

# Introduction to Programming (III) Project 3 - Reversi Tree Search

Deadline: 2018/1/11 (四) 23:59:59 (以 ilms 上傳時間為準)

↑ Y方向座標

9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	0	0	0	0	0	0	0	0	-1
7	-1	0	0	0	0	0	0	0	0	-1
6	-1	0	0	0	0	0	0	0	0	-1
5	-1	0	0	0	2	1	0	0	0	-1
4	-1	0	0	0	1	2	0	0	0	-1
3	-1	0	0	0	0	0	0	0	0	-1
2	-1	0	0	0	0	0	0	0	0	-1
1	-1	0	0	0	0	0	0	0	0	-1
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0	1	2	3	4	5	6	7	8	9

→ X方向座標

## Final Project

很開心各位同學能夠堅持到學期末，本學期的 Final Project 是有關於黑白棋的 AI 程式設計，主要是用來練習 Tree Search 的實作，你可以用各種創意去設計你的 Tree Search 來讓 AI 變厲害。

## 作業說明

你需要寫的地方是 I2P\_ID2 function (ID2 改成自己的學號，例如: 105062999)，這個 function 預設是隨機落子，你可以改掉裡面的選擇下棋的方法，去增強你的 AI。下面介紹 I2P\_ID2 function:

pair<int, int> I2P\_ID2 (int Board[N][N], int player, vector<pair<int, int>> ValidPoint)

- **int Board[N][N]**: 棋盤，詳見最上面的棋盤圖  
(1 表示黑棋, 2 表示白棋, 0 是空格, -1 是棋盤外面的一圈)  
(-1 是不能下棋的地方，使用 -1 圍一圈的目的是比較容易藉此判斷棋盤的邊界)  
(你唯一能下的點，是空格的地方，但是下棋時仍要符合黑白棋的規則)  
(因此並不是每個空格你都可以下，必須是吃的到棋子的空格，才能下棋子)
- **int player**: player=1 表示輪到黑棋，player=2 表示輪到白棋
- **vector<pair<int, int>> ValidPoint**: 這個裡面存放了所有你目前可以下棋的點(valid point)  
(預設是從 ValidPoint 隨機挑出一個點)  
(你可以直接從 ValidPoint 裡面挑出一個你覺得不錯的點)

※黑白棋並不是每一個點都可以下，可以下的點必然是吃的到對手棋子的點

(吃子的規則，以及相關遊戲規則詳閱 WIKIPEDIA-Reversi)

## 遊戲規則 [編輯]

棋盤共有8行8列共64格。開局時，棋盤正中央的4格先置放黑白相隔的4枚棋子（亦有求變化相鄰放置）。通常黑子先行。雙方輪流落子。只要落子和棋盤上任一枚己方的棋子在一條線上（橫、直、斜線皆可）夾著對方棋子，就能將對方的這些棋子轉變為我已方（翻面即可）。如果在任一位置落子都不能夾住對手的任一顆棋子，就要讓對手下子。當雙方皆不能下子時，遊戲就結束，子多的一方勝。

## 使用說明

將主程式(main.cpp)配上兩個檔案(SmileReversi00.cpp, [SmileReversi01.cpp](#)) 放在一起編譯並執行，該路徑檔案夾會產生一個棋譜檔案(Chess.txt)，可以點開來看對奕結果。

你需要寫的部分是 [SmileReversi01.cpp](#)。

main.cpp 是主程式，不做任何修改，有需要任何函數，可以自行引入到 SmileReversi01.cpp。

SmileReversi00.cpp 是預設電腦 AI (random 下棋)，有需要的話，你可以把預設電腦 AI 設計的強一些，讓你的對弈實驗有一個好的對照組。

## 評分說明

		example	example score
Basic	3.0 分 Tree Search		3.0
	2.0 分 與 random 下棋電腦對戰 (暫定 10 盤)	8 勝 2 敗 勝率 80%	$2.0 * 80\% = 1.6$
	2.0 分 與淺層 search 電腦對戰 (暫定 10 盤)	6 勝 4 敗 勝率 60%	$2.0 * 60\% = 1.2$
Report	3.0 分	認真寫	2.4
比賽	自由參加	X	X 0.0
Total			8.2

### Basic (7.0)

3.0 分 Tree Search: 實作可以利用 Recursion + Value function

- step 1: 取得當下對局者盤面中，可以下的點。(vector<pair<int, int> > BoardPointValid())
- step 2: 複製 t 個盤面(void BoardCopy()), t 是 step 1 可下點的數目
- step 3: 自行撰寫函數，將這 t 個盤面傳到下一層，  
在下一層試下(bool PlayReversi())這 t 個可以下的點，  
並且在下一層使用自行設計的 value function 去評估這手棋的分數。  
(example: value 可以設定成，這步棋吃了幾顆棋子，但是這不一定是好的設計)
- step 4: 輪到對手下棋(int player 更換)  
(括號內的部分，在 main.cpp 主程式有對應的 function，可以直接複製過去修改使用。)

將這 4 個步驟重複執行，並設定 depth(搜尋深度)作為截止條件。

value function 由下一層盤面做一些運算而得到 (ex. max, min, mean... etc)

alpha-beta tree search 是一種進階，效果不錯的方法，亦可參考其 value function 的定義方式。

```
01 function alphabeta(node, depth, α, β, maximizingPlayer)
02     if depth = 0 or node is a terminal node
03         return the heuristic value of node
04     if maximizingPlayer
05         v := -∞
06         for each child of node
07             v := max(v, alphabeta(child, depth - 1, α, β, FALSE))
08             α := max(α, v)
09             if β ≤ α
10                 break (* β cut-off *)
11         return v
12     else
13         v := +∞
14         for each child of node
15             v := min(v, alphabeta(child, depth - 1, α, β, TRUE))
16             β := min(β, v)
17             if β ≤ α
18                 break (* α cut-off *)
19         return v
```

(alpha-beta tree search 的精神是可以減去不重要的分枝，

如果使用 BFS 的話，會把這些不重要的分枝，也都搜尋過一遍)

如果使用 alpha-beta tree search 進行實作，並且在 report 詳細解釋其運作方式，助教視實作完整度 + report 解釋詳細程度，給予適當的 bonus 加分。

參考資料: [https://en.wikipedia.org/wiki/Alpha-beta\\_pruning](https://en.wikipedia.org/wiki/Alpha-beta_pruning)

## 2.0 分 與 random 下棋電腦對戰：

助教一開始給的 code 就是 random 電腦，可以多跟它對弈，驗證自己的下棋策略是不錯的。

## 2.0 分 與淺層 search 電腦對戰：

淺層 search 的部分，助教會用一些很簡單的算法，進行測試，

如果 AI 設計妥當的話，這部分也能拿到不錯的分數。

## Report (3.0)

- 詳解方法：  
請詳細說明你的 tree search 方法、value function 定義方式，以及你的方法有什麼樣的特色。
- 其他努力(任何你曾經嘗試過的新方法，以及任何創新的想法，都可以寫在這地方)  
(以及你在程式裡面，比別人多做了些什麼樣的設計)
- 心得：  
你在這次的 project 裡面學到了些什麼，有什麼感想，或是有什麼特別的發現。  
碰到了些什麼困難，你是怎麼解決它。  
其他對於 project 的建議。

## 比賽 (自由參加，比賽時間 1/12 (五))

請自備隨身碟與筆電，現場抽籤安排賽程，對戰的兩隊將 code 用隨身碟傳到同一台筆電，使其對弈，並在對弈完成後，現場與助教回報戰績。

### 比賽獎勵：

如果參賽隊伍數有 16 隊以上的話，取前 10 名進行加分獎勵。

- 第一名 學期總成績 +5.0 分
- 第二名 學期總成績 +4.5 分
- 第三名 學期總成績 +4.0 分
- 第四名 學期總成績 +3.5 分
- 第五名 學期總成績 +3.0 分
- 第六名 學期總成績 +2.5 分
- 第七名 學期總成績 +2.0 分
- 第八名 學期總成績 +1.5 分
- 第九名 學期總成績 +1.0 分
- 第十名 學期總成績 +0.5 分

若參賽隊伍少於 16 隊的話，視情況調整加分獎勵。

(此外，現場視情況會用 random 電腦對戰測試 code，對戰 random 電腦勝率達到 70% 的 AI，才能獲得加分獎勵，若對戰 random 電腦勝率過低者，該名次得以從缺，參賽前務必確保自己寫的 AI 對戰 random 有著不錯的勝率。)

## 注意事項

- 你需要寫的部分是 SmileReversi01.cpp。  
main.cpp 是主程式，不做任何修改，有需要任何函數，可以自行引入到 SmileReversi01.cpp  
SmileReversi00.cpp 是預設電腦 AI (random 下棋)，有需要的話，你可以把預設電腦 AI 改寫的強一些，讓你的對弈實驗有一個好的對照組。
- 把主程式第五行 (#define SECOND ID2) 中的 ID2 改成自己的學號
- 所有在 I2P\_ID2 function 外要使用的函數與變數，請在 namespace s\_ID2 (ID2 改成自己的學號) 進行宣告，違反者扣五分 (以下以 void hi() 作為範例)

●

```
namespace s_ID2
{
    int N;
    void hi();
}
```

使用時，利用 `s_ID2::hi()` 來呼叫 function。

● 編譯使用 C++11

### 違反下列事項者零分

- 每一手棋時限 2.0s (2000ms)，測量時間使用的是 `clock_t` (單位:ms)。
- 下棋下在不能下的點  
(黑白棋並不是每一個點都可以下，可以下的點必然是吃的到對手棋子的點)  
(可以下的點會在 `ValidPoint` 裡面)
- 禁止抄襲  
(使用助教 `main.cpp` 裡面的 code 是允許的，這裡指的抄襲是指抄同學、網路上的答案。)
- 禁止撰寫會讓電腦跑很久的無窮迴圈，以及禁止任何與 `tree search`、下棋策略無關的 code
- 程式碼無法執行
- 使用非 C++11 的 standard library 的 library
- 使用 `asm`
- 使用網路
  - 如 MPI
- 使用非 CPU 的資源
  - 如 GPU
- 使用非 CPU 指令
  - 如 SSE
- 

### 上傳檔案(檔案擺放方式)

106123456.zip

//106123456 的位置填入自己的學號

//上傳時把程式碼檔案(`SmileReversi01.cpp`)，名字改成 s+自己的學號.cpp

//上傳 `Report_學號.pdf`

違反者扣五分

106123456

├── s106123456.cpp

└── Report

    └── Report\_106123456.pdf

### 如何讓 AI 變強

- Rule Based: 設定一些已知的下棋規則、技巧，這些技巧需要一些黑白棋專業知識。  
關於黑白棋的規則，可以詳見維基百科，<https://en.wikipedia.org/wiki/Reversi>  
練習下棋的網站：<http://qk-gamer.com/games/gk2465.php>
- 與其他組別切磋對弈，下友誼賽。