# 數學、資訊科學與數字遊戲

**數學是科學之母，也是資訊科學的重要基礎。沒有相當的數學基礎，也就很難設計出一個好的遊戲程式。**

**劉昭麟**

資訊科學與工程近幾年來隨著網際網路的盛行而受到社會各界高度的重視。從資訊科學相關技術在新聞媒體的高曝光率，從它們在新新人類的日常生活中所佔的比重，甚至從股票市場投資者的討論節目中，都可以簡單看到這一學科對我們的生活方式正在產生重大的影響。

在大學的資訊科學與工程科系中，同學們可以學到許多電腦軟、硬體的技術。在硬體知識方面，有電子學、數位系統和計算機組織等；在基礎軟體系統的知識方面，有系統程式、作業系統、資料庫和編譯器設計等；在程式設計的方面有教導 C、 Java 和 C++ 等程式語言和資料結構等課程；此外還有比較進階的課程，如網際網路、電腦圖學、資訊安全、人工智慧和計算機結構等課程。資訊科學與工程科系會提供同學系統化的課程，讓同學深入了解電腦的工作原理，學習如何製作程式，讓電腦為人類服務，並進而做好進入職場從事資訊相關工作的準備。

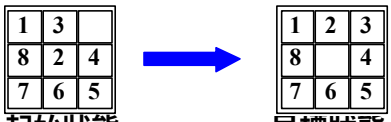
除了上述和資訊直接相關的專業科目之外，資訊科系的大學部也會提供同學修習如微積分、機率統計、線性代數甚至工程數學等數學相關的基礎課程。這些科目表面上看起來似乎和寫程式、製作電腦沒什麼直接關係，所以常常被同學們忽略而沒有認真的學習。

事實上，數學常常是製作功能很強大的程式的重要基礎。例如，能夠聽懂我們所講的話的語音辨認程式和能夠辨認手寫中文的程式，都用到很複雜的機率和統計的知識；能夠畫出生動、逼真的三度空間動畫的電玩程式中，則要用到許多的線性代數的知識；更有許多人應用複雜的數學模式，去預測經濟景氣的走向和股價的變化趨勢。

數學是科技之母，即使是資訊科技也絲毫不能例外。學會許多程式語言，就好像學會了如何舞刀弄劍的基本技術；若想要成為功夫高手，還得從特殊的武功密笈中進一步提昇自己的武功等級。而數學正是資訊科學中進修武功密笈和製作高級程式不可或缺的一個重要工具。以下我們就透過兩個利用電腦程式玩遊戲的例子，來感覺一下數學在設計遊戲程式中可以扮演的奇妙功用。

## 排列數字遊戲

這個遊戲是要將三乘三方格中的數字由所給定的排列，經過合法的移動步驟，變成所要的排列方式。如圖一所示，我們要將左圖起始狀態中的排列，變成右圖的目標狀態。在九個小方格中，會放 1 到 8 的八個數字和一個空白格。而所謂合法的移動步驟，是指我們只能將數字移到緊鄰的空白格中。數字移動之後，原本的位置則由空白取代。圖一所舉的例子相當地簡單。只要依序將數字 3 和 2 依順時鐘方向各移動一格就可以達成目標了。




起始狀態　　目標狀態

**圖一　排列數字遊戲**

但是對於比較複雜的問題，我們可就得大費周章，靠嘗試錯誤的方法找答案了。各位能夠很快找出將圖二的兩個起始狀態轉變成圖一目標狀態的步驟嗎？


起始狀態二　　起始狀態三

**圖二　你能看出答案嗎？**

如果你不能解決這個問題的話，讓我們一起來看看電腦是如何幫我們解決的。電腦長處之一就是它們擅長快速執行既枯燥又重複的工作；所以，我們可以設計一個程式，讓電腦幫我們搜尋這個遊戲的答案。事實上，我們可以從資訊科學的文獻中，找到許多排列數字問題的解決方法。現在就利用圖一的例子來看一個比較容易瞭解的方法。

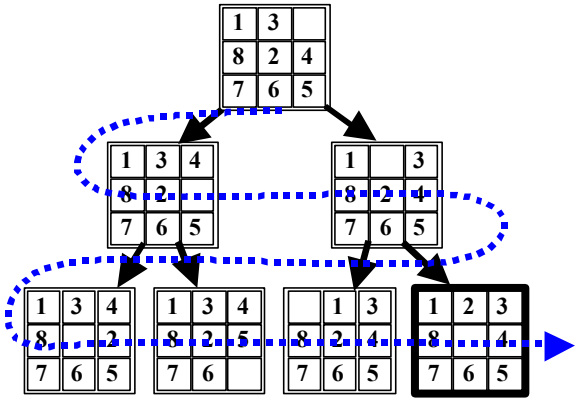
當我們通知電腦程式起始和目標狀態後，電腦可以有系統地靠窮舉法找出答案。從圖一的起始狀態開始，

我們只有兩個合法的移動步驟：移動數字 3 或 4。如果程式選 4，會得到圖三的左圖；如果程式選 3，則會得到圖三的右圖。但不管程式選擇哪一個，圖三的兩個狀態都不是我們想要的目標狀態。雖然如此，程式還是可以從這些中間狀態繼續搜尋答案。從圖三的中間狀態一出發，程式可以選擇移動數字 4、2 或 5。但是，這幾個方向都不會立即讓我們達到目標狀態。（如果此刻選 4，就會回到最原始的起始狀態，所以聰明的程式是不會做這樣不聰明的動作的。）而從中間狀態二出發，程式可以選擇移動數字 1、2 或 3。如果選擇移動 1 或 3，也不會立即達到目標狀態。但是如果選擇移動 2，我們的程式，就成功找到圖一問題的答案了：先移動 3 再移動 2。


中間狀態一　　中間狀態二

**圖三　搜尋答案的部分過程**

我們剛剛是利用一種叫做寬度優先搜尋法(Breadth First Search)的演算法來找到圖一的答案。圖四用圖形來總結這個搜尋過程。



**圖四　利用寬度優先搜尋法尋找排列數字遊戲的答案。虛線的箭頭方向表示程式搜尋的先後順序。**

前面提過，實際上有許多的搜尋法可以用來找排列數字遊戲的答案。這些搜尋法的設計，大致上都是藉由有系統的方法，找遍所有從起始狀態開始而可以藉著合法的移動步驟所達成的中間狀態，並檢查這些中間狀態是否是我們所要的目標狀態。當搜尋程式找到一個和目標狀態一樣的中間狀態時，程式就找到問題的答案了。

我們可以利用同樣的演算法，幫我們來找尋圖二中兩個起始狀態的答案。它可以找到起始狀態三的答案：依序移動數字 15314254267812。但出乎許多人意料的是，經過奮力的搜尋之後，我們的程式會跟你報告，它無法將起始狀態二轉變成目標狀態！為什麼呢？是不是我們的程式寫錯了呢？有趣的是：不是的！實際上，的確沒有人能夠依據遊戲規則，將起始狀態二轉變成目標狀態。

這個排列數字遊戲由來已久，早在一百多年前就已經風行美國了，只不過當時流行的是四乘四的形式。西元 1879 年由美國數學學會所出版的論文期刊 *American Journal of Mathematics* 甚至還有文章討論這個遊戲哩！而數學家早就找到方法能夠證明：依據我們所陳述的遊戲規則，絕對無法將一些狀態（如圖二的起始狀態二）轉成另一些狀態（如圖一的目標狀態）。依據數學家們所得的結果，我們不需要靠窮舉法和電腦快速的運算，就可以知道沒有任何人能夠依據遊戲規則，將起始狀態二轉變成目標狀態。

延用圖一的例子，我們來看看這個神奇的方法。首先，我們用數字 9 代表空白，將起始狀態和目標狀態，從上而下、從左而右，分別改寫成下面數列的形式：

1 3 9 8 2 4 7 6 5 和 1 2 3 8 9 4 7 6 5。

然後，我們算一算數列中有多少對數字是比較小的數字是出現在比較大的數字的右邊，將這個總數寫下來。以起始狀態的數列為例：1 的右側都沒有比 1 小的數，2 也沒有，3 的右側有一個數字小於它（也就是 2），4、5、6、7、8 和 9 的右側分別有 0、0、1、2、5 和 6 個比較小的數字；所以總數是 0+0+1+0+0+1+2+5+6=15。重複這個步驟，我們可以算出來目標狀態數列的總數是 11；起始狀態二和起始狀態三的總數則分別是 22 和 23。

眼尖的你或許已經找到結論了：如果總數一樣是奇數，這些狀態才是相通的，而問題才有答案。沒錯！你已經看到一半的答案了。

除了檢驗總數是奇數或偶數之外，我們還要檢驗起始狀態中空白格的位置和目標狀態中空白格的位置。如果這兩個空白格的最近距離是偶數格，那麼總數一樣是偶數的狀態才能互通，總數一樣是奇數的狀態也能互通，但是這兩類狀態不能互通。如果這兩個空白格的最近距離是奇數格，則是奇偶互通，其他的組合不通。前面所舉的例子，都屬於空白格的最近距離是偶數格的情形；所以只有起始狀態二不能轉變成目標狀態。

由這個排列數字遊戲，我們可以看到數學和資訊科

學相輔相成的例子。單靠搜尋程式，電腦必須費很大的功夫才能確定所給的問題有沒有答案；而數學家的發現，雖然可以讓我們用很快的方法，確定問題是否有答案，但是，如果問題有答案，數學理論並不會告訴我們答案是什麼。綜合兩者後，既可以快速確定問題有沒有答案，也可以在問題有答案時找到確實的答案。

## 猜數字遊戲

接著，再來看一個也是許多人都已經很熟悉的遊戲。這個猜數字遊戲要由兩個人玩，雙方各先從 0 到 9 的數字中挑出 4 個不同的數字，組合成 4 位數；而遊戲的目標是要猜到對方的數字；遊戲的過程則由雙方輪流猜對方的數字，被猜的一方有義務據實提供另一方所猜數字的正確性。至於數字的正確性由「幾 A 幾 B」表示：如果對方猜的數字是你所選的數字，且位置正確，則以 A 表示；如果對方猜的數字是你所選的數字，但位置不正確，則以 B 表示。假設你選擇 1234 作為底牌，而對方猜的是 3894；那麼，你必須跟對方說 3894 是 1A1B，因為對方猜中了你的 3 和 4，而且 3 的位置不正確（得一個 B），4 的位置正確（得一個 A）。依此規則，如果對方猜 1298，則你應該回覆 2A（因為 1 和 2）；如果對方猜 3194，則你應該回覆 1A2B（因為 1、3 和 4）。如果四個數字都被猜到了，而位置也正確，則是遊戲結束的時候（當然還是得讓雙方都輪完才決定勝負）。

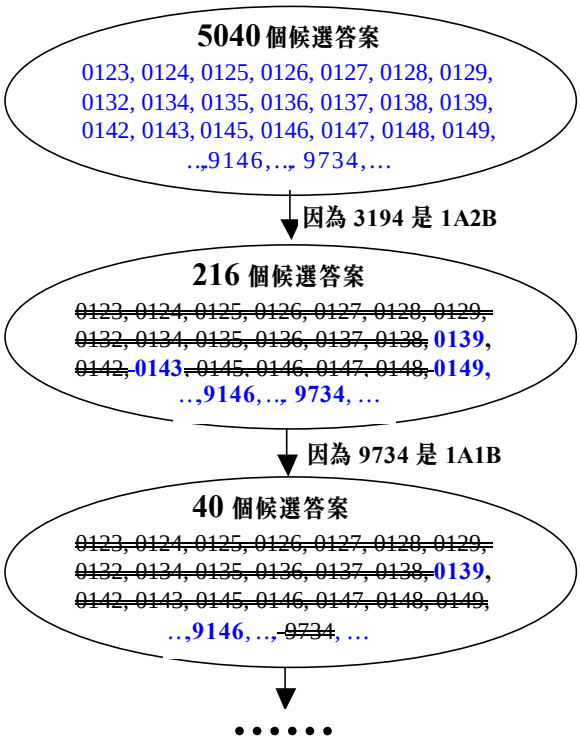
玩過這個猜數字遊戲的人，或許會覺得要寫一個電腦程式來玩這樣的遊戲有點困難。如果電腦不需要猜你的數字，而只是產生一個數字由你來猜，並且負責回應，那麼程式就很好寫了。比較困難的是，如何寫一個能猜你的數字的程式。

讓我們好好地思考一下這個問題：當我們猜 3194，而對方回覆 1A2B 時，我們該如何應用這樣的資訊呢？透過這一項訊息，我們可以知道 1234 可能是答案；因為如果對方的底牌真的是 1234，那麼我們猜 3194 時就會得到 1A2B。當然答案也可能是 1534、4793、9147 ..等共 216 個可能的答案。

為什麼當我們得知 3194 是 1A2B 時，我們就知道可能的答案只有 216 個呢？這時排列組合的知識就能派上用場了！首先回想一下遊戲規則；因為對方要從 10 個數字中找出 4 個不同的數字作排列。所以原本對方總共有 10!/6!=5040 種可能的答案。當我們得知 3194 是 1A2B 時，對方的底牌必須有 3 個數字是來自 3194，這三個數字必定是 319、314、394 或 194 等 4 種組合的其中一個。不管實際上是這 4 種情形的哪一種，這 3 個數字必須有一個是位置正確(有 3 種選擇)，另外兩個則是位置不正確（也有 3 種選擇)。此外，因為 3194 只包含了對方答案的 4 個數字的 3 個數字，所以對方底牌中必須有一個數字是來自於 3194 之外的一個數字，因此有 6 種可能。依此推論，我們可以依據排列組合的知識，原本 5040 個可能的答案中，只有 4*3*3*6=216 個符合這樣的描述。依照類似的方法，我們可以推算出，如果你猜的第一個數字得到 1A3B 和 0A0B 時，可能的答案的個數就分別變成 8 個和 360 個了。

依據以上的討論，我們已經發現了能讓電腦程式猜對手數字的基礎了。首先在程式中建立一個包含所有 5040 個可能的答案的資料庫。當程式首次猜對方的數字，並且得到回應之後，就可以依照所得的回應從 5040 個答案中刪除不可能的答案。譬如說，程式首先猜 3194 並且得到 1A2B 的回應時，程式就可以一一檢驗所有的 5040 個可能的答案。從 0123 開始：如果 0123 是答案的話，那麼程式猜 3194 時，就應該得到 1B 的回應。假設對手遵守遊戲規則，且不犯錯的情形下，0123 就絕對不是答案（否則對手必須跟程式講 3194 是 1B)；因此程式可以將它從現有的可能的答案的資料庫中刪除。程式接著檢驗 0124：如果 0124 是答案的話，則程式猜 3194 時，對方應該回應 2A；所以 0124 必定也不是答案，程式因此可將它自資料庫中清除。接著程式會去檢驗 0125、0126、0127、0128、0129、0132、0134、0135、0136、0137 和 0138，並且將它們自可能的答案的資料庫中清除。當程式檢驗到 0139 時，程式會發現它是一個可能的答案，因為對 0139 而言，3194 的正確性是 1A2B，所以程式將 0139 保留在資料庫中。程式就是這樣依序依據對手的回應檢查所有的可能答案，刪除不可能的候選答案，保留可能的答案。依據這樣的步驟，程式檢驗過現有 5040 個可能的答案後，自然會剔除 4824 個已經不可能是答案的數字，只留下仍可能是答案的 216 個數字。當下一次輪到程式猜數字時，程式就從這剩下的 216 個可能的答案中任選一個去猜，然後再重複上述的過程，依據最近一次所得的回應，再一次篩選和清理資料庫。如下圖所示，這個程式基本上就是不斷從資料庫中選一個數字猜測對方答案，然後再依據對手的回應清理可能

答案的資料庫，直到猜到對方數字為止。




**圖五　利用篩選法玩猜數字遊戲（假設對手底牌是 9146）**

這個方法看起來很簡單，實際上也很有效。依據上面的描述所建立的程式執行得很快。在 Pentium 200MHz 的機器上，最耗時間的第一次檢驗，只需要一秒鐘左右就可以完成了 5040 個數字的篩選工作。而依據筆者的實驗，用這個方法在一萬次不同的測試中猜對手的數字，平均猜 5.2 次就可以找到對手的答案。由於運氣的關係，這個程式曾經猜測 2 次，就找到對手的答案了。不過，還沒有過只用一次就猜到對手底牌的紀錄。（雖然理論上有可能發生首次就猜中的情形，但是這樣的機會非常地小，只有 1/5040 的機會喔！）另外在筆者的測試中，這個方法最多花 8 次的猜測才找到對手的答案。但是比起一般人的功力和耐力的話，這個程式的整體表現，已經稱得上是既快又準了，而且還有可以連續玩一萬次的超人耐力呢！

透過排列組合的觀念導引，我們已經找到一個可以玩猜數字遊戲的程式。而且經過實驗證實，我們所討論的方法，不僅執行時間很短，準確性也相當高。

事實上，我們的猜數字遊戲程式的準確性仍然有改善的空間。它在已經算相當老舊的 Pentium 200MHz 的機器上執行時，最多只讓對手等一秒鐘而已，只要引用比較複雜的方法，就可以進一步減少程式猜中對手底牌所需要猜的次數了。舉例來說，在前述的程式設計中，當我們的程式第二次猜對手底牌時，程式是從 216 個剩下的可能答案中，任意選一個來猜對方的數字。其實我們可以用特殊的方式，進一步比較這 216 個可能答案，給每一個數字一個分數來代表它們進一步引導我們儘快找到對手底牌的可能性。然後選擇分數最高的一個數字來猜對手的底牌。

在資訊科學和通訊科技中，有一門叫做「資訊理論」(Information Theory)的課；另外在人工智慧和決策科學中還有一個叫做「決策理論」(Decision Theory)的研究主題。這些和機率理論密切相關的專門知識，都可以提供我們相關的知識，來計算這樣的相對的分數。雖然依據這些進階理論所建立的程式，並不保證每一次都會用比較少的次數猜到對手的底牌；但是平均而言，加強後的程式只需要比較少的次數就可以猜到對手的底牌。所以從精確度來說，利用進階理論所建構的猜數字程式的功力，還是會比先前的版本要好。

平均猜測次數的改進真的有好處嗎？答案是肯定的。舉例來說，如果猜數字雙方言明猜贏的一方得 10 分，輸的一方得 0 分；而且雙方的勝負以十回合的總分為依據。假設甲方平均花 6 次會猜中對手的底牌，而我們的程式從平均 5.2 次增強到平均 4.9 次。那麼我們的程式在個別回合中得勝的機會就會提高，得到較高總分的機會因此升高，取得最後勝利的機會也就跟著提高了。

## 結語

許多人常抱怨學校裡所學的數學理論都是象牙塔裏的知識，誤以為離開學校後，這些數學就真的變回理論而不能應用了。實際上，數學可以應用到許多地方，只是我們不自知罷了！例如對資訊科學的學習者和從業人員，良好的數學訓練其實是非常重要，而且有其實用性的。

寫程式誠然是一件不容易的事，要有寫好程式的功力，更需要經年累月的訓練和經驗的累積。因此資訊科系的同學必須花相當的時間，接受撰寫程式的訓練。因為這個緣故，非本科系的人常誤以為資訊科技的訓練就是寫程式的訓練。其實，程式設計師較像是資訊產業中的基層人員，他們最重要的任務是接受系統分析和設計師的指揮，撰寫合於系統規格的程式。而想要從事系統設計，甚至規劃的工作，除了資科專業的知識之外，良

好的數學基礎是一項不可或缺的本錢。

本文藉由建立兩個玩數字遊戲的程式，讓大家瞭解抽象的數學如何幫我們建立好玩的電腦程式，希望能藉此增強現在的和將來的資訊人學習數學的興趣。

劉昭麟任教於政治大學和海洋大學資訊科學系

感謝國立台灣海洋大學資訊科學系的許玉平教授和林英仁教授，對排列數字遊戲相關的數學理論的指導，及已經畢業的黃俊穎同學辛苦地撰寫程式，實際驗證起始狀態二無法轉變成圖一中的目標狀態。

## 參考資料

1. Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

2. Sherman K. Stein, *Mathematics: The Man-Made Universe; An Introduction to the Spirit of Mathematics*, W. H. Freeman, 1963.