AI HW3

tags: ***AI HW***

41047025S 王重鈞

檔案說明

一、

編譯方式:

> 可使用我提供的makfile進行編譯，或使用下面的指令進行編譯
> g++ game.cpp -O3 -o game -std=c++17

執行方式:

> 在game.cpp上方有兩個定義 `INPUT_PATH` 及 `OUTPUT_PATH` ，是用來設定輸入文件及輸出文件的，預設為 `./input.txt` 及 `./output.txt` ，助教可依照需求去更改路徑。
> 記得在更改路徑後重新編譯一次。

> 利用makefile編譯過後，可直接執行 `./game` 。
> 使用者介面中，你可以選則1-3的模式，如果助教只需要改功課，可以使用mode (3) One step，這個模式將只會跑一步最佳解，並輸出一步到輸出文件中。
> 另外兩個模式則是 PVP 及 PVE 模式，助教可自行遊玩，輸出文件將會是**當前**遊玩步驟的最佳解。

使用C++版本:

> C++17

硬體設備作業系統:

> MacOS 記憶體8GB (僅有該電腦設備)

聯絡電話:

> 0905866533

二、

已有增加註解，但建議閱讀上方文件執行方式較為合適

三、

我撰寫了一個python程式 `case_generator.py` ，可以自行產生測資，可透過 `python3 case_generator.py -h` 查看他的參數，如果沒有參數，測資將會固定為10個，長寬隨機，輸出位置是當前資料夾，並命名第n筆資料為input[n].txt

四、

在此次功課，我使用 alpha-beta purning 找尋最佳解
因為盤面僅64個點，因此我利用bitboard技術（注：bonus 1.），使我用一個 `uint64_t` 便可儲存整個盤面
此外，我撰寫了一個 `take_mask` 的陣列，使提子速度從 O(max(n, m)) 變成 O(1)（注：bonus 2.）
在計算提子數量時，我使用 bit counting （注：bonus 3.）使計算提子數量的速度提升
為了維持文件整齊，測資我將放在文件的最後

五、

- alpha-beta pruning資料參考: **https://fu-sheng-wang.blogspot.com/2017/02/ai-16-alpha-beta-pruning.html** (https://fu-sheng-wang.blogspot.com/2017/02/ai-16-alpha-beta-pruning.html)
- bit counting: **https://www.geeksforgeeks.org/count-set-bits-in-an-integer/** (https://www.geeksforgeeks.org/count-set-bits-in-an-integer/)

六、

在確認測資是否正確時，經常會有人腦compile不出最佳解的情況，或是誤解最佳解是錯誤的，造成花大量的時間在 de 無意義的 bug。

Bonus

1. bitboard減少儲存空間

在此功課中我利用bitboard技術去儲存盤面，僅需64bit便可儲存盤面，大大減少記憶體用量，此外還可利用下方技術使速度更快

2. take_mask加速提子速度

在我們要移除棋子（下稱提子）時，我們必須耗用 O(max(n, m)) 的時間去計算提子後的結果。而我利用遮罩加快速度

```
0xFFFFFFFFFFFFFF00, 第一行遮罩
0xFFFFFFFFFFFF00FF, 第二行遮罩
0xFFFFFFFFFF00FFFF, 第三行遮罩
0xFFFFFFFF00FFFFFF,
0xFFFFFF00FFFFFFFF,
0xFFFF00FFFFFFFFFF,
0xFF00FFFFFFFFFFFF,
0x00FFFFFFFFFFFFFF,
0xFEFEFEFEFEFEFEFE, 第一列遮罩
0xFDFDFDFDFDFDFDFD, 第二列遮罩
0xFBFBFBFBFBFBFBFB, 第三列遮罩
0xF7F7F7F7F7F7F7F7,
0xEFEFEFEFEFEFEFEF,
0xDFDFDFDFDFDFDFDF,
0xBFBFBFBFBFBFBFBF,
0x7F7F7F7F7F7F7F7F
```

這些遮罩讓我提子時只需要將原本盤面與特定的遮罩做and運算，便可將該列或該行的棋子移除。

3. xor計算提子數量

但儘管如此，我們仍需耗用 O(max(n, m)) 去計算提子的數量，因此我將原盤面與提子後的盤面做xor運算，便會得到提子的位置，再利用 bit counting 演算法計算提子數量，使時間複雜度降低

Test Case

EXAMPLE TEST:

test1

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


Row#: 3
3 points
Total run time = 0.000015 seconds
```

test2

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


Row#: 1
0 points
Total run time = 0.000015 seconds
```

test3

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

```
Column#: 4
2 points
Total run time = 0.000130 seconds
```

RANDOM TEST:

test1

```
8 8
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1


Row#: 1
0 points
Total run time = 0.800627 seconds
```

test2

```
3 8
1 0 0 1 0 1 1 0
0 1 1 1 1 0 1 1
0 0 0 1 1 1 1 1


Row#: 2
5 points
Total run time = 0.000025 seconds
```

test3

```
2 3
0 0 1
0 1 1



Row#: 2
1 points
Total run time = 0.000012 seconds
```

test4

```
8 6
0 1 1 0 1 0
1 0 0 0 0 1
0 1 1 1 1 1
0 1 0 1 0 0
0 0 1 1 1 1
0 1 1 1 0 0
0 0 1 1 0 0
1 1 1 1 0 1


Row#: 3
2 points
Total run time = 0.077194 seconds
```

test5

```
5 8
1 0 1 1 1 0 1 1
0 0 0 0 0 0 1 1
0 0 0 1 0 1 0 0
1 0 1 0 1 1 0 1
1 0 0 1 0 0 0 0


Row#: 1
3 points
Total run time = 0.000202 seconds
```

以下皆為8*8測試

test6

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


Row#: 1
1 points
Total run time = 0.580583 seconds
```

test7

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


Row#: 3
4 points
Total run time = 0.021522 seconds
```

test8

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



Row#: 8
4 points
Total run time = 0.208534 seconds
```

test9

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


Row#: 7
3 points
Total run time = 0.154276 seconds
```

test10

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


Row#: 2
4 points
Total run time = 0.050594 seconds
```