。我們這樣做的原因, 一方面是圖論的問題很容易就可改爲判斷性的問題, 而且原問題屬於 P 之充分而且必要的條件是改成的判斷性問題屬於 P。例如給定任一圖 G, 求 G 之著色數 $\chi(G)$ 之問題 (將圖 G 的點塗上顏色使得有邊相接的任二點均塗不同的顏色, 則稱爲 G 之一著色。G 之任意著色所用到的顏色數中最小的值稱爲 G 之著色數), 可改爲給定任一圖 G 是否存在使用 k 個顏色的著色的問題; 而且這種改法, 若原問題有多項式型的演算法求出 $\chi(G)$, 則該演算法即能使我們得到當 $k < \chi(G)$ 時改得的問題答案爲否, 而 $k \geq \chi(G)$ 時改得的問題的答案爲是; 反之, 若改得的問題對 $k = 1, 2, \ldots, n$ 均有時間複雜度爲 $O(n^\ell)$ 之演算法, 則原問題就有時間複雜度爲 $O(n^{\ell+1})$ 之演算法。故我們將問題限定成判斷性的問題並不影響我們對 P 中問題的討論。

3. 非確定性的演算法

關於圖論的問題, 有很多都有多項式型的演算法來解它們; 可是卻也有很多的問題, 到現在都沒有辦法提出多項式型的演算法來解它們, 例如求任一圖之著色數、邊著色數、全著色數的問題, 都沒有發現多項式型的演算法可用來求解。那麼, 到底是努力不夠呢? 還是這些問題根本就不屬於 P 呢? 由於對這一問題的思考, Cook發展出了 NP 完備性的觀念。爲了了解這一觀念, 我們先要有非確定性的演算法的概念。

一般我們所說的演算法, 我們把它歸類爲確定性的演算法, 它的特色是當我們在執行演算法的過程中, 若遇到分岐點時, 我們必須每一分支都檢驗過才能確定我們的答案。而非確定性的演算法則是一種想像中的演算法, 它與確定性演算法唯一不同的地方就是, 當非確定性演算法, 遇到分岐點時, 此演算法能自動的選擇出一個適當的分支並保證只要檢驗此一分支即可確定答案。

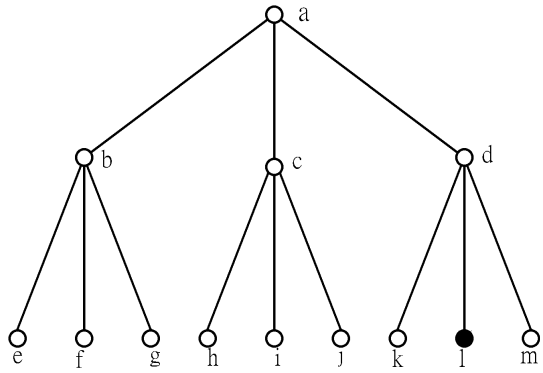


圖2(a)

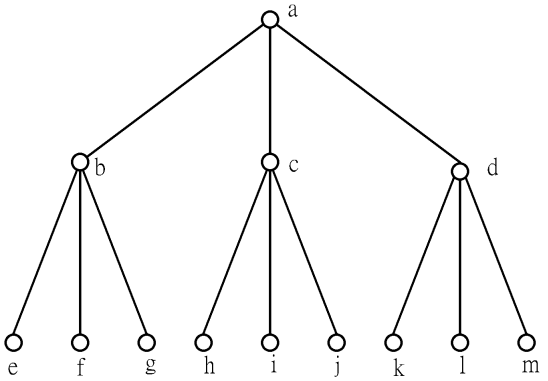


圖2(b)

圖2(a),(b) 分別表示給定一樹圖, 欲判斷此樹圖上是否有點塗黑的問題的二例。當非確定性演算法遇到圖2(a) 時會自動選擇出 $a-d-l$ 路徑並判斷出該圖中有塗黑的點; 而該演算法遇到圖2(b) 時, 則可能選出路徑 $a-b-g$ 而判斷出此樹圖中沒有點被塗黑; 這二情況, 非確定性演算法只要檢驗三點有無被塗黑。而確定性演算法遇到圖2(a) 時, 可能會檢驗過 $a-d-m$ 及 $a-d-l$ 二路徑而判斷出有塗黑的點; 而遇到圖2(b) 時, 則會檢驗過所有的9個路徑後判斷出沒有塗黑的點; 確定性的演算法處理這二例, 分別需檢驗4點及13點。

對於非確定性的演算法, 我們一樣的可仿照確定性演算法定義它的時間複雜度。一問題 q 若存在有時間複雜度爲 $O(p(n))$ 之非確定性的演算法求解其答案, 則稱此問題 q 爲非確定性多項式型問題並記做 $q \in NP$, 其中 NP 即爲由所有非確定性多項式型問題所成的問題類。

爲什麼非確定性演算法有這麼大的選擇能力呢? 或者說爲什麼我們要考慮這麼奇怪的演算法呢? 大概可以用下面二層面來解釋:

1. 若想像在每個分岐點上, 有幾個分支就配置幾個計算機, 則以這種模式作平行處理的話, 處理問題所需要的時間就是非確定性的演算法的時間複雜度, 因此此演算法可看成是平行處理的一種推廣。
2. 當我們對一問題答「是」的充分條件及必要條件有相當充分的了解時, 我們在分岐點上常常可以做適當的選擇。若一問題可經由充分、必要條件的討論而提供出問題參數 n 之多項式 $P_1(n)$ 種選擇, 而此問題又有時間複雜度爲 $P_2(n)$ 之非確定性演算法加以求解, 則該問題自然就有時間複雜度爲 $O(P_1(n)P_2(n))$ 之演算法求出答案。

由 P 與 NP 之定義, 自然可知 $P \subseteq NP$。但 P 會是 NP 的眞子集合嗎? 由非確定性演算法的強有力的選擇能力, 直覺上我們可能會認爲 P 應該是 NP 的眞子集合; 可是由上面 (ii) 的層面上看, 感情上我們又希望 $P = NP$。事實上, 直到目前爲止, 我

們即未能證明 $P = NP$, 也未能提出反例證明 $P \neq NP$; 不過, 對這一問題的討論卻引導出一蠻有趣的觀念 — NP 完備性。

4. NP 完備性

設 L 爲一判斷性的問題。則 L 中的例子, 即稱爲 L 的一個輸入, 而 L 之所有的輸入所構成的集合則稱爲 L 的輸入空間。令 L_1 與 L_2 爲二判斷性問題, 而 U_1 及 U_2 分別爲 L_1 與 L_2 之輸入空間。若存在有一時間複雜度爲多項式函數的演算法將 U_1 中的任一輸入 u_1 轉變爲 U_2 中的輸入 u_2 使得 u_1 在 L_1 中的答案爲是的充分而且必要的條件爲 u_2 在 L_2 中的答案爲是, 則稱 L_1 可多項式時間內化爲 L_2。

在看例子之前, 讓我們先瞭解圖論中的兩個常見的術語。設 G 爲一圖, 其點集合爲 V 而其邊集合爲 E。令 A 爲 V 之子集合。若 A 中任二點 a_1, a_2 均沒有 E 中的邊連接它們, 則稱 A 爲 G 中之一獨立集。反之, 若 A 中任二點 a_1, a_2 均存在有 E 中的邊連接它們, 則稱 A 爲 G 中之一團。若 B 爲 V 之子集合且 E 中任一邊均與 B 中某一點相連, 則 B 稱爲 G 之一邊覆蓋。如圖3中之圖 G 中, $A_1 = \{a, b, c, d\}$ 爲 G 之一團。

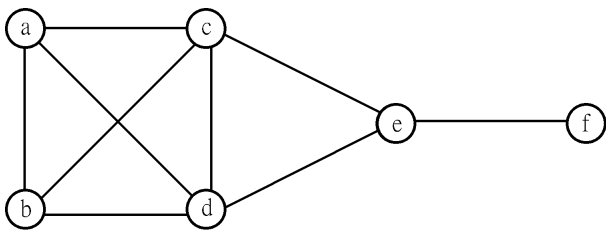


圖3. G 圖

$A_2 = \{b, e\}$ 爲 G 之一獨立集, 而 $B = \{a, c, d, e\}$ 爲 G 之一邊覆蓋。

例: 令 L_1 表給定任一圖 G_1 及非負整數 k_1 (小於或等於 G_1 之點數), 判斷 G_1 有沒有多於或等於 k_1 個點的團的問題; 而 L_2 表給定任一圖 G_2 及非負整數 k_2, 判斷 G_2 有沒有少於等於 k_2 的邊覆蓋的問題。則 L_1 之輸入空間 $U_1 = \{(G_1, k_1) : \text{若 } G_1 \text{ 之點數爲 } n_1, \text{ 則 } 0 \leq k_1 \leq n_1\}$ 而 L_2 之輸入空間 $U_2 = \{(G_2, k_2) : k_2 \geq 0\}$。考慮將 U_1 中之輸入 (G_1, k_1) 化爲 U_2 中之輸入 (G_2, k_2) 之演算法 $\mathcal{A}$, 其中 $k_2 = n_1 - k_1$, 而 G_2 爲 G_1 之補圖, 即以 G_1 之點集合 V 爲 G_2 之點集合, 而任二點在 G_1 中有邊相連之充分而且必要的條件爲此二點在 G_2 中沒有邊連接它們 (如圖4, G_2 爲 G_1 之補圖)。則 $\mathcal{A}$ 確將 U_1 中任一元素均轉化爲 U_2 中之元素, 且 $\mathcal{A}$ 之時間複雜度 $\psi(n_1) = O(n_1^2)$。而且若 G_1 有多於或等於 k 個點的團 A_1, 則 G_2 中之任一邊均不可能連接 A_1 中之二點, 即 G_2 中之任一邊均至少連接 A_1 以外的一點。故 $V - A_1$ 即爲 G_2 之點數少於或等於 $n_1 - k_1 = k_2$ 之邊覆蓋。反之, 若 A_2 爲 G_2 中點數少於或等於 $k_2 = n_1 - k_1$ 的邊覆蓋, 則 G_2 中之任一邊均至少與 A_2 中某一點相連接, 即 G_2 之任一點均不可能連接 A_2 以外之二點, 故 $V - A_2$ 爲 G_1 之一團。故 L_1 可多項式時間內化爲 L_2。

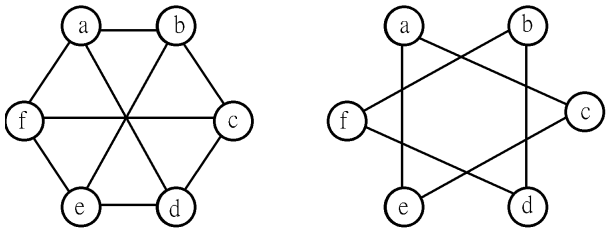


圖4

設 L_1 可多項式時間內化爲 L_2。若 L_2 存在有時間複雜度爲多項式函數的演算法解之, 則只要結合這兩個演算法而得一新的演算法, 則此新的演算法即爲 L_1 之時間複雜度爲多項式函數的演算法。故顯然直覺上會認爲 L_2 比 L_1 較困難。故我們接受下面的定義:

定義: 令 X 爲一問題, 且 NP 中的任一問題均可多項式時間內化爲 X, 則稱 X 爲一 NP 難的問題。若更進一步, $X \in NP$, 則稱 X 爲 NP 完備的問題。

Cook建立起 NP 完備性的觀念後, 在1971的一篇論文裡證明了一個很引人注目的定理, 這大概也是他獲得計算機科學中之一個大獎 − Turing 獎的原因之一吧! 在介紹他的定理之前, 讓我們看一下布爾代數的一些概念。

設 S 爲一 n 變數的布爾代數式, 則 S 可表爲此 n 變數及其補數的和的乘積 (這種式子稱爲合取正規式)。如 $S' = (x_2 + (x_1 \cdot x_2)) \cdot \overline{((x_1 + x_3) \cdot x_2)}$ 可表爲 $S' = (x_2 + x_1) \cdot (x_2) \cdot (\overline{x}_1 + \overline{x}_2) \cdot (\overline{x}_3 + \overline{x}_2) = x_2 \cdot (\overline{x}_1 + \overline{x}_2) \cdot (\overline{x}_3 + \overline{x}_2)$。若我們給此 n 變數指定其值爲0或1, 則 S 亦有值爲0或1。若可以適當的指定變數之值而使 S 之值爲1時, S 稱爲可滿足的。如若指定 $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, 則 $S' = 1$。故 S' 爲可滿足的。

SAT問題: 給定一寫成合取正規式的布爾代數式 S, 判斷 S 是否爲可滿足的。

Cook定理: SAT 問題是 NP 完備的。

5. NP 完備性證明的例子

Cook證明了 SAT 問題是 NP 完備的第二年 (1972), Karp一口氣提出了24個重要的 NP 完備的問題 (後來 Karp 也獲得 Turing 獎)。從此以後, NP 完備的問題就像雨後春筍一般在圖論裡到處被發現。但是仍然有些問題我們依舊無法判斷它是屬於 P? 還是屬於 NP 完備? 下面就讓我們看看一些證明問題是 NP 完備的例子。當然, 當我們已知一些 NP 完備的問題以後, 我們要證明一問題是 NP 完備的就簡單多了, 我們只要證明此問題屬於 NP, 且有一已知 NP 完備的問題可多項式時間內化爲此問題即可; 不必像 Cook 當年證明 SAT 問題是 NP 完備的那麼辛苦。這也難怪到1979年發現的 NP 完備的問題就有約300種。

3SAT問題: 給定一寫成每一項都是3個變數或其補數的和的合取正規式的布爾代數式 S, 判斷 S 是否爲可滿足的?

定理: 3SAT 問題是 NP 完備的。

證明: (1) 設 S 爲給定的布爾代數式。由於非確定性演算法可適當的指定各變數的

值, 故而要判斷 S 的值是1或是0, 只要 S 之項數 n 之 $O(n)$ 的時間複雜度內即可完成, 故3SAT 問題屬於 NP。

(2) 設 S 爲任一寫成合取正規式的布爾代數式。令 $C=(x_1+x_2+\cdots+x_k)$ 爲 S 中之一項。考慮將 C 化爲每一項均爲三變數或其補數之和的合取正規式 C' 的演算法分別如下:

(i) 當 $k\geq 4$ 時, 取 $C'=(x_1+x_2+y_1)\cdot(x_3+\overline{y}_1+y_2)\cdot(x_4+\overline{y}_2+y_3)\cdot\cdots\cdot(x_{k-1}+x_k+\overline{y}_{k-3})$, 其中 $y_1,y_2,\ldots,y_{k-3}$ 爲新加入的變數。若 C 爲可滿足的, 則可指定每一 x_i 之值使 C 之值爲1, 當然必有某一 x_j 之值被指定爲1; 故可指定 x_i 爲原來的值, 而 $y_1,y_2\cdot\cdots\cdot y_{j-2}$ 爲1而其餘 y 之值爲0, 則 C' 之值爲1, 故 C' 爲可滿足的。反之, 若 C' 爲可滿足的, 則可適當的指定 x_i 及 y_j 之值使得 C' 之值爲1, 顯然必有某一 x_ℓ 之值被指定爲1, 否則 $y_1\cdot(\overline{y}_1+y_2)\cdots(\overline{y}_{k-3})$ 爲1, 此將使 y_{k-3} 與 $\overline{y}_{k-3}$ 均爲1, 矛盾; 故原 x_i 之指定值即能使 C 之值爲1。故 C 爲可滿足的充要條件爲 C' 爲可滿足的。

(ii) 當 $k=3$ 時, 取 $C'=C$。顯然, C' 與 C 同時爲可滿足的或同時爲不可滿足的。

(iii) 當 $k=2$ 時, 取 $C'=(x_1+x_2+z)\cdot(x_1+x_2+\overline{z})$。若 $C=(x_1+x_2)$ 爲可滿足的, 則可指定 x_1 與 x_2 之值使 x_1+x_2 之值爲1, 故可任意指定 z 之值 使 C' 之值爲1。反之, 若 C' 爲可滿足的, 則可指定 x_1, x_2 及 z 之值使 C' 之值爲1, 則該指定即能使 C 之值爲1, 故 C 爲可滿足的。故 C 與 C' 同爲可滿足的或同爲不可滿足的。

(iv) 當 $k=1$ 時, 取 $C'=(x_1+y+z)\cdot(x_1+\overline{y}+z)\cdot(x_1+y+\overline{z})\cdot(x_1+\overline{y}+\overline{z})$。則 C 與 C' 顯然同時爲可滿足的或同時爲不可滿足的。

故由 (i)-(iv) 的討論知 SAT 問題可多項式時間內化爲3SAT 問題。

故知3SAT 問題是 NP 完備的。

看完了 NP 完備性的1個證明後, 我們再看看圖論中證明 NP 完備性的例子。

3−著色問題: 給定任一點集合爲 V、邊集合爲 E 之圖 G, 判斷 G 是否可用3種顏色著色?

定理: 3−著色問題是 NP 完備的。

證明: (1) 設 G 爲給定之任一圖。由非確定性演算法可適當的選取3顏色該著的各點, 故只要判斷此選取是否爲一著色, 即可分辨 G 是否可用3種顏色著色, 而判斷該選取是否爲一著色, 可在 $O(e)$ 或 $O(n^2)$ 時間複雜度內完成 (其中 e,n 分別表 G 之邊數及點數)。故3-著色問題屬於 NP。

(2) 欲證: 3SAT 問題可多項式時間內化爲3-著色問題。令 E 爲任一寫成每項均爲三變數或其補數和之合取正規式的布爾代數式。我們考慮將 E 如下化爲圖 G 之演算法 $\mathcal{A}$:

(i) 若 E 共有 $x_1,x_2,\ldots,x_n$ 共 n 個變數, 則構造 G 圖之中心部分如圖4所示。

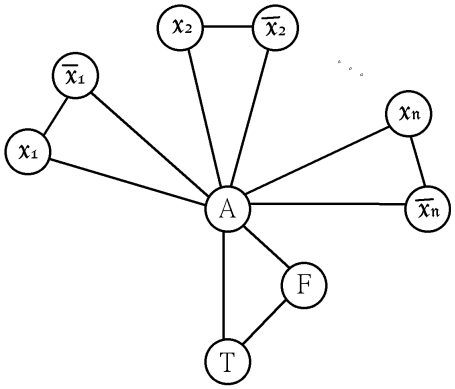


圖4

(ii) 若 E 有一項 $(x_{i_1}+x_{i_2}+x_{i_3})$, 則 G 除中心部分之外, 增加一個由另外6點加上 $x_{i_1}, x_{i_2}, x_{i_3}$ 及 T 所構成如圖5所示的部分。

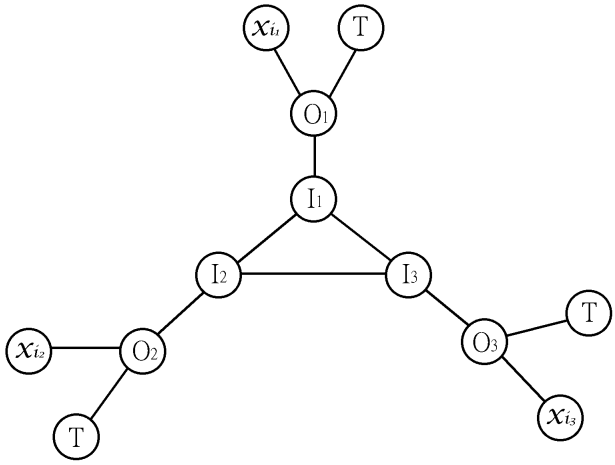


圖5

圖6表示當 $E=(x+\overline{y}+z)\cdot(\overline{x}+y+\overline{z})$ 時所化成的圖 G。

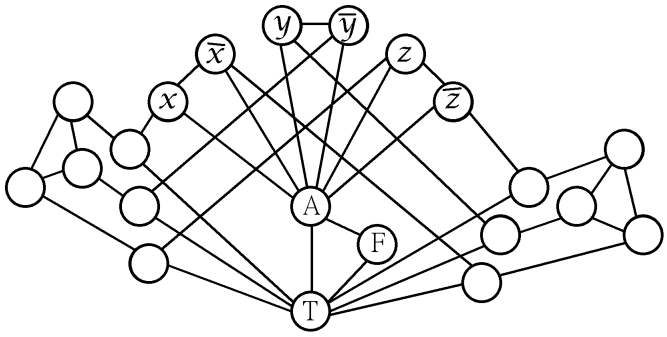


圖6

若 E 爲可滿足的, 則可適當的指定 $x_1, x_2, \ldots, x_n$ 之值使 E 之值爲1, 按照"若 x_i 指定爲1則 x_i 著 T 且 $\overline{x}_i$ 著 F, 若 x_i 指定爲0則 x_i 著 F 且 $\overline{x}_i$ 著 T " 將 G 之中心部分加以著色; 又由於 $x_1, x_2, \ldots, x_n$ 之值的指定, 能使 E 之每一項 $(x_{i_1}+x_{i_2}+x_{i_3})$ 中至少有一 x_i 爲1, 就說是 x_{i_1} 爲1吧! 則將此項對應的部分中 O_1 著 F, O_2、O_3 都著 A 而 I_1 著 A, I_2、I_3 分別著 T、F, 則顯然構成 G 之用3種顏色的著色, 故 G 可用3種顏色著色。反之, 若 G 可用三種顏色著色, 則指定與 T 點著同色的變數爲1而與 F 同色的變數爲0, 則 E 之每一項 $(x_{i_1}+x_{i_2}+x_{i_3})$ 中必有某一變數 x_{i_j} 爲1, 否則此項對應的部分中 I_i $(i=1,2,3)$ 三點則不可能著不相同的三色, 矛盾。故 E 爲可滿足的。

由定義知此問題爲一 NP 完備的問題。

看完了證明某一問題是 NP 完備的兩個例子後, 相信對 NP 完備性的概念已有了一個初步的認識。事實上 NP 完備的問題相信只要 $P \neq NP$ 未被確定之前都會一再被發現出來。因此, Johnson從1981年起在演

算法期刊 (Journal of Algorithms) 開闢有一個 NP 完備性的專欄, 專門報導關於 NP 完備性的最新消息。

6. 結語

建立了 NP 完備性的概念後, 也許有一個問題一直遺留在我們的腦海裡: 到底 $P = NP$ 還是 $P \neq NP$? 從 NP 完備性的定義看來, 若是 $P = NP$, 則只有能給某一 NP 完備的問題提出一個多項式時間複雜度的演算法, 則過去所有發現的 NP 完備的問題都一口氣通通被解決了, 似乎是故事裡的情節, 現實生活裡不太容易出現吧! 可是, 若 $P \neq NP$, 則 Ledner 卻又證明了在 P 與 NP 困難的問題之間將有無限困難度等級的問題類介於此二極端之間, 似乎又與我們當初考慮 NP 問題的動機相違背! 這就是本文介紹 NP 完備性的目的, 希望能引起讀者的興趣而解決這一疑問, 畢竟我們讀者中有些人是注定將要解決一些重要且困難的問題的!

7. 參考資料

1. S. A. Cook, The complexity of theorem proving procedures, Third Annual ACM Symposium on Theory of Computing, New York (1971), pp.151-158.
2. R. M. Karp, Reducibilities among combinatorial problems, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York (1972), pp.85-103.
3. R. E. Ladner, On the structure of polynomial time reducibility, Journal of the ACM, 22 (1975), pp.155-171.
4. U. Manber, Introduction to algorithms, Addison-Wesley, New York (1989).

—本文作者任教於大同工學院應用數學系—