投稿類別：資訊類

篇名：

與電腦對戰-用極大極小演算法寫 AI!

作者：

張書銘。國立台中高工。資訊三乙

指導老師：

葉水福老師

壹●前言：

（一）研究動機

在玩電腦遊戲時，不論此遊戲是線上多人對戰，或是單機大作，常常會有與電腦對戰的模式，並分成好幾種難度。去年誕生的 AlphaGo，還打贏了世界數一數二頂尖的棋士。對於這些與人 PK 的電腦遊戲的程式是怎麼寫的，邏輯是怎麼思考的，我非常想學習，在 Google 搜尋了一下，發現有好幾種不同的演算法，各有優劣與適合使用的狀況，所以想先從極大極小演算法(Minimax Algorithm)起步。

（二）研究目的

1、瞭解極大極小演算法的概念

2、簡單實作於程式中(C++)

（三）研究方法

對於極大極小演算法：以 Wiki 的資料、Youtube 上國外的教學影片以及各個網站學習此演算法的概念。

對於實作於 C++程式：嘗試將 Wiki 上的虛擬碼付諸於 C++，並做一個簡單的井字遊戲，以此演算法作為 AI 的邏輯。寫程式的過程中遇到的困難，以搜尋書上的語法，以及詢問老師或同學解決。

貳●正文：

（一）極大極小演算法的概念

在每場賽局中，每個玩家都試著選擇對自己最有利的決定，而這個演算法的功用在於尋找損失最少、獲利最多的決定，以下實際以極大極小演算法的概念講解一個進入末期的井字遊戲，如何用極大極小演算法尋找對自己最有利的選擇：

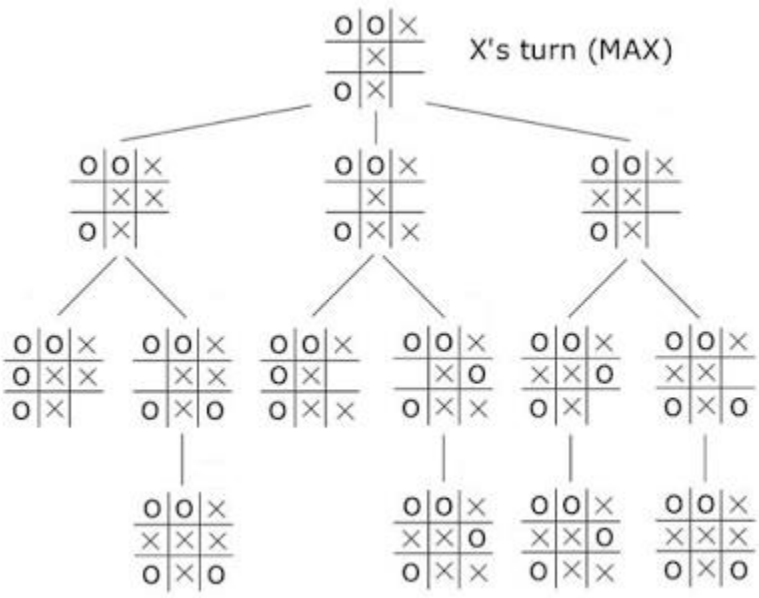

圖 1 井字遊戲末期的遊戲樹

（圖 1）所示的是：依據最頂端的現況，所有可能的結果。假設玩家是使用 X，此演算法的第一個步驟：將玩家勝利的局面在旁邊畫記為+1，敗北的局面畫記為-1，平手的畫記為 0（如圖 2-1）。

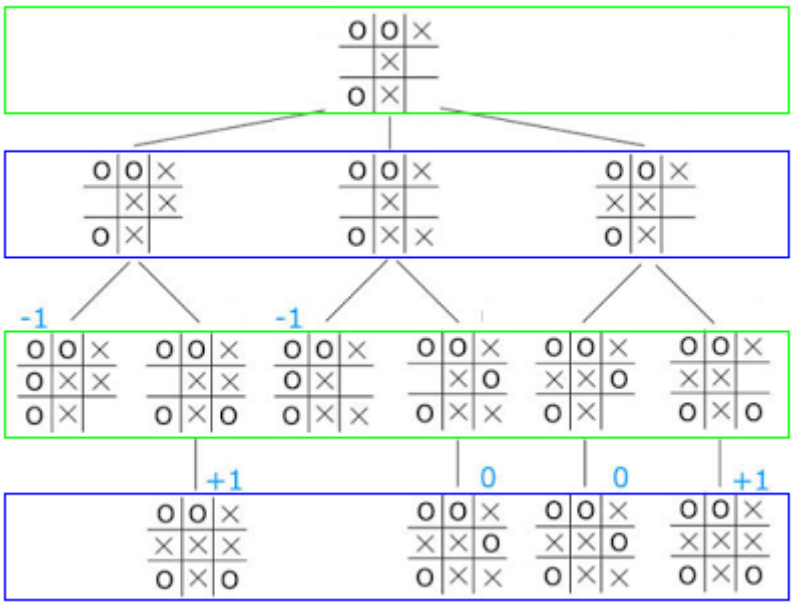

圖 2-1 遊戲樹分析 1

（圖 2-1）中，為了講解方便，將原本的遊戲樹分層，綠色框格表示為 X 的回合，藍色框格表示為 O 的回合。

青藍色數字為第一個步驟標示的分數，可以理解成：+1 是對 X 有利的局面；-1 是對 X 不利的局面。

此演算法的第二個步驟：剩下的未標示分數的局面，從下層開始分析。若是在綠色框格，表示是 X 的回合，則選擇在所有可能性中，對 X 最有利的局面，也就是分數最大的局面；若是在藍色框格，表示是 O 的回合，則選擇在所有可能性中，對 X 最不利的局面，也就是分數最小的局面。並在局面的旁邊畫記選擇的局面的分數（如圖 2-2）。

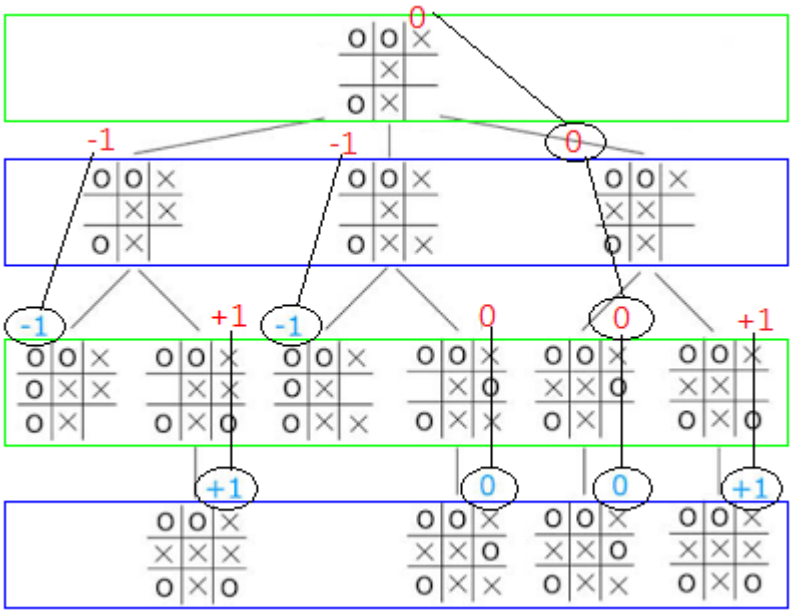

圖 2-2 遊戲樹分析 2

（圖 2-2）中第二層藍色框格中，最左邊的局面，有-1、+1 兩種選擇，因為是 O 的回合，會選擇對 X 最不利的局面，固選擇-1，並依此邏輯將第二層的分數都確定，完成後，第一層，也就是一開始的局面，將會有-1、-1、0 三種選擇，因為是 X 的回合，會選擇對 X 最有利的局面，固選擇 0。而位於第二層，分數為 0 的局面，便是目前這個情況最佳的選擇。

由於這種演算法會將所有結果都模擬出來，決定出每個局面的分數，並依據是哪個玩家，選擇極大或者是極小的值作為選擇，所以總是可以選擇出最佳的選擇，但是也因為會模擬所有結果，若選擇性太多，可能在速度上會令人不滿意，所以又有了一種加快極大極小演算法的方式，即為 Alpha-Beta Pruning（剪枝法），其原理是透過加入 A、B 值作為是否要修剪遊戲樹的參考標準，不過因為本文注重於極大極小演算法的基礎，以及篇幅考量，這裡簡單提起就好。

（二）實作於 C++ Code

1、基本說明：

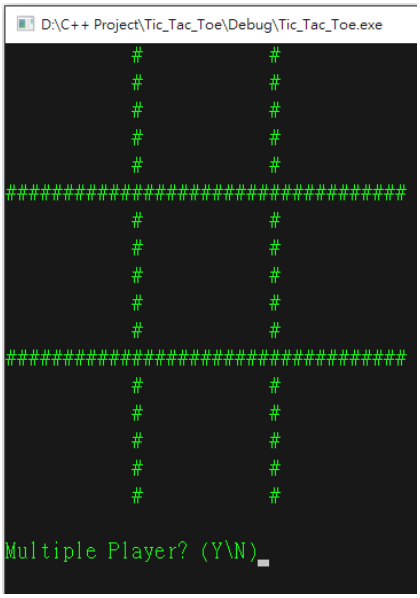

圖 3-1 遊戲畫面 1-詢問遊玩模式

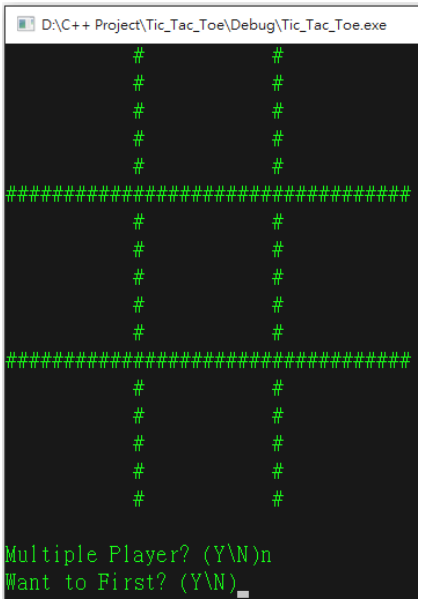

圖 3-2 遊戲畫面 2-詢問優先方

本程式還有支援兩人遊玩，所以遊戲一開始會詢問是否多人遊玩(如圖 3-1)，選擇進入單人遊戲後，還會詢問是否先手（如圖 3-2），進入遊戲後，使用者將要輸入 X 與 Y 座標（如圖 3-3），並試著將自己的標誌連成一條線。

我們將以單人與電腦對戰的模式講解，並且注重於 AI 是如何計算，在一些偵測使用者輸入錯誤、判定勝負，以及設定、輸出棋盤目前配置的程式碼就不贅述了。

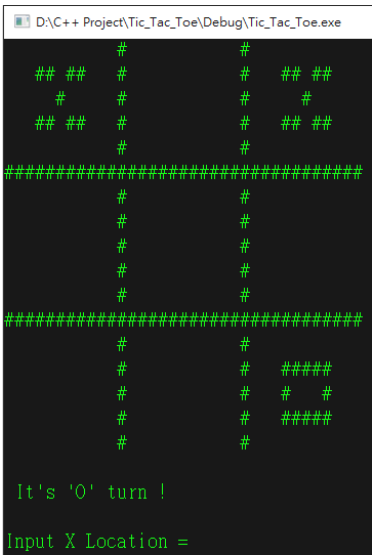

圖 3-3 遊戲畫面 3-與 AI 對戰

圖 3-4 遊戲畫面 4-勝負已定

2、程式流程圖：

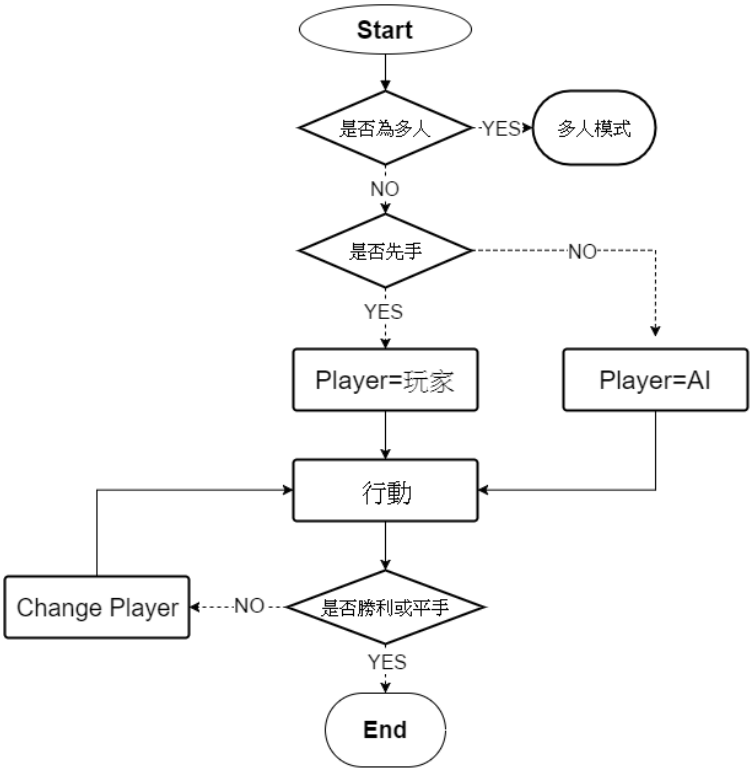

圖 4 程式流程圖

3、程式碼說明：

```
//是否為多人遊戲
if (MultiplePlayer){ ... }
else{
    AI _AI;

    do{ ... } while (IsValid == false);//詢問是否先手

    do {
        if (_player == Player_Mark)
            IsVictory = _user_move(_player, _board);//使用者行動
        else
            IsVictory = _AI.ai_move(_board);//AI行動

        _player = change_player(_player);//改變_player

        count_tie++;//檢查是否平手，若平手則脫離迴圈
        if (count_tie == 9) {
            IsTie = true;
            break;
        }
    } while (!IsVictory);//若遊戲未結束(未分出勝負或是未平手)就繼續迴圈

    _board.print();//輸出目前棋盤配置

    //判定勝負
    if (IsTie){ ... }
    else{ ... }

}
```

圖 5-1 程式碼說明 1-主程式片段

（圖 5-1）顯示的是主程式的片段，圖中出現的 AI Class 緊接著說明。

```
struct AiMove {
    AiMove() {};
    AiMove(int Score) :score(Score) {} //設定分數
    int x;
    int y;
    int score;
};
```

圖 5-2 程式碼說明 2-Struct AiMove

```
class AI {
public:
    //命令AI移動
    bool ai_move(board &_board);

private:
    //尋找最佳選擇
    AiMove ai_GetBestMove(board &_board, int player);

};
```

圖 5-3 程式碼說明 3-Class AI

```
bool AI::ai_move(board &_board) {
    AiMove move = ai_GetBestMove(_board, AI_Mark);//取得最佳選擇
    _board.setMarkPrint(move.x + 1, move.y + 1, AI_Mark);//設定於棋盤上
    _board.print();//輸出棋盤

    return _board.CheckVictory();//回傳是否勝利
}
```

圖 5-4 程式碼說明 4-ai_move

為了之後程式碼的方便，而定義了 Struct Aimove（如圖 5-2）。在自訂類別 AI（如圖 5-3），將 ai_move 設定為 Public，在主程式中即可實作此類別，並呼叫 ai_move 讓 AI 進行動作。

（圖 5-3）中可以看到還有自訂類別 board，但礙於篇幅以及主題性，相關程式碼就不顯示，大致上，此類別專門做與棋盤有關的事，例如：紀錄玩家棋子的配置情形、定義棋盤初始值、輸出棋盤、判斷棋盤是否有某方勝利。

接下來要提到的，就是實作此演算法的程式碼，為整個程式的精華，由於此 Function 過長，以下將整個 Function 再依據功能切分成（圖 5-5）、（圖 5-6）和（圖 5-7）分開說明。

```
AiMove AI::ai_GetBestMove(board &_board, int player) {
    bool IsVictory = _board.CheckVictory();
    bool IsTie = true;

    if (IsVictory) {//檢查上個player是否勝利
        if (player == AI_Mark)
            return AiMove(-10);//如果AI_Mark的上個player勝利，則該局面為-10分
        else if (player == Player_Mark)
            return AiMove(10);//反之則該局面為+10分
    }

    //判斷是否平手 平手則該局面為0分
    for (int n = 0; n < 9; n++){ ... }
    if (IsTie)
        return AiMove(0);
    //若沒有某方勝利、平手，則表示此局面非最後的局面，所以會繼續往下面的程式執行、判斷各局面分數
```

圖 5-5 程式碼說明 5-ai_GetBestMove（上）

（圖 5-5）的程式碼是前面提到的，極大極小演算法的第一個步驟：將所有結束的局面計分。若沒有某方勝利或平手，則會繼續執行到後面的程式。

```
    std::vector<AiMove> moves;//宣告動態增加的陣列以存放各個子選擇局面的分數

    for (int y = 0; y < 3; y++) {
        for (int x = 0; x < 3; x++) {
            if (_board.Mark[y * 3 + x] == No_Mark) {
                AiMove move;
                move.x = x;
                move.y = y;
                _board.Mark[y * 3 + x] = player;
                if (player == AI_Mark)
                    move.score = ai_GetBestMove(_board, Player_Mark).score;
                else
                    move.score = ai_GetBestMove(_board, AI_Mark).score;
                moves.push_back(move);
                _board.Mark[y * 3 + x] = No_Mark;
            }
        }
    }
```

圖 5-6 程式碼說明 6-ai_GetBestMove（中）

（圖 5-6）所示的程式碼的功能，是遞迴地呼叫 ai_GetBestMove，以搜尋所有子選擇局面的分數，並將所有子選擇的分數儲存進 moves。

```
    int bestMove = -1;
    if (player == AI_Mark) {//取得分數最大的子選擇
        int bestScore = -100;
        for (int i = 0; i < moves.size(); i++) {
            if (moves[i].score > bestScore) {
                bestMove = i;
                bestScore = moves[i].score;
            }
        }
    }
    else {//取得分數最小的子選擇
        int bestScore = 100;
        for (int i = 0; i < moves.size(); i++) {
            if (moves[i].score < bestScore) {
                bestMove = i;
                bestScore = moves[i].score;
            }
        }
    }
    //回傳代表該局面的值
    return moves[bestMove];
}
```

圖 5-7 程式碼說明 7-ai_GetBestMove（下）

（圖 5-7）所示之程式碼的功能，就是在 moves 陣列裡搜尋最佳的分數，依據 player 來決定代表該局分數是要挑最大值或是最小值。

綜合（圖 5-6）和（圖 5-7），兩者一起完成的功能就是前面提到的，極大極小演算法的概念中的第二步驟：搜尋剩餘局面的分數。再配上（圖 5-5）所完成的第一步驟，整個演算法概念就完成實作於程式碼中了。

參●結論：

經過一番搜尋與研究，終於將學到的概念寫到程式裡了，這個不會輸的 AI 也讓我獲得蠻大的成就感，過程中遇到蠻多困難的，第一個困難點是對於指標、傳址、Class 還有 Struct 的陌生，透過同學和一些網路的搜尋，我才慢慢熟悉，第二點是遞迴地呼叫函數，想了許久、寫了好幾張計算紙，才把想法釐清。寫完這支程式，我發現要將人的思考方式完全寫到程式中，還真是複雜，不過也就是因為複雜，寫程式才會充滿挑戰與樂趣。寫完這次的小論文，讓我的程式功力得到不小的進步。

肆●引註資料：

1、Minimax Wiki。網址：
https://en.wikipedia.org/wiki/Minimax

2、AL － CH4 極大極小搜尋法與剪枝 Minimax Algorithm and Alpha-beta Pruning。網址：
http://mropengate.blogspot.tw/2015/04/ai-ch4-minimax-alpha-beta-pruning.html

3、C++/Game Tutorial 40: AI for Tic-Tac-Toe with Minimax Algorithm。網址：
https://www.youtube.com/watch?v=CwziaVrM_vc

4、蔡明志（2012）。C++ Primer Plus 5/e 中文精華版。碁峰出版社。