比爾蓋茲在大學裏所寫的論文

林耀鈴

靜宜大學資管系

yllin@pu.edu.tw

William H. Gates
Chairman and Chief Software Architect,
Microsoft Corporation

Christos H. Papadimitriou

Professor, Computer Science Division,
University of California at Berkeley

比爾·蓋茲在你心目中的印象是什麼？大概每個人的印象都是：微軟的老闆、世界最有錢的人。的確，比爾·蓋茲三年來一直穩居世界首富之位，到 2000 年其身價達到了 765 億美元。雖然由於今年（2001 年）全球科技股普遍下跌，比爾·蓋茲的身價也隨之降到 500 億美元上下。因此 2001 年 04 月《星期日泰晤士報》所列出的年度世界富豪排行榜時，沃爾瑪特連鎖超市總裁瓦爾頓取代了蓋茲而成爲世界新首富。不過，白手起家，一位哈佛中輟生與高中好友在車庫中成立的小小微軟公司，日後變成叱剎風雲的微軟而形成今天電腦軟體界無人能項其背的「巨」軟仍然是人人津津樂道的傳奇。

許多人也都知道，蓋茲先生在 20 歲，哈佛只唸到大三時就中途輟學，與他的好朋友、兒時玩伴 Paul Allen 跑去開了一家叫做 Microsoft 的公司（1975 年）。有些人也許會以爲蓋茲一定是在學校混不下去，沒辦法適應哈佛的高標準大學生活，才出此下策；只是幸運地趕上時代潮流，迷迷糊糊就成了世界首富。事實上，我們的蓋茲先生在學術方面的潛力可一點點都不會輸給他的大學同學。我們這篇文章要介紹給讀者的，就是：蓋茲在哈佛輟學之前，曾經與他當時的老師 Christos H. Papadimitriou 共同研究過一個很有趣的數學問題，並且在 1977 年時寫了一篇科學（或者，更精準的說是離散數學）論文。這篇文章在2年後刊登在 Discrete Mathematics 27 (1979), pp 47-57. 文章名稱爲： “Bounds for sorting by prefix reversal.”

這篇文章是這樣開始的：一位餐廳侍者在送出一疊煎餅 (pancakes) 到顧客之前，發現廚師實在太混了，這些煎餅大小不一，混雜在一起堆成一疊，客人實在不會有太多好感。因此，在送出這些煎餅之前，這位侍者會使用一片鍋鏟將這些煎餅重新排成一疊由小而大排列的煎餅。不過，由於盤子太小，我們不能夠將煎餅平鋪後再重新排一次，而只能用鍋鏟卡在某個煎餅的下方，將上頭全部的煎餅一起做翻面的動作。比如說，這裡有三塊煎餅，最大的一塊叫做 3,中間大小的是 2,最小的一塊就是 1。如果 1 號在盤子的最下面，3 號煎餅疊在它上面，最上面的煎餅就是 2 號；由上而下的排列就是：(2,3,1)。那麼，侍者就可以先用鍋鏟卡在第二個煎餅的下方，將前 2 個煎餅做翻面的動作，這三個煎餅順序就成了：(3,2,1)；然後，侍者就可以再用鍋鏟卡在第 3 個煎餅的下方，將全部煎餅做翻面的動作，這三個煎餅順序就成了：(1,2,3)。好了，我們這位高興的侍者就可以趕緊將這些排好的煎餅送出去給客人了。

看到一個可愛的問題，離散數學家都會反射動作式地想要思考更難、更一般化的問題。比如說，如果原來的是 4 個煎餅，順序是：(2, 4, 3, 1)，那麼……我們要用鍋鏟做多少個動作呢？有一個很簡單的想法是這樣子：我們可以先找到最大的那片煎餅所在的位置，在它下面用鍋鏟做翻面的動作；如此一來，最大的一塊餅就會跑到最上面來，接著我們用鍋鏟將整疊煎餅翻轉過來，於是最大的一塊餅就會跑到最下面。爲了說明方便，我們用 [2] 這個符號代表我們想要將前 2 個煎餅翻面。所以，我們就用 [2] -> (4, 2, 3,1) 來表示這個動作與結果。然後，在第二個動作我們得到：[4]->(1,3,2,4)。再來，我們再用兩個動作：[2]->(3,1,2,4) ; [3]->(2,1,3,4) 就可以將 3 號煎餅丟到 4 號上面。再來，最後一個動作就是 [2]->(1,2,3,4)。

很明顯地，如果我們有 n 個煎餅，每次最多花 2 個動作就可以將最大號的煎餅丟到下面。因此，對於 3 號到 n 號煎餅（共 n-2 塊煎餅），我們可以在 2(n-2) 次動作將它們搞定，剩下來的 1 號 2 號煎餅，最多只要花 1 次翻面的動作就行了。也就是說，n 個煎餅，我們保證可以在 2n – 3 次動作內，完成正確排序動作。剛剛的 4 塊煎餅，我們就用了 5 個動作。

一個有趣的問題就來了：我們怎麼知道這樣做是最好的方法呢？我的意思是：5 次動作會不會太「浪費」了？有沒有可能少於 5 個動作就能夠把它排完？答案是可以。例如：2431 [3]-> 3421 [2]-> 4321 [4]-> 1234 就可以只用 3 個動作就將它搞定。可是，問題還在，我們怎麼知道這樣做是一定是最好的方法呢？ 我們還是問，3 次動作會不會太「浪費」了？

爲了回答這個問題，我們仔細看看原來的 4 個煎餅：(2, 4, 3, 1)。我們發現，前兩個煎餅 2 與 4 連起來不是兩個連續數字。我們就把這種現象稱爲一個「斷點」。一個斷點代表什麼意義？注意到：如果我們不用鍋鏟插入 2, 4 之間，那麼，這個斷點就永遠在這裡，最後的結果就不可能是一疊排好的煎餅了！（請留意最後的結果是 (1, 2, 3, 4) 上面完全沒有任何斷點。）因此，原來的排列 (2, 4, 3, 1) 就有： 2-4, 3-1, 再加上最後的 1 與 「鍋底」之間也算是一個斷點，這個排列就剛剛好有 3 個斷點。結論

是：要將 (2, 4, 3, 1) 排成 (1, 2, 3, 4)，我們最少最少也要用 3 次鍋鏟動作才能去掉所有的斷點。不過，我們剛才不是用了 3 個動作將它排好嗎？因此，我們就可以很得意地說：我們剛剛的 3 個動作算是最節省，最有效率的方法之一。

如果把最後一個煎餅與鍋底的斷點也算進去，那麼 n 個煎餅最多可能會造成 n 個斷點。這是不是說，如果我們能夠一此「解決」掉一個斷點，最後就能夠把這些煎餅排成正確順序呢？問題倒也沒有那麼容易，例如，6 個煎餅可能排成：536142 這個排列，讀者可以自行嘗試一下，非得要用 7 個動作才能排成 123456，只用 6 個動作是不可能的。

目前爲止，我們仍然不知道如何用計算機（更別提用人腦）來有效率地計算出最佳解法。Gates 在這篇文章中的貢獻之一就是提出一個電腦可以執行的方法（我們的行話叫做「演算法」），對 n 個煎餅的任何排列，都能找出一組鍋鏟動作，並且保證可以在 (5n+5)/3 次動作之內得到由小而大的正確排列。比較遺憾的是，Gates 與 Papadimitriou 所提出的演算法，雖不能說是非常困難，不過也實在是太過繁瑣，沒有辦法詳細介紹給讀者分享。不過，基本上他們用的策略就是用剛剛提到的「去掉斷點」這個精神，想辦法先去除掉大部分的斷點（事實上是去掉 n-2 個斷點）；最後，再對剩下來的兩個斷點做處理，由此得到一組「還不錯的」解答。這種策略在演算法裡頭是叫做一種「經驗法則」(heuristics)。這裡頭最困難的技巧還是在如何說明這個 heuristic 方法爲什麼最多產生 (5n+5)/3 次動作。筆者猜測是 Gates 想到這個 heuristics，而 Papadimitriou （一位在今日世界計算機理論界仍是大大有名的人物）最後是用了線性規劃理論裡頭的 duality theorem 而推導出最後 (5n+5)/3 這個數據。

得到一個 heuristics 說明 (5n+5)/3 動作就一定可以將 n 個煎餅排好順序的確不錯。但是究竟「最佳解」需要多少次動作？我們就用函數 f(n) 來表示：給定 n 個煎餅的任何順序，對最難排好的排列而言，我們最少需要的鍋鏟動作數。也就是說，對於任何 n 個排列，我們用 f(n) 次鍋鏟動作保證可以將它排好，但是 f(n)+1 次就一定是太浪費了。很明顯地，利用剛剛提到 Gates 的 heuristics ，我們知道 f(n) 最多就是到 (5n+5)/3, 也就是說，$f(n) \leqq (5n+5)/3$. 同時，既然 n 個煎餅最多有可能有 n 個斷點，很明顯地 $n \leqq f(n)$. Gates 與 Papadimitriou 在這篇文章中的另一個貢獻就是提出 f(n) 在 n 夠大之後不會只有 n 那麼小。論文中，他們提出一些觀點，基本上，他們用的方法是利用這個反覆一段長度爲16的基本序列：

(1, 7, 5, 3, 6, 4, 2, 8, 16, 10, 12, 14, 11, 13, 15, 9)

反覆 k 次之後，他們論證：至少需要 17k 以上的翻面動作才能將這些 16k 個煎餅排好。因此證明了 $(17/16)\,n \leqq f(n)$, 文章結論就是說：$(17/16)\,n \leqq f(n) \leqq (5n+5)/3$。

知道 f(n) 最少是 (17/16)n ，而且也有一個 heuristics 說明 (5n+5)/3 動作就一定可以將 n 個煎餅排好順序的確不錯。但是無論如何，人們還是會想知道「最佳解」f(n) 的確切答案。很明顯的， f(1) = 0, 一個煎餅本身就已經排好; f(2) = 1, 兩個煎餅

最多翻一次就行。 f(3) = 3, 因爲 132 怎麼翻都需要 3 次 （注意到它其實只有兩個斷點。）下面這個表格列出 f(n) 的前 13 個數值。

n	1	2	3	4	5	6	7	8	9	10	11	12	13
f(*n*)	0	1	3	4	5	7	8	9	10	11	13	14	15

其實，這個表格得來不易：前7個數字，是由在計算機理論界鼎鼎大名的 Garey and Johson (以及 S. Lin) 在 1977 年算出來的。 f(8) 與 f(9) 則由 Robbins 在1979 年算出。一直到18年後，f(10) 到 f(13)才由M. Heydari and I. H. Sudborough 在 1997 年算出來的。事實上，他們最後還改進了原來 Gates 他們所提出了原來 f(n) 的下限。他們提出類似的論證：利用這個反覆一段長度爲14的基本序列：

(1, 7, 5, 3, 6, 4, 2, 8, 14, 12, 10, 13, 11, 9)

反覆 k 次之後，他們論證：至少需要 15k 以上的翻面動作才能將這些 14k 個煎餅排好。因此證明了 (15/14)n ≦ f(n) 。比較有趣的是，20 年都已經過去， f(n) 的下限只提升了一點點，而 Gates 的演算法， f(n) ≦ (5n+5)/3 的 f(n) 上限到現在還是沒有任何改進。

最近，這個「玩具問題」，其實也已經脫離了純粹只是與比爾·蓋茲掛勾的趣味問題。在生物資訊領域中，生物細胞內部的 DNA 序列由於各種分子生物學的理由，而造成 DNA 序列中的兩個鹼基的連接關係斷裂，一個片段可能由原來的位置漂流出來後，以「相反的方向」與原先的兩個斷點反向回黏回去。也就是說，原先可能是一段 DNA：

AATGCGTTAAGCCTAAGATCCTTA

在回黏之後會變成：

AATGCGTATCCGAATAGATCCTTA

這問題可以看成用「煎餅夾」夾住原來煎餅堆的中間一疊煎餅，並將它們翻轉過來。這個問題，計算機科學家已經可以證明這問題幾乎不容易得到有效率的演算法；並有其他的變形問題可以容易地得到最佳解。不過，這些問題似乎已經脫離我們這篇文章的主題太遠，有興趣的讀者可以參考下面的參考文獻，自行查證。

結語

前面提過 Papadimitriou 在今日世界計算機理論界仍是一號鼎鼎大名的人物，很多計算機科學界的學者對他是敬重有加。因此，在這行裡曾經流行一個笑話：話說有位學者有一天在路上遇見 Papadimitriou ，他也知道從前 Gates 跟他做了這篇研究論文。於在閒聊中就奉承地說：「唉啊！Gates 當初如果跟著您繼續學藝，今天他的學問必然是不可限量；他當初沒有追隨大師繼續做下去實在是太可惜了。」沒想到 Papadimitriou 卻回答說：「哪裡，我那時沒有跑去追隨 Gates 到 Microsoft， 我才是真正的可惜！」

參考文獻

- W.H. Gates and C. Papadimitriou, "Bounds for sorting by prefix reversal." *Discrete Mathematics* 27 (1979), pp 47-57.
- M. Heydari and I. H. Sudborough, "On the Diameter of the Pancake Network," *Journal of Algorithms*, 25 (1997), pp. 67-94.
- V. Bafna and P. Pevzner "Genome rearrangments and sorting by reversal," *SIAM J. Comput.*, 25(2), pp. 272-289, 1996.
- Caprara, A. "Sorting permutations by reversals and Eulerian cycle decompositions." *SIAM J. Discrete Math.* **12**(1), pp. 91-110, 1999.
- Hannenhalli, S. and Pevzner, P.A. "Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals." *J. ACM* **46**(1), 1-27, 1999.

相關網站：

http://www.microsoft.com/billgates/bio.asp

http://www.cs.berkeley.edu/~christos/

http://www.cut-the-knot.com/SimpleGames/Flipper.html

http://www-math.mit.edu/~tchow/mathstuff/prefixrev