Project 1 for CS 414, Fall 2010
Due 11/15, 6am

This project will focus on using code from the RL 2009 competition. The assignment will be written for the Mario domain, although Tetris is also part of the same package and should not be difficult to use. You may use Tetris instead of Mario, but check with Matt to make sure that your interpretation of this project for the new domain is correct.

The product of this project will be

1. a writeup (most important)
2. all the code in one zip file (not very important)
3. an in-person demonstration of the code (a little important)
4. an in-person discussion of your writeup (a little important)

Step 0:
Estimate for Matt: 5 minutes
Install the RL competition code and run Mario. You should be able to see a visualizer and run with a demo agent. Change the agent so that it **only** runs to the right. This is not hard, but will force you to install everything and get started early ☺

Step 1:
Estimate for Matt (only human time): 2 hours
Test how well the ExMarioAgent.java agent plays. Run two experiments, using the following parameters:

Level Seed = 121
Level Type = 0 and 1
Level Difficulty = 0
Instance = 0

Run thirty trials for level type 0 and thirty trials for level type 1. Report the average and standard deviation for this simple agent on both level types.
Hint: You will want to set up a script to run this test and report the final reward for each trial. If you try to use runDemo.bash, this project will take *much* longer.

Step 2: (Monday, October 25, 6am)
Estimate for Matt: 0.5 hour
Given the information used by getAction() in ExMarioAgent, what do you think would make a good set of state variables to describe Mario's state?

Example: Suppose I defined Mario's state variables as follows:
1: Is there a pit to my right? (as done in ExMarioAgent)
2: Is there a smashable block above me? (using the getTileAt() function)

Then, Mario could potentially learn to jump if he's under a smashable block or if there's a pit next to him. However, he would ignore all monsters and would only notice pits at the last moment. Thus a better representation of state might involve putting some information about the nearest monster into the state, and/or the distance to any pit to Mario's right.

Step 3:
Estimate for Matt: 0.25 hours
Using the state representation you designed in step 2 (after taking into account any feedback from Matt about the state representation), decide on a tabular representation of the action-value function. Matt recommends somewhere between 100 and 10,000 states. What are the (dis)advantages of having a small or a large state space?

Step 4:
Estimate for Matt: 1 hour
Figure out how you can debug your (to be developed) learning algorithm without relying on Mario. For instance, you may want to design your own very simple test environment so that you can give the agent a state and a return and see if the learning update is working correctly.

Step 5: (Monday, November 1, 6am)
Estimate for Matt: .5 hour to program, 1.5 hours to test, debug, and tune
Using the tabular approximation of the action value function, program a Sarsa. You will need to modify the start, step, end, and getAction functions.
Hint: Tune multiple learning rates (alpha) and exploration rates (epsilon).

Step 6:
Estimate for Matt (only human time): 30 min
Test your learning algorithm on Mario. Let Mario play for many episodes – the reward you receive should increase. Run 10 trials using your algorithm on level type 0, using the same parameters as in Step 1. Plot the average reward vs. Episode Number, along with the standard deviation.

Step 6:
Estimate for Matt: 1 hour
Implement Q-Learning or Monty Carlo. Tune the learning parameters and compare with Sarsa. Graph and explain your results.

Step 7:
Estimate for Matt: 2 hours (mostly for testing, debugging, and tuning)
Using the same state features, change from a tabular representation to using function approximation. Matt suggests a neural network or a CMAC, and will suggest how to set it up for your state representation. Tune the parameters of the function approximator and compare to learning with the same algorithm in the tabular representation. Graph and explain your results.

Step 8:
Estimate for Matt: 2 hours
Update one or more of your algorithms to use eligibility traces. Tune the value of lambda that you use, and then compare the learning results to learning with lambda=0. Graph and explain your results.

Step 9:
Estimate for Matt: 1 hour
Using one of your learning methods, allow Mario to learn in level 0 and save your action-value function. Now, compare learning in level 1 between A) learning as normal and B) beginning with the old action-value function. This is an example of transfer learning – if the two levels are similar, what Mario learned on level 0 should help.

Step 10:
Estimate for Matt: 4 hours
In this step, you are *teaching* the computer to play Mario. First, develop a keyboard interface for the Mario game. Second, write to a text file that records all the states the agent sees, and what action you took (if any). Third, use you ID3 algorithm from project 0 to learn to classify all of your data into a policy (i.e., given a state, what action would you most likely take). Fourth, use this learned policy to play Mario. How well does it do? Does the amount of demonstration that you give the agent affect its performance?

Pass: You generate a learning curve in step 6.
Maximum benefits from the following conditions:
+1/2 letter grade: The learning curve has a positive slope (i.e., it does *learn*)
+1/2 letter grade: Mario is able to learn to do better than the example policy
+1/2 letter grade: overall thoroughness, presentation quality, insight, etc.
+1/2 letter grade: in person demo + discussion with Matt goes well
+1 letter grade: Step 6, Step 9
+2 letter grades: Step 7, Step 8
+3 letter grades: Step 10

-1/2 letter grade: October 25th or November 1st checkpoint missed

Example:
Suppose you miss the checkpoints at both the 25th and 1st. You're able to get Sarsa to learn better than the demo policy (+.5 +.5), write a so-so report (+.25), have a really good demo of the project (i.e., you're able to explain what you did and answer a few questions about your design decisions = +.5), and do an OK good job on step 7(+1), and an OK job on step 9 (+0.5). You would receive: -.5 -.5 + pass + .5 +.5 +.25 +.5 +2 + .5 = pass + 3.25 grade letters = 60 + 32.5 = A-

I hope that you'll do more than the minimum number of steps, as my hope is that this will be fun, and to make sure that you get a good grade. The maximum grade I will give for this project is a 100%, but there are other, less tangible reasons, for showing off (e.g., "geek street cred "and good rec letters).