Data Structure Assignment 2
日本將棋對弈程式 Result Report

通訊二 110503518 李秉宸

December 14, 2022

1 Version 版本紀錄

v1.0 以陣列型態完成基本規則判定、持駒、持駒打入、升變、悔棋、棋譜紀錄、棋譜讀取與基本計時器。 v1.1 修復攻方可移動敵方部分棋子的問題。
v2.0 將對局過程改以Linked list存取。
v3.0 更新以Libev顯示計時器(包含累積以及單手花費時間)

2 Usage 程式明與使用教學

2.1 部分變數定義

Table 1: 部分變數定義

變數名稱	定義
state	表示程式狀態，初始化為0，當程式開始執行時，變為1並記錄開始時間。
turn	表示第幾手，初始化為1。
MAX_turn	表示執行讀取棋譜時，最後一手的大小，初始化為0。
checkerboard	10*10的二維陣列，存取棋盤所有資訊，初始化為0（每一格皆為空)。
checkerboard_memory	500*10*10的三維陣列，用於push和pop棋盤所有資訊。
X_chess_memory	500*20的二維陣列，用於push和pop X方駒台的資訊。
Y_chess_memory	500*20的二維陣列，用於push和pop Y方駒台的資訊。
game_mode	表示運行模式，1表示開新局、2表示讀取棋譜、3表示遊戲結束。
top	代表checkerboard_memory最後一筆資料位置。
X_chess_top	代表X_chess_memory最後一筆資料位置。
Y_chess_top	代表Y_chess_memory最後一筆資料位置。

2.2 棋子編號定義

Table 2: 棋子編號定義

棋子	藍方（X方）	紅方（Y方）	棋子	藍方（X方）	紅方（Y方）
香車	1	-1	步兵	8	-8
桂馬	2	-2	成香	9	-9
銀將	3	-3	成桂	10	-10
金將	4	-4	成銀	11	-11
王將	5	-5	成步	12	-12
飛車	6	-6	龍王	13	-13
角行	7	-7	龍馬	14	-14

藉由正負號定義棋子編號，可簡化持駒與打入的編號轉換問題，在判定棋子類型時也可用abs取對簡化判定。

2.3 計時器

本版本計時器使用Libev函式庫製作，示意圖如下

1、每一手開始後，攻方玩家紀錄開始時間，輸入完畢後記錄完成時間。

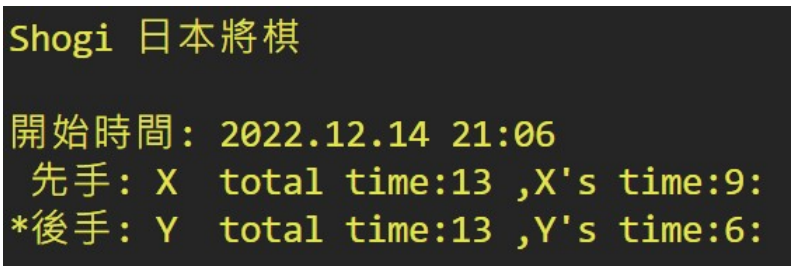

Figure 1: 計時器示意圖

2.4 基礎規則判定

輸入欲移動的棋子座標時，系統會依下列順序依序判定是否正確：

1、開始計時，檢測輸入是否為”0”、”-1”或”10”，若不是，則繼續檢測。

2、檢初始座標是否為攻方棋子，若是，則繼續檢測。

3、檢目的座標是否為敵方棋子或為空，若是，則繼續檢測。

4、檢方向、移動距離是否符合規則，以及利用迴圈檢路徑中間是否有其他棋子，若符合規則，則繼續執行。

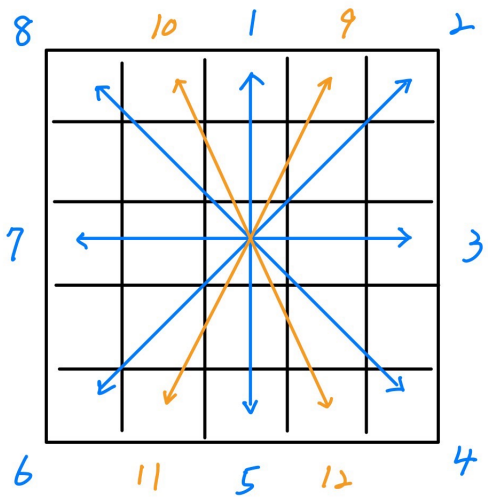

Figure 2: 方向判定明圖

Figure 2為方向判定明圖，整個棋盤皆適用於該方向表，下方為我方（藍方）、上方為敵方（紅方），其中方向9-12為桂馬專用方向，其他則為常規的八個方位。

程式將透過初始與終點座標判定出方向，再確認該棋子是否可行走該方向（如我方銀將僅能走12468方向），且移動步數是否正確，若涉及長距離移動（如：香車、飛車、角行、龍王、龍馬），則額外判斷經過路徑是否有其他棋子

5、更新棋盤狀態（初始位置變為0，新位置變為新棋子代號）。

6、若涉及吃子，則先判定被吃的子是否已經升變。

7-1、已升變：將狀態變回原始狀態，更換敵我方代號（*(-1)），並存取至攻方持駒陣列中。

7-2、未升變：直接更換敵我方代號（*(-1)），並存取至我方持駒陣列中。

8、將資料push入stack中，並將top加一，turn加一。

9、更新計時，重新印出棋盤並由一方繼續行棋。

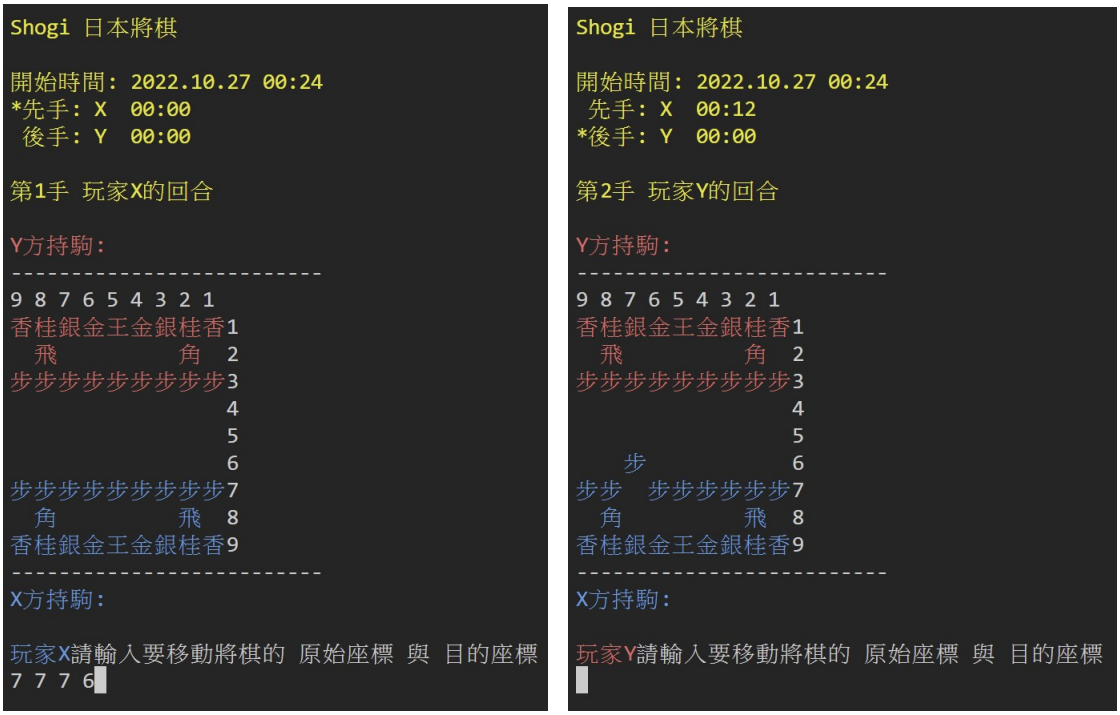

Figure 3: 對局示意圖

2.5 悔棋

悔棋時，系統會依下列順序依序判定是否正確：

1、開始計時，檢測輸入是否為"0"，若是，則繼續檢測。

2、確認stack是否為空（為空表示回到第一手），若不是，則繼續檢測。

3、將stack中的資料pop出來，並將top減一，turn減一。

4、更新計時，重新印出棋盤並由一方繼續行棋（悔棋前所累積之時間不會消除）。

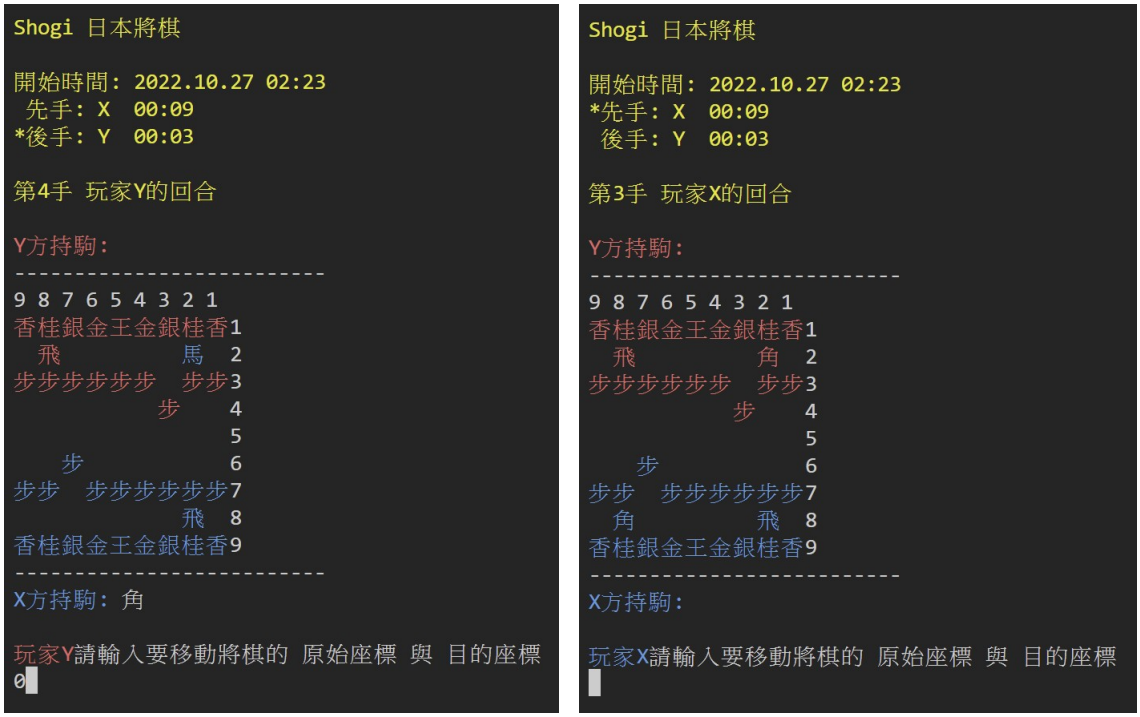

Figure 4: 悔棋示意圖

2.6 持駒

吃子後，系統會依下列順序依序執行：

1、將攻方駒台陣列（X.chess或Y.chess）的index（X_chess_top或Y_chess_top）加一，並將基礎規則判定中第7點所更換完的代號儲存至駒台陣列中。

2、將資料push入stack中，並將top加一，turn加一。

2.7 持駒打入

攻方回合時，依下列順序執行可進行持駒打入：

1、開始計時，檢測輸入是否為"10"，若是，則繼續檢測。

2、確認攻方駒台陣列是否為空（為空表示尚未擁有持駒），若是，則顯示錯誤並要求重新執行，反之則繼續執行。

3、使用者輸入整數1-N（N為駒台陣列數量，也就是持駒數量）以選擇所要打入的持駒。

4、使用者輸入要打入的位置，並確認該位置為空，若是，則繼續檢測。

5、確認打入的位置符合基礎規則（包含不可一筋二步、不可打入死棋），若符合，則繼續執行。

6、清空打入後的駒台陣列元素，並利用for迴圈將後續的持駒資料向前平移一格陣列。

7、更新計時，重新印出棋盤並由一方繼續行棋。

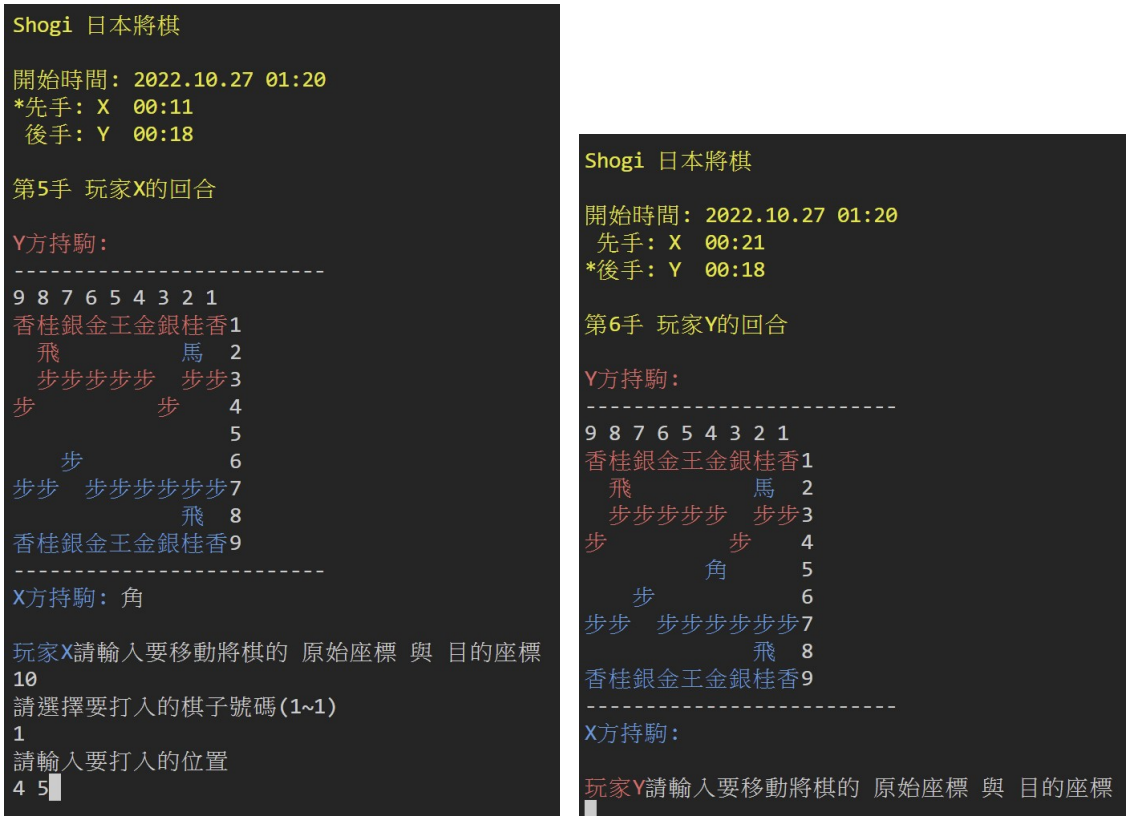

Figure 5: 持駒打入示意圖

2.8 升變與強制升變

Table 3: 升變後棋子文字明

棋子	步兵	香車	桂馬	飛車	角行	銀將	金將	王將
升變前	步	香	桂	飛	角	銀	金	王
升變後	と	杏	圭	竜	馬	全	無法升變	無法升變

行棋後，依下列順序執行可進行升變：
1、行棋後，若目的座標滿足升變條件（進入敵陣、在敵陣移動或是得強制升變），程式會詢問使用者是否進行升變。
2-1、非強制升變情況下，使用者輸入”Y”或”y”，則將棋子進行升變（更新棋子代號）。
2-2、非強制升變情況下，使用者輸入”N”或”n”，則棋子不進行升變。
2-3、強制升變情況下，使用者無需輸入任何容，棋子會強制進行升變。

備註1：強制升變是指該棋子已無其他前進方式，如步兵或香車走到敵陣的最後一列。
備註2：升變後的棋子被吃掉時，需變回原型態。

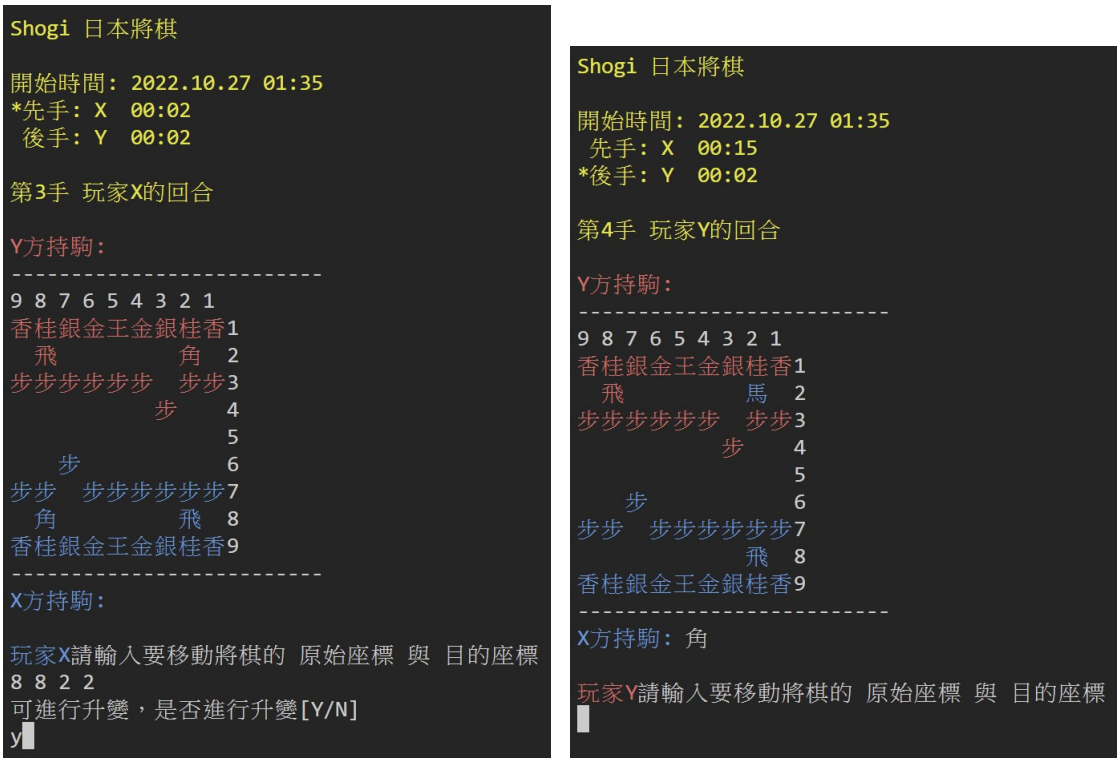

Figure 6: 升變示意圖

2.9 寫入棋譜

新局進行過程中，程式依下列順序執行棋譜寫入：

1、棋譜格式請使用.txt檔，格式如下：

第幾手/目的座標（筋段）/初始移動棋子/初始座標（筋段）/移動後是否升變

如：1 7 6 步 7 7 N 代表第1手，步兵從77走到76，未升變

2、開檔案，檔名為使用者在getopt函式中自定義的名稱。

3、寫檔會在玩家輸入完畢座標，並確認移動規則正確後才進行紀錄。

4、若玩家進行”悔棋”，則利用fseek函式將光標（讀寫頭）上移一排並清除該行資料。

5、若玩家進行”打入”，則寫檔時的初始座標預設為(9, 持駒矩陣index)，方便讀取棋譜時，系統可從駒台提取對應的持駒打入。

6、玩家輸入”-1”或將死而結束遊戲時，關閉檔案。

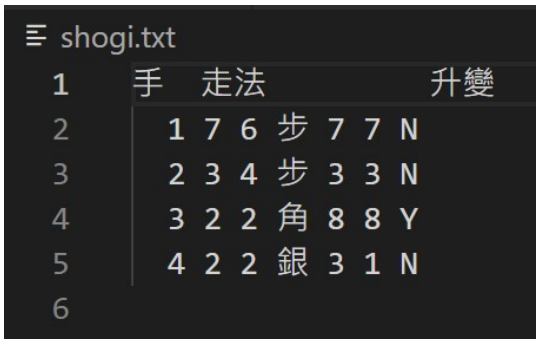

Figure 7: 寫入棋譜

2.10 讀取棋譜

複盤模式中，程式依下列順序執行棋譜讀取：

1、開檔案，檔名為使用者在getopt函式中自定義的名稱。（須為已存在且格式正確的txt檔）

2、程式在背景將每一手的移動方式完成並存於link list中（對局過程），與stack中（駒台）。

3、程式回到第一手顯示棋盤，使用者可輸入"f"移動至下一手或"s"結束程式。（英文字母大小寫皆可）

4、當使用者輸入"f"時，linked list 的node移動為node指向right，駒台的stack top會+1並pop出原本的資料。

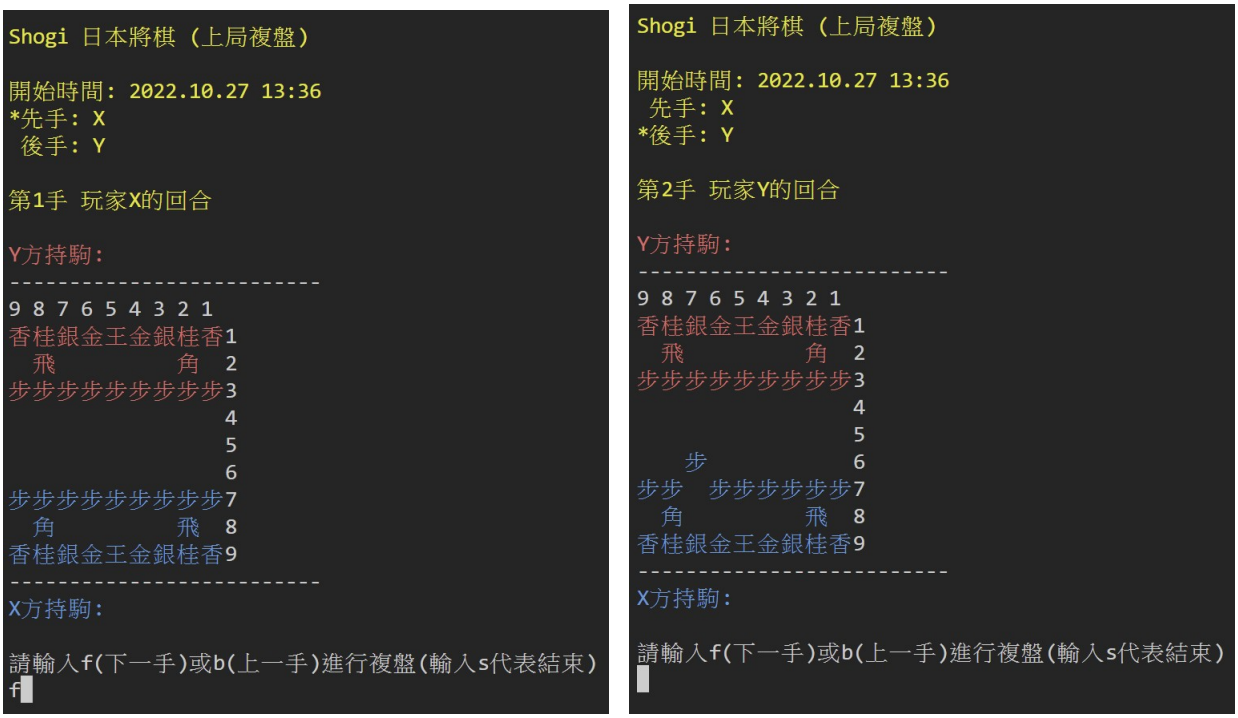

Figure 8: 讀取棋譜

2.11 Linked list使用位置

為了達到可以在回放時使用”f”以及”b”變更前後手，我使用Double linked list存取對局過程，同時在讀檔時可減免Reverse的過程。唯目前在輸入”b”時，發生預期外的部分資料失問題（如存取的座標自動變為0，但其餘資料正常存取），尚在尋找問題。

```
if(top >= STACKSIZE){
    printf("儲存空間已滿\n");
}
else{
    top++;

    if((tmp=(NODE*)malloc(sizeof(NODE))) != NULL){
        tmp->initial_row_memory = initial_row;
        tmp->initial_column_memory = initial_column;
        tmp->initial_chess_memory = initial_chess;
        tmp->goal_row_memory = goal_row;
        tmp->goal_column_memory = goal_column;
        tmp->goal_chess_memory = goal_chess;
        tmp->left = NULL;
        if(stack != NULL){
            tmp->right = stack;
            stack->left = tmp;
        }
        stack = tmp;
    }

    for(i=0; i<20; i++){
        X_chess_memory[top][i] = X_chess[i];
        Y_chess_memory[top][i] = Y_chess[i];
    }

}
```

Figure 9: Double Linked List(Push)

```
if(stack == NULL){
    printf("無法再悔棋\n");
    move_chess();
}
else{
    fseek(new_shogi_game, -18, SEEK_CUR);

    tmp = stack;
    stack = stack->right;
    checkerboard[tmp->initial_row_memory][tmp->initial_column_memory] = tmp->initial_chess_memory;
    checkerboard[tmp->goal_row_memory][tmp->goal_column_memory] = tmp->goal_chess_memory;
    free(tmp);

    for(i=0; i<20; i++){
        X.chess[i] = X_chess_memory[top][i];
        Y.chess[i] = Y_chess_memory[top][i];
        if(X.chess[i] != 0){
            Xmax++;
        }
        if(Y.chess[i] != 0){
            Ymax++;
        }
    }
    X_chess_top = Xmax;
    Y_chess_top = Ymax;
    top--;
    return TRUE;
}
```

Figure 10: Double Linked List(Pop)

3 Todo 待改善問題

3.1 王手放置

在基礎將棋規則中，若任一方已經王手（下一手可將死），則一方應優先拯救王將離險境，可使用的方法包括但不限於移動王將、移動我方其他棋子以阻擋路線、打入駒台上的棋子等。也就是，受威脅的那方不得任王將在下一手被將死，除非任何方式都無法拯救王將，此時的情況稱為"詰"，代表遊戲結束。

3.2 Libev 計時器問題

讀取棋盤(回放)時，尚無法顯示原始對局花費的時間，預計在未來將棋譜新增對局開始與結束時間，每一手的攻方，每一手行棋所花費的時間與總累積時間等資料。

3.3 細節顯示問題

因程式中使用system("clear")函式，導致部分訊息顯示瞬間即被清除，此部分仍在尋找解決方法。

3.4 Double linked list問題

目前未知原因，讀取棋譜後輸入"b"，會導致部分資料失，已確認push資料過程中，linked list連結沒問題。

4 Reference 參考資料

[1] 日本將棋走法玩法 https://shogi.hk/Gameplay-of-Japanese-Chess-Shogi/

[2] C語言tips：帶顏色的輸出 https://www.796t.com/article.php?id=190246

[3] Cursor Movement https://tldp.org/HOWTO/Bash-Prompt-HOWTO/x361.html

[4] getopt函式範例 https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html