Theory of Computer Game Final Project 資工所碩一 R05922068 彭宇劭

source code: main.cc / Evaluation.h / HASH.h / makefile / original template code

環境： Ubuntu 14.04 LTS

編譯執行方式及環境：

```
cd src
make
cd ../EnterRoom or CreateRoom
./DarkChess_linux 1
```

檔案說明：

- main.cc 程式主控區、實作NegaScout，並在過程中呼叫使用下面兩個檔案
- Evaluation.h 實作審局函數，主要分為移動棋子及翻子兩部分
- HASH.h 定義Transposition Table相關操作函數，以及Hash Node裡的資料結構形式
- makefile compile程式並將執行檔search複製到[Create],[EnterRoom]/Search資料夾中

實作部分：

- NegaScout
- Transposition Table
- Iterative Deepening (Simple Dynamic)
- Evaluation Function
  - 走步審局
    - 不同子力大小權重
    - 兵卒權重根據數量以及對方將帥存活調整調整
    - 將軍差距（為了彌補兵卒被降低價值的空缺）
    - 殘局加分（當對方棋子少於四子時）
    - 雙方最大子棋力相同情形（ 關廁所）
    - 自己最大子無剋星加分
  - 翻子審局
    - 砲位置分數
    - 鄰居分數

Hash Table

Size: 0xFFFFFFF (27 bits)

Data Structure

Data Type & Name	說明
int depth	還需要搜尋的層數
int score	目前盤面的分數
int flag	上面的分數是Exact(1) 還是Bound(2) 或是空的(0)
MOV bestMov	這個盤面搜到的最好走步

Function

HASH()

constructor 分配Hash Table空間

void initial_hash_table()

初始化Hash Table每個Node的Flag=0 (空)

void insertHash(uint64_t key, int flag, int cut, int score, MOV bestM)

將目前盤面資訊存入Hash Table
若Flag=0(空)直接存入
若Flag!=0且剩餘搜尋深度大於Hash Table內存的，更新
若Flag!=0但剩餘搜尋深度小於Hash Table內存的，不變

int getFlag(uint64_t key, int cut)

輸入目前盤面計算出來的key，以及所剩餘的搜尋深度cut
查看Hash Table對應的位置比較flag及cut，回傳對應的Flag值

int getExactVal(uint64_t key)

根據輸入的key，取後27bit當index，輸出對應HashTable裡的score值
因為在呼叫這個Function之前有先取得比較過Flag，所以這裡直接取值

int getBound(uint64_t key)

根據輸入的key，取後27bit當index，輸出對應HashTable裡的score值
因為在呼叫這個Function之前有先取得比較過Flag，所以這裡直接取值
跟上面Function相同

int getBestMov(uint64_t key)

取得當前盤面獲得最佳分數的那一個走步（Best Child）

Algorithm & Relative Function

- **Iterative Deepening & Time Control**
  根據目前所剩餘時間做調整思考時間
  根據目前合法走步數量調整搜尋深度
  設定後利用for-loop由淺至深搜尋
  當如果搜到獲勝棋步則不繼續搜下去

```
// 當時間只剩下30秒 每步思考時間縮短為1秒
if (remain_time < 30*1000)
    DEFAULTTIME=4;

// 當時間只剩下100秒 每步思考時間縮短為3秒
else if (remain_time < 100*1000)
    DEFAULTTIME=6;

// 當合法步數小於10 深度減少
if (lst.num < 10)
    ITER_DEEP=7;
else
    ITER_DEEP=12;

// Iterative Deepening
for (int i=1; i<ITER_DEEP; i++){
    scout_val = NegaScout(B, -INF, INF, 0, i);
    if (scout_val==WIN)
        break;
}
```

目前搜尋深度　剩餘搜尋深度

- **SCORE NegaScout(const BOARD &B, int alpha, int beta, int dep, int cut)**
  依照上課投影片提供的虛擬碼實作
  1. 判斷目前盤面是否輸了，若輸了直接回傳-WIN
  2. 判斷是否搜到設定最深層||時間到||沒有合法步，根據目前min/Max回傳Eval值
  3. 取得當前盤面hash值，並利用此key取得這個Hash Node的Flag
  4. 根據回傳的Flag，直接回傳Exact值 || 更新Lower Bound || 不做事
  5. 都設定完成後，進入For Loop跑過目前的合法走步
  6. 期間如果發生cut-off須判斷是否需要將值存入或更新Hash Table
  7. 出回圈將目前的最佳分數，判斷要不要存入Hash Table，並回傳目前最佳值

- **void generate_random_state_turn()**
  在對局一開始時，會先產生15*32+2個64bit random number
  (15: 雙方棋子+未翻開的　32: 棋盤所有位置　2: 現在輪到誰)
  因為蓋著的棋子對其他子來說是個路障，所以也需要考慮，而最後兩個random number而是目前輪到誰，同樣的盤面輪到不同人下，是不同的情形

- **uint64_t getZobristKey(const BOARD &B, int depth)**
  將目前盤面上的每個子及位置、輪到誰，對應上面Function產生的random number，所有值做exclusive-or回傳

```
uint64_t getZobristKey(const BOARD &B, int depth){
    uint64_t key = 0x0;
    for (int i=0; i<LOCATION; i++){
        if (B.fin[i]<15){
            key = key^state[B.fin[i]][i];
        }
    }
    key = key^turn_who[depth%PLAYER];
    return key;
}
```

Evaluation Function and Knowledge

變數及函數說明

int material[16] = {1000,1200,200,50,10,600,50, 1000,1200,200,50,10,600,50, 0,0} 每個棋子的子力，依序為{將,士,象,車,馬,砲,兵,未翻,空}，兵在之後會動態加分
int fin_my_cannon[7] = {100, 100, 30, 20, 10, -100, 5} **int fin_en_cannon [7] = {-100, -100, -10, -10, 0, 30, 0}** 我方及敵方在翻子時若翻出砲獲得的獎勵或懲罰 Ex. 當離我這個位置兩格且中間有東西的地方有隻帥，如果翻到砲就賺到
int fin_my_neighbor[7] = {-50, 10, 5, 3, 1, -1000, 10} **int fin_en_neighbor[7] = {10, -50, -10, -5, -5, 100, -10}** 我方及敵方在翻子時若周圍環境分數 Ex. 當這個位置旁邊有隻敵方的砲，我得100分，因為很有可能可以吃掉
int material_value (int dep) 目前棋盤走步的審局函數，下方詳述
int get_fin () 翻子的審局函數

走步分數計算（material_value）

- **pure_material**
  計算雙方子力大小差距

- **pins_dynamic_val**
  動態的增加兵卒的分數，當對方的將帥還活著，數量越少分數越高

- **king_live**
  為了平衡上面當將帥被吃掉時所掉的分數
  原本兵卒因為數量稀少可能分數非常高，但將帥被吃掉會很多掉分

- **dis_val**
  進入殘局時（對手不到四子），加入距離條件
  並把吃子當成首要工作 [line 70]
  接著將對手殘子跟自己的棋子計算分數，距離越近子力差距越大分數越多
  並且處理將帥跟兵卒的關係

- **end_val**
  當雙方最大子力棋子等級相同時，加入更高且較警慎的距離分數
  需要越來越接近，但又不能剛好走到敵人的旁邊
  而當自己的最大子無剋星時另外加大量分數

翻棋分數計算（get_fin）

由棋盤頭掃到尾，每個尚未翻開的點算出一個分數，最後翻出最大分的棋子
分為砲擊位置、鄰居位置
砲擊位置左右加減2 上下加減8，找出砲擊目標，並取得分數
鄰居位置左右加減1 上下加減4，判斷鄰居可能的弱點或威脅

遇到問題及解決方法

寫好NegaScout時一直無法提升能力
仔細去Trace Code後才發現，原本給的Evaluation Function的觀點是根據當前輪到誰來記分，一開始沒注意到這點，導致min Max層在加負號會分數剛好顛倒。

修正上述問題後，在殘局一直無法贏
這時候去看原生的CheckLose並Trace code，這才發現CheckLose的觀點也是對當下盤面，所以在NegaScout裡面不用根據目前是min Max層反轉，一律都 return -WIN即可

比賽結果（2W 3L 9D）

比賽中遇到很多和局，有幾局都是殺到對方剩下少數子，無法收下勝利，在敵方離自己很遠時，常常會被自己的棋子擋住，而在左右游移不定，造成長抓和局。
在賽前Debug、調整參數都是跟助教提供的TCG1做測試，勝率大約在8成左右，比較少遇到這種情況，我想可能的原因是，助教提供的程式，會比較隨機的移動，而如果遇到跟我類似的同學就會一起左右游移，導致和局。
在殘局部分的審局少考慮到了一些情形，我在殘局有特別寫當雙方最大子力如果相同時，會使用關廁所法，利用距離加分，而且絕對不能主動走到對手旁邊，不過這裡忽略了其他子力權重的分數問題，有時候會過度保護棋子不被吃掉，而小心翼翼的不跟對方換不重要的棋子。