Daniel Chai (1 team member)

CSE 415

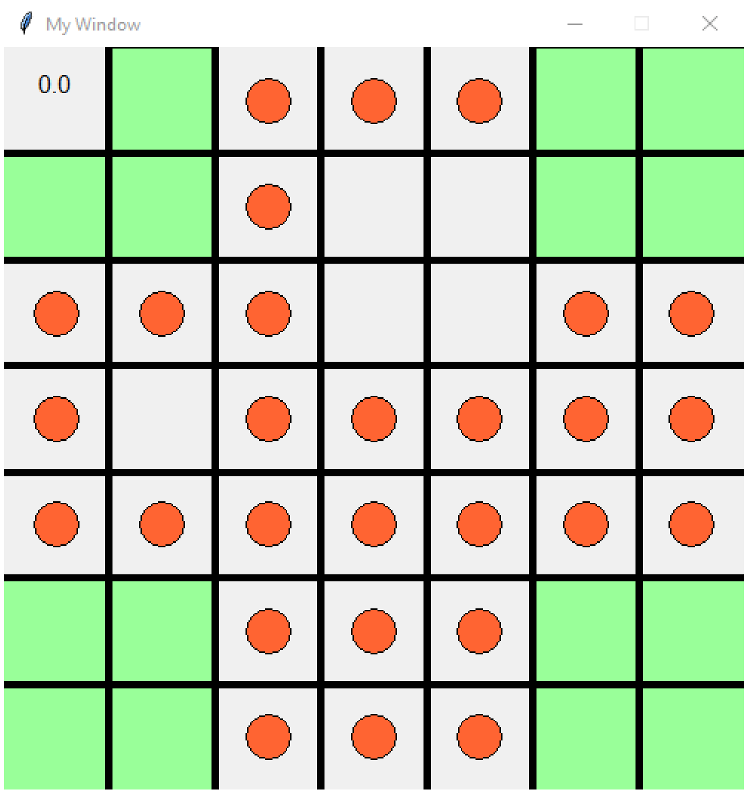
June 2nd, 2017

Title: 33-Peg Solitaire

3. The program is supposed to find the solution to the puzzle through Q-Learning.

4. I primarily used Q-Learning in trying to find the solution to the puzzle. I first formulated the puzzle by making a state, which contains the current board. Additionally, I added all the possible moves as Operators to look at all the possible outcome in a board. The goal state was a board with just one peg. The failure state was when there were more than one pegs but no more moves make. The Q-Learning part then took the initial state learned from subsequent states. Then, with the Q value dictionary that was created from Q-Learning, I made a visual representation of what the program has calculated so far. Unfortunately, my program did not have any optimal policy implemented to find the solution. Also, there was a problem of my Q-Learning never reaching the goal state. This was not allowing the Q-Learning to properly make correct decisions. During my peer evaluation with a group that did 2x2x2 rubik's cube, they mentioned that this could be happening because of the number of combinations that was possible for each state.

5. One of the states, with Q-Value in the top right corner



6. In order to use the program, go the visual.py and change the values in the learn.QLearning (discount, niterations, epsilon). Then, run the visual.py and it will take some time to load the image depending on the number of iterations you entered (good number is ~100).

7. I liked the QValues_string() function because this draws the visualization. This is the first time I've tried making graphics on python, so it was fun and new. The code first initializes the canvas and I calculated the size of each square for the 7x7 board. Then I draw rectangles in the illegal states of the board because I didn't want to draw the rectangle every single time the pegs were drawn. Then the grid line were drawn. Next, I sorted the Q dictionary in order of highest to lowest Q value, and then iterated through every single square to draw the orange peg in each squares.

8. I learned more of what I was capable of doing. I could explore further into AI to work on my personal projects.

9. If I had more time I would implement:

    a. optimal policy – to only display the correct moves

    b. Working reversely – I would work from goal state and work reversely in order to figure out which states can lead to the goal state. This would allow me to initialize the Q dictionary with more values that lead to the goal state, lowering the chance of not finding any solutions to the puzzle.

    c. Better animation - to actually go through the game, from beginning to the end.