人工智慧 作業二

資工112 40847015S 紀軒宇

1. 請說明你所使用之機器軟硬體規格、所用的作業系統、開發之程式語言版本

等相關資訊以及你為何選擇這樣的規格。另外請提供你的連絡電話，以便不

時之需。

CPU: Intel® Core™ i7-8750H

GPU:Nvidia GeForce GTX 1060

ram:24g ddr4 2666

OS: Windows 10

開發軟體：Python 3.9.6

會選擇這樣的規格是因為當時考量未來可能會學習機器學習相關知識，選擇了有獨立顯示卡的電腦，方便日後學習。

開發軟體：Python 3.9.6

連絡電話：0988-056-412

2. 詳細說明你所使用之機器軟硬體規格及作業系統、開發軟體版本相關資訊。

另外請提供你的連絡電話，以便不時之需。

CPU: Intel® Core™ i7-8750H

GPU:Nvidia GeForce GTX 1060

ram:24g ddr4 2666

OS: Windows 10

開發軟體：Python 3.9.6

連絡電話：0988-056-412

3. 首先你先設法製作輸入檔若干個。其難易度由你決定。請說明你如何製作這

些測試用的輸入檔。

以下測試檔製作皆透過<https://www.mad-weasel.de/sliding.html>中的 sliding game，以final state開始手動做方塊的移動，計算最佳步數

(i) 皆由1x1的方塊所組成3x3的棋盤，最佳解共20步

3 3

5 7 8

4 1 2

6 3 0

(ii) 皆由1x1的方塊所組成3x4的棋盤，最佳解共36步

3 4

7 6 5 3

1 4 10 11

2 8 9 0

(iii) 題目的盤面5x4，最佳解共4步

5 4

1 2 2 3

1 2 2 3

4 0 5 5

4 0 7 6

9 10 8 6

(iv)盤面5x4，最佳解共15步

5 4

1 0 2 2

1 0 2 2

5 5 10 3

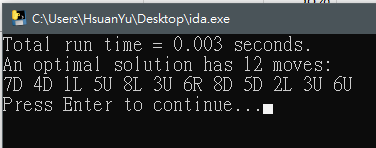
4 7 6 3

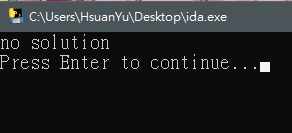
4 9 6 8

4. 兩支程式之原始碼中應加註解，請說明如何執行這兩支程式。

為執行效率，兩支程式原始碼中監控時間及計算節點相關程式碼皆已註解或刪除，兩支程式 ids.exe和ida.exe執行方法皆相同，將input.txt和output.txt放在C:\下，執行檔圖例如下，執行的輸入input.txt和output.txt皆儲存在，因為 input.txt 和 output.txt 皆儲存在 C槽，需要存取時需要一定的系統權限，所以要使用 exe檔時要對檔案點右鍵->以系統管理員身分執行，執行部分，每次迭代前會檢查程式執行時間，如果執行時間超過一小時，或是IDS深度超過100步、IDA\*的f-limit超過1000，即會回傳 no solution，回傳輸出會出現在執行視窗與C:\output.txt，執行結束後會有 Press Enter to continue…提示程序已結束。







執行完後按下會顯示輸出及輸出結果到C:\output.txt中，執行結束按下Enter即可結束執行

5. 說明第一支程式IDS你使用甚麼方法、甚麼資料結構、甚麼技術(操練要項)來解決這個問題，並請你說明你測試一些盤面時的表現如何、耗用的時間及空

間為何、是否一定會找到最佳解。請你用一些例子輔助說明。(解題愈快、說

明愈清楚的程式成績愈高。)

- 方法

使用深度限制的DFS，深度超過就traceback，並逐步增加深度限制直到找到解

- 資料結構

在這個部分，我對每個節點（node）作出了一個 class，class中有幾個部分

- table：list

用來記錄整個盤面。

- last\_inverse\_op：str

用來記錄上一步展開的動作的反動作，避免再search時再走回去一樣node

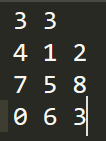
- block\_data：dict

用來記錄所有方塊（block）的資料

- block\_data的資料結構如下

{ 'block\_number' : { 'location' : [0, 2], 'shape' : [1, 1] }, …}

block\_number用來標記方塊編號  
location記錄該方塊最左上方塊的位置

shape 用來記錄方塊形狀

- 技術（操練要項）

- 盤面表示部分，以list來表示，右圖的輸入在程式中如下所示

[[4, 1, 2], [7, 5, 8], [0, 6, 3]]

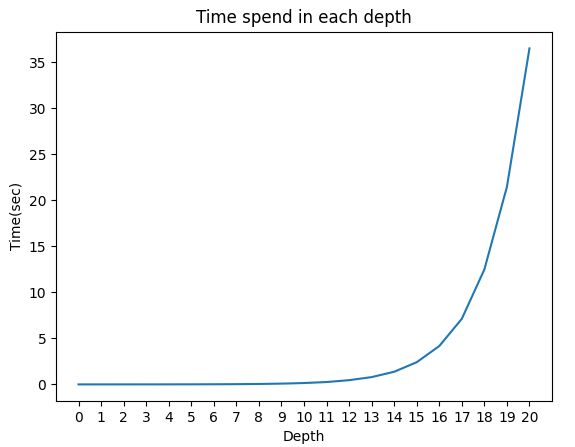
- 走步先以successor() 計算該點有哪些動作可以走，在對每個點做dfs搜尋

- 節點所存的資訊都在上面資料結構的class node中所示

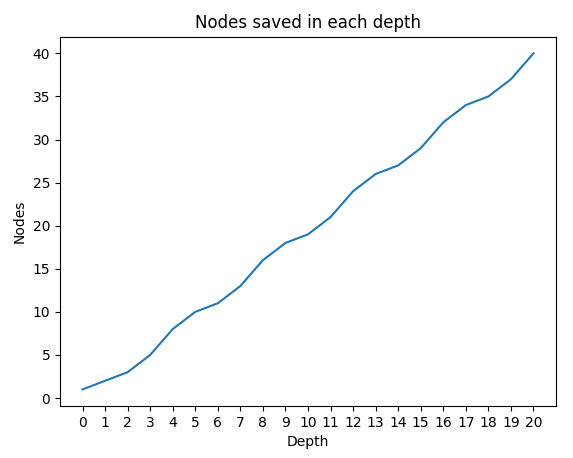
- 判斷重複部分，有在過程中加入visited的set判斷這個節點的盤面是否在之前出現過，但是在實驗過後，對於執行時間沒有太多的進步，反而因為要加入set的動作增加了許多運作時間

- 時間與空間相關分析在以下會說明

- 所得結果與手動生出的盤面倒著回去做的步數相較，比對多個盤面測試資料後皆為最佳解。

 - 耗用的時間

耗用時間部分，這邊以第3題中的 (i) 盤面做測試，理論步數為20步，在每個深度搜尋後測試其計算時間，將其製成圖表後如下，可以看出其計算時間隨深度呈現指數成長，只要盤面複雜一點，可能就會需要上百分鐘甚至一天才能搜尋完。

- 耗用的空間  
 耗用空間部分，也是以第3題中的 (i) 盤面做測試，理論步數為20步，在每個深度搜尋後測試儲存的節點量，將其製成圖表後如下，可以明顯觀察到節點儲存量是呈現倍數成長每次成長皆在2到3之間。

- 其餘盤面的時間使用

- (ii) No answer  
 （無法在時間內跑出來）

- (iii) Total run time = 0.011 seconds.

An optimal solution has 4 moves:

5L 6U 7L 8U

- (iv) Total run time = 26.674 seconds.

An optimal solution has 15 moves:

2L 3U 3U 8U 8U 6R 10D 8L 6U 10D 8D 5R 4U 9L 10L

- 是否一定會找到最佳解

理論上是一定會找到最佳解的，但是以上題的圖表來看，唯一的問題可能會發生在計算時間遠超使用者所預估的時間。

6. 說明第二支程式IDA\*你使用甚麼方法、甚麼資料結構、甚麼技術(操練要項)

來解決這個問題，並請你說明你測試一些盤面時的表現如何、耗用的時間及

空間為何、是否一定會找到最佳解。請你用一些例子輔助說明。(解題愈快、

說明愈清楚的程式成績愈高。)

- IDA\*

A\* search中加入搜索f(n) 限制 f-limit，若超過就traceback，並在每次迭代中逐漸放寬搜索限制 f-limit直到找到最終解

- 資料結構

在這個部分，我對每個節點（node）作出了一個 class，class內部大致與IDS中的相同，但是有多一樣屬性，其餘皆相同

- gn：int

用來記錄該節點之深度。

- 技術（操練要項）

- 盤面表示部分，也與IDS相同

- 走步先以successor() 計算該點有哪些動作可以走，將每個點做move加入succ的list中對每個點做深度優先搜尋，不在f(n)超過bound就traceback，持續迭代，直到找到答案。

- 節點所存的資訊同IDS

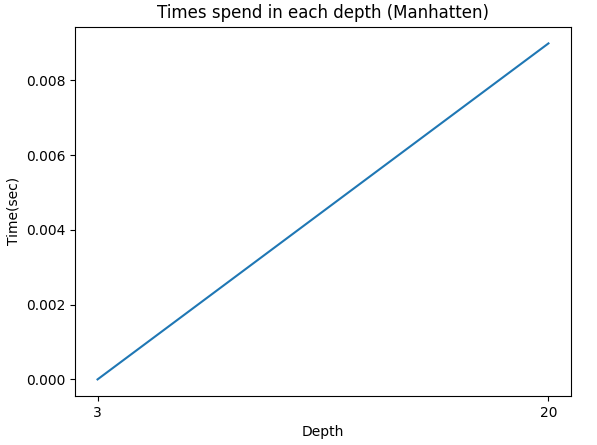
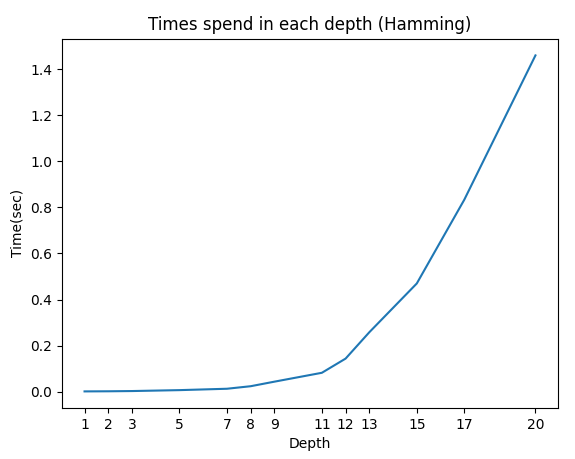
- 時間與空間相關分析在以下會說明

- 所得結果與手動生出的盤面倒著回去做的步數相較，比對多個盤面測試資料後皆為最佳解

- 不同heuristic的分析會在以下兩點作分析

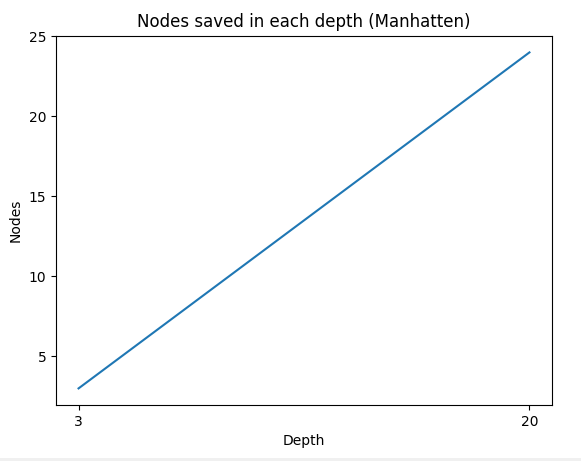
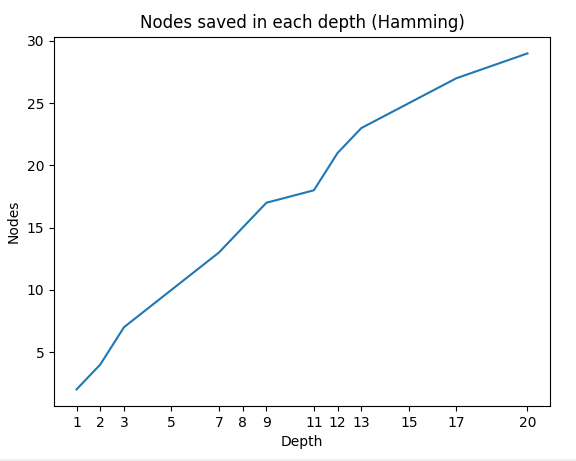
- 耗用的時間

耗用時間部分，這邊使用Hamming Distance和Manhatten Distance兩種heuristic，同樣使用第3題中的 (i) 盤面做測試，理論步數為20步，在每個round搜尋後看其搜索的最大深度及其計算時間，將其製成圖表後分別如下，可以看出兩種計算時間在同樣深度讀的情況下，皆比IDS快，再來比較兩種heuristic，將 Hamming Distance和Mahatten Distance做比較，使用Mahatten Distance的執行速度更快。



- 耗用的空間

耗用時間部分，這邊使用Hamming Distance和Manhatten Distance兩種heuristic，同樣使用第3題中的 (i) 盤面做測試，理論步數為20步，在每個round搜尋後看其搜索的最大深度及其儲存節點數量，將其製成圖表後分別如下，可以看到隨回合數增加，儲存的節點數也會上升，兩個heuristic在每個深度所需儲存的節點數量幾乎都不超過IDS儲存的節點數量，再來相較Hamming Distance和Manhatten Distance，使用Manhatten Distance所需儲存的節點數更少。



- 其餘盤面所花時間

- (ii) Total run time = 19.43 seconds.

An optimal solution has 36 moves:

9R 8R 4D 10L 8U 9L 11D 8R 9U 4R 10D 9L 5D 6R 7R 1U 9L 5L 4U 10R 2R 9D 5L 2U 10L 11L 8D 4R 6D 7R 2U 6L 7D 3L 4U 8U

- (iii) Total run time = 0.002 seconds.

An optimal solution has 4 moves:

5L 6U 7L 8U

- (iv) Total run time = 0.01 seconds.

An optimal solution has 15 moves:

2L 3U 3U 8U 8U 6R 10D 8L 6U 10D 8D 5R 4U 9L 10L

- 是否一定會找到最佳解

理論上是一定會找到最佳解的，相較IDS，同樣的問題在都可以找到最佳解的情況下都可以有更快的結果。

7. 請說明你做此作業所踫到的一些狀況及困難。

這次作業是第一次使用 python 的 class，所以在使用上還有許多不熟悉的地方，要常常去翻python相關的document，而且這次也是第一次對於執行速度上有越快越好的需求，所以為了達到更快的執行效率，在寫完之後還會去對了許多資料的資料結構或參數的使用方法去進行調整，在最後調整完後有較讓自己滿意的執行時間。

8. 請列出你的參考文獻(含網站)來源，並請說明參考了那些部份用於作業中。

- IDS實作部分參考：<https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/> 中的 Algorithm

- IDA\*實作部分參考： [https://en.wikipedia.org/wiki/Iterative\_deepening\_A\*](https://en.wikipedia.org/wiki/Iterative_deepening_A*) 上的Pseudocode