EECS 3401 – Project1 [20 Points]

Due by 28th November 11:59pm EST

In the second project, you will design agents for the zero-sum game called **Four in a Row**. This is a two-player board game, in which the players choose a color and then take turns dropping colored discs into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. Four in a Row is a solved game. The first player can always win by playing the right moves. See here for more details.

For this project, you will implement the following 3 algorithms:

- **Minimax** - Here you will implement depth-limited minimax which searches to an arbitrary depth as it's not feasible to search the entire game tree.
- **Alpha-Beta Pruning** - This will speed-up the search as the depth of the tree increases, as it allows for more efficient exploration of the minimax tree
- **Expectimax –** This models probabilistic behavior of opponents that may make suboptimal decisions. For this you will have chance nodes that will replace the MIN nodes (AI-Agent2 or Human Player). For this implementation, consider you will only be running against an adversary which chooses actions uniformly at random, because Minimax and Alpha-Beta algorithms assume that MAX plays against an adversary who makes optimal decisions.

Pseudocodes for the above-mentioned algorithms have been provided in the lecture slides. Note that they only return the best utility value, whereas you need to select the action that is associated with this value. To implement this, please consider for the MAX player, all valid actions at the root of the tree, and then you return the action that leads to the best value.

There are some incomplete function definitions provided that need to be completed for the complete implementation of the game.

A GUI has been provided to simulate the game and test the performance of these algorithms.

The starter code that has been provided to you, includes two main files:

- **four_in_a_row.py** - This file includes all functions of the game. Also, an evaluation function has been provided to score the leaves of your minimax tree, in order to treat them as terminal nodes.
    - You have a choice to implement your own evaluate function, if you are able to think of a better one. A better evaluation function would mean it has lower computation time and/or higher winning percentage for a fixed depth compared to what has been provided. So in case you're up for the challenge, please explain your strategy with regards to choosing the evaluation function in a readme.txt file.
    - For an AI-Agent to make its next move, it needs to run an adversarial search as a MAX player. In the case of Minimax and Alpha-Beta, the opponent is considered a MIN player. Whereas in the case Expectimax the opponent is considered a CHANCE player. For a Random-Player making a move, the move is a random valid move.

    - At the maximum tree depth utilities of the nodes should always be evaluated from MAX's point of view, because during this process the adversary never considers its own score at all. Also, the adversary tries to minimize the score of the MAX player that initiated the game search is an assumption that minimax algorithm works under.

- **game_gui.py** - This file provides the GUI for the game and includes all types of players that can play the game. This file helps to display the game environment and select different game modes including AI-Agent1 vs AI-Agent2, and AI-Agent vs Human-Player, AI-Agent vs Random-Player, etc.

Types of players that can participate in the game:

- **AI-Agent** – This will be the agent that you implement
- **Random Player** – This kind of player chooses valid columns with equal probability
- **Human Player** – This will be a human playing the game.

To run the game, use the following command: python four_in_a_row.py

Make sure your code is well commented.

Submission Instructions: Your submission should be a zipped file, having the name Firstname_Lastname_studentid_Project2.zip including your modified four_in_a_row.py file to Project 2 on eclass.