COMPSCI 270	**Instructor:** Kris Hauser
Homework 6	**Due:** 04/24/2017

1 Instructions

Read this assignment carefully, complete the programming assignment and answer all of the written questions. Place all of your code in the `value_iteration_agents.py` and `q_learning_agents.py` files as indicated and your written answers in the `hw6_answers.py` file in the appropriate locations. Alternatively, you may submit your answers to the written questions in a PDF file called `hw6_answers.pdf`. Submit all of these files separately (i.e., not zipped into a single file) on Sakai before 4:20pm on 04/24/2017.

1.1 Homework Policies

Homework and lab assignments **must be completed individually**. Students are permitted and even encouraged to discuss assignments. However, any attempt to duplicate work that is not your own – for example, in the form of detailed written notes, copied code, or seeking answers from online sources – is strictly prohibited and will be considered cheating.

Homework assignments are due by the end of class on the due date unless otherwise specified. **No extensions will be granted** except for extenuating circumstances. Extensions are granted at the instructor's discretion, and valid extenuating circumstances include, for example, a debilitating illness (with STINF), death in the family, or travel for varsity athletics. Extensions will not be granted for personal or conference travel, job interviews, or a heavy course load.

2 Overview

In this assignment, you will be implementing algorithms for use with Markov Decision Processes (MDPs) and in reinforcement learning. These algorithms will be applied in a version of Sutton and Barto's "cliff walking" domain.

3 Preparation

For this assignment, you will again need to have Project Malmo installed and working. Refer to HW0 as a reference for doing this, if needed.

4 Problems

4.1 Programming

For this section, you will be implementing value iteration and Q-learning agents.

4.1.1 Problem 1

Implement a value iteration agent in the `ValueIterationAgent` class, part of which has been written for you in `value_iteration_agents.py`. You will need to complete the class methods that are incomplete (those where it is indicated that you code should be placed). Note that this includes the constructor (`__init__`), where you should run value iteration based on the given MDP for the given number of iterations.

You can test your code by running the `hw6_0.py` file with `python hw6_0.py`. You should make changes to the `__main__` section to perform further testing and to complete the written questions.

4.1.2 Problem 2

Implement a Q-learning agent in the `QLearningAgent` class. The skeleton for this class has been provided for you in `q_learning_agents.py`. Be sure to complete all of the methods which indicate that your code is required.

You can test your code by running the `hw6_1.py` file with `python hw6_1.py`. You should make changes to the `__main__` section to perform further testing and to complete the written questions.

4.2 Written

4.2.1 Problem 3

Consider the MDP from Problem 1. Starting with a value function that is uniformly zero, perform two iterations of value iteration and include the results as a table which represents the game grid (14 rows x 5 columns).

4.2.2 Problem 4

Reproduce the final value function from Problem 1 as a table (again, representing the game grid). Compare this table to the table from Problem 3. What do you notice about how the value function changed by continuing to iterate?

4.2.3 Problem 5

Run your value iteration agent again, but use a discount factor of 0.7 and include the final value function as a table. Compare this table to the table from Problem 4 - what do you notice about how the value function has changed?

4.2.4 Problem 6

Include a table for the value function learned by your Q-learning agent.

4.2.5 Problem 7

The `epsilon` value that is passed to your Q-learning agent governs the trade off between exploration and exploitation. Initially, this is set to 0.5 in `hw6_1.py`. Test your agent by running it with `epsilon` values of 0.25 and 0.75. Alter the number of episodes which are used for training (`numTraining`) and record how many training episodes are required for the agent to learn an optimal policy (one that moves along the shortest distance to the goal) at each of the three values of `epsilon`. What does this tell you about this trade off?