人工智慧 HW3

TAGS: `111-02`

1.

姓名：黃定凡
學號：41047032S
電話：0965334650

作業系統：Mac OS 10.15.7
python環境：Python 3.10.5
c++環境：c++11 / apple clang version 12.0.0

2.

首先用 `make` 來編譯出 `./main` 執行檔，接著將輸入放進 `./input.txt` ，或用 `randinput.py` 來產生隨機初始盤面，詳細說明於第三題。環境備妥後，執行 `./main` ，即進到遊戲介面。
首先，需分別輸入兩位玩家的名字，若希望該玩家由AI來控制，即輸入 `AI` 。
進到下棋畫面後，照著提示輸入大寫字母或數字來指定要選擇的欄/列。

3.

下面五筆測試輸入皆為 python 程式隨機產生，若要產生隨機盤面，請用 python3 執行 `./randinput.py` ，接著分別輸入列數和欄數，若希望隨機則留空。
決定列/欄數後，程式將根據給定大小隨機產生 `input.txt`

input 1

```
3 4
1 1 1 0
0 0 1 0
1 1 1 1
```

input 2

```
4 8
0 1 1 1 0 0 1 1
1 0 0 0 0 1 1 1
0 1 1 0 0 1 1 1
1 0 0 0 0 0 1 0
```

input 3

```
0 1 0 0 0 1
1 0 1 0 0 0
0 1 0 1 0 1
0 1 0 0 0 0
1 0 1 1 0 0
1 0 1 0 1 0
```

input 4

```
6 6
7 7
1 0 1 1 0 1 0
0 0 1 1 1 1 0
0 0 1 0 0 0 0
0 0 1 1 0 0 0
0 1 0 1 1 0 1
1 1 1 1 0 1 0
0 0 1 0 1 1 1
```

input 5

```
8 8
0 0 1 0 0 1 1 0
0 0 1 1 0 0 0 0
1 1 1 0 0 1 1 0
0 0 0 0 0 1 0 1
0 1 1 0 1 1 0 0
0 1 1 1 0 1 0 1
0 0 1 1 1 1 0 1
1 0 1 0 1 0 0 0
```

4.

在此程式中，我定義了兩個 Class，分別是 `Board` 及 `Game`，`Game` 即包含一個 `Board` 及兩個玩家的資訊。
每個 `Board` 代表一個盤面，其中 `uint64_t board` 用來儲存盤面資訊，前 $64-mn$ 個 bit 為 0，後 mn 個 bit 則用來作 bitboard。
當 AI 要下棋的時候，使用 Minimax 演算法再加上 Alpha-Beta pruning 來做決策，每個 node 會將其所得的分數(顆數)傳遞給其 child，terminate node 則以最終雙方的顆數差(可正可負)當作權值回傳。

表現的部分，若沒有加入 alpha-beta pruning，只能解到 input 1，input 2 即需要非常久的時間，加入了剪枝之後，input 4 需要一秒多，input 5則需30秒左右，不過都能正確有效地找出最佳解。

5.

miniMax：https://www.freecodecamp.org/chinese/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm/
alpha-beta pruning：https://fu-sheng-wang.blogspot.com/2017/02/ai-16-alpha-beta-pruning.html
c++ 檔案讀寫：https://hackmd.io/@ndhu-programming-2021/BkZukG4jK
以上參考資料僅作為理解用，並沒有直接用在實作程式碼中。

6.

在寫物件導向的程式較困難的地方還是不好 debug，要寫到一定程度後才可以執行，因此第一次執行通常都滿滿的 error，不過這次一項一項修好後就可以運行了。

這次的實作卡住比較久的地方是我寫好後，雖然範例測資都過了。但在複雜的盤面中，同樣的盤面我卻下的 AI 好（即沒有找到最佳解），而且 8*8 的有時會跑不完。後來才發現我的 alpha-beta pruning 寫錯了，使用 move 的編號而非 score 來當作 pruning 的標準，將此 bug 修掉後程式即運行得非常順利。

除此之外沒有遇到其他困難了。