

食譜、廚師及鯉魚 (Recipe, Cook and Carp)

by Jehn-Ruey Jiang (江振瑞) at NCU (中央大學) (Copyright 2005-2013)

1. 食譜 (recipe) :

食譜是世界上流通最廣的演算法(algorithm)。根據韋伯字典的定義，凡是為解決某個特定問題的一步一步程序(a step-by-step procedure for solving a problem)均可稱為演算法。食譜可以一步一步解決我們日常三餐的做菜問題，因此也可稱為是一種演算法。

世界上有許多待解的問題(例如判斷一個整數 n 是否為質數的問題)，有時也存在一些特定的演算法可以解決這些問題。為比較這些演算法執行的快慢效率(efficiency)，我們以時間複雜度(time complexity)來衡量之。時間複雜度是一個相依於問題規模(problem size) n 的函數，而當問題規模 n 不大時，各演算法的執行時間很難分出高下，因此我們都考慮 n 很大時($n \rightarrow \infty$)的情形，我們採用大 O 記號(big O notation)來表示之。使用大 O 記號，演算法執行的快慢效率高下立見：

$$O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

以上所列時間複雜度，除最後兩個以外均為問題規模 n 的多項式(polynomial)，屬於這種時間複雜度的演算法稱為多項式時間演算法(polynomial time algorithm)；而最後兩個為問題規模 n 的指數式(exponential)，屬於這種時間複雜度的演算法稱為指數式時間演算法(exponential time algorithm)。一般而言，多項式時間演算法執行起來比較有效率(比較快)；而指數式時間演算法執行起來比較沒有效率(比較慢)。

2. 廚師 (cook) :

一個食譜的好壞，可以由其是否能引導人們完成味道正確(correctness)的菜，以及食譜做菜的快慢效率(efficiency)來判斷之。我們在前段中已提及，演算法可以使用大 O 記號來衡量快慢效率高下，但對於某個特定的問題，我們怎麼知道我們已經找到最佳(optimal)解法(演算法)了呢?因此，我們必須探索問題的解題步驟數下界(lower bound)。例如，以下是一個已被證明的結論：排序(sorting)問題的解題步驟數下界為 $\Omega(n \log n)$ 。(這裡採用 Ω 記號，因為談的是步驟數下界；而大 O 談的是步驟數上界。)

針對有些問題，我們目前找不到在最壞情況下(worst case)的多項式時間演算法，也就是說，它們目前僅有指數式時間的解法；但同時我們目前也無法證明它們的最壞情況下的解題步驟數下界為指數式。因此研究學者想盡辦法要為這些問題找出在最壞情況下的多項式時間演算法，或證明這些問題的最壞情況下的解題步驟數下界為指數式。然而，研究學者經常是白忙一場而一無所獲。

Dr. Cook 在 1971 年發表論文，使用非確定性(nondeterministic)演算法的模式來探討上述這些問題之間的關係。日常生活中實際可用的演算法為確定性的(deterministic)，因為它的每個步驟的執行均為明確的(definite)、有效的(effective);而 nondeterministic 演算法可以使用一個不明確的執行步驟 — Guess (或是 Choice)。Guess 可以在所有給定的狀況中，挑中對的狀況(答案)；而若無對的狀況，則 Guess 會隨意挑一個狀況。可想而知，現實生活中不可能真正存在 Guess

這種執行步驟，它僅可做為理論探討之用。

那麼，nondeterministic algorithm 有什麼用處呢?它的用處可大了，它讓 Dr. Cook 在 1982 年得了 Turing Award 大獎 (Computer Science 界的極高榮譽獎)。Dr. Cook 有一個以他名字命名的定理 — Cook Theorem (1971)

$$NP = P \quad \text{iff} \quad SAT \in P$$

(SAT 代表 the satisfiability problem ; NP 代表所有可以用 polynomial time nondeterministic algorithms 解決的問題所構成的集合; P 代表所有可以用 polynomial time deterministic algorithms 解決的問題所構成的集合)

Dr. Cook 證明：

$$\forall X \in NP: X \alpha SAT \quad (\alpha: \text{polynomially reduces to})$$

Dr. Cook 定義 NPC (NP-Complete)問題，並證明 SAT 問題是一個 NPC。一個 NPC 問題的定義如下：

$$Y \in NPC \quad \text{iff} \quad (Y \in NP) \wedge (\forall X \in NP: X \alpha Y)$$

3. 鯉魚 (Carp) :

SAT 是第一個 NPC，它有什麼用處?它的用處也很大，它讓 Dr. Karp(與 Carp 諧音)在 1985 年也得到 Turing Award。

Dr. Karp 利用 SAT 及 reduction (α)的遞移性，證明了另外 21 個問題也是 NPC。例如，他證明 $SAT \alpha 3\text{-SAT} \alpha CN$ (chromatic number problem):

$$\therefore \forall X \in NP: (X \alpha SAT)$$

$$\therefore \forall X \in NP: (X \alpha 3\text{-SAT}) \wedge (X \alpha CN)$$

$$\therefore 3\text{-SAT} \in NPC \quad \wedge \quad CN \in NPC$$

NPC 除了使人得獎之外，還有什麼用處嗎?有用的，因為每一個 NP 問題均可 reduces to NPC 問題。因此，一旦任何 NPC 問題可以使用 deterministic algorithm 在 polynomial 時間解決掉，則所有的 NP 問題均可以使用 deterministic algorithm 在 polynomial 時間解決掉，也就是 $NP=P$ 。

到目前為止，仍然沒有任何 NPC 問題被證明是 P 問題，而似乎也很難會有(也就是極有可能 $NP \neq P$)。一般認為 NPC 問題是難的(intractable)，因為到目前為止，任何解決 NPC 問題的 deterministic algorithm 在最壞情況下都具有 exponential 時間複雜度。但是仍然沒有人能夠證明 $NP \neq P$ ，這需要證明某個 NPC 問題的 worst case lower bound 是屬於 exponential 量級的，這仍是未解的難題。