CS 188 Introduction to Summer 2015 Artificial Intelligence

Midterm 1

- 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.

First name	
Last name	
SID	
edX username	
Name of person on your left	
Name of person on your right	

For staff use only:

Q1.	Search	/12
Q2.	Game Trees and Pruning	/9
Q3.	Encrypted Knowledge Base	/6
Q4.	MDPs - Farmland	/8
Q5.	Reinforcement Learning	/9
	Total	/44

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [12 pts] Search

Each True/False question is worth 1 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth −1 points.**

(a) Consider a graph search problem where for every action, the cost is at least ϵ, with $\epsilon > 0$. Assume the used heuristic is consistent.

(i) [1 pt] [*true* or *false*] Depth-first graph search is guaranteed to return an optimal solution.
False. Depth first search has no guarantees of optimality. Further, it measures paths in length and not cost.

(ii) [1 pt] [*true* or *false*] Breadth-first graph search is guaranteed to return an optimal solution.
False. Breadth first search has no guarantees of optimality unless the actions all have the same cost, which is not the case here.

(iii) [1 pt] [*true* or *false*] Uniform-cost graph search is guaranteed to return an optimal solution.
True. UCS expands paths in order of least total cost so that the optimal solution is found.

(iv) [1 pt] [*true* or *false*] Greedy graph search is guaranteed to return an optimal solution.
False. Greedy search makes no guarantees of optimality. It relies solely on the heuristic and not the true cost.

(v) [1 pt] [*true* or *false*] A* graph search is guaranteed to return an optimal solution.
True, since the heuristic is consistent in this case.

(vi) [1 pt] [*true* or *false*] A* graph search is guaranteed to expand no more nodes than depth-first graph search.
False. Depth-first graph search could, for example, go directly to a sub-optimal solution.

(vii) [1 pt] [*true* or *false*] A* graph search is guaranteed to expand no more nodes than uniform-cost graph search.
True. The heuristic could help to guide the search and reduce the number of nodes expanded. In the extreme case where the heuristic function returns zero for every state, A* and UCS will expand the same number of nodes. In any case, A* with a consistent heuristic will never expand more nodes than UCS.

(b) Let $h_1(s)$ be an admissible A* heuristic. Let $h_2(s) = 2h_1(s)$. Then:

(i) [1 pt] [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to be an optimal solution.

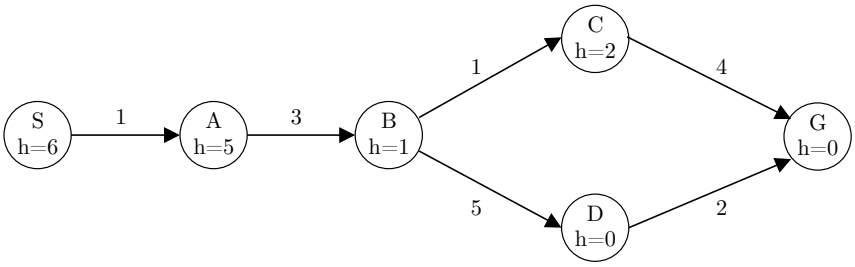
False. h_2 is not guaranteed to be admissible since only one side of the admissibility inequality is doubled.

(ii) [1 pt] [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to have a cost at most twice as much as the optimal path.
True. In A* tree search we always have that as long as the optimal path to the goal has not been found, a prefix of this optimal path has to be on the fringe. Hence, if a non-optimal solution is found, then at time of popping the non-optimal path from the fringe, a path that is a prefix of the optimal path to the goal is sitting on the fringe. The cost $\bar{g}$ of a non-optimal solution when popped is its f-cost. The prefix of the optimal path to the goal has an f-cost of $g + h_0 = g + 2h_1 \leq 2(g + h_1) \leq 2C^*$, with C^* the optimal cost to the goal. Hence we have that $\bar{g} \leq 2C^*$ and the found path is at most twice as long as the optimal path.

(iii) [1 pt] [*true* or *false*] The solution found by A* graph search with h_2 is guaranteed to be an optimal solution.
False. h_2 is not guaranteed to be admissible and graph search further requires consistency for optimality.

(c) [2 pts] The heuristic values for the graph below are not correct. For which single state (S, A, B, C, D, or G) could you change the heuristic value to make everything admissible and consistent? What range of values are possible to make this correction?

State: B Range: [2,3]


S
h=6
1
A
h=5
3
B
h=1
1
C
h=2
4
5
D
h=0
2
G
h=0

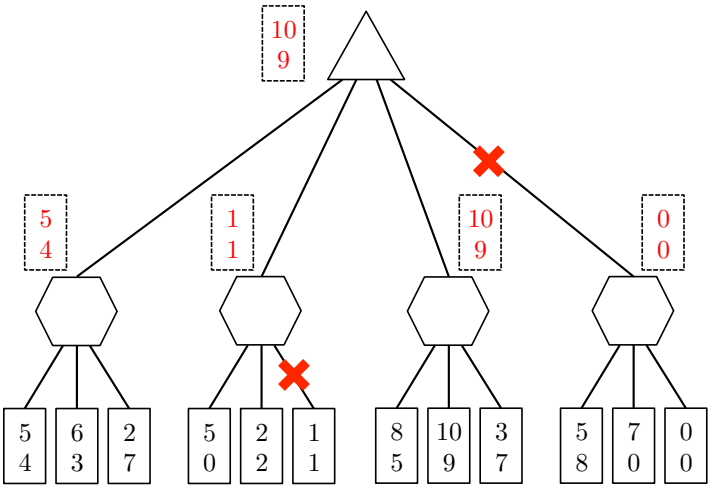
Q2. [9 pts] Game Trees and Pruning

You and one of the 188 robots are playing a game where you both have your own score.

- The maximum possible score for either player is 10.
- You are trying to maximize your score, and you do not care what score the robot gets.
- The robot is trying to minimize the absolute difference between the two scores. In the case of a tie, the robot prefers a lower score. For example, the robot prefers (5,3) to (6,3); it prefers (5,3) to (0,3); and it prefers (3,3) to (5,5).

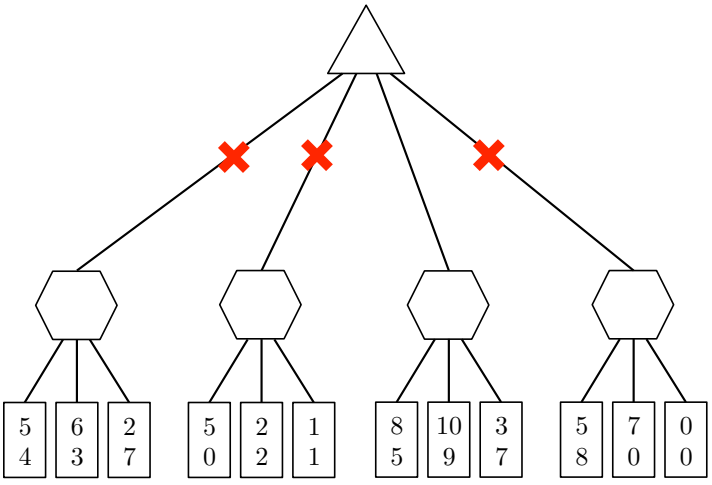
The figure below shows the game tree of your max node followed by the robots nodes for your four different actions. The scores are shown at the leaf nodes with your score always on top and the robots score on the bottom.

(a) [3 pts] Fill in the dashed rectangles with the pair of scores preferred by each node of the game tree.



(b) [3 pts] You can save computation time by using pruning in your game tree search. On the game tree above, put an 'X' on line of branches that do not need to be explored. Assume that branches are explored from left to right.

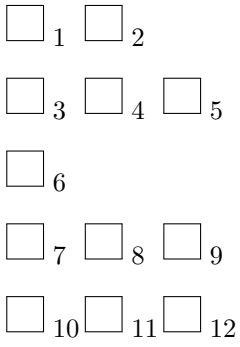
(c) [2 pts] You now have access to an oracle that tells you the order of branches to explore that maximizes pruning. On the copy of the game tree below, put an 'X' on line of branches that do not need to be explored given this new information from the oracle.



Q3. [6 pts] Encrypted Knowledge Base

We have a propositional logic knowledge base, but unfortunately, it is encrypted. The only information we have is that:

- Each of the following 12 boxes contains a propositional logic symbol (A, B, C, D, or E) or a propositional logic operator and
- Each line is a valid propositional logic sentence.



(a) [3 pts] We are going to implement a constraint satisfaction problem solver to find a valid assignment to each box from the domain $\{A, B, C, D, E, \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$.

Propositional logic syntax imposes constraints on what can go in each box. What values are in the domain of boxes 1-6 after enforcing the unary syntax constraints?

Box	Remaining Values
1	$\neg$
2	A B C D E
3	A B C D E $\neg$
4	$\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
5	A B C D E
6	A B C D E

(b) [2 pts] You are given the following assignment as a solution to the knowledge base CSP on the previous page:

$$\neg A$$
$$B \Rightarrow A$$
$$D$$
$$C \vee B$$
$$D \vee E$$

Now that the encryption CSP is solved, we have an entirely new CSP to work on: finding a model. In this new CSP the variables are the symbols $\{A, B, C, D, E\}$ and each variable could be assigned to $true$ or $false$.

We are going to run CSP backtracking search with forward checking to find a propositional logic model M that makes all of the sentences in this knowledge base true.

After choosing to assign C to false, what values are removed by running forward checking? On the table of remaining values below, cross off the values that were removed.

Symbol	Remaining Values
A	F
B	T ~~F~~
C	F
D	T
E	T F

Forward checking removes the value false from the domain of B. Forward checking does not continue on to make any other arcs consistent.

(c) [2 pts] We eventually arrive at the model $M = \{A = False, B = False, C = True, D = True, E = True\}$ that causes all of the knowledge base sentences to be true. We have a query sentence α specific as $(A \vee C) \Rightarrow E$. Our model M also causes α to be true. Can we say that the knowledge base entails α? Explain briefly (in one sentence) why or why not.

No, the knowledge base does not entail α. There are other models for which the knowledge base could be true and the query be false. Specifically $\{A = False, B = False, C = True, D = True, E = False\}$ satisfies the knowledge base but causes the query α to be false.

Q4. [8 pts] MDPs - Farmland

In the game FARMLAND, players alternate taking turns drawing a card from one of two piles, PIG and COW. PIG cards have equal probability of being 3 points or 1 point, and COW cards are always worth 2 points. Players are trying to be the first to reach 5 points or more. We are designing an agent to play FARMLAND.

We will use a modified MDP to come up with a policy to play this game. States will be represented as tuples (x, y) where x is our score and y is the opponent's score. The value $V(x, y)$ is an estimate of the probability that we will win at the state (x, y) **when it is our turn to play** and both players are playing optimally. Unless otherwise stated, assume both players play optimally.

First, suppose we work out by hand V^*, the table of actual probabilities.

		Opponent				
		0	1	2	3	4
	0	0.75	0.5	0.5	0	0
	1	1	1	0.5	0	0
You	2	1	1	0.75	0.5	0.5
	3	1	1	1	1	1
	4	1	1	1	1	1

According to this table, $V^*(1, 2) = 0.5$, so with both players playing optimally, the probability that we will win if our score is 1, the opponent's score is 2, and it is our turn to play is 0.5.

(a) [2 pts] At the beginning of the game, would you choose to go first or second? Justify your answer using the table.

You should choose to go first. Since $V^*(0, 0) = 0.75$, if it is your turn and the scores are both 0, the probability that you will win is 0.75.

(b) [3 pts] If our current state is (x, y) (so our score is x and the opponent's score is y) but it is **the opponent's turn to play**, what is the probability that we will win if both players play optimally **in terms of V***?

$$1 - V^*(y, x)$$

(c) [3 pts] As FARMLAND is a very simple game, you quickly grow tired of playing it. You decide to buy the FARMLAND expansion, BOVINE BONANZA, which adds loads of exciting cards to the COW pile! Of course, this changes the transition function for our MDP, so the table V^* above is no longer correct. We need to come up with an update equation that will ultimately make V_∞ converge on the actual probabilities that we will win.

You are given the transition function $T((x, y), a, (x', y))$ and the reward function $R((x, y), a, (x', y))$. The transition function $T((x, y), a, (x', y))$ is the probability of transitioning from state (x, y) to state (x', y) when action a is taken (i.e. the probability that the card drawn gives $x' - x$ points).

Since we are only trying to find the probability of winning and we don't care about the margin of victory, the reward function $R((x, y), a, (x', y))$ is 1 whenever (x', y) is a winning state and 0 everywhere else. As in normal value iteration, all values will be initialized to 0 (i.e. $V(x, y) = 0$ for all states (x, y)).

Write an update equation for $V_{k+1}(x, y)$ in terms of T, R and V_k.
Hint: you will need to use your answer from part b.

$$V_{k+1}(x, y) \leftarrow \max_a \sum_{x'} T((x, y), a, (x', y))[R((x, y), a, (x', y)) + (1 - V_k(y, x'))]$$

Q5. [9 pts] Reinforcement Learning

(a) Each True/False question is worth 1 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth −1 points.**

(i) [1 pt] [*true* or *false*] Temporal difference learning is an online learning method.
Temporal difference learning is used when we don't have the full MDP model and must collect online samples.

(ii) [1 pt] [*true* or *false*] Q-learning: Using an optimal exploration function leads to no regret while learning the optimal policy.
In order to learn the optimal policy, you must explore, and exploring in general has a non-zero chance of regret.

(iii) [1 pt] [*true* or *false*] In a deterministic MDP (i.e. one in which each state / action leads to a single deterministic next state), the Q-learning update with a learning rate of $\alpha = 1$ will correctly learn the optimal q-values (assume that all state/action pairs are visited sufficiently often). Remember that the learning rate is only there because we are trying to approximate a summation with a single sample. In a deterministic MDP where s' is the single state that always follows when we take action a in state s, we have $Q(s,a) = R(s,a,s') + \max_{a'} Q(s',a')$, which is exactly the update we make.

(iv) [1 pt] [*true* or *false*] A small discount (close to 0) encourages greedy behavior.
A discount close to zero will place extremely small values on rewards more than one step away, leading to greedy behavior that looks for immediate rewards.

(v) [1 pt] [*true* or *false*] A large, negative living reward ($\ll 0$) encourages greedy behavior.
A negative living reward adds a penalty for every step taken. If that penalty is large, the agent will prefer to find an exit as soon as possible despite potential rewards on longer paths.

(vi) [1 pt] [*true* or *false*] A negative living reward can always be expressed using a discount < 1.
While both negative living rewards and discounts can encourage similar behavior, they are mathematically different. A discount has a multiplicative effect at each step, whereas a living reward only has an additive effect.

(vii) [1 pt] [*true* or *false*] A discount < 1 can always be expressed as a negative living reward.
While both negative living rewards and discounts can encourage similar behavior, they are mathematically different. A discount has a multiplicative effect at each step, whereas a living reward only has an additive effect.

(b) [2 pts] Given the following table of Q-values for the state A and the set of actions $\{Forward, Reverse, Stop\}$, what is the probability that we will take each action on our next move when we following an ϵ-greedy exploration policy (assuming any random movements are chosen uniformly from all actions)?

$Q(A, Forward) = 0.75$
$Q(A, Reverse) = 0.25$
$Q(A, Stop) = 0.5$

Action	Probability (in terms of ϵ)
Forward	$(1-\epsilon) + \frac{\epsilon}{3} = 1 - \frac{2\epsilon}{3}$
Reverse	$\frac{\epsilon}{3}$
Stop	$\frac{\epsilon}{3}$