TIC-TAC-TOE

INSTRUCCIONES

Descargue el código tictactoe.zip

Instale el paquete pygame

Hay dos archivos principales para este proyecto: runner.py y tictactoe.py.

El segundo contiene toda la lógica para jugar y hacer las movidas óptimas. Runner.py ya está implementado y contiene todo el código para correr la interfaz gráfica del juego. Una vez que usted complete todas las funciones requeridas en tictactoe.py, se puede correr en Python runner.py para jugar.

tictactoe.py define tres variables: x, o y EMPTY, para representar los posibles movimientos.

La función initial_state regresa el estado inicial. Para este problema se escoge representar el tablero como una lista de tres listas que representan las 3 filas del tablero, donde cada lista interna contiene tres valores que son x, o o EMPTY. Las siguientes funciones deben ser implementadas:

player, actions, result, winner, termina, utility, y minimax:

- The player function should take a board state as input, and return which player's turn it is (either X or O).
  - In the initial game state, X gets the first move. Subsequently, the player alternates with each additional move.
  - Any return value is acceptable if a terminal board is provided as input (i.e., the game is already over).
- The actions function should return a set of all of the possible actions that can be taken on a given board.
  - Each action should be represented as a tuple (i, j) where i corresponds to the row of the move (0, 1, or 2) and j corresponds to which cell in the row corresponds to the move (also 0, 1, or 2).
  - Possible moves are any cells on the board that do not already have an X or an O in them.
  - Any return value is acceptable if a terminal board is provided as input.
- The result function takes a board and an action as input, and should return a new board state, without modifying the original board.
  - If action is not a valid action for the board, your program should raise an exception.
  - The returned board state should be the board that would result from taking the original input board, and letting the player whose turn it is make their move at the cell indicated by the input action.
  - Importantly, the original board should be left unmodified: since Minimax will ultimately require considering many different board states during its computation. This means that simply updating a cell in board itself is not a correct implementation of the result function. You'll likely want to make a deep copy of the board first before making any changes.
- The winner function should accept a board as input, and return the winner of the board if there is one.

    - If the X player has won the game, your function should return X. If the O player has won the game, your function should return O.
    - One can win the game with three of their moves in a row horizontally, vertically, or diagonally.
    - You may assume that there will be at most one winner (that is, no board will ever have both players with three-in-a-row, since that would be an invalid board state).
    - If there is no winner of the game (either because the game is in progress, or because it ended in a tie), the function should return None.
- The terminal function should accept a board as input, and return a boolean value indicating whether the game is over.
    - If the game is over, either because someone has won the game or because all cells have been filled without anyone winning, the function should return True.
    - Otherwise, the function should return False if the game is still in progress.
- The utility function should accept a terminal board as input and output the utility of the board.
    - If X has won the game, the utility is 1. If O has won the game, the utility is -1. If the game has ended in a tie, the utility is 0.
    - You may assume utility will only be called on a board if terminal(board) is True.
- The minimax function should take a board as input, and return the optimal move for the player to move on that board.
    - The move returned should be the optimal action (i, j) that is one of the allowable actions on the board. If multiple moves are equally optimal, any of those moves is acceptable.
    - If the board is a terminal board, the minimax function should return None.

Para todas las funciones que aceptan board como entrada, se debe asumir que es un tablero válido (contiene las tres filas, cada una con tres posibles valores x, o, EMPTY. No se deben modificar las declaraciones de funciones con el orden o número de argumentos.

Una vez que todas las funciones estén implementadas correctamente, se debe poder correr runner.py y ya se podría jugar. El empate es lo mejor que se puede obtener cuando se juega contra el programa de inteligencia artificial.