食譜、廚師、鯉魚及琥珀 (Recipe, Cook, Carp, and Bernstein)

by Jehn-Ruey Jiang (江振瑞) at NCU, Taiwan (國立中央大學)

摘要:

這份〈食譜、廚師、鯉魚及琥珀〉是一篇以生活化比喻介紹計算複雜度（computational complexity）核心概念的短文。全文依序用「食譜（Recipe）」、「廚師（Cook）」、「鯉魚（Carp/Karp）」與「琥珀（Bernstein）」四個主題，帶讀者理解演算法（algorithm）、時間複雜度（time complexity）、P、NP、NP-complete（NPC）、NP-hard，以及量子計算中的 BQP 等概念。全文的主線是：先從演算法效率談起，再引入 Cook 對 NP 與 SAT 的奠基性結果，接著說明 Karp 如何利用歸約（reduction）把更多問題證成 NPC，最後再補充量子計算中的 BQP 類別。

更具體地說，第 1 部份把食譜視為一種一步一步解決問題的程序，因此可類比為演算法；並用大 O 記號介紹常見時間複雜度的等級，說明多項式時間（polynomial time）通常被視為有效率，而指數時間（exponential time）通常較沒效率。第 2 部份以 Cook 為核心，說明確定性（deterministic）與非確定性（nondeterministic）演算法的差別，並引入 Cook theorem 與 SAT 的關鍵地位：SAT 是第一個被證明為 NP-complete 的問題。第 3 部份轉到 Karp，說明他如何利用 SAT 與多項式歸約的遞移性（transitivity），證明更多問題也是 NP-complete。第 4 節則延伸到量子計算，介紹 BQP 是量子電腦在有界錯誤下可於多項式時間求解的問題類別，並說明其與 P、BPP、PSPACE 等類別之間的關係。

1. 食譜 (Recipe)：

食譜可說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(例如判斷一個大於 1 的整數 n 是否為質數的問題)，通常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。時間複雜度衡量演算法的執行步驟數，是一個相依於問題規模(problem size) n 的函數。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 很大的情形(例如，$n\to\infty$時)。我們採用趨近記號或漸近記號(asymptotic notation)中的大 O 記號(big O notation)來描述演算法的時間複雜度。使用大 O 記號可以得到一個函數的漸進上界，演算法執行的快慢效率因此可以高下立見：

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

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)函數，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)函數，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快慢效率最佳(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，以下是一個已被証明的結論：排序(sorting)問題的解題步驟數下界為Ω(n log n)。注意，這裡採用漸近記號中的大Ω記號(big omega notation)，因為大Ω談的是步驟數下界；而大 O 談的是步驟數上界。

針對有些特定問題，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証明它們在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。注意，我們在這裏特別強調最差狀況時間複雜度。因為假如一個問題在最差狀況情況下可以使用某個演算法在多項式時間複雜度解答，那其他狀況當然也可以使用多項式時間複雜度解答，但反之不必然成立。舉例而言，某些問題可以使用特定演算法在平均狀況下以多項式時間複雜度解決，但是在最差狀況下，所有解決該問題的演算法依然具有指數時間複雜度。

Dr. Cook 在 1971 年發表論文，使用目前尚且無法實作的非確定性(nondeterministic)演算法模型來探討上述特定問題之間的關係。日常生活中實際可用的演算法為確定性的(deterministic)，因為它具有輸入、輸出、並且是有限的(finite)，而且它每個步驟的執行均為明確的(definite)、有效的(effective)；但是，nondeterministic 演算法可以使用一個不明確的執行步驟 ─ Guess (或是 Choice)。Guess 可以在一個步驟中就可以在給定的選項中挑中對的選項進行後續的檢查(Check)，讓演算法以傳出成功訊息的方式結束；而若無對的選項，則 Guess 會隨意挑一個選項進行後續檢查，讓演算法以傳出失敗訊息的方式結束。可想而知，現實生活中不可能真正存在像 Guess 這麼強大的執行步驟，因此目前非確定性演算僅可做為理論探討之用。或許等待未來有新的計算設備出現，例如量子計算機(quantum computer)及生化電腦(biochemical computer)等，才有可能實現 Guess 這種步驟。

那麼，nondeterministic algorithm 有什麼用處呢?它的用處可大了，它讓 Dr. Cook 在 1982 年得了圖靈大獎(Turing Award)，這是 Computer Science/Engineering 界的最高榮譽獎。Dr. Cook 有一個以他名字命名的定理 ─ Cook Theorem (1971)，簡單描述如下:

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

(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 證明以下兩個命題(statement)：

1. $\forall X \in NP: X \alpha SAT$ (α: polynomially reduces to) (在此證明 SAT 是 NP-hard)
2. SAT 可以使用 nondeterministic algorithm 在多項式間解決 (在此證明 SAT 是 NP)

根據以上兩個命題，可以證明 SAT 問題是一個 NPC 問題。

Dr. Cook 是第一個定義 NPC (NP-Complete)問題的學者，他並證明 SAT 問題是一個 NPC 問題。因此 SAT 問題是第一個被證明是 NPC 問題的問題。一個 NPC 問題的定義如下：

$Y \in NPC \quad \text{iff} \quad (Y \in NP) \wedge (\forall X \in NP: X \alpha Y)$, where X and Y are problems.

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。

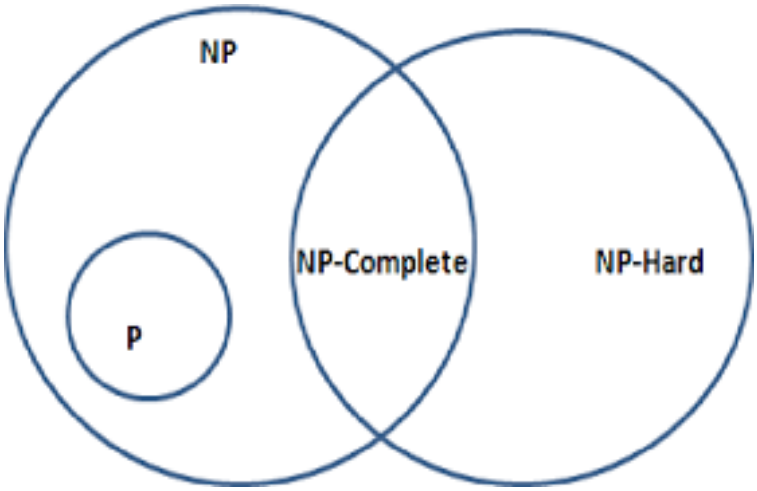
Dr. Karp 利用 SAT 及 reduction (α)的遞移性(transitivity)，証明了另外 21 個問題也是 NPC。例如，他證明 SAT α 3-SAT α CN (chromatic number problem)，而我們又很容易證明 (3-SAT∈ NP) Λ (CN∈NP)，因此我們有以下推論 ：

∀X∈ NP: (X α SAT)	(1) (By Cook Theorem)
∀X∈ NP: (X α 3-SAT) Λ (X α CN)	(2) (By reduction transitivity)
3-SAT∈NPC Λ CN∈NPC	(3) (By Eq. (2) and by NPC definition)

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 polynomially 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。這可說是一人得道，雞犬升天(一人: 某個 NPC 問題；雞犬: 所有的 NP 問題)。

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

綜合以上的說明，我們可以畫出以下的問題類別關係圖:

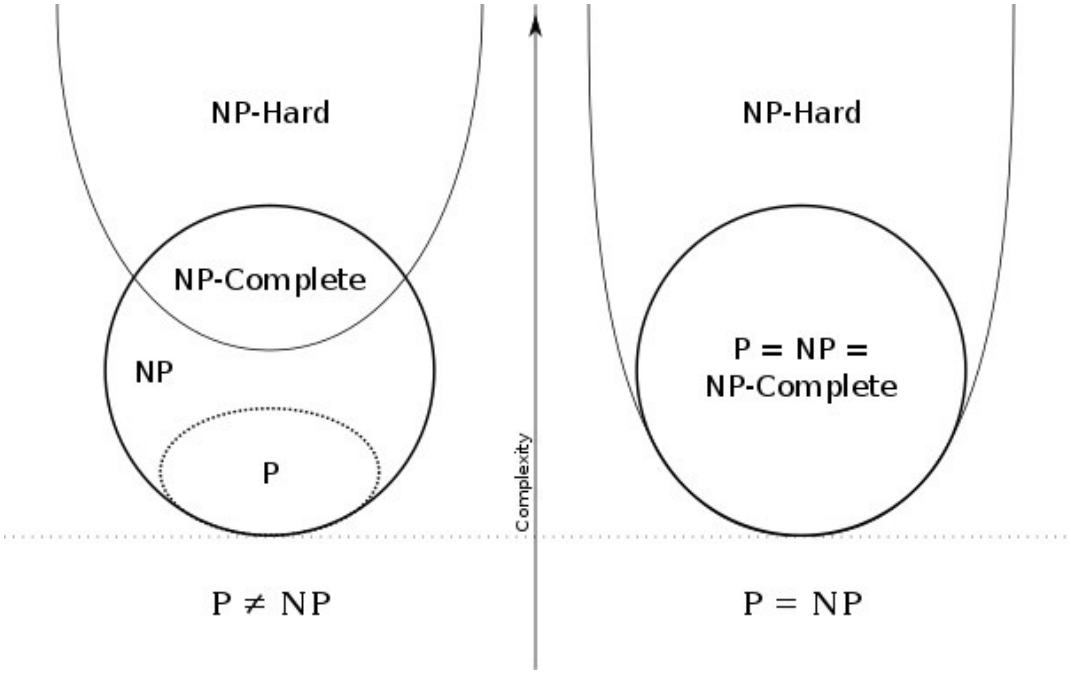


(Source: https://d3i71xaburhd42.cloudfront.net/d19cce0996d8236b2158123b5c9d98ea0b190e94/15-Figure1-1.png)

以下到分隔線之間內容為修改自 ChatGPT 對 P、NP 及 NP-hard 的說明:
P 代表了可以在多項式時間內解決的問題，NP 代表了可以在多項式時間內驗證解答的問題。NP-hard 則是指那些至少和 NP 中的問題一樣困難的問題。

根據定義，P 是 NP 的子集，這意味著所有 P 類型的問題同時也是 NP 類型的問題。NP-hard 則是一個更廣泛的類型，包含了所有和 NP 問題同樣難解的問題。

至於 P=NP 或 P≠NP 則可以透過以下的圖來了解:



(Source: https://commons.wikimedia.org/wiki/File:P_np_np-complete_np-hard.svg, by Behnam Esfahbod (CC BY-SA 3.0))

實際上，P≠NP 或是 P=NP 是千禧年大獎難題(Millennium Prize Problems, MPP)的七道難題之一。MPP 由美國的克雷數學研究所(Clay Mathematics Institute, CMI)於 2000 年 5 月 24 日公佈，其解題總獎金為 700 萬美元。解題沒有時間限制，但是解答必須發表在知名的國際期刊，並經過各方驗證，只要通過兩年驗證期和專家小組審核，每解破一題可獲獎金 100 萬美元。俄國數學家佩雷爾曼(Grigori Perelman)於 2003 年解出其中的一道題，獲得國際數學聯合會的菲爾茲獎，但他名聲與金錢不感興趣，因此拒絕領菲爾茲獎與獎金，也拒絕領 MPP 獎金。

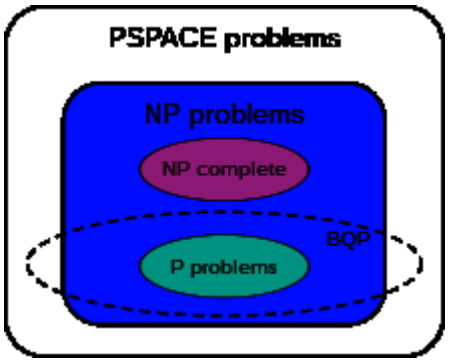
4. 琥珀 (Bernstein)： (以下到分隔線之間內容部份修改自 ChatGPT)

Bernstein 在德文中的意思為琥珀。在 1993 年，計算機科學家伊桑・伯恩斯坦（Ethan Bernstein）和烏梅斯・瓦齊拉尼（Umesh Vazirani）定義了一個新的複雜性類型，稱為有界錯誤量子多項式時間(Bounded-error Quantum Polynomial time, BQP）。

BQP 是一個描述使用量子計算機在多項式時間內解決問題的複雜性類型。這意味著 BQP 類型的問題可以在多項式時間內使用量子計算機找到接近正確解的解答。在這個定義中，允許有一定的錯誤概率，但這個錯誤概率是有界的，即在多項式時間內可以控制在一個較小的範圍內。

BQP 的定義與其他複雜性類型相比具有重要的意義。例如，P 類型是在多項式時間內可以使用傳統計算機解決的問題，而 BQP 類型則擴展了計算的能力，利用量子計算機的特性解決更複雜的問題。此外，BQP 類型還與其他複雜性類型如 NP 類型和 PSPACE 類型之間存在著關聯。

==========



BQP 與各類型問題的可能關係圖

(Source: https://commons.wikimedia.org/wiki/File:BQP_complexity_class_diagram.svg by Booyabazooka, Public Domain)

PSPACE 是計算複雜度理論中能被確定型圖靈機(deterministic Turing machine)利用多項式空間解決的判定問題集合，是 Polynomial SPACE 的簡稱。(PSPACE is the set of all decision problems that can be solved by a Turing machine using a polynomial amount of space.)

BQP，在計算複雜性理論中，代表著「有界誤差、量子、多項式時間」(Bounded error, Quantum, Polynomial time）。它指的是可以在多項式時間內由量子電腦解決的問題，對於所有實例，錯誤機率最多為 r, 0< r < 1/2 (註: 許多研究論文採用 r = 1/3 的設定)。換句話說，有一個保證在多項式時間內運行的演算法可供量子電腦使用。在演算法的任何一次執行中，它錯誤的機率一定比 1/2 小。如此經過多次執行演算法會使大多數執行結果都錯誤的機率會呈現指數量級地減小，在演算法執行趨近無窮多次時，這個機率會趨近於 0。因此，可以透過執行演算法非常多次，然後以投票的方式，以大多數的結果為最終結果。

目前有一些實際有使用價值的問題被認為屬於 BQP，但被懷疑可能不屬於 P。目前，只有三個這樣的已知問題：整數質因數分解問題(以 Shor 演算法解決)、離散對數問題(以 Shor 演算法解決)、量子系統模擬(以通用型量子電腦之量子線路解決）。

BQP 這個類別是為了量子電腦而定義的。對於一般圖靈機加上一個隨機來源(ordinary Turing machine plus a source of randomness)的對應類別是有界錯誤隨機多項式時間(Bounded-error Probabilistic Polynomial time, BPP)。BQP 包含了 P 和 BPP，並被 PSPACE 所包含。有機會我們再多談談 BPP 與其他問題類別及它們與 BQP 的關係。