AI Assignment 2

- 學生: 蘇子權
- 學號: 40947023S
- 系級: 資工 113

Directory Tree

```
// Assignment 2 directory tree

├── gameLib
|   ├── basic.cpp
|   ├── basic.h
|   ├── board.cpp
|   ├── board.h
|   ├── game.cpp
|   └── game.h
├── IDAStar
|   ├── IDASTAR
|   ├── IDAStar_count_cluster_heuristic.cpp
|   ├── IDASTAR.cpp
|   ├── input.txt
|   └── makefile
├── IDS
|   ├── IDS
|   ├── IDS.cpp
|   ├── input.txt
|   └── makefile
├── testcase_IDAStar
|   ├── input1.txt
|   ├── input2.txt
|   ├── input3.txt
|   ├── input4.txt
|   └── input5.txt
└── testcase_IDS
    ├── input1.txt
    ├── input2.txt
    ├── input3.txt
    ├── input4.txt
    └── input5.txt
```

1

- 硬體規格
    - CPU：AMD Ryzen 5 5600X 6 Core
    - 顯卡：NVIDIA GeForce RTX 3600
    - RAM：16G
    - OS: Windows 家用版
    - 選用該硬體規格和此電腦，而非我的筆電主要原因是 GPU 加速運算。

    - 選用該作業系統原因是本身對於使用 Windows 較為熟悉。
    - 寫程式的環境是 WSL2 的 Arch Linux
    - 

- 連絡電話
    - 0976960352
- 如何執行程式
    - 本次作業是使用 C++ 撰寫，也有寫 makefile 方便助教批改，但如果沒辦法 make，folder 中我也有附上執行檔以便不時之需。
    - IDS 和 IDA* 兩支程式分別是 IDS folder 下的 `IDS.cpp` 和 IDAStar folder 下的 `IDASTAR.cpp` 。
        - 若要批改 IDS，如果是使用 make 的情況...
            1. 進入 IDS folder 中。
            2. 將指定要測試的 testcase (`../IDS_testcase` folder 中 `input<1~5>.txt` 指定的一筆內容全部複製並全部覆寫到現在 IDS folder 目錄下的 `input.txt`)。
            3. `make clean`
            4. `make`
            5. `./IDS`
        - 若要批改 IDS，如果是使用執行檔的情況...
            1. 進入 IDS folder 中。
            2. 將指定要測試的 testcase (`../IDS_testcase` folder 中 `input<1~5>.txt` 指定的一筆內容全部複製並全部覆寫到現在 IDS folder 目錄下的 `input.txt`)。
            3. `./IDS`
        - 若要批改 IDA*，如果是使用 make 的情況...
            1. 進入 IDAStar folder 中。
            2. 將指定要測試的 testcase (`../IDAStar_testcase` folder 中 `input<1~5>.txt` 指定的一筆內容全部複製並全部覆寫到現在 IDAStar folder 目錄下的 `input.txt`)。
            3. `make clean`
            4. `make`
            5. `./IDASTAR`
        - 若要批改 IDA*，如果是使用執行檔的情況...
            1. 進入 IDAStar folder 中。
            2. 將指定要測試的 testcase (`../IDAStar_testcase` folder 中 `input<1~5>.txt` 指定的一筆內容全部複製並全部覆寫到現在 IDAStar folder 目錄下的 `input.txt`)。
            3. `./IDAStar`

2

- 前情提要：由於 IDS 和 IDA* 的執行 efficiency 不同，因此測資是分開來的(見置頂 Directory Tree)。
- IDS
  - 小的測資大約長度 < 5
  - 大的到長度 14
  - 其餘的皆在兩者中間，並且有全部 1 或是 0、1 交錯的測資
- IDA
  - 小的測資大約長度 < 5
  - 大的到長度 20
  - 其餘的皆在兩者中間，並且有全部 1 或是 0、1 交錯的測資

3

- 根據 gameLib 的 folder，我先將遊戲本題的 core 和 API 建立好
  - `board.h`、`board.cpp`
    - 定義 game board 上的每一格的狀態以及修改該值的 api
      - class name -> boardUnit
    - 將整個 game board 以 `vector<boardUnit>` 的形式儲存
      - class name -> gameBoardUnit
      - 可以方便往下 board unit init board 的值
      - 與 game class friend
  - `game.h`、`game.cpp`
    - 將整個 game board 以 `vector<boardUnit>` 的形式儲存
    - 設計「 constructor、init game board、得到 game board 的大小 api、得到該狀態下 board 上 1 的數量的 api、game board 是否全數清成 0 的 api、條件是否達成的 api、下一回合的所有操作(含清掉指定位置的細菌、細菌左右分裂)的 api。
  - `basic.h`、`basic.cpp`
    - 主要是將測資 `.txt` file 轉成 string 再將1、0 parse 出來用 `vector<int>` 儲存。
- IDAStar
  - 讀好資料、construct and init game board 後 call IDAStar function。
  - heuristic function 的設計是 base on 還有多少 1 在 game board 裡，而且顯然保證不會 overestimated。
  - 演算法流程中詳細的 annotation 在 codebase 中。
- IDS
  - 讀好資料、construct and init game board 後 call IDDFS function。
  - 演算法流程中詳細的 annotation 在 codebase 中。

4

- 什麼方法
  - IDDFS 演算法遞迴求解
- 什麼資料結構
  - 第 3 題有完整說明到
- 什麼技術

    - 盤面怎麼表示
        - 0's, 1's value in 1*n `vector<int>`
        - 第 3 題有完整說明到
    - 棋盤全部資訊要存在每個節點嗎
        - 是的，基本上就我設計的 gameLib 是會在每個節點上存取 game board
    - 節點要存哪些局部資訊
        - 呈上，我有設計 api 去 get 到 board size, 1's total number(for heuristic function), change element state 等資訊(詳細可見第 3 題的 `game.h`, `game.cpp`)
    - 走步要如何產生
        - 呈上，有設計 api 去生成指定位置下一步的 game board 狀態，並存在 vector 中
    - 需要判別重複?
        - 不需用
    - 最後答案如何取出來
        - 過程中有一個 solution 的陣列在存遞迴當下每一步操作，也有一個參數(gx -> round) 在記錄第幾回合(層數)。
    - 會不會跑不停
        - 如果 game board length 超過 20，會超過 100 秒，我將其視為跑不停。
    - 記憶體會不會爆掉
        - 就 gameLib 設計上不會
    - 所得結果會是最佳解嗎
        - 是
    - 使用哪一種 heuristic 結果比較好
        - 我目前採用兩種方法，然而第 2 點在長度為 20 的 game board 下比第 1 點慢將近 16 秒。因此相較起來第 1 點較好。
            1. 計算當 game board 上 1 的總數
            2. game board 上 disjoint 1's cluster 的數量
    - 碰到 dead end 可提早 backtracking?
        - 在有 threshold 的情況下會先被 prune 掉。

5

- 什麼方法
    - IDA* 演算法遞迴求解
- 什麼資料結構
    - 第 3 題有完整說明到
- 什麼技術
    - 盤面怎麼表示
        - 0's, 1's value in 1*n `vector<int>`
        - 第 3 題有完整說明到
    - 棋盤全部資訊要存在每個節點嗎
        - 是的，基本上就我設計的 gameLib 是會在每個節點上存取 game board
    - 節點要存哪些局部資訊
        - 呈上，我有設計 api 去 get 到 board size, change element state 等資訊(詳細可見第 3 題的 `game.h`, `game.cpp`)

    - 走步要如何產生
        - 呈上，有設計 api 去生成指定位置下一步的 game board 狀態，並存在 vector 中
    - 需要判別重複?
        - 我沒有特別做判別重複
    - 最後答案如何取出來
        - 過程中有一個 solution 的陣列在存遞迴當下每一步操作，也有一個參數(gx -> round) 在記錄第幾回合(層數)。
    - 會不會跑不停
        - 如果 game board length 超過 20，會超過 100 秒，我將其視為跑不停。
    - 記憶體會不會爆掉
        - 就 gameLib 設計上不會
    - 所得結果會是最佳解嗎
        - 是
    - 碰到 dead end 可提早 backtracking?
        - 我沒有特別做判別重複，所以無法提早 backtracking
- IDS IDA* 實作比較圖

編號	測資	長度	花費時間	編號	測資	長度	花費時間
IDA*-1	1 1 1 0	4	0.000000 sec	IDS-1	1 1 1 0	4	0.000000 sec
IDA*-2	1 1 1 1 1 1 1 1 1 1	10	2.120000 sec	IDS-2	1 1 1 1 1 1 1	7	16.151000 sec
IDA*-3	1 0 1 0 1 0 1 0 1 0	10	0.000000 sec	IDS-3	1 0 1 0 1 0 1 0 1 0	10	0.108000 sec
IDA*-4	1 1 1 1 1 1 1 1 1 1 1 1 1	12	930.806000 sec	IDS-4	0 1 0 1 0 1 0 1 0 1 0 1 0	13	40.513000 sec

編號	測資	長度	花費時間	編號	測資	長度	花費時間
IDA*-5	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	20	87.696000 sec	IDS-5	0 1 0 1 0 1 0 1 0 1 0 1 0 1	14	641.009000 sec

6

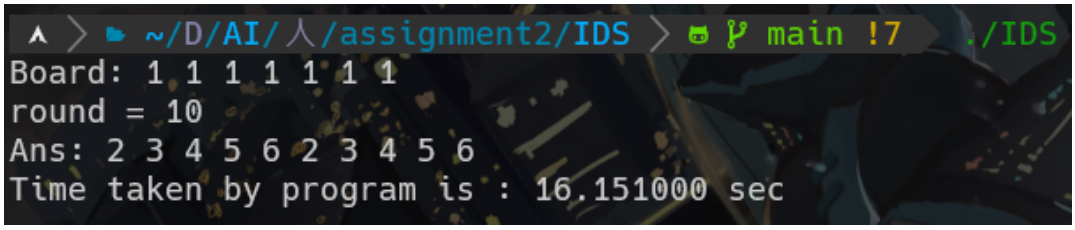
- 整體實作下來遇到的狀況基本上是一開始在建構資料結構和如何有層次性的拉好程式的架構
- 但中途實作 IDA* 和 IDS 時遇到的狀況較小
- 可是 IDA* 的 heuristic function 我真的除了「1的數量、disjoint 1's cluster」的想法到後面實作有成功，其他我還想的 1 isolated、1 左右 0 的數量之類的做法都是以失敗告終。我會繼續想其他做法的原因是我有聽到其他同學的 game board 長度可達 60 以上並且 10 sec 內可以跑完，因此我卡在想不到更好的 heuristic function。
- 備註：IDAStar folder 下的 `IDAStar_count_cluster_heuristic.cpp` 就是另一個不是效率較好的實作程式。

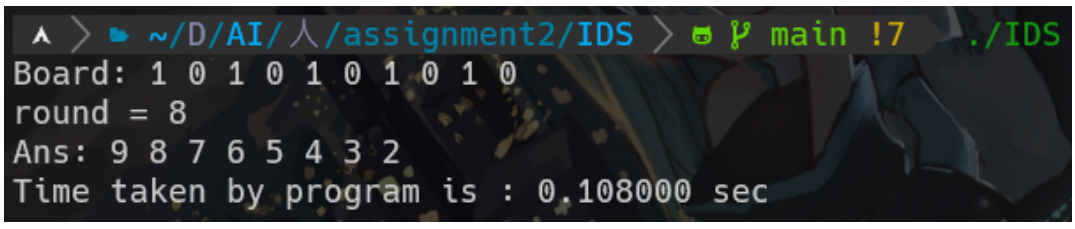
7

- 參考資料
  - https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/
  - https://web.ntnu.edu.tw/~algo/State.html#4

IDS

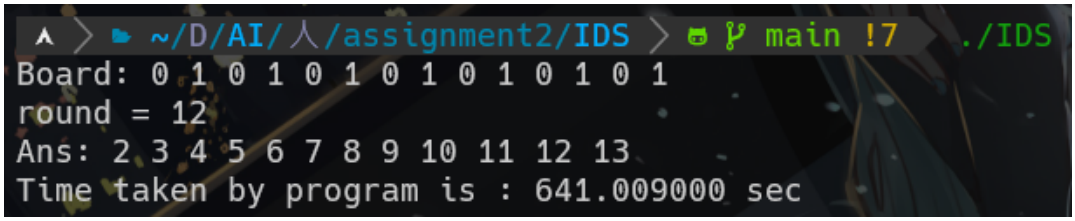
- testcase 1
  - 

- testcase 2
  - 

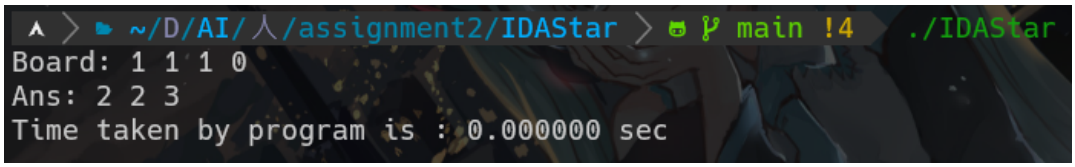
- testcase 3
  - 

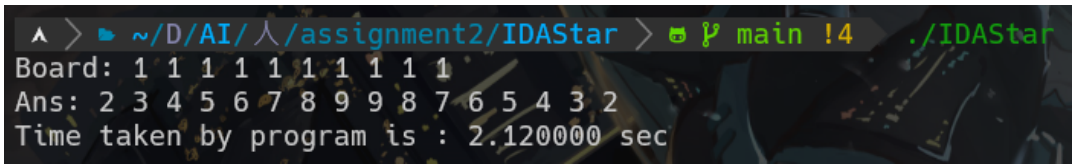
- testcase 4

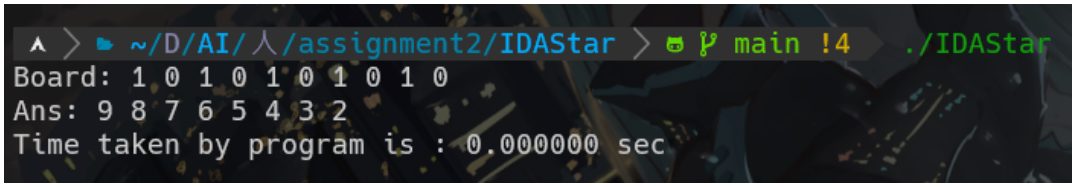
- 

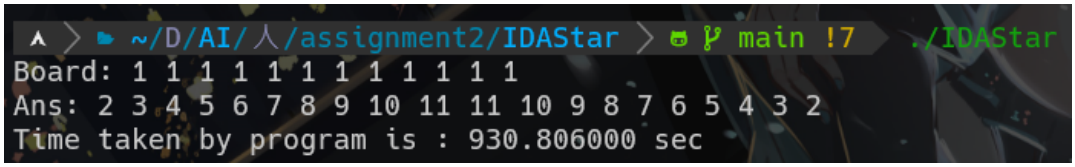
- testcase 5
  - 

IDAStar

- testcase 1
  - 

- testcase 2
  - 

- testcase 3
  - 

- testcase 4
  - 

- testcase 5
  -