Project 1: Connect Four

Due 02/27/17

Introduction

In this assignment, you'll write a computer program to play the classic game Connect Four. This is a great way to practice the minimax and heuristic algorithms we've covered in class.

How to Play

Connect Four is a tic-tac-toe variant played on a grid. Players alternate turns dropping coins into one of the seven different columns. Unlike tic-tac-toe, Connect Four worlds are affected by gravity and you may only place coins at the lowest possible positions in each column. In other words, moves in Connect Four are made by dropping the coins into the columns, rather than placing them into specific squares.

As the name implies, the goal of Connect Four is to get four of your colored coins in a row, either horizontally, diagonally, or vertically. The first player to do so wins. If all locations are filled without a player getting four in a row, the game is a draw.

Connect Four is known to be **biased in favor of the first player**, so when testing your AI make sure that you have it play as both the first and second player. A decent AI program will never lose as the first player, and a good AI program will be able to win as the second.

Instructions

As with PathFinder, we've provided you with a good amount of starter code to handle most of the complicated setup. The provided starter code is a working Connect Four engine that allows you to mix and match human and computer opponents using the command line. Your assignment is to create an instance of the AIModule class that plays Connect Four. To do so, you'll need to familiarize yourself with the workings of the GameState and AIModule classes.

Because even a simple minimax player can play perfectly given unlimited time, in this assignment part of your task will be to create a player that can work in limited time conditions. During game play, your player will have to select a move within a given time frame. **Make sure you understand what the chosenMove field of the** AIModule **class is for before writing your player. Take a look at** MonteCarloAI **for an example of how to write a working AI.**

The Provided Framework

The starter code we've provided will work out of the box and should require no changes on your part.

The Connect Four program has several different command-line switches that you can use to control how the game is played. By default, the two players are human-controlled. You can choose which AI modules to use by using the -p1 and -p2 switches to select the AIModules to use as the first and second player. For example, to pit the RandomAI player against the MonteCarloAI player, you could use:

java Main -p1 RandomAI -p2 MonteCarloAI

Any unspecified players will be filled in with human players.

You can also customize how much time is available to the AI players. By default, each computer has 500ms to think and this is what we'll use in the competition. However, you can use the -t switch to change this. Use the --help switch to learn more about the options available to you.

Questions

In all of these questions you will be playing connect-4 on a 7x6 board. Please form groups of two, choose a name and clearly in the assignment state who is part of your group (include student ids). **You should not divulge your evaluation function to any of your**

opponents.

1. In this question you are to clearly (but in no more than one page), describe your evaluation function.

Your answer must cover:
a) A detailed motivation on why you believe your evaluation function is reasonable (one paragraph)
b) Your evaluation function as a numerical expression with definitions of all variables.
c) One worked example showing a board state and your evaluation function score.

(20 points)

2. Implement the mini-max algorithm and your evaluation function following the interface defined above. Call your java file that is to be submitted minimax_<team-name>.java.

(15 points)

3. Play your algorithm five times against the given players: StupidAI.java and RandomAI.java. Your algorithm should always beat these two players. Play your algorithm ten times against MonteCarloAI.java, your algorithm should beat this player the majority of times. Use the -text option and the –seed options (seeds 1 thru 10) and submit **just the final board** for each of the twenty games. **Clearly report how many of the twenty games you won/lost/drawn.**

(20 points – 1 point for a win, 0.5 points for a draw)

4. In the next question you shall implement alpha-beta pruning. In this question you are to describe your successor function and how you shall order moves to ensure that the best case situation will occur. In this question you are expected convince the grader that your scheme will work. Unconvincing answers will receive little or no points.

(25 points)

5. In this question the TAs will play your algorithm against all other groups twice. In the first game your algorithm will be p1 and in the second it will be p2.

Implement the alpha-beta pruning algorithm **and your evaluation function** and submit your code as alphabeta_<team-name>.java.

You will score 0.5 points for a win, 0.25 points for a draw for a maximum of 35/35. So you'll score full points if you beat about half of the class.

The winner amongst all players will get 20 bonus points, 15 bonus points for second, 10 bonus points for third and 5 bonus points for fourth.

(55 points, maximum for tournament results).

1.5.7.1