電腦科學邏輯基礎 期末報告 107703001 資科三 鄭家宇

1. 基本的搜尋架構：

搜尋架構是採用 alpha-beta search，我參考了老師提供的井字遊戲的 alpha-beta search 架構，去延伸更改成能應用在五子棋棋盤上，棋盤的部分無法像井字遊戲一樣列舉出所有盤面可能，所以我每下一顆棋就 assert 他，最後遞迴結束回來再把它 retract。推測三步棋之後的盤面可能到最後棋子數量越來越多會超過五秒，保險起見我只下兩步就結算一個盤面，途中如果有一方勝利就直接結束那層展開。

2. 評估盤面的方式：

針對盤面的評估我是先掃過盤面上每顆棋子，看看有沒有勝利或敗北狀況，如果勝利就回傳 700000 分，敗北就回傳-700000 分，若無一方勝利或敗北，則看每顆棋子是否有以下五種狀況並依狀況給分：活四 300 分、死四 20 分、活三 30 分、跳三 8 分、活二 15 分，並且把當回合下棋方的分數加總扣掉另一方的分數加總（例如該回合是黑方下棋，就把黑方每顆子得到的分數扣掉白方每顆子得到的分數，即為此盤面黑方得到的分數）。而每回合的可能落子點選擇我設定為，當前盤面上每個棋子往外八個方向擴張一格的所有點，雖然往外擴張兩格的表現會比較好，但因為時間限制所以最後還是選擇使用前者。

3. 速度優化：

速度優化上就像前面說的，沒有考慮棋盤上所有空著的點，只考慮有落子點向外擴張一格的所有點，藉此來減少不必要的搜尋，table 上因為每次落子都會 assert 所以也沒辦法使用，剪枝則是照著 alpha-beta search 的概念下去實作。

4. 其他心得：

一開始我上網查資料時看到有人用 C++寫的一個盤面評估程式，他是先將盤面上每一個點用一個字元表示（例如黑子是'b'，白子是'w'，空子是'h'），再用一個 15*15 的矩陣表示整個盤面，然後用 KMP 演算法匹配字串來評估盤面並給分（例如活四是"hbbbbh"或"hwwwwh"，若盤面上有符合的子字串則代表某方有一個活四），但我將這個方法實作在 prolog 上之後，發現要額外新建一個 fact 來記錄每個點的字元，還要花很多時間在字串的轉換上（因為每下一個點就要更新整個盤面 15*15 的字串），導致程式隨便一跑就超過五秒，所以最後還是回來使用遞迴的方式，判斷每顆落子會造成幾顆連續，並且中斷連續的點是空點還是對方落子點等等情況，加上使用 cut 結束一些不必再繼續的遞迴來加速程式執行，雖然一開始用了不適合的方法多花了點時間，但我覺得整個過程還是很有趣的，也因為這樣我學到了更多 prolog 內建的 function，雖然之後不一定還會回來碰這門語言，但最後把這個東西做出來，成就感很高！