Homework 3

Due: 18 February 2020

For both of the problems this week, if you don't know the general rules of the "standard" game referred to, make sure to look it up and/or talk to other students about it!

Problem 3.1

In this problem, consider the following variant of Othello (aka Reversi):

Players play with standard pieces, on a standard board (but where the columns are numbers 1-8). Before each turn, the player rolls an 8-sided die. If the player has any valid moves in that numbered column, they must take one of those moves (but may choose among them). If the player has no valid moves in that column, but has valid moves elsewhere on the board, they may choose any of their valid moves.

Assume a standard state model representation (whose turn it is, plus a 2D grid of $\{-1, 0, 1\}$) and a standard action model representation (coordinates of move), neither of which needs to change for this variant. At the moment where someone chooses their action, the die has already been rolled for that turn (so that constraint is known), but of course for *future* moves the rolls of the dice could be anything.

Use this to explain and illustrate, as if to someone who has learned about minimax but never tried to apply it to a game with random elements (e.g. someone who has completed CMSC262), how the Expectiminimax algorithm works. **Include diagrams** of at least one state space lookahead graph: the start state should *not* be the initial state, but rather a state plausibly 2–5 moves into the game; and the graph should include at least some lookahead that goes at least 2–3 moves deep (to fully explain the algorithm), although you do not need to draw the *entire* state space to that depth (which would already be getting prohibitively large).

Problem 3.2

In this problem, consider the following variant of (American) Checkers:

Each player has their own board, with their own pieces on it (initially in the standard Checkers starting layout) and the other side empty. Each player cannot see the other player's board; there is a neutral judge/arbiter that can see both. Because the starting position is fixed, *initially* both players know exactly where the other player's pieces are.

To move, a player puts their finger on one piece (so the arbiter can see which piece is selected) and announces which square they are attempting to move to (again by standard Checkers rules). The arbiter announces one of three results to this:

- Piece successfully moves to the location (if the square is empty)
- Piece jumps opponent at location (if the square is occupied and the next square in that diagonal direction is empty)
- Piece fails to move to location (if the square is occupied and the next square is also occupied)

In the first two cases, the player updates their own board according to the successful move (and in the jump case, the opponent also removes the piece that was formerly at that square). If the move carried the player into the final row, the piece is converted into a king as in standard Checkers. In the third case, the failed move is treated as a pass and the next player gets to take their turn.

Because a move to a location can come from either of two directions (or four, later on when kings are involved), each player's mental model of the state of the game is not complete after the game has begun. Devise **two different plans** for implementing an AI for this game: primarily and especially, how exactly is the game state modeled—actually laying out the data with instance variables and types/descriptions—but also give brief notes on how this state would get updated as new info about attempted/actual moves would come in, and how that would be used to adapt minimax to play the game.

Remember, two different plans. Comment on which one seems like it will be more effective, and why.