CS 4700, Foundations of Artificial Intelligence
Spring 2020
Solutions to Quiz 9

1. There were two possible questions:
   a. True/False: Consider a single-state MDP with two actions, A and B. Action A gives you a reward of 2 every time you do it. Action B gives you a reward of 1 every time you do it. (The optimal policy is obviously to always do action A.) Imagine that you ran Q learning on this problem with Ne (the minimum number of times you need to attempt each action in a state) set to 0. Then Q Learning is guaranteed to learn the optimal policy.
   False. With Ne=0 it means there's no exploration and at each step Q Learning just takes whichever action has the highest Q value. The initial Q values are both 0, so for the first step it is random which action is selected. If it is action B it gets a reward of 1. The Q value for that action from the given state is updated to be 1. We're now back in the same state, and pick whichever action has the higher Q value, which is B. And these keeps happening, without ever trying action A. The random selection of B as the initial moved locks you into always doing action B without ever doing action A, the optimal action to take.

   b. Imagine that instead of starting off Q learning with each Q(s,a) estimate being set to 0 you instead initialized all Q values to the values that the algorithm is supposed to converge to. At the end of each iteration of Q Learning the algorithm will always pick the optimal action to take. (If you are concerned about situations where ties occur assume the algorithm always breaks ties in the same fashion - on other words, your answer shouldn't be about some funky situation that might arise involving tie breaking.)
   False. Even though the algorithm has the correct estimates, as long as the number of times an action has been selected is less than Ne it will ignore the Q value and pick an action that hasn't been tried enough times.

2. There were two possible questions:
   a. Imagine that Policy Iteration is initialized with a policy that is actually the optimal policy. Policy Iteration will never change that policy as it runs.
   True. As discussed in class and in the textbook, Policy Iteration converges (and, it's guaranteed to get to the optimal policy when it does so). That means when it reaches a policy that doesn't change it's the correct optimal policy, even if the policy happened to be the one you started with.

   b. Once Policy Iteration reaches the optimal policy if you continue to run its update loop it will no longer change either the policy or the U values it assigns to each state.
   False. As discussed in class Policy Iteration can get to the optimal policy even before the U estimates are correct. If you keep running Policy Iteration the U estimates get refined to be closer and closer to the correct value even though the policy stays as it was.

3. There were three possible questions, only differing terms of the probabilities (shown in red:

   Imagine you have a two state MDP with states A and B. They both have the same single action available: STAY. If you do the STAY action in either state with probability {0.9, 0.8, 0.7} you will stay in your same state and receive reward 10, and with probability {0.1, 0.2, 0.3} you will switch to the other state and receive reward 0. (There is obviously only one policy for this MDP since in each state there are no options about what action to take (and it is obviously the optimal policy).) What is the expected cumulative discounted reward of state A and state B if γ=0.5? Use 2 significant digits after the decimal (drop any zeroes at the end, so that 9.496 become 9.5 and 9.995 becomes just 10).

The two states for this problem end up being identical – you have the same actions and same rewards associated with each action. With probability {0.9, 0.8, 0.7} you get reward of 10, and with probability {0.1, 0.2, 0.3} you get reward of 0. Let's called the probability of getting reward 10 p, meaning you get zero with probability (1-p). That means that the cumulative discounted reward is

$$\sum_{t=0}^{\infty} \gamma^t [p \times 10 + (1-p) \times 0] = 10p \sum_{t=0}^{\infty} \gamma^t = \frac{10p}{(1-\gamma)}$$

Since $\gamma = 0.5$ this simplifies to 20p. So if p=0.9 your answer should be 18 for both states, if p=0.8 it should be 16, and if p=0.7 it should be 14.

(A different way to solve this arises if you didn't notice that the two states were the same. Let's use a to refer to the value of state A and b for the value of state B. Then you get the following equation for a:

$$a = [p \times (10 + \gamma b) + (1-p) \times (0 + \gamma b)] = 10p + p\gamma b + (1-p)\gamma b = 10p + \gamma b.$$

The identical calculation takes place for b, swapping the a's and b's, yielding

$$b = 10p + \gamma a.$$

We now have two equations with two unknowns. If you plug in the formula for b into the formula for a you get:

$$a = 10p + \gamma(10p + \gamma a) = (1+\gamma)10p + \gamma^2 a$$

Getting the a's on one side and simplifying gives:

$$(1-\gamma^2)a = (1+\gamma)10p$$

$$a = \frac{(1+\gamma)10p}{(1-\gamma^2)}$$

Since $\gamma = 0.5$ this simplifies to $a = \frac{1.5 \times 10p}{0.75} = 20p$. You can plug this in to the formula for b to similarly get $b = 20p$. If $p = 0.9$ this gives $a = b = 18$, if it is 0.8 you get $a = b = 16$, and if it is 0.7 you get $a = b = 14$.)