Sanjeev Janarthanan
Andrew Sang
sanjrj_sang76_dbg_agent.py
sanjrj_sang76_sbg_agent.py
Assignment 3 for CSE 415, Winter 2020, University of Washington

**Deterministic Simplified Backgammon Agent**

For this agent I (Sanjeev) created an agent that could play a deterministic version of backgammon. This first thing I needed to do was create a method that could create a set of possible moves. I had to look at every edge case involving the bar and bearing off and account for all these cases. I had a simple static evaluation function. The closer red or white's checkers were to their goal, the more the value was increased or decreased respectively. I additionally adjusted the value based on if one color had checkers beared off (a good thing for that color) or on the bar (a bad thing for that color). Next, I implemented minimax search using the pseudocode from the slides to associate with each move. I had a standard implementation of alpha-beta pruning. Once I had a score associated to each move, I found the best possible moves and returned them in move().

**Stochastic Simplified Backgammon Agent**

This agent implements the expectiminimax algorithm, a recursive adversarial search, in order to find the best move for a player in a game of backgammon where the roll of the dice isn't predetermined. It operates by implementing a minimax search, except on every other layer, there is consideration given to chance, essentially a "move by nature" to account for the random chance of the dice. I (Andrew) wrote the expectiminimax function from pseudo-code from wikipedia, and the rest of the code was taken from my partner's pre-written code.

**Partnership Retrospective**

Sanjeev Janarthanan

I think the main issue with this project is that it didn't lend itself well to partner work. Everything except for the search functions were the same in each agent, so I unintentionally ended up with a lot more work than my partner. The biggest lesson I learned is to properly assess the workload of each part of the assignment and split it up appropriately.

Andrew Sang

One thing that was unexpected for us in this assignment was how front-loaded the work in the work would be. We split the assignment down the logical middle line between the SBG and DBG agents, where Sanjeev would do the DBG and I would implement SBG afterwards on a similar codebase. We just didn't expect how much work was necessary to implement the preliminary methods (possibleMoves, etc) to even get started on the algorithms. Because of that, Sanjeev did more work than I did because he had to implement the auxiliary methods before even getting started on the minimax search, and I just extended his code. In particular, the possibleMoves function was very difficult to account for all of backgammon's cases.