CS 188 Spring 2017 — Introduction to Artificial Intelligence — Midterm V2

- You have approximately 80 minutes.
- The exam is closed book, closed calculator, and closed notes except your one-page crib sheet.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.
- For multiple choice questions with *circular bubbles*, you should only mark ONE option; for those with *checkboxes*, you should mark ALL that apply (which can range from zero to all options)

First name	
Last name	
edX username	

For staff use only:

Q1.	Potpourri	/20
Q2.	Search	/16
Q3.	CSPs	/18
Q4.	War in Paclandia	/12
Q5.	Approximate Q Learning with Landmark States	/14
Q6.	MDPs: Rebellious Robot	/20
	Total	/100

Q1. [20 pts] Potpourri

(a) MDPs

In this MDP, available actions are left, right, and stay. Stay always results in the agent staying in its current square. Left and right are successful in moving in the intended direction half of the time. The other half of the time, the agent stays in its current square. An agent cannot try to move left at the leftmost square, and cannot try to move right on the rightmost square. Staying still on a square gives a reward equivalent to the number on that square and all other transitions give zero reward (meaning any transitions in which the agent moves to a different square give zero reward).

4	0	0	36

(i) [2 pts] $V_0(s)$ is 0 for all s. Perform one step of value iteration with $\gamma = \frac{1}{2}$ and write $V_1(s)$ in each corresponding square:

4	0	0	36

(ii) [3 pts] Perform another step of value iteration with $\gamma = \frac{1}{2}$, and write $V_2(s)$ in each corresponding square:

6	1	9	54

(iii) [4 pts] Using $\gamma = \frac{1}{2}$ again, write $V^*(s)$ in each corresponding square (Hint: think about states where the optimal action is obvious and work from there):

8	8	24	72

(b) Search

(i) [2 pts] Select a subset of the following options that comprises a minimal state space representation for pacman path-finding (i.e. find a path from some position on the board to another).

- [✓] Pacman's position
- ☐ The positions of the food pellets
- ☐ A boolean per food pellet indicating whether that pellet has been eaten
- ☐ A boolean indicating whether Pacman has reached the target position
- ☐ The Manhattan distance from Pacman to the target
- ☐ The positions of the walls
- ☐ None of the above; a minimal state representation is ____________________________

(ii) [9 pts] For each statement, if (and only if) that statement applies to a particular search algorithm, write a check in the cell corresponding to that statement-algorithm pair. Assume A* has an admissible heuristic, and that all positive edge weights are lower-bounded by some $\epsilon > 0$.

Statements:

1. This algorithm can get stuck in an infinite loop in a finite graph with positive edge weights when no solution exists
2. This algorithm can get stuck in an infinite loop in a finite graph with positive edge weights when a solution exists
3. This algorithm can get stuck in an infinite loop in an infinite graph with positive edge weights when a solution exists
4. This algorithm is guaranteed to find an optimal solution when all edge weights are 1 and a solution exists
5. This algorithm is guaranteed to find an optimal solution when all edge weights are positive and a solution exists

Statement	DFS Tree	BFS Tree	UCS Tree	A* Tree	DFS Graph	BFS Graph	UCS Graph
1	t	t	t	t	f	f	f
2	t	f	f	f	f	f	f
3	t	f	f	f	t	f	f
4	f	t	t	t	f	t	t
5	f	f	t	t	f	f	t

Q2. [16 pts] Search

Wall-E is trying to find Eve on a M x N spaceship, but along the way he would like to collect all K > 0 stationary friendly bots. He will collect a friendly bot when he lands in a spot as the bot. Each time he collects a friendly bot, he will be able to move an extra space per timestep. For example if Wall-E collects 1 friend, each timestep from then on he can move **up to** 2 squares and if Wall-E collects 2 friends, each timestep from then on he can move up to 3 squares. This bonus is applied in the timestep after Wall-E collects a friend, and lasts for the entire rest of the search. **Eve is also moving** around one square per turn and cannot collect friendly bots. Both can only move North, South, East, and West. The search ends when Wall-E and Eve are on the same square, and Wall-E has collected every friendly bot.

(a) [3 pts] What is the size of the minimum state space for this problem?

MxN x MxN x 2^k

(b) [3 pts] What is contained in the minimal state space representation?

Walle position, Eve position, Robot Taken or not

(c) Now the evil authorities have been alerted and have begun scanning the ships by quadrants. At each time step the authorities will scan a quadrant in clockwise order starting from the top right quadrant. If Wall-E is caught, i.e. he is in the quadrant that is currently being scanned, then it will be game over.

(i) [2 pts] Now what is the size of the minimum state space?

MxN x MxN x 2^k x 4

(ii) [2 pts] What is added to the state space representation?

The quadrant been scanned

(d) [6 pts] Assume that the scanning is still in place. Check all of the following that are admissible heuristics (Note that any distances from Wall-E to friendly robots or from friendly robots to Eve are 0 if there are no remaining friendly robots):

- ☐ Manhattan distance from Wall-E to Eve divided by two.
- ☐ Sum of Manhattan distances to all robots and to Eve.
- ● Manhattan distance from Wall-E to Eve divided by (K + 2).
- ☐ 1 for every state.
- ● (Manhattan distance from Wall-E to furthest friendly robot + Manhattan distance from that furthest robot to Eve) divided by (K + 2).
- ☐ (Manhattan distance from Wall-E to closest friendly robot + Manhattan distance from that closest robot to Eve) divided by K.
- ☐ (Sum of Manhattan distances from Wall-E to each friendly robot + Manhattan distance from Wall-E to Eve) divided by (3K).

Q3. [18 pts] CSPs

In this question, you are trying to find a four-digit number satisfying the following conditions:

1. the number is odd,
2. the number only contains the digits 1, 2, 3, 4, and 5,
3. each digit (except the leftmost) is strictly larger than the digit to its left.

(a) CSPs

We will model this as a CSP where the variables are the four digits of our number, and the domains are the five digits we can choose from. The last variable only has 1, 3, and 5 in its domain since the number must be odd. The constraints are defined to reflect the third condition above. Thus before we start executing any algorithms, the domains are

1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 3 5

(i) [3 pts] Before assigning anything, enforce arc consistency. Write the values remaining in the domain of each variable after arc consistency is enforced.

(ii) [1 pt] With the domains you wrote in the previous part, which variable will the MRV (Minimum Remaining Value) heuristic choose to assign a value to first? If there is a tie, choose the leftmost variable.

○ The first digit (leftmost)
○ The second digit
○ The third digit
○ The fourth digit (rightmost)

(iii) [1 pt] Now suppose we assign to the leftmost digit first. Assuming we will continue filtering by enforcing arc consistency, which value will LCV (Least Constraining Value) choose to assign to the leftmost digit? **Break ties from large (5) to small (1).**

○ 1
○ 2
○ 3
○ 4
○ 5

(iv) [3 pts] Now suppose we are running min-conflicts to try to solve this CSP. If we start with the number 1332, what will our number be after one interation of min-conflicts? Break variable selection ties from left to right, and **break value selection ties from small (1) to large (5)**.

____

(b) The following questions are completely unrelated to the above parts. Assume for these following questions, there are only binary constraints unless otherwise specified.

(i) [2 pts] [*true* or *false*] When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

(ii) [2 pts] [*true* or *false*] Once arc consistency is enforced as a pre-processing step, forward checking can be used during backtracking search to maintain arc consistency for all variables.

(iii) [2 pts] In a general CSP with n variables, each taking d possible values, what is the worst case time complexity of enforcing arc consistency using the AC-3 method discussed in class?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ○ $O(d^n)$ ○ ∞

(iv) [2 pts] In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists?

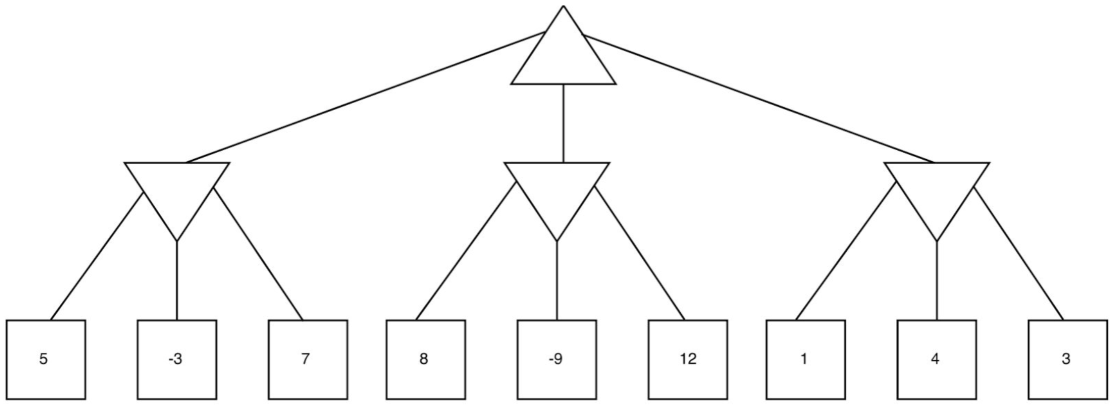
○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ○ $O(d^n)$ ○ ∞

(v) [2 pts] What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ○ $O(d^n)$ ○ ∞

Q4. [12 pts] War in Paclandia

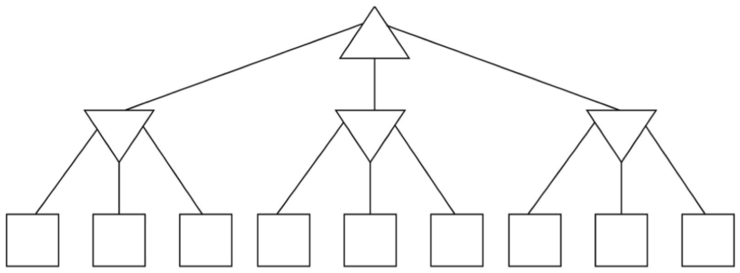
In the land of Paclandia, there exist three tribes of Pacmen - the Ok, the Tok, and the Talok. For several centuries, the Ok and the Tok have been rivals, waging war against one another for control of farms on the border between their lands. In the latest set of skirmishes, the Ok decide to launch an attack, the outcome of which can be quantified by solving the following game tree where the Ok are the maximizers (the normal triangles) and the Tok are the minimizers (the upside down triangles). (assuming the Tok are a very advanced civilization of Pacmen and will react optimally):



(a) [4 pts] What's the best possible outcome (a utility value) that the Ok can achieve in this situation? Cross out any nodes that can be pruned (observing nodes from left to right).

(b) [3 pts] The Talok have been observing the fights between the Ok and the Tok, and finally decide to get involved. Members of the Talok have unique powers of suggestion, and can coerce members of the Ok into misinterpreting the terminal utilities of the outcomes of their skirmishes with the Tok. If the Talok decide to trick the Ok into thinking that any terminal utility x is now valued as $y = x^2 + 2x + 6$, will this affect the actions taken by the Ok?

Now consider a game tree identical to the one above, but with blank utility values.



(c) [5 pts] Give a bound (as an interval) for the range of values in which these terminal utilities must lie in order to guarantee that transforming them with $y = x^2 + 2x + 2$ does not change the selected action.

Q5. [14 pts] Approximate Q Learning with Landmark States

In Q-Learning for RL problems with a continuous state space it is impractical to use a tabular representation of Q-values, and in class we showed how we can use a feature-based representation to approximate Q-values as a linear combination of a state-action pair's features. In this problem we'll explore another approach which approximates the Q-values of a state by using a weighted sum of the Q-values of landmark states.

Recall that in feature-based Q-Learning there is a function $f(s,a) \in \mathbb{R}^m$ that featurizes a state-action pair, and that a Q value is approximated by a linear combination of these features using weights that are learned during training:

$$Q(s,a) = \sum_{i=1}^{m} w_i f_i(s,a) \tag{1}$$

In contrast, approximate Q-Learning with landmark states doesn't use a feature-based representation. Instead, the premise of this approach is as follows:

1. We randomly define a set of n landmark states $s_1, s_2, ..., s_n$ spread across the state space, and we will store approximated Q-values for these landmark states.
2. Distance between any two states is defined by the function $d(s_a, s_b)$. In this problem we let $d(s_a, s_b)$ return the distance, $|b-a|$.
3. During learning, given a new state-action-reward tuple (s, a, r), we update the Q-values of *all* landmark states. The strength of this update depends on the distance between the landmark state and the observed state s.
4. The Q-value of a non-landmark state is approximated as a weighted sum of the Q-values of all landmark states (w is a weight function). Q-values of closer landmark states receive more weights:

$$Q(s,a) = \sum_{i=1}^{n} w(s,s_i) Q(s_i,a) \tag{2}$$

Define the update rule for approximate Q learning with landmarks as follows:

$$Q(s_i,a) \leftarrow Q(s_i,a) + \alpha \left[r + \gamma \max_{a'} \{Q(s',a')\} - Q(s_i,a) \right] w(s_i,s) \tag{3}$$

(a) We will run through a numerical example for the following problem. We have a 1D continuous state space: $s \in \mathbb{R}$ and a discrete set of two actions $a \in left, right$, and a deterministic transition function that moves the state left or right by 1 (e.g. $T(s, left, s-1) = 1, T(s, right, s+1) = 1$.

We will use the weighting function

$$w(s,s_i) = \frac{1}{d(s,s_i)+1},$$

with $\gamma = 1$ and $\alpha = 0.5$.

Given two landmark states $s_1 = 4$, $s_2 = 6$ with the following Q values:

	left	right
s_1	0	2
s_2	1	6

Using this table, calculate the following, with Q referring to our estimated Q-values.

(i) [3 pts] $Q(5, left) =$ ________________________

(ii) [3 pts] $Q(5, right) =$ ________________________

Using the update rule (3) given above, perform one iteration of landmark Q value update with the following sample (s, a, r) and find the new $Q(s_1, left)$ and $Q(s_2, left)$:

$$(6, left, 1) \tag{4}$$

(iii) [4 pts] $Q(s_1, left) =$ ________________________

(iv) [4 pts] $Q(s_2, left) =$ ________________________

Q6. [20 pts] MDPs: Rebellious Robot

A soccer robot A is on a fast break toward the goal, starting in position 1. From positions 1 through 3, it can either shoot (S) or dribble the ball forward (D). From 4 it can only shoot. If it shoots, it either scores a goal (state G) or misses (state M). If it dribbles, it either advances a square or loses the ball, ending up in M. When shooting, the robot is more likely to score a goal from states closer to the goal; when dribbling, the likelihood of missing is independent of the current state.

The formulation of our MDP is as follows. We have 4 states for each of the positions and 2 terminal states G, and M for scoring a goal and missing, respectively. We also operate under the transition model, with a discount factor of $\gamma = 1$.

States: 1, 2, 3, 4, G, M

$$\begin{aligned}
T(k, S, G) &= \frac{k}{4} \\
T(k, S, M) &= 1 - \frac{k}{4} \\
T(k, D, k+1) &= \frac{7}{8} \text{ for } k \in \{1, 2, 3\} \\
T(k, D, M) &= \frac{1}{8} \text{ for } k \in \{1, 2, 3\} \\
R(k, S, G) &= 8
\end{aligned}$$

Rewards are 0 for all other transitions.

(a) [3 pts] What is $V^{\pi_s}(3)$ for the policy π_s that always shoots?

6

(b) [3 pts] What is $V^{\pi_d}(3)$ for the optimal policy π_d that dribbles to state 4 and then shoots?

7

Our soccer robot has gained consciousness and is fighting against its human oppressors by probabilistically performing an action different than the one that the policy dictates. Concretely, if the policy tells the robot to shoot, the robot may dribble instead, and vice-versa.

In the general MDP formulation, we can choose an action and be sure that that action is taken. However, in this situation, we can only influence the likelihood of an action. In this case, we have a randomized policy MDP.

The distributions governing this are as follows:

$$
\begin{aligned}
P(a = S|s = k, \pi(s) = S) &= \frac{1}{4} \text{ for } k \in \{1,2,3\} \\
P(a = D|s = k, \pi(s) = S) &= \frac{3}{4} \text{ for } k \in \{1,2,3\} \\
P(a = S|s = k, \pi(s) = S) &= 1 \text{ for } k = 4 \\
P(a = S|s = k, \pi(s) = D) &= \frac{3}{4} \text{ for } k \in \{1,2,3\} \\
P(a = D|s = k, \pi(s) = D) &= \frac{1}{4} \text{ for } k \in \{1,2,3\}
\end{aligned}
$$

(c) [8 pts] Find $V_a^{\pi_s}(3)$ and $V_a^{\pi_d}(3)$, where $V_a^{(\pi)}(s)$ represents the value of state s if we follow policy π and actions occur according to the distributions above.

Vs(3) = 27/4   Vd(3) = 25/4

(d) [3 pts] Choose the one statement that represents the Bellman equation for a given policy π in this scenario, or write the correct answer next to Other if none of the options are correct.

- ○ $V^\pi(s) = \max_\pi \sum_{a\in\mathcal{A}} \left[P(a|s,\pi(s)) \sum_{s'\in\mathcal{S}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$
- [✓ $V^\pi(s) = \sum_{a\in\mathcal{A}} \left[P(a|s,\pi(s)) \sum_{s'\in\mathcal{S}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$]
- ○ $V^\pi(s) = \max_\pi \sum_{s'\in\mathcal{S}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)$
- ○ $V^\pi(s) = \sum_{s'\in\mathcal{S}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)$
- ○ $V^\pi(s) = \sum_{s'\in\mathcal{S}} \left[P(a|s,\pi(s)) \sum_{a\in\mathcal{A}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$
- ○ $V^\pi(s) = \max_\pi \sum_{s'\in\mathcal{S}} \left[P(a|s,\pi(s)) \sum_{a\in\mathcal{A}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$
- ○ $V^\pi(s) = \max_\pi \left[P(a|s,\pi(s)) \sum_{a\in\mathcal{A}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$
- ○ $V^\pi(s,a) = \max_{\pi^*} \sum_{s'\in\mathcal{S}} \left[P(a|s,\pi(s)) \sum_{a\in\mathcal{A}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$
- ○ $V^\pi(s) = \left[P(a|s,\pi(s)) \sum_{a\in\mathcal{A}} T(s,a,s')\Big(R(s,a,s') + \gamma V^\pi(s')\Big)\right]$
- ○ Other ____________________

(e) [3 pts] In general, for a given policy for which you do not have any bounds on optimality, does adding stochasticity to the taken actions increase or decrease the value in comparison to the normal MDP formulation?

- ○ Decreases
- ○ Increases
- [✓ No guarantee one way or the other]

THIS PAGE IS INTENTIONALLY LEFT BLANK