tic-tac-toe by alpha-beta or min-max algorithm

113321531游志信

程式說明:
主要就是參考老師上課講解的alpha beta演算法進行修改，評估的函數為O獲勝的分支+1分，X獲勝的分支-1分，平局則0分，最後透過alpha beta進行剪枝優化搜索效率，alpha為最大化玩家的最優分數，beta為最小化玩家的最優分數，如果beta <= alpha就可以剪枝，後續的搜索無法改變結果沒有意義，然後在做的途中有發現到一個問題，就是電腦對下的時候，不管下幾局所有的盤面都會一致，所以就增加了隨機初始步的函數增加對局的多樣性

函數介紹:
initialize_board(): 創建一個 3x3 的二維列表，代表井字棋的初始棋盤
print_board(board): 輸出當前棋盤狀態，以數字列顯示行和列，便於玩家識別位置
check_winner(board): 檢查是否有玩家獲勝或平局
minimax(board, depth, is_maximizing, alpha, beta): 使用 minimax 演算法進行遞迴搜尋，並使用 alpha beta剪枝優化
computer_move(board, player): 根據 minimax 函數的評分，決定電腦的最佳移動
human_move(board, row, col): 執行玩家的移動
play_human_vs_computer(): 人機對下的邏輯
random_first_move(board, player): 隨機初始步
simulate_self_play(games): 電腦對下的邏輯

執行結果:

```
$ python alpha-beta-homework.py
Choose mode: (1) Human vs Computer, (2) Self Play: 1
Initial Board:
  1 2 3
1 _ _ _
2 _ _ _
3 _ _ _

Enter your move (row col): 2 2
After your move:
  1 2 3
1 _ _ _
2 _ X _
3 _ _ _

Computer's move:
  1 2 3
1 O _ _
2 _ X _
3 _ _ _

Enter your move (row col): 1 3
After your move:
  1 2 3
1 O _ X
2 _ X _
3 _ _ _

Computer's move:
  1 2 3
1 O _ X
2 _ X _
3 O _ _

Enter your move (row col): 2 1
After your move:
  1 2 3
1 O _ X
2 X X _
3 O _ _

Computer's move:
  1 2 3
1 O _ X
2 X X O
3 O _ _

Enter your move (row col): 1 2
After your move:
  1 2 3
1 O X X
2 X X O
3 O _ _
```

```
Computer's move:
  1 2 3
1 O X X
2 X X O
3 O O _

Enter your move (row col): 3 3
After your move:
  1 2 3
1 O X X
2 X X O
3 O O X

It's a draw!
```

```
$ python alpha-beta-homework.py
Choose mode: (1) Human vs Computer, (2) Self Play: 2
Enter the number of games: 5
Results: Wins: 0, Losses: 0, Draws: 5

Game 1 Result: Draw
  1 2 3
1 O X X
2 X X O
3 O O X

Game 2 Result: Draw
  1 2 3
1 X O X
2 X O O
3 O X X

Game 3 Result: Draw
  1 2 3
1 O X O
2 X O X
3 X O X

Game 4 Result: Draw
  1 2 3
1 X O X
2 O O X
3 X X O

Game 5 Result: Draw
  1 2 3
1 X O X
2 O O X
3 X X O
```