Theory of Computer Games, Fall 2018
Homework 2

B04902012 Han-Sheng Liu, CSIE, NTU
E-mail: b04902012@ntu.edu.tw

1 Compilation and Execution

```
cd source
g++ b04902012.cpp -O2 -o b04902012
./b04902012
```

2 Implementation

2.1 Encode/Hash

一個盤面是由 5×5 的棋盤，以及若干個棋子組成。其中，棋子可以分成四類：

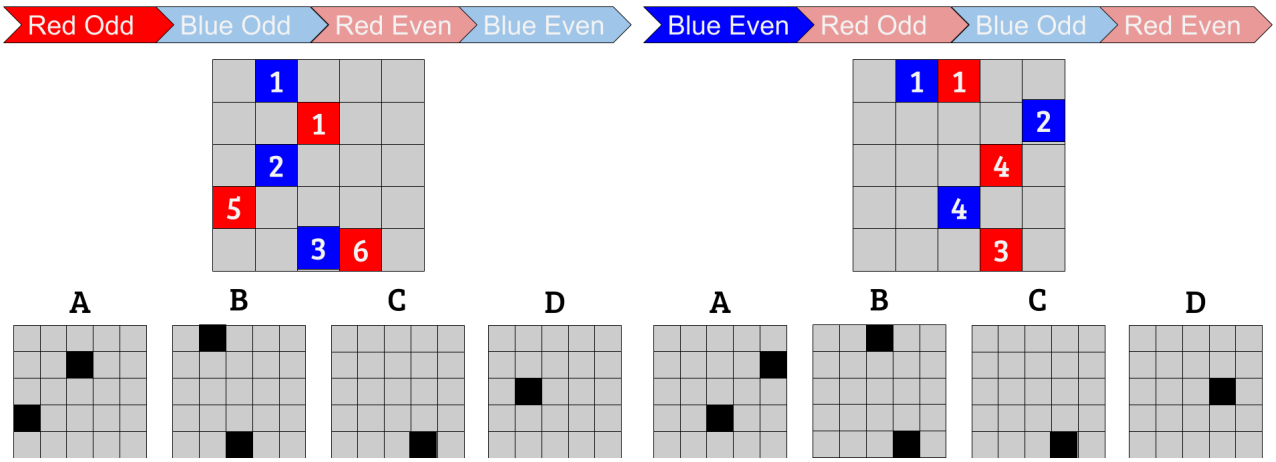
- A: 本回合可移動的棋子
- B: 下回合可移動的棋子
- C: 下下回合可移動的棋子
- D: 下下下回合可移動的棋子

如果有兩個盤面，它們的四種棋子的位置分佈都相同，那麼它們便應該被等價地看待。注意到位置是以「該回合的玩家」的視角爲準，而不是棋盤的絕對位置。對於紅色玩家和藍色玩家，可以分別把 25 個格子以對稱的方式編號 0 至 24。

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

24	23	22	21	20
19	18	17	16	15
14	13	12	11	10
9	8	7	6	5
4	3	2	1	0

舉例來說：以下的兩個盤面雖然輪到的人不同，棋子的擺放也不一樣，但它們的四種棋子的位置分佈，對於當前回合的玩家而言，都相同，因此可以視爲等價。



$A = \{7, 15\}$
$B = \{2, 23\}$
$C = \{23\}$
$D = \{13\}$
我的編碼方式是先把 A, B, C, D 分別以上述的順序轉換成一個長度爲 25 的 bitset，再把這四個 bitset 接起來，形成一個長度爲 100 的 bitset。
$Enc(A) = 0000000001000000010000000$
$Enc(B) = 0100000000000000000000100$
$Enc(C) = 0100000000000000000000000$
$Enc(D) = 0000000000010000000000000$
$Enc(\text{Board}) = Enc(D)Enc(C)Enc(B)Enc(A)$
Board 在一手之後的盤面會是 $Enc(A')Enc(D')Enc(C')Enc(B')$。

2.2 Information Storage

我建立了一個 `bitset -> pair<double, double>` 的 hash table `trans`，一個盤面的勝場數和敗場數會被紀錄在 `trans[`$Enc(\text{Board})$`]` 中。這樣的好處是，可以重複使用等價盤面的信息量。

2.3 Searching Strategy

定義 $Move(Enc(\text{board})) = \{Enc(\text{all possible board after one move})\}$. 採用 *Monte-Carlo Tree Search*，並對勝/敗場數的平均與標準差以 *Progressive Pruning* 作爲優化。$Enc(\text{board})$ 的小孩是所有 $Move(Enc(\text{board}))$ 內的元素。由於 Enc 的編號是以「該回合的玩家」爲標準，造成的效果就像是每一手都把盤面旋轉 180 度。因此我的作法比起 Min-Max，更像是 Nega-Max。

2.4 Parameters

- Simulating times per child when expanding: 100
- Searching timeout: 8 sec.
- r_d: 1.0
- σ_e: 0.3