Assignment 2: Game Playing
Due Oct. 9 at 11:59pm

This assignment can be done in teams of two or individually.

Important Note: The univeristy policy on academic dishonesty (cheating) will be taken very seriously in this course. Students can work in pairs on this assignment. You may not discuss the specific questions in this assignment, nor their solutions with any other group. You may not provide or use any solution, in whole or in part, to or by another group.

You are encouraged to discuss the general concepts involved in the questions in the context of completely different problems. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

Part 0: Starting Out

Download the Java backgammon code from the course website. Compile the code, and try running the driver program Bg.class to play a game by yourself.

Part 1: Implement Expectiminimax Search

Create a new class **ExpectiminimaxBackgammonAgent.java**, a subclass of **BackgammonAgent.java**, that performs minimax search. You will need to search a game tree containing nodes corresponding to game states with branches corresponding to possible moves and possible rolls of the dice. Use the evaluation function **NaiveEvaluation.java** to evaluate board configurations in the game tree that are at the cutoff depth.

Feel free to subclass NaiveEvaluation.java if it makes your testing easier.

Your code should support n-ply search, for any $n \in \mathbb{N}$.

Part 2: Develop a Better Evaluation Function

Create a new class, **BackgammonEvaluationXX_YY.java**, an extension of **Evaluation.java**, where XX and YY are your group members' student numbers. Improve upon the naive evaluation function distributed with the code. The following website contains more information on backgammon for those who are not familiar with the game: `http://www.bkgm.com`

For example, a simple evaluation function could count the total number of spaces that a player's checkers must move in order clear them from the board. A more advanced evaluation function could count the number of "blots" (isolated checkers) left by a player, and give a negative score to them.

The XX_YY will be used to identify your backgammon playing agent in a tournament involving the agents developed by the class. The best backgammon agent will earn a prize for its creators. The tournament will be run using our own ExpectiminimaxBackgammon-

Agent.java, so even if there is a problem with your implementation in Part 1 you will be able to participate in the tournament. The tournament will be run using 2-ply look-ahead. Agents using evaluation functions that do not return values within reasonable amounts of time, crash, or try anything shady will be disqualified.

Testing Notes

Testing these search algorithms can be time consuming. Please use these hints to make testing simpler:

- Modify `DIE_MAX` in `Dice.java` to be a small number.
- Turn off doubles by setting `use_doubles` to false in `BackgammonBoard.java`.
- Set up the board with only a few pieces in a particular configuration.

For testing Expectiminimax, it is sufficient to show a few small example cases. Print out a visualization of your search tree, showing your algorithm choosing the correct move.

Submitting Your Assignment

You must create your assignment in electronic form. For each part of the assignment, submit the source code. In addition, submit a **brief** report with the following components:

- Testing output with descriptions of the cases being tested for Part 1
- Summary of the semantics of the evaluation function, and explanation of choices for features and weights

This report should be in one of these formats:

- ASCII Text
- HTML
- PDF (Portable Document Format)

You must include a completed cover sheet along with all the files from your completed assignment. This file is available from the course web page: `http://www.cs.sfu.ca/~mori/courses/cmpt310/cover_sheet_a2.txt` This file will be used by the TA to give you feedback on your assignment, so don't forget to complete it.

All files should be labeled appropriately so that the markers can easily follow and find your work. Create an archive (either .zip or .tar.gz) containing all relevant assignment files, plus the cover sheet. When you are done, go to the Assignment Submission Page at `https://submit.cs.sfu.ca/` and follow the instructions there.

You must also submit a **compiled** copy of your **BackgammonEvaluationXX_YY.class** on `https://submit.cs.sfu.ca/` under the Assignment2_part2 "assignment." If your evaluation class requires any auxilliary files, include them in this submission. Their filenames must include XX_YY, as all of these files will be dumped in one directory for running the tournament.

Submission Checklist

In summary, submit the following items:

- All source code
- Brief report
- Cover sheet
- BackgammonEvaluationXX_YY.class (under Assignment2_part2)

Grading Scheme

- Expectiminimax Implementation (25 marks)
- Evaluation Function (15 marks)
- Coding style (10 marks)
- Testing (10 marks)