CS 188 Fall 2008 Introduction to Artificial Intelligence Midterm Exam

INSTRUCTIONS

- You have 80 minutes. 70 points total. Don't panic!
- The exam is closed book, closed notes except a one-page crib sheet, non-programmable calculators only.
- 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.
- **Question 0: Fill out the following grid and write your name, SID, login, and GSI at the top of each subsequent page.** (-1 points if done incorrectly!)

Last Name	
First Name	
SID	
Login	
GSI	
All the work on this exam is my own. **(please sign)**	

For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Total
/12	/11	/15	/17	/15	/70

THIS PAGE INTENTIONALLY LEFT BLANK

1. (12 points.) Search: Mr. and Ms. Pacman

Pacman and Ms. Pacman are lost in an NxN maze and would like to meet; *they don't care where*. In each time step, *both* simultaneously move in one of the following directions: {NORTH, SOUTH, EAST, WEST, STOP}. They do *not* alternate turns. You must devise a plan which positions them together, somewhere, in as few time steps as possible. Passing each other does not count as meeting; they must occupy the same square at the same time.

(a) (4 points) Formally state this problem as a *single-agent* state-space search problem.

States:

Answer: The set of pairs of positions for Pacman and Ms. Pacman:
$\{((x_1, y_1), (x_2, y_2)) \mid x_1, x_2, y_1, y_2 \in \{1, 2, \ldots, N\}\}$

Maximum size of state space:

Answer: N^2 for both pacmen, hence N^4 total

Maximum branching factor:

Answer: Each pacman has a choice of 5 actions, hence $5^2 = 25$ total

Goal test:

Answer: $isGoal((x_1, y_1), (x_2, y_2)) := (x_1 = x_2) \wedge (y_1 = y_2)$

(b) (3 points) Give a non-trivial admissible heuristic for this problem.

Answer: Manhattan distance between Pacman and Ms. Pacman DIVIDED BY 2 (since both take a step simultaneously)

(c) (3 points) Circle all of the following graph search methods which are guaranteed to output optimal solutions to *this problem*:

(i) DFS
(ii) BFS
(iii) UCS
(iv) A* (with a consistent and admissible heuristic)
(v) A* (with heuristic that returns zero for each state)
(vi) Greedy search (with a consistent and admissible heuristic)

Answer: BFS, UCS, A* (with a consistent and admissible heuristic), A* (with heuristic that returns zero for each state)

(d) (2 points) If h_1 and h_2 are admissible, which of the following are also guaranteed to be admissible? Circle all that apply:

(i) $h_1 + h_2$
(ii) $h_1 * h_2$
(iii) $max(h_1, h_2)$
(iv) $min(h_1, h_2)$
(v) $(\alpha)h_1 + (1 - \alpha)h_2$, for $\alpha \in [0, 1]$

Answer: $max(h_1, h_2)$, $min(h_1, h_2)$, $(\alpha)h_1 + (1 - \alpha)h_2$, for $\alpha \in [0, 1]$

2. (11 points.) CSPs: Finicky Feast

You are designing a menu for a special event. There are several choices, each represented as a variable: (A)ppetizer, (B)everage, main (C)ourse, and (D)essert. The domains of the variables are as follows:

A: (v)eggies, (e)scargot
B: (w)ater, (s)oda, (m)ilk
C: (f)ish, (b)eef, (p)asta
D: (a)pple pie, (i)ce cream, (ch)eese

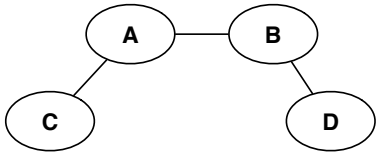
Because all of your guests get the same menu, it must obey the following dietary constraints:

(i) Vegetarian options: The appetizer must be veggies or the main course must be pasta or fish (or both).

(ii) Total budget: If you serve the escargot, you cannot afford any beverage other than water.

(iii) Calcium requirement: You must serve at least one of milk, ice cream, or cheese.

(a) (3 points) Draw the constraint graph over the variables A, B, C, and D.



(b) (2 points) Imagine we first assign A=e. Cross out eliminated values to show the domains of the variables after forward checking.

A	[		e		]
B	[	w	s	m	]
C	[	f	b	p	]
D	[	a	i	ch	]

Answer: The values s, m, and b should be crossed off. "s" and "m" are eliminated due to being incompatible with "e" based on constraint (ii). "b" is eliminated due to constraint (i).

(c) (3 points) Again imagine we first assign A=e. Cross out eliminated values to show the domains of the variables after arc consistency has been enforced.

A	[		e		]
B	[	w	s	m	]
C	[	f	b	p	]
D	[	a	i	ch	]

Answer: The values s, m, b, and a should be eliminated. The first three are crossed off for the reasons above, and "a" is eliminated because there is no value for (B) that is compatible with "a" (based on constraint (iii)).

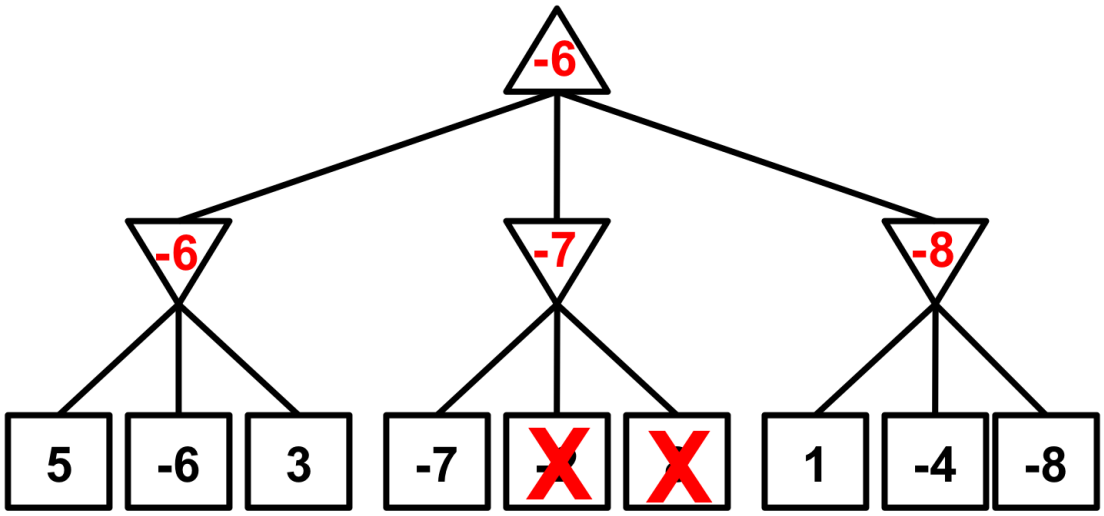
(d) (1 point) Give a solution for this CSP or state that none exists.
Answer: Multiple solutions exist. One is A=e, B=w, C=f, and D=i.

(e) (2 points) For general CSPs, will enforcing arc consistency after an assignment *always* prune at least as many domain values as forward checking? Briefly explain why or why not.
Answer: Two answers are possible:
Yes. The first step of arc consistency is equivalent to forward checking, so arc consistency removes all values that forward checking does.
No. While forward checking is a subset of arc consistency, after any assignment, arc consistency may have already eliminated values in a previous step that are eliminated in that step by forward checking. Thus, enforcing arc consistency will never leave more domain values than enforcing forward checking, but on a given

step, forward checking might prune values than arc consistency by pruning values that have already been pruned by arc consistency.

3. (15 points.) Game Trees: The Balancer

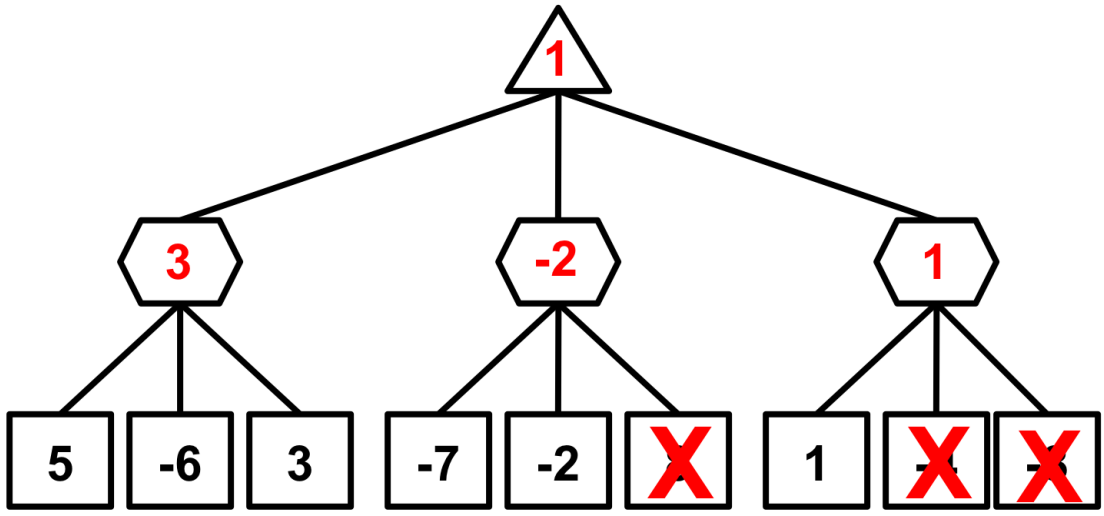
Consider the following zero-sum game, in which the utilities $U_A(s)$ are shown for the first player (A). Assume the second player (B) is a minimizer: B holds the opposite utilities to A, $U_B(s) = -U_A(s)$. In this case, B's maximization of U_B is equivalent to minimization of U_A (i.e. the computation is standard minimax).



(a) (2 points) In each node, write $U_A(s)$, the (minimax) utility of that state for player A, assuming that B is a minimizer.
Answer: Displayed above.

(b) (3 points) Cross off any nodes which will be skipped by alpha-beta pruning, assuming left-to-right ordering.
Answer: Displayed above.

Assume now that B is not a minimizer, but a *balancer*. A balancer does not try to minimize A's score, but rather wishes the outcome of the game to be as balanced as possible. Formally, assume B's utility for a state s is defined as $U_B(s) = -|U_A(s)|$. The game tree is shown here, with hexagons indicating player B's control.



(c) (3 points) In each node, write $U_A(s)$, the utility of that state for player A, assuming that B is a balancer.
Answer: Displayed above.

(d) (3 points) Write pseudocode for the functions which compute the $U_A(s)$ values of game states in the general case of multi-turn games where B is a balancer. Assume you have access to the following functions: `successors(s)` gives the possible next states, `isTerminal(s)` checks whether a state is a terminal state, and `terminalValue(s)` returns A's utility for a terminal state. *Careful:* As in minimax, be sure that both functions compute and return player A's utilities for states – B's utility can always be computed from A's utility.

Answer: Below. Note that for balanceValue(s), we must return the utility the maximizer's perspective.

```
def maxValue(s): // compute $U_A(s)$ assuming that A is next to move.
  if isTerminal(s): return terminalValue(s)
  return max([balanceValue(succ) for succ in successors(s)])

def balanceValue(s): // compute $U_A(s)$ assuming that B is next to move.
  if isTerminal(s): return terminalValue(s)
  maxVal = -infty
  for succ in successors(s):
    val = maxValue(succ)
    if math.abs(val) < math.abs(maxVal): maxVal = val
  return maxVal
```

(h) (2 points) Consider pruning children of a B node in this scenario. On the tree on the bottom of the previous page, cross off any nodes which can be pruned, again assuming left-to-right ordering.

Answer: Answers above.

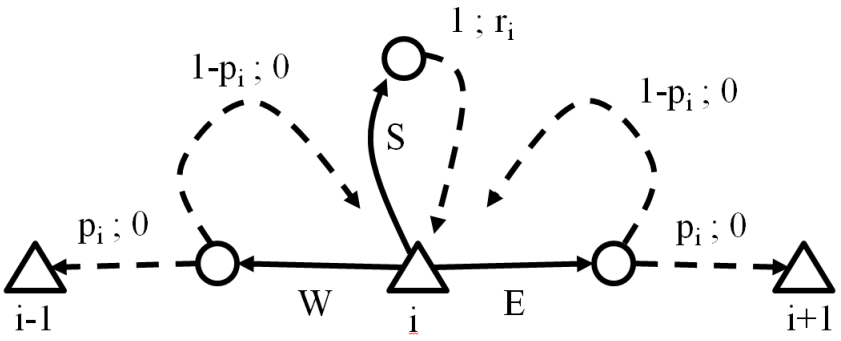
(i) (2 points) Again consider pruning children of a B node s. Let α be the best option for an A node higher in the tree, just as in alpha-beta pruning, and let v be the U_A value of the best action B has found so far from s. Give a general condition under which `balanceValue(s)` can return without examining any more of its children.

Answer: $|v| < \alpha$.

4. (17 points.) MDPs and RL: Wandering Merchant

There are N cities along a major highway numbered 1 through N. You are a merchant from city 1 (that's where you start). Each day, you can either travel to a neighboring city (actions *East* or *West*) or stay and do business in the current city (action *Stay*). If you choose to travel from city i, you successfully reach the next city with probability p_i, but there is probability $1 - p_i$ that you hit a storm, in which case you waste the day and do not go anywhere. If you stay to do business in city i, you get $r_i > 0$ in reward; a travel day has reward 0 regardless of whether or not you succeed in changing cities.

The diagram below shows the actions and transitions from city i. Solid arrows are actions; dashed arrows are resulting transitions labeled with their probability and reward, in that order.



(a) (2 points) If for all i, $r_i = 1$, $p_i = 1$, and there is a discount $\gamma = 0.5$, what is the value $V^{stay}(1)$ of being in city 1 under the policy that always chooses *stay*? Your answer should be a real number.

Answer: for all cities (states) $i = 1, \ldots, N$, we have that the optimal value behaves as follows:

$$V^{stay}(i) = r_i + \gamma V^{stay}(i)$$

(remember, this is like the Bellman equation for a fixed policy). Plugging in values, we get $V^{stay}(i) = 1 + 0.5V^{stay}(i)$. Now we can just solve for $V^{stay}(i)$ using algebra to obtain $V^{stay}(i) = 2$. In particular, $V^{stay}(1) = 2$.

(b) (2 points) If for all i, $r_i = 1$, $p_i = 1$, and there is a discount $\gamma = 0.5$, what is the optimal value $V^*(1)$ of being in city 1?

Intuitive Answer: since all the cities offer the same reward ($r_i = 1$), there is no incentive to move to another city to do business, so the optimal policy is to always stay, yielding $V^*(1) = 2$.

More Formal Answer:

For all cities (states) $i = 1, \ldots, N$, writing out the Bellman equations:[1]

$$V^*(i) = \max\{\underbrace{r_i + \gamma V^*(i)}_{\text{stay}}, \underbrace{p_i \gamma V^*(i-1) + (1-p_i)\gamma V^*(i)}_{\text{left}}, \underbrace{p_i \gamma V^*(i+1) + (1-p_i)\gamma V^*(i)}_{\text{right}}\}$$

Since $p_i = 1$, this drastically simplifies:

$$V^*(i) = \max\{\underbrace{r_i + \gamma V^*(i)}_{\text{stay}}, \underbrace{\gamma V^*(i-1)}_{\text{left}}, \underbrace{\gamma V^*(i+1)}_{\text{right}}\}$$

From this, we see that $V^*(i)$ is the same for all i, so the max is obtained always with the stay action.

(c) (2 points) If the r_i's and p_i's are known positive numbers and there is almost no discount, i.e. $\gamma \approx 1$, describe the optimal policy. You may define it formally or in words, e.g. "always go east," but your answer

[1] For $i = 1$, omit the left action; for $i = N$, omit the right action.

should precisely define how an agent should act in any given state. *Hint:* You should not need to do any computation to answer this question.

Basically Right Answer: the optimal policy is to always move towards the city with the highest reward. Once there, stay there and do business forever.

Technical Answer: The only complication is due to possible ties. Let $r^* = \max_{1 \le i \le n} r_i$ be the maximum reward out of all the cities. The optimal policy from city i is as follows: if $r_i = r^*$, stay; otherwise, move towards the closest city j that has $r_j = r^*$, where distance between i and $j > i$ is the the expected number of moves to get there ($\sum_{k=i}^{j-1} 1/p_k$).

Suppose we run value iteration. Recall that $V_k(s)$ is the value of state s after k rounds of value iteration and all the values are initialized to zero.

(d) (2 points) If the optimal value of being in city 1 is positive, i.e. $V^*(1) > 0$, what is the largest k for which $V_k(1)$ could still be zero? Be careful of off-by-one errors.

Answer: Assuming $r_i > 0$, then the largest k is 0, because $V_1(s) = \max\{r_i + 0, \cdots\} > 0$.

(Intended) Answer: If we don't assume $r_i > 0$, then the largest k is $N - 1$. Proof: since $V^*(1) > 0$, at least one of the r_i's must be strictly positive. After one iteration, $V_1(i) > 0$; after two iterations, $V_2(i-1) > 0$; finally after i iterations, $V_i(1) > 0$. In the meantime, if all $r_j = 0$ for $j < i$, then $V_j(1) = 0$ for all $j < i$. In the worst case, $i = N$, so $V_{N-1}(1) = 0$ is possible, but $V_N(1) > 0$.

(e) (2 points) If all of the r_i and p_i are positive, what is the largest k for which $V_k(s)$ could still be zero for some state s? Be careful of off-by-one errors.

Answer: Since $r_i > 0$, the largest k is 0, because $V_1(s) = \max\{r_i + 0, \cdots\} > 0$.

Suppose we don't know the r_i's or the p_i's, so we decide to do Q-learning.

(f) (3 points) Suppose we experience the following sequence of states, actions, and rewards: (s=1, a=*stay*, r=4), (s=1, a=*east*, r=0), (s=2, a=*stay*, r=6), (s=2, a=*west*, r=0), (s=1, a=*stay*, r=4, s=1). What are the resulting $Q(s, a)$ values if the learning rate is 0.5, the discount is 1, and we start with all $Q(s, a) = 0$? Fill in the table below; each row should hold the q-values after the transition specified in its first column. You may leave unchanged values blank.

(s,a,r,s')	Q(1,S)	Q(1,E)	Q(2,W)	Q(2,S)
initial	0	0	0	0
(1,S,4,1)				
(1,E,0,2)				
(2,S,6,2)				
(2,W,0,1)				
(1,S,4,1)				

After $(1, S, 4, 1)$, we update $Q(1, S) \leftarrow 0.5[4 + 1 \cdot 0] + 0.5(0) = 2$.
After $(1, E, 0, 2)$, we update $Q(1, E) \leftarrow 0.5[0 + 1 \cdot 0] + 0.5(0) = 0$.
After $(2, S, 6, 2)$, we update $Q(2, S) \leftarrow 0.5[6 + 1 \cdot 0] + 0.5(0) = 3$.
After $(2, W, 0, 1)$, we update $Q(2, W) \leftarrow 0.5[0 + 1 \cdot 2] + 0.5(0) = 1$.

After $(1, S, 4, 1)$, we update $Q(1, S) \leftarrow 0.5[4 + 1 \cdot 2] + 0.5(2) = 4$.

Circle *true* or *false*; skipping here is worth 1 point per question.

(g) (2 points) (*True/False*) Q-learning will only learn the optimal q-values if actions are eventually selected according to the optimal policy.

Answer: False. As long as the policy used explores all the states (even a random policy will work), Q-learning will find the optimal q-values.

(h) (2 points) (*True/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.

Answer: True. 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.

5. (15 points.) Probability and Utilities: Wheel of Fortune

You are playing a simplified game of Wheel of Fortune. The objective is to correctly guess a three letter word. Let X, Y, and Z represent the first, second, and third letters of the word, respectively. There are only 8 possible words: X can take on the values 'c' or 'l', Y can be 'a' or 'o', and Z can be 'b' or 't'.

Before you guess the word, two of the three letters will be revealed to you. In the first round of the game, you choose one of X, Y or Z to be revealed. In the second round, you choose one of the remaining two letters to be revealed. In the third round, you guess the word. If you guess correctly, you win. The utility of winning is 1, while the utility of losing is 0.

You watch the game a lot and determine that the eight possible words occur with the probabilities shown on the right. Your goal is to act in such a way as to maximize your chances of winning (and thereby your expected utility).

$P(X, Y, Z)$

X	Y	Z	P
c	a	b	0.10
c	a	t	0.10
c	o	b	0.20
c	o	t	0.20
l	a	b	0.05
l	a	t	0.15
l	o	b	0.05
l	o	t	0.15

(a) (3 points) What is the distribution P(Y, Z)? Your answer should be in the form of a table.

Answer:

P(X=c,Y=a)=0.2

P(X=c,Y=o)=0.4

P(X=l,Y=a)=0.2

P(X=l,Y=o)=0.2

(b) (2 points) Are the second and third letters (Y and Z) independent? Show a specific computation that supports your claim.

Answer: No, since P(X=c) = 0.6, P(Y=a) = 0.4 but P(X=c,Y=a)=0.2 which is not P(X=c)P(Y=a)=0.24 (other counterexamples exist too)

(c) (2 points) Are the second and third letters (Y and Z) independent if you know the value of the first letter (X)? Show a specific computation that supports your claim.

Answer: Yes. $P(Y = a, Z = b|X = c) = P(X = c, Y = a, Z = b)/P(X = c) = 1/6.$

$P(Y = a|X = c) = (0.1 + 0.1)/0.6 P(Z = b|X = c) = (0.1 + 0.2)/(0.6) = 1/2.$

Thus, $P(Y = a, Z = b|X = c) = 1/6 = P(Y = a|X = c)P(Z = b|X = c)$. To be certain, you have to also check for all pairs (not required for full credit). Alternatively, you can show that $P(Y|X, Z) = P(Y|X)$

Suppose that in the first round, you ask about X and are told that X = c. It is the second round and you can now either ask the host to reveal Y or to reveal Z.

(d) (2 points) If you ask the host to reveal Y, what is the probability that you will win in the third round?

Answer: Since Y and Z are independent conditioned on X, no matter what Y comes out to be, $P(Z = b|X = c, Y)$ will be 0.5 Thus, you'll guess arbitrarily and win with probability 0.5

(e) (1 point) What letter *should* you ask the host about in the second round to maximize your chance of winning, Y or Z?

Answer: Z, since you'll be able to win 2/3 of the time (see part f)

(f) (3 points) What is your expected utility if you act optimally from the state where X=c?

Answer: Since Y and Z are conditionally independent given X, knowing Z won't give you any additional information about Y. So, you'll guess the most likely value of Y given $X = c$, which is o since $P(Y = o|X = c) = 2/3$ and win 2/3 of the time.

(g) (2 points) Suppose that the host is allowed to pick any distribution over the three variables but has to tell you what the distribution is before the game starts. What distribution should the host pick to minimize your chances of winning? Justify your answer briefly.

Answer: Uniform: you need some distribution where each letter has 50% chance of occuring and the values of each variable are independent of the others, so knowing the revealed values doesn't give any information about the hidden one.