CS 4700 - Foundations of Artificial Intelligence	Homework 3
Fall 2019	Due Tuesday, November 12 (2:40pm)

1. **MDP's and Utility functions (20 Points)**

You are a freshman at Cornell during O-Week, and you are still adjusting to the complicated campus. What you want to do is get from North Campus (0, 0) to CollegeTown (2, 4) as efficiently as possible. Being a future Computer Science student, you decide that it would be a good idea to formulate this as a Markov Decision Process problem.

As you probably know, there are a number of routes to Collegetown, but for the purpose of this problem we'll simplify the expansive map of Cornell to a 3x5 grid:

	0	**1**	**2**
0	North Campus	MVR	Empty
1	AAP	Vet School	Empty
2	West Campus	Arts Quad	Cornell Store
3	104 West	Empty	Ho Plaza
4	Law School	Snee Hall	CollegeTown

Formally, we define the states as a set:

$S = \{$*North Campus, MVR, AAP, Vet School, West Campus, Arts Quad, Cornell Store, 104 West, Ho Plaza, Law School, Snee Hall, Collegetown*$\}$

Where each state is represented as a vector of its coordinates. For example, if you were in the start state "North Campus", it would be represented as [0 0], and the end state "Collegetown', it would be represented as [2 4].

The actions a are drawn from the set $A = \{$*Up, Down, Left, Right*$\}$, where they only apply if they are valid. For example, going Up or $Left$ from "North Campus" is not valid. Nor is going into an "Empty" space.

The transition function is given as $P(\vec{s'} \mid a, \vec{s}) = 0.7$ if s' is the state you intended to go to by using action a in state s, and 0.1 in each other valid direction (if a direction is not valid, then the probability 0.1 is for returning to the same state s for that action).

The reward function is given as

$$R(\vec{s}, a, \vec{s'}) = -\left\|\vec{s_g} - \vec{s'}\right\|_2$$

Where $\vec{s_g}$ is the vector which represents the goal state: in this case, CollegeTown at [2 4].

Note: The $\|\cdot\|_2$ notation indicates the **L2-norm** or **Euclidean norm** of the vector $\vec{s_g} - \vec{s'}$. Think about $\left\|\vec{s_g} - \vec{s'}\right\|_2$ as the distance between the goal state $\vec{s_g}$ and your current state $\vec{s'}$. For example, suppose that the goal state has coordinates (x_g, y_g), and your current state has coordinates (x_c, y_c). Then,

$$\left\|\vec{s_g} - \vec{s'}\right\|_2 = \sqrt{(x_g - x_c)^2 + (y_g - y_c)^2}$$

This is just the classic distance formula. Don't forget to **take its negative** to obtain the reward.

(a) Represent this MDP as a directed graph, complete with actions and probabilities on the edges, and rewards in the states / nodes. (8)

(b) You are given a policy $\pi_{DR}(\vec{s})$ = For the given s, if it is possible to use the action *Down*, then do so, else use the action *Right*. Write the utility $U^{\pi}(s)$ equations of your policy π for the states $S = \{$*North Campus, Vet School, Ho Plaza, 104 West*$\}$. Let $\gamma = 0.5$. Note, you do *not* need to explicitly calculate the utility of the policy at each given state, but rather simply state the equations. (12)

2. **Discounting Rewards (14 Points)**

At first, it seems strange that when we calculate the overall reward of a sequence of states/actions, we **discount** rewards in future time-steps by a factor of γ^t. In this problem we'll review why we do this.

Suppose we want to calculate the utility of a **specific** infinite-length sequence of states and actions $[s_0, a_0, s_1, a_1, s_2, \dots]$. Define $U_h([s_0, a_0, s_1, a_1, s_2, \dots])$ to be the utility of that **fixed** sequence of states and actions.

(a) Briefly describe two reasons why it is undesirable to simply sum up all the rewards we receive, as follows:

$$U_h([s_0, a_0, s_1, a_1, s_2, \dots]) = \sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})$$

You should give one sentence per reason. **Hint:** If you're unsure, check out section 16.1.1 of the textbook (available on the course website) (2).

(b) Assume that every reward $R(s_t, a_t, s_{t+1})$ lies between the range $[R_{min}, R_{max}]$, where R_{min} and R_{max} are finite constants. Also suppose that $0 < \gamma < 1$. Show that the **sum of discounted rewards**

$$U_h([s_0, a_0, s_1, a_1, s_2, \dots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$$

is always finite, and provide a lower and upper bound for it (7).

(c) Suppose that receiving a reward of $+1$ at time-step $t = 5$ is twice as good as receiving a reward of $+1$ at time-step $t = 10$. What should my γ be? Round to 3 decimal places (3).

(d) Now, suppose I change my mind and I now think that getting the reward of $+1$ at $t = 5$ is **more** than twice as good as receiving the reward of $+1$ at $t = 10$. Should I increase or decrease γ? (A short English sentence is enough.) (2)

3. **Solve MDP using policy iteration (16 Points)**

A robot is tasked with collecting empty soda cans in an office building, and placing them in a bin. It runs on a rechargeable battery.

As a simple example, assume that the robot can only distinguish between two charge levels; the set of states is $S = \{high, low\}$.

In each state, the agent can decide whether to

- actively search for a can for a certain period of time
- remain stationary and wait for someone to bring it a can
- head back to its home base to recharge its battery.

We write the action set as $A = \{search, wait, recharge\}$.

Suppose this system can be represented as an MDP with the following transition probabilities and rewards.

s	a	s'	$P(s'\|s,a)$	$R(s,a,s')$	Inuition (optional)
high	search	high	0.7	+4	If the robot actively searches for cans, the expected number of cans it collects will be higher (+4). However, searching runs down the robot's battery. There's a 70% chance that the battery will stay high, but a 30% chance that it will go to low.
		low	0.3	+4	
high	wait	high	1.0	+1	If the robot stays still, the expected number of cans it collects is lower (+1). However, the battery is guaranteed to stay high.
		low	0.0	+1	
high	recharge	high	1.0	0	If we charge, we don't collect any cans (so reward is 0). We always end up with high battery after charging.
		low	0.0	0	
low	search	high	0.4	−10	With probability 0.4, if the robot has low battery and searches, its battery will completely die. Then a human will need to find the robot and charge it, which is costly (reward −10).
		low	0.6	+4	Otherwise with probability 0.6, the robot's battery stays alive, and we collect cans from searching.
low	wait	high	0.0	+1	
		low	1.0	+1	Waiting will not affect the battery level, so it will stay low. The expected numbers of cans collected from waiting is +1.
low	recharge	high	1.0	0	If we charge, we don't collect any cans (so reward is 0). We always end up with high battery after charging.
		low	0.0	0	

(a) How many unique policies are there in this example (3)?

(b) In general, if there are s states and a actions possible from each state, how many unique policies are there? (After finding your answer, it should be clear why it's often impossible to iterate over every single possible policy.) (3)

(c) Manually run policy iteration for this MDP. Set the initial policy to be the policy of always searching (this is a bad policy). Initialize the utility of both states to be 0. Let $\gamma = 0.5$. For each iteration, calculate the updated utilities and updated policy for each state, and show your work. (10)

You only need to show 2 iterations. By then, the policy should have converged. Box your final policy(10).

4. **Q-learning (15 Points)**

Recall that a Q-function $Q(s, a)$ describes the expected reward for taking action a in state s. In particular, utility and an ideal Q function are related via...

$$U(s) = \max_a Q(s, a)$$

If one takes an action a at state s and reaches a new state s', then the update rule is as follows...

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

γ : discount factor such that $0 \leq \gamma \leq 1$
α : learning rate such that $0 < \alpha \leq 1$
$P(s'|s, a)$: transition probability for state s' after taking action a in state s
s : state
a : action
s' : next state

For the correct Q-values, this should converge to an assignment such that the following holds...

$$Q(s, a) = \sum_{s'} P(s'|s, a) \cdot (R(s, a, s') + \gamma \max_{a'} Q(s', a'))$$

(a) Comment briefly on how the convergence relationship connects to expected utility, and why this is a good description of what we want from a Q function. Hint: A good answer should include the words "expected" and "utility" (4).

(b) Why does α need to be between 0 and 1? What happens if $\alpha = 1$? (4)

(c) What does the max do in these equations? (2)

(d) What is the Q function for the North Campus state from Question 1? (5)