EECS 348 Winter 2018 Lab 2: Minimax & AlphaBeta

Assignment

For this assignment you will implement Minimax and AlphaBeta players to play Konane (also known as Hawaiian Checkers). Konane is played on an NxN board of light and dark pieces. The board shown below is 8 × 8, and uses **x** for dark and **o** for light:

```
    0 1 2 3 4 5 6 7
0   x o x o x o x o
1   o x o x o x o x
2   x o x o x o x o
3   o x o x o x o x
4   x o x o x o x o
5   o x o x o x o x
6   x o x o x o x o
7   o x o x o x o x
```

First, the dark player removes a dark piece (an **x**) at position (0, 0), (7, 7), (3, 3), or (4, 4). (If N is odd, then only the center position or the two corners can be chosen.) Next, the light player removes a light piece adjacent to the space created by the first move. Then the players alternate moves, each jumping one of their own pieces over one horizontally or vertically adjacent opponent's piece, landing in a space on the other side, and removing the jumped piece. The game ends when one player can no longer move, and that player is considered the loser. For more information about Konane (particularly the rules of play), see:

https://en.wikipedia.org/wiki/Konane#Rules_and_gameplay

or

http://www.konanebrothers.com/How-to-Play.html

For each game, you will be given a partially played Konane board of arbitrary size not to exceed 16x16, a maximum number search plies (*e.g.*, maximum depth) and a maximum number of moves. You will also arbitrarily be assigned as player 'x' or player 'o'. At the completion of each turn, your player is expected to return a Python list coordinates for the move in the format of:

move = [[x_from, y_from], [x_to, y_to]]

or

a Python '**None'** if there are no moves

Your player will play against an autograder player which uses a variety of deterministic algorithms. You will be graded based upon the move sequences generated by your player matching a "gold" sequence for each game.

Modified Rules

To simplify implementation, you can assume that each starter board will already have the initial moves

for ‘x’ and ‘o’ already made, and that your player has the next move on the board.

Implementation Details

You have been provided with starter code for this assignment. This code contains `konane.py` which handles housekeeping tasks for the Konane board, `main.py` which runs the tests to evaluate your player, `human.py` which contains code for you to play against the computer (`python human.py` will start up a game) and `student.py` which contains the interfaces for the student player to interact with the game.

Each of the players in the student code implement simple deterministic algorithms that can be used to familiarize yourself with the game and starter code. There is also an algorithm for a random player which may help you to test your code, but will not be used for grading.

You will need to complete the `getMinimaxMove()` and `getAlphabetaMove()` member functions of the `player` class. You may (and are encouraged) to add as many additional helper member functions as needed, but you may not change any code in `konane.py`, `main.py` or the existing interfaces in `student.py` (feel free to modify the code in `human.py` for your own understanding). A heuristic (utility function) is defined for you in the `player` class in `student.py` (the heuristic is defined as a function helpfully called `heuristic`). For instances where there are ties for the utility function, your player should select the first move from the best moves in order specified by `possibleNextMoves()`.

Your code will be expected to run in less than 30 seconds for each game board provided. In addition to passing the tests provided in the started code, you are encouraged to construct additional test cases.

A few helpful things to think about:

- recursion may be useful while exploring the search tree, and
- making deep copies of boards during search may simplify logistics of maintaining board integrity

Portions of this assignment and starter code are based upon materials from Jim Marshall and Deepak Kumar.