Assignment 8

002311001041

Writeup

Question:

Consider the classical Tic-Tac-Toe problem and its solution. Implement the tic-tac-toe game for human vs computer using minimax algorithm/ any constraint satisfactory algorithm.

Overview:

For this assignment, I created a Tic-Tac-Toe game where the computer plays against a human using alpha-beta pruning. I chose alpha-beta pruning because it's a smarter version of the minimax algorithm—it helps the computer make good decisions faster by skipping over moves that won't change the outcome. In my code, the ab() function does the heavy lifting: it looks ahead at all possible moves, but if it finds a move that's clearly worse than one already considered, it stops checking those branches. This makes the AI both efficient and challenging to play against. The game also lets you pick your symbol (X or O) and choose the difficulty level, so you can play as easy, medium, or hard depending on how tough you want the computer to be.

Key Features Added:

- Implemented alpha-beta pruning for the computer's decision-making.
- Added difficulty levels: easy (random moves), medium (blocks player wins), and hard (uses alpha-beta pruning).
- Allowed the user to choose whether to play first or second.

Main Functions Used:

- *print_b()*: Displays the current state of the board.
- *win()*: Checks if a player has won.
- *full()*: Checks if the board is full.
- *score()*: Evaluates the board for the AI.
- *ab():* Performs alpha-beta pruning to select the best move.
- *easy_move(), medium_move(), hard_move()*: Select moves based on the chosen difficulty.
- *main()*: Handles the game loop, user input, and overall flow.

Important Variables:

- b: Represents the game board as a 2D list.
- user, ai: Symbols for the human and computer players.
- diff: Stores the selected difficulty level.
- turn: Tracks whose turn it is.

Process and Compliance:

- All intermediate steps and results of the game are displayed during execution and saved to output.txt, as required by the assignment.
- The entire output is interactive and prompts the user for the player symbol and difficulty level; no inputs are hardcoded.

Submitted files: 041_Assignment_8_Sushar_Hembram.ipynb (notebook), output.txt (game log), and this write-up.

Additional Details:

The game logs each move and the final result to a text file for record-keeping.

Conclusion:

This implementation demonstrates the use of search algorithms in game AI and provides a user-friendly interface with adjustable difficulty.