COSC 4550/5550 - Introduction to Artificial Intelligence
Fall 2023
University of Wyoming
Homework #4

Due: Sunday, November 05, 2023, 11:59 p.m.

Instructions: Answer all of the following questions. This homework is a programming assignment. For this, you will complete ***code_hw4.py*** and also prepare a PDF document named ***written.pdf*** and include your analysis figures and discussion in the written file. *Please show the details of your work.* Feel free to use a calculator (or better yet, the Python interpreter!) to compute the final numeric answers. Show at least 4 decimal places in your answers. Be sure to use floating point division in the Python interpreter (one way to make sure is write things like 1./2 instead of 1/2).

You will submit the completed **written.pdf** and **code_hw4.py** files. Your submission should only have the above mentioned two files.

***Programming* (Takes Time!)**

1. For this question, you will be designing and implementing a program that plays Tic-Tac-Tical (a variant of tic-tac-toe). It will exemplify the minimax algorithm, and alpha-beta pruning, and the use of heuristic (evaluation/static) functions to prune the adversarial search. Usage of any other existing AI techniques (e.g. machine learning, search strategies) and/or techniques that you develop for this problem *along with* the Minimax algorithm will earn extra credit and potentially improve the performance.

   Tic-Tac-Tical is a two-player game (one of them being your computer program) which is played on a 5x4 grid. Figure 2 shows the board configuration at the start of the game. Players take turns placing their pieces on the board until one of the players win. The player who first manages to place three of their symbols/pieces together in consecutive positions, either horizontally, vertically, or diagonally on the board is the winner.

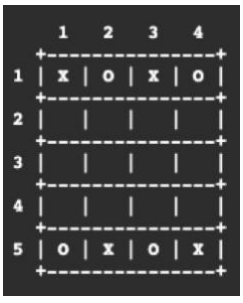

Figure 2: Initial Board Configuration

Symbols: Player 1 uses ‘x’ and Player 2 uses ‘o’ pieces and will place on the 5x4 board.

Game Rule: A player may move one of their tokens one space horizontally or vertically, forward or backward (but NOT diagonally) on a single turn.

You will complete the implementation of the given **code_hw4.py** program that plays Tic-Tac-Tical game. A few important points about the program representation are given below:

- The squares of the board are numbered by row and column, with '1 1' in the upper left corner, '1 2' directly to the right of '1 1', etc.

- The board is represented by a 2-dimensional matrix named ***Board***. The visible portion of the board is ***Board[1…5][1…4]***. In order to simplify calculations, the Board has been declared with dimensions ***Board[0..6][0..5]***, so that there are extra rows and columns surrounding the visible portion. The surrounding cells all contain the value ***OutOfBounds***. The visible cells all contain the values **x**, **o**, or **Empty**. These values are named global constants with numerical (not alphabetic) values.

- Moves are of the form **'i j m n'**, where **(i, j)** is a square occupied by the piece to be moved, and **(m, n)** is the square to which you move it. Routines are provided to initialize the Board, print the board, determine all possible moves for ***Player*** (whose value is either x or o), to prompt a human player to input a (legal) move and check it for legality, and to apply a legal move to the current ***Board***.

- You need to complete the function $\boldsymbol{Win}(\boldsymbol{Player}, \boldsymbol{Board})$, which determines whether there are three consecutive tokens with value **Player** anywhere on **Board**.

- You need to complete the function $\boldsymbol{GetComputerMove}(\boldsymbol{Player}, \boldsymbol{Board})$, which determines what move to make when it is the computer's turn to move, and the computer's token is ***Player***. Currently, that function simply determines all possible moves for Player, and chooses one at random. To do this, you should implement $\boldsymbol{Minimax}$**, alpha-beta pruning,** and use a heuristic to determine the Player's move.

- You need to write a heuristic function named $\boldsymbol{userid\_h}(\boldsymbol{Player}, \boldsymbol{Board})$, where ***userid*** is your UWyo userId. This heuristic should return a high value for board states that are favorable for ***Player***, and a low value for board states that are unfavorable.

- You need to collect data on number of states evaluated with and without pruning, amount of time used, and depth achieved. Summarize your results in a table in **written.pdf** file.

Your program should adhere to the following conventions:

- You can write new supporting functions given the new function names do not raise a name conflict with existing functions.

- Your program should be able to make either the first or second move.

- Your program should be able to be either the player with X or O tokens.

- Note that by allowing each player to be either human or a computer, this permits your program to do testing with both players as human, or with both players as computer, which should be helpful for you while developing and evaluating your heuristic.