第二章　倉庫番遊戲主要問題

第一節　人腦解題與電腦解題的不同

在我們開始討論如何以電腦來解決倉庫番問題之前，讓我們先來比較一下以人腦解題與電腦解題，分別有什麼樣的優缺點。以人腦解題最大的優勢就是人類所見到的遊戲盤面是二度空間的狀態，而電腦接受到的遊戲盤面是一度空間的。因此人腦與電腦在解題時，人腦不管是盤面搜尋(Pattern Search)或是死結狀況的偵測(Deadlock Detect)都比電腦來的容易，本章的第二節將介紹電腦偵測死結盤面的分析與策略。

以人腦解題，我們常以經驗法則來解題，例如我們尋找到一個箱子的搬移路徑，人腦會根據此一經驗去解決其他箱子的搬移。雖然電腦也可以做到經驗的累積，可是人腦對於經驗的利用具有歸納演繹的彈性，遇到不同的情況，人腦可以對於既有經驗稍做修正加以套用，而電腦卻沒有這樣的彈性。對於 Andreas 及 Jonathan【1】【2】先前利用人工智慧經驗法則的解題策略，未必能套用於所有倉庫番盤面的解題。並且，以經驗法則所得到的答案未必是最佳解。

因此，以電腦解題的優勢在於電腦可以記錄每一個移動的過程，並利用快速的計算來尋找最佳的搬移演算法，不過由於電腦的記憶空間是有限的，因此無限制地以嘗試錯誤的方式來解決倉庫番遊戲是愚蠢而不智的，因此在資料結構與演

算法策略方面(詳見本論文第三章、第四章)，必須加以設計，來達到節省記憶體空間與加快搜尋速度的目標。

第二節　死結狀況(Deadlock)的偵測

所謂的死結(Deadlock)就是在倉庫番遊戲過程中，搬運工將箱子推到某些位置，造成部份箱子不能再被移動或是移動範圍受限制的情況。在過去的研究中，認為死結狀況的偵測是以電腦解決此一問題時最大的挑戰【1】。歸納有下列五種基本死結狀況(如圖 2-1 所示)，及 1 種特殊死結情況(如圖 2-6 所示)，分別介紹說明如下：

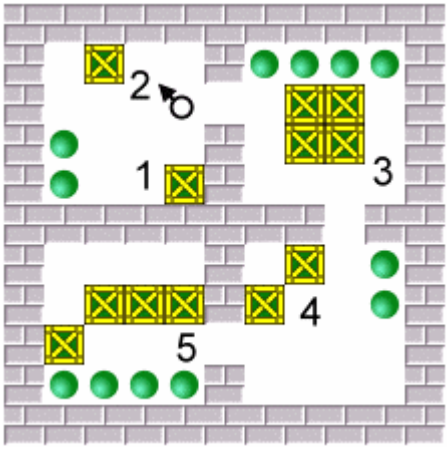


圖 2-1　五種基本的死結情況

第一種基本死結情況：「進入死角」

最基本的死結狀況，箱子被推到了牆角不能再移動。偵測此一死結狀況也是最容易的，只要在事先將盤面上所有不是目標位置的轉角標記起來(Marked)，在推箱子的過程中不要將箱子推入標記的位置即可。

第二種基本死結情況：「左右為難」

雖然箱子並未進入牆角，也沒有與其他箱子互相牽制，但是卻只能沿著牆邊左右移動(或上下移動)，永遠無法移動到目標區，此種死結情況可能推廣到另一種較不易偵測的情況，就是目標區剛好在牆邊，但是並不是所有的箱子都能靠牆邊運動，例如在某一面的牆上有兩個目標點，這一面牆就容許有兩個箱子可以在這面牆做平行的運動(如圖 2-2 所示)，但是如果有第三個箱子進入此面牆就形成死結(如圖 2-3 所示)。所以欲偵測此一死結狀況必須對盤面上每一面牆可以推入的箱子數目做紀錄及計算。

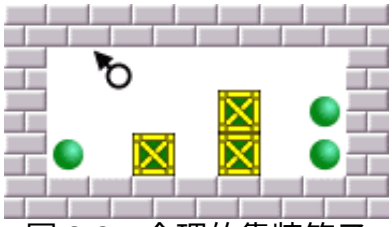


圖 2-2　合理的靠牆箱子

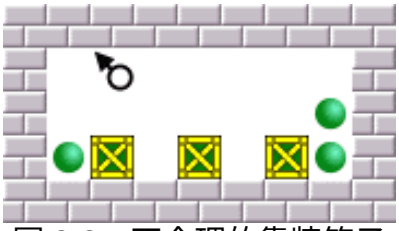


圖 2-3　不合理的靠牆箱子

第三種基本死結情況：「四角方塊」

四個箱子剛好被搬運工推成一個緊密相連的方塊，造成四個箱子都無法再移動。欲偵測此一死結狀況，在推動箱子之前，必須掃描盤面上是不是已經有三個箱子相連，並避開形成方塊的推動路徑。不過，並不一定要四個箱子才能形成四角方塊，如果其中幾個箱子以牆壁取代(如圖 2-4 所示)，同樣也會形成死結，而如果四角方塊中有三個位置以牆壁取代或是圖 2-4(c)的情況，就相當於一個牆角，所以第一種基本死結屬於四角方塊死結中的一種特例。

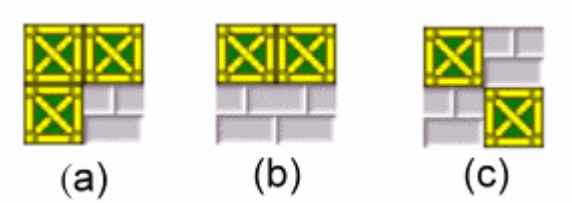


圖 2-4　四角方塊與牆壁形成的死結說明

第四種基本死結情況：「牆角牽制」

兩個箱子被搬運工推到牆角兩旁，雖然兩個箱子並未進入牆角(第一種基本死結情況)，如果該牆角不是一個指定的目標位置，這兩個箱子勢必要往遠離牆角的方向推動，因此搬運工必須移動到該牆角來推動箱子，但是搬運工卻受到兩個箱子的互相牽制而不能順利進入牆角使箱子往目標區移動。欲偵測此一死結狀況，除了要考慮目前所推動的這一個箱子所要推行的方向之外，還需要搜尋其他箱子的相對位置。

第五種基本死結情況：「區域牽制」

算是第四種基本死結情況的延伸，數個箱子被搬運工推到與牆壁形成矩形或沿著牆邊形成其他封閉形狀(如圖 2-5 所示)，在這個封閉形狀內，如果沒有足夠的目標位置讓箱子搬移進入，則會有部分或是所有的箱子，必須往封閉區以外的區域推動，因此搬運工必須藉由推動箱子來破壞封閉區域，進入封閉區域中把箱子從裡面往外面推，但是如果搬運工不論怎麼推動箱子都無法破壞封閉性，順利進入封閉區域中把箱子往外推，就形成了死結。欲偵測此一死結狀況，是所有情況中較困難的，因為形成死結的區域可能非常大，以人眼來辨識是較容易的，但

是以電腦要辨識此狀況就需要多次掃描盤面才能偵測，十分耗時。過去的研究者 Andreas 及 Jonathan【1】藉著在展開盤面中動態建立這樣的死結狀態知識庫(Pattern)來避免再次陷入死結情況，隨著箱子數目及活動範圍的擴大或變形，建立的資料會越來越多，越來越複雜。

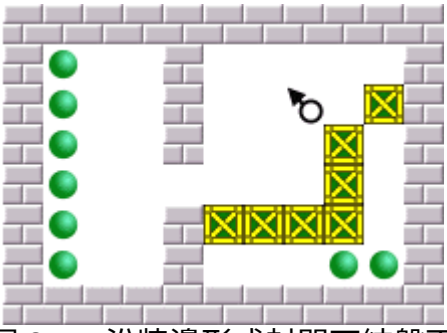


圖 2-5　沿牆邊形成封閉死結盤面

以上五種死結情況是屬於基本類型，對於玩倉庫番遊戲的玩家而言，都算是遊戲中的基本常識，而在以電腦解題方面，在過去 Andreas 及 Jonathan 的研究中【1】，他們在遊戲中動態建立死結盤面的組合(Pattern)，藉由比對的方式來避免死結的發生，不過在我們的研究中，我們發現有一種特別的死結情況對這樣的電腦解題策略而言不能夠判斷出來，如圖 2-6 所示，因為畫面右方的兩個箱子 E3 及 F3 把出入口堵死了，加上搬運工沒有第二條路可以到畫面右邊，所以形成死結，如果在 E3 移動之前，先把 F3 的箱子移動到 G3 或 H3，則死結情況就可以避免。這樣的死結情況在偵測上的困難點在於出現死結的箱子，例如圖 2-6 中的 F3 箱子，不在搬運工的活動範圍內，而發生死結的原因並不是加上了目前這一個箱子才形成死結，而是好幾個步驟之前其他箱子的擺放位置不好，到了這一個盤面才形成死結，因此有可能中間許多搬移動作都是浪費的。對於電腦解題而言，

圖 2-6 中的箱子 F3 單獨存在時並不是死結，但是當 E3 位置有箱子時就會形成死結，依照 Andreas 及 Jonathan 的研究，他們的方法是將 E3、F3 的位置建立成為一個判斷死結的規則，當有 F3 在時，E3 的位置不能把箱子推入，避免形成死結，但是問題並不是出在 E3 可不可以把箱子推入，而是要先把 F3 的箱子移動到其他位置，因此對於這樣的死結不能夠加以判斷。

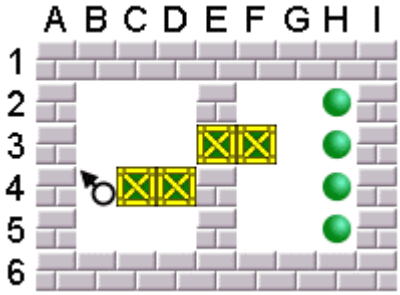


圖 2-6　特別的死結情況

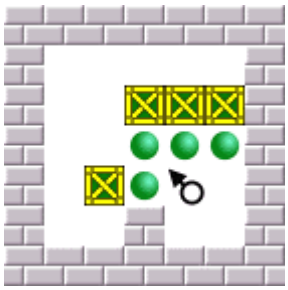


圖 2-7　拉目標所形成的死結

我們發現，由於死結的情況主要來自於搬運工將箱子「推」到與其他箱子或牆壁形成限制箱子移動的情況，而這樣的牽制範圍可能遍及整個盤面，為了偵測這些情況，我們必須做許多的計算，每次展開一個新盤面就必須計算是否形成死結，大量的盤面計算累積下來就成為解題過程上很大的時間負擔。但是我們發現如果我們在電腦解題的過程中以反向思考，採取讓搬運工去把目標位置「拉」到箱子位置的方式，則死結盤面將只有一種情況：搬運工將目標拉成一個封閉區域，將自己的活動範圍壓縮(如圖 2-7 所示)，因為搬運工只能「拉目標」，如果要把圖 2-7 的目標都拉到箱子的位置，搬運工必須破壞封閉性，到封閉區域外去拉目標接近箱子，但是因為越拉目標，空間就越壓縮，反而無法走到封閉區域外，便形成死結。這種情況與先前我們以「推箱子」方式所形成的第五種基本死結十

分相似，只是以「推箱子」的方式，搬運工相對於形成的死結的是開放性的，但是以「拉目標」的方式，搬運工處於有限的封閉空間內，在偵測計算上會容易許多。對於這一推一拉間的奧妙，我們將在本論文下一節做進一步討論。

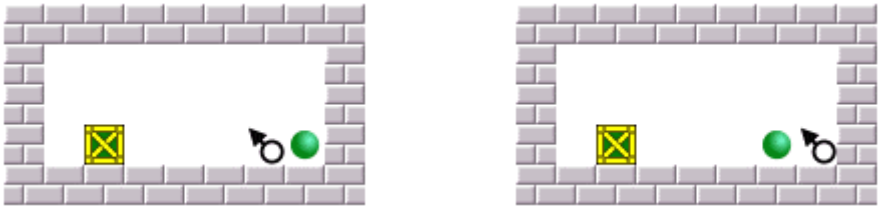
第三節　反向搜尋遊戲盤面

在上一節的內容中，我們提到死結情況的偵測是倉庫番遊戲解題上一個挑戰所在，以正向的方式，讓搬運工去「推」箱子，所造成的五種基本死結情況及一種特別死結情況，以電腦一維的搜尋方式，需要做許多次的盤面掃描或是必須建立知識庫，以盤面比對的方式才能達成。因此我們便有一個構想：如果我們改採反向搜尋的方式，從終止盤面上已經在目標位置上的箱子，反向「拉」回到起始盤面的位置，是不是能減少死結情況的發生呢？答案是肯定的，我們將詳細介紹如下：

為了避免讀者的混亂，在本論文往後的討論中，在圖例中如果看到  的圖樣，表示是一個被拉動的目標位置，我們的目的就是將所有的  圖樣，以「拉」的方式，拉到起始盤面上所有箱子  的位置，因此當我們說明反向拉目標的做法時，  的位置上並不是真的有一個箱子，而是表示拉目標時的目的地。

讓我們先來討論前一章所提到的五種基本死結情況，第一種情況「進入死角」

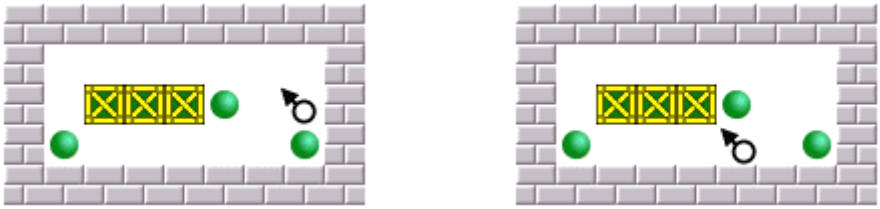
，當我們反向以拉目標的方式來尋找答案時，由於將目標拉完之後，目標會在原來搬運工站的位置，而搬運工必須往後退一步，因此在搬運過程中，我們只會把目標位置從死角拉出來，如圖 2-8(a)所示，不會把目標位置拉進死角，因為如果目標要拉進死角，搬運工將沒有位置可以後退，如圖 2-8(b)所示，因此不會造成第一種「進入死角」的死結情況。



(a) (b)

圖 2-8　以拉目標避免進入死角的範例

第二種死結情況「左右為難」，同樣是因為搬運工在拉目標的時候，搬運工必須要有位置可以後退，因此只有從牆邊把目標拉離開牆邊，如圖 2-9(a)所示，不會有需要把目標拉進牆邊的情況，如圖 2-9(b)中，正中間目標不可能往下拉到牆邊，因此當我們順向將箱子依照「拉目標」的路徑去搬移時，也不會發生第二種「左右為難」的死結情況。



(a) (b)

圖 2-9　以拉目標避免左右為難的範例

第三種死結情況「四角方塊」，由於搬運工在拉目標後退時需要空間，因此如同圖 2-10 所示，四個目標位置不可能拉成一個四角方塊，也就是當我們順向

將箱子依照拉目標的路徑移動時，也不會造成四角方塊的死結情況。

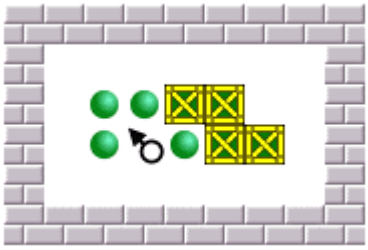
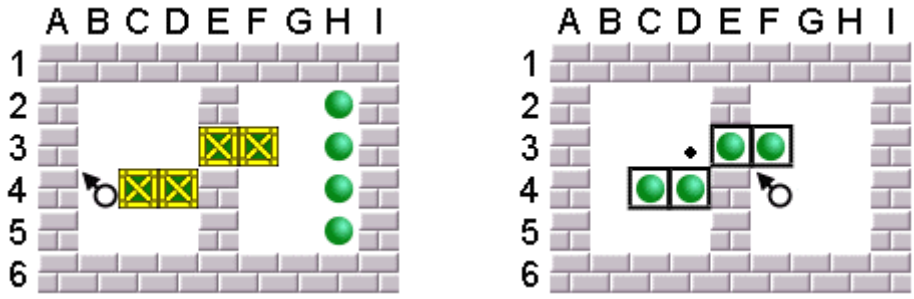


圖 2-10　以拉目標避免四角方塊的範例

至於第四種、第五種及特殊的死結情況，我們可以把他歸類成因為數個箱子將盤面分成封閉區域，造成搬運工沒辦法破壞其封閉性而產生的死結，如同我們在上一節最後所介紹的，當我們以「拉目標」的方式去尋找將目標拉回箱子的路徑時，所造成的死結封閉區域，搬運工是處在封閉區域內企圖將箱子往封閉區域外的方向拉，搬運工可以活動的範圍內所接觸到的目標，一定是造成封閉的目標之一，因此在搜尋時我們會先針對造成封閉的目標群進行計算，只要發現沒辦法藉由拉任一目標的方式使搬運工脫離封閉區域，就屬於反向搜尋的死結。而如果我們採用順向「推箱子」的方式，造成的死結封閉區域，搬運工是處於封閉區域外企圖將箱子從封閉區域內向外推，在搬運工可以活動的範圍內所接觸到的箱子，不見得是造成死結的禍首，因此在搜尋時可能會展開到沒有造成死結的箱子，但事實上如果沒有將造成死結的箱子做處理，怎麼移動其他箱子都是沒有用的，而所展開的盤面也都是浪費的。我們以圖 2-11 加以說明，當我們以「推箱子」的方式造成圖 2-11(a)的死結情況，就是因為搬運工沒辦法破壞封閉性進入出問題的右半部的區域，但如果我們採「拉目標」的方式，在圖上 E3、F3 的位置造成

死結時，搬運工一定是位於出問題的右半部，只要搬運工把 F3 拉開就解除死結。同樣的例子圖 2-11(b)，採「拉目標」時有沒有可能搬運工會在左半部呢？答案是否定的，因為如果搬運工在左半部，E3 的位置一定不可能有目標，因為如果 E3 位置有目標，必是從 F3 的位置拉到 E3 的位置，然後搬運工停在我們打上♦符號的 D3 位置，如果 F3 位置上已經有目標，E3 位置就一定不可能有目標。



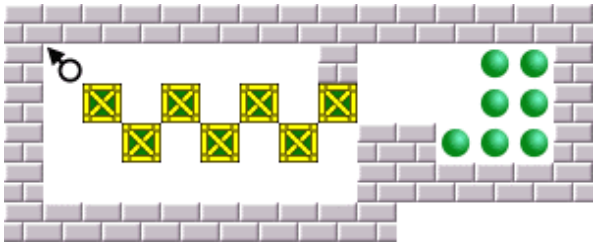
(a)　　(b)

圖 2-11　以拉目標避免封閉死結的範例

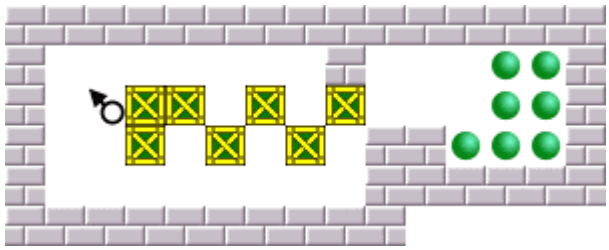
由此可知，採取反向拉目標接近箱子的方式，會比順向推箱子去接近目標的方式，在死結情況的偵測上容易許多，甚至部分的死結偵測根本不必做，例如「進入死角」、「左右為難」、「四角方塊」三種死結，因為根本不會發生。而封閉型的死結類型，以拉目標的方式所造成的封閉也因為搬運工一定處於造成死結的封閉區域內而容易處理。

由於我們採用反向的搜尋方式，避開了大量的死結盤面，因此真正需要展開的盤面相對較少，以圖 2-12(a)為例，以正向方式來搜尋，需要展開 99,829 個盤面(使用 Andreas 及 Jonathan 的單一代理人搜尋法【1】【2】)，但是以我們的方法反向搜尋就只需要展開 10,244 個盤面即可，差別就在於當我們順向去推箱子

的時候，很容易一開始就展開圖 2-12(b)的情況，雖然搬運工只把三個箱子推在一起，不屬於我們先前所提到的死結之一，但事實上。為了解除這三個箱子相鄰的情況，必須多展開許多盤面來找到路徑。



(a)



(b)

圖 2-12　正向、反向搜尋方式比較之範例盤面

雖然以反向搜尋的方式來解題，可以減少死結偵測的時間也可以減少展開的盤面，不過，以反向搜尋倉庫番遊戲與正向搜尋時有一些不同，在解題過程中我們有兩個需要注意的地方：(一)多個起始盤面，由於反向搜尋就是將所有的目標以「拉」的方式往起始盤面上箱子的位置搬移，如同從圖 2-13(b)，將所有的目標拉到如圖 2-13(c)的位置，因此展開時可能的第一個盤面並不唯一，以圖 2-13(b)為例，可能的起始位置包括座標 B4、C4、E4、F4 四個位置，這四個可以開始「拉」目標的位置，我們都必須考慮進去。(二)終點位置判斷。當我們把所有的目標都拉到指定的地點後，還需要判斷搬運工是否能回到最原始給定的位置，以下頁圖

2-13(a)為例，由於搬運工的起始位置在盤面 F7 的位置，所以當我們從圖 2-13(b)的情況把所有目標拉到圖 2-13(c)的情況，還需要考慮能否把搬運工移動到 F7 的位置，以此盤面來說，如果最後搬運工的位置在畫面的左半部 B2、B3、B4、B5、B6、B7、C2、C3、C5、C7 這些位置，以致於搬運工不能回到 F7 的位置，都算是失敗的，也就是不能把這樣的結果當做可行的答案，必須繼續搜尋。

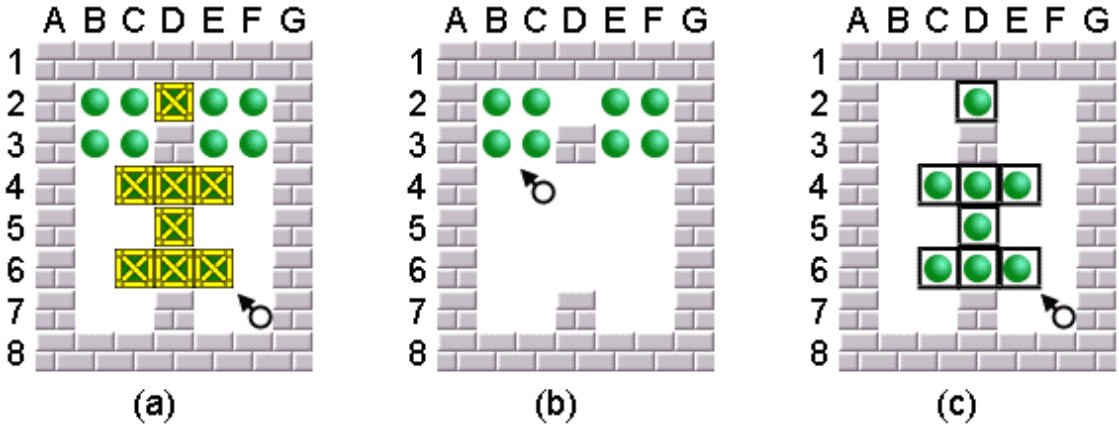


圖 2-13　反向搜尋時多起始盤面及終點位置說明圖例

第四節　龐大的搜尋空間

雖然倉庫番遊戲已經被證明為 PSPACE-complete 的問題，不過倉庫番遊戲與其他益智遊戲不同的地方在於它的搜尋空間與真實世界問題接近(Real world like problem)，隨著盤面可移動位置與箱子個數的增加，搜尋空間便會快速成長。例如一個內部有 42 個移動空間、6 個箱子、1 個搬運工的盤面，光是 6 個箱子可能的位置組合就高達 C_6^{42}，而搬運工的位置一定在其中一個箱子旁邊的位置，因此真正的可能的盤面組合數介於 C_6^{42} (5,245,786)到 C_7^{42} (26,978,328)之間，如果我們一個盤面以 10 個位元組(10bytes)來記錄，如果我們沒有做盤面展開的判斷，

也沒有對死結盤面加以偵測，以暴力窮舉所有情況就需要約 50MBytes 到 250MBytes 之間的記憶體空間，如果我們再多一個箱子，可能產生的盤面所需要的記憶體空間將增加到 1GBytes 以上。因此，所有的研究者都往兩個方向來減少搜尋的空間，一個就是對於死結的偵測，一個就是設計展開盤面時的判斷策略，對於死結的偵測方面，沒有將死結盤面完全偵測，只是增加搜尋的時間，但是展開盤面的判斷策略如果不適用於所有情況，則將可能會錯失最佳解的情況。對於死結盤面的預防或是偵測，我們利用反向搜尋的技巧得到不錯的效果，而在展開盤面時的判斷策略，由於現在我們將研究的重點放在倉庫番遊戲的最佳搬移上，因此我們也將對於所提出來的解題策略，分析是不是會造成錯失最佳解的情況，在本論文第四章第二節及第三節將會對此加以討論。