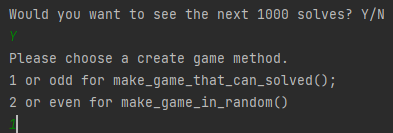
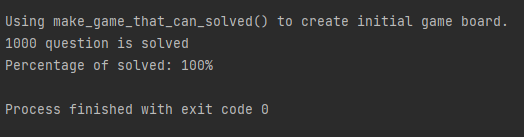
**電腦對局理論-8 Puzzle作業**

資訊工程碩一 711083104 李旻軒

1. 編譯方法  
   　　我使用一個二維陣列來實施BFS，因為我認為direct address可以更靈活的運用在BFS中，該二維陣列會存上兩種資料，移動方向及當前node的parent node，這樣的模式可以讓我不使用linked list，卻又能夠獲得其特色。  
   　　從root開始，首先會創造出4種方向的node，先判斷這樣的走法是否合乎規則，若該方向無法移動，其移動方向會被定義為stop，且未來再也不會生成新node；若該方向可以移動，會開始將該方向的值放入一個stack中，並利用二維陣列所提供的parent node，繼續將之前走過的方向存入stack中，直到遇到root，此時的stack獲得了「可以到達當前node」的所有移動方向，並且可以利用stack去計算出當前node是否已經達成目標。  
   　　雖然結果顯示我可以解出不少game board，但是仍然有個大問題，就是當樹越來越大時，程式很容易崩潰（可能是超出陣列的最大值），因此只要最短路徑大於某個數字時，就會無法完整地給予解答，我認為有可能是因為我設置的條件太少，我只有消除掉上一步往上、下一步往下的那種移動方式，卻沒有移除繞大圈圈的情形，或許將他移除後會有更好的解答比例。
2. 操作說明
   1. 按下執行或開啟.exe，首先會先跑出一個初始game board以及逐步列出該版面的解答，包括初始畫面、是否有解出、及列印所有移動過程。
   2. 接下來會問你需不需要看接下來1000個遊戲的解答，如果需要請打Y or y，若是打其他的字元，將會直接進入程式終點。  
      
   3. 請選擇製作初始game board的方法，1或奇數是必有解的情況，2或偶數則是不一定有解。
   4. 程式便會開始執行，其輸出值的順序為：
      1. 每個game board的初始畫面。
      2. 是否有解？  
         若有解，則印出The game is solved. Now start print the steps to solved the problem. 。  
         若無法解出則跳出The tree is too big.的提示。
      3. 若（2）顯示有解，則會開始印出每個步驟，值到印出所有的步驟，才會開始製作下一個board game。
      4. 最後將會輸出已解決的次數，程式結束。  
         
3. 解決問題的比例:  
   本次我有使用兩種生成game board的function：  
   (1) 是利用隨機方式生成出game board；  
   (2) 是利用goal往回推，讓game board呈現必有解的情況。

|  |  |
| --- | --- |
| make\_game\_in\_random (1) | make\_game\_that\_can\_solved (2) |
| 119/1000 | 1000/1000 |

解決問題的比例如上表所示，可以看到利用(1)生成初始版面，可能會導致無解的狀況發生，或是雖然有解但需要的記憶體太多而讓程式崩潰，因此我有設定一些停損點，使程式在一定的範圍下停止計算。而利用(2)把最終board回朔到初始board的解題率則是100 %，這代表當必有解的時候我的程式的邏輯是對的，未來如果還要更精進自己的話，可能要在剃除一些條件（如繞圈圈），使分配給樹的記憶體不會因為可以更妥善的運用。