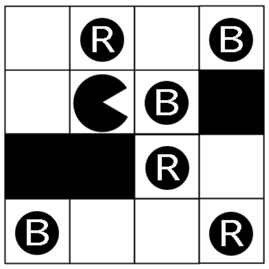
CS 188 Spring 2011 | Introduction to Artificial Intelligence | Midterm Exam Solutions

Q1. [11 pts] Foodie Pacman

There are two kinds of food pellets, each with a different color (red and blue). Pacman is only interested in tasting the two different kinds of food: the game ends when he has eaten 1 red pellet and 1 blue pellet (though Pacman may eat more than one of each pellet). Pacman has four actions: moving up, down, left, or right, and does not have a "stay" action. There are K red pellets and K blue pellets, and the dimensions of the board are N by M.



$K = 3$, $N = 4$, $M = 4$

(a) [1 pt] Give an efficient state space formulation of this problem. Specify the domain of each variable in your state space.

$(x \in [1 : N], y \in [1 : M], eaten_R \in \{T, F\}, eaten_B \in \{T, F\})$

(b) [2 pts] Give a tight upper bound on the size of the state space.

$4 \times N \times M$

(c) [2 pts] Give a tight upper bound on the branching factor of the search problem.

4

(d) [1 pt] Assuming Pacman starts the game in position (x,y), what is the initial state?

(x, y, F, F)

(e) [1 pt] Define a goal test for the problem.

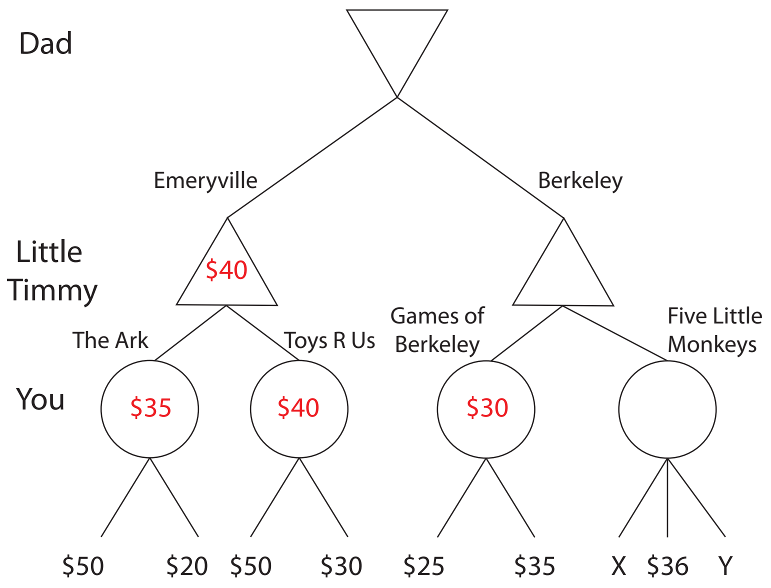
$(eaten_R == T)\&\&(eaten_B == T)$

(f) [4 pts] For each of the following heuristics, indicate (yes/no) whether or not it is admissible (a correct answer is worth 1 point, leaving it blank is worth 0 points, and an incorrect answer is worth -1 points).

Heuristic	Admissible?
The number of pellets remaining	No
The smallest Manhattan distance to any remaining pellet	No
The maximum Manhattan distance between any two remaining pellets	No
The minimum Manhattan distance between any two remaining pellets of opposite colors	No

Q2. [9 pts] Expectimax

Your little brother Timmy has a birthday and he was promised a toy. However, Timmy has been misbehaving lately and Dad thinks he deserves the least expensive present. Timmy, of course, wants the most expensive toy. Dad will pick the city from which to buy the toy, Timmy will pick the store and you get to pick the toy itself. You don't want to take sides so you decide to pick a toy at random. All prices (including X and Y) are assumed to be nonnegative.



(a) [1 pt] Fill in the values of all the nodes that don't depend on X or Y.

(b) [3 pts] What values of X will make Dad pick Emeryville regardless of the price of Y?

$\frac{x+y+36}{3} > 40 \Leftrightarrow x > 84 - y \Leftrightarrow x > 84$

(c) [3 pts] We know that Y is at most \$30. What values of X will result in a toy from Games of Berkeley regardless of the exact price of Y?

$\frac{x+y+36}{3} < 30 \Leftrightarrow x < 54 - y \Leftrightarrow x < 24$

(d) [2 pts] Normally, alpha-beta pruning is not used with expectimax. However, with some additional information, it is possible to do something similar. Which **one** of the following conditions on a problem are required to perform pruning with expectimax?

1. The children of the expectation node are leaves.
2. All values are positive.
3. $\boxed{\text{The children of the expectation node have specified ranges.}}$
4. The child to prune is last.

Q3. [6 pts] Forced Random Policy in MDP

(a) [6 pts] Assume you are asked to act in a given MDP $(S, A, T, R, \gamma, s_0)$. However, rather than being able to freely choose your actions, at each time step you must start by flipping a coin. If the coin lands heads, then you can freely choose your action. If the coin lands tails, however, you don't get to choose an action and instead an action will be chosen for you uniformly at random from the available actions. Can you specify a modified MDP $(S', A', T', R', \gamma', s'_0)$ for which the optimal policy maximizes the expected discounted sum of rewards under the specified restrictions on your ability to choose actions? (Hint: you may not need to change all entities in the MDP.)

$S' = S$

$A' = A$

$T' = \quad \forall s \in S, s' \in S, a \in A, T(s,a,s') = P(\text{heads})T(s,a,s') + P(\text{tails}) \sum_{a' \in A} \frac{1}{|A|} T(s,a',s')$
Here $|A|$ denotes the number of elements in the set A, i.e., the number of actions.

$R' = R$

$\gamma' = \gamma$

$s'_0 = s'_0$

Q4. [8 pts] Search

(a) [4 pts] The following implementation of graph search may be incorrect. Circle all the problems with the code.

```
function GRAPH-SEARCH(problem, fringe)
    closed ← an empty set,
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop
        if fringe is empty then
            return failure
        end if
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem,STATE[node]) then
            return node
        end if
        ADD STATE[node] TO closed
        fringe ← INSERTALL(EXPAND(node, problem), fringe)
    end loop
end function
```

1. $\boxed{\text{Nodes may be expanded twice.}}$
2. $\boxed{\text{The algorithm is no longer complete.}}$
3. The algorithm could return an incorrect solution.
4. None of the above.

(b) [4 pts] The following implementation of A* graph search may be incorrect. You may assume that the algorithm is being run with a consistent heuristic. Circle all the problems with the code.

```
function A*-SEARCH(problem, fringe)
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop
        if fringe is empty then
            return failure
        end if
        node ← REMOVE-FRONT(fringe)
        if STATE[node] IS NOT IN closed then
            ADD STATE[node] TO closed
            for successor IN GETSUCCESSORS(problem, STATE[node]) do
                fringe ← INSERT(MAKE-NODE(successor), fringe)
                if GOAL-TEST(problem,successor) then
                    return successor
                end if
            end for
        end if
    end loop
end function
```

1. Nodes may be expanded twice.
2. The algorithm is no longer complete.
3. $\boxed{\text{The algorithm could return an incorrect solution.}}$
4. None of the above.

Q5. [14 pts] Probability

(a) [3 pts] Consider the random variables A, B, and C. Circle all of the following equalities that are **always** true, if any.

1. $\mathbf{P}(A,B) = \mathbf{P}(A)\mathbf{P}(B) - \mathbf{P}(A|B)$

2. $\mathbf{P}(A,B) = \mathbf{P}(A)\mathbf{P}(B)$

3. $\mathbf{P}(A,B) = \mathbf{P}(A|B)\mathbf{P}(B) + \mathbf{P}(B|A)\mathbf{P}(A)$

4. $\boxed{\mathbf{P}(A) = \sum_{b \in B} \mathbf{P}(A|B=b)\mathbf{P}(B=b)}$

5. $\mathbf{P}(A,C) = \sum_{b \in B} \mathbf{P}(A|B=b)\mathbf{P}(C|B=b)\mathbf{P}(B=b)$

6. $\boxed{\mathbf{P}(A,B,C) = \mathbf{P}(C|A)\mathbf{P}(B|C,A)\mathbf{P}(A)}$

Now assume that A and B both can take on only the values true and false ($A \in \{\text{true}, \text{false}\}$ and $B \in \{\text{true}, \text{false}\}$). You are given the following quantities:

$$\begin{aligned} \mathbf{P}(A = \text{true}) &= \tfrac{1}{2} \\ \mathbf{P}(B = \text{true} \mid A = \text{true}) &= 1 \\ \mathbf{P}(B = \text{true}) &= \tfrac{3}{4} \end{aligned}$$

(b) [3 pts] What is $\mathbf{P}(B = \text{true} \mid A = \text{false})$?

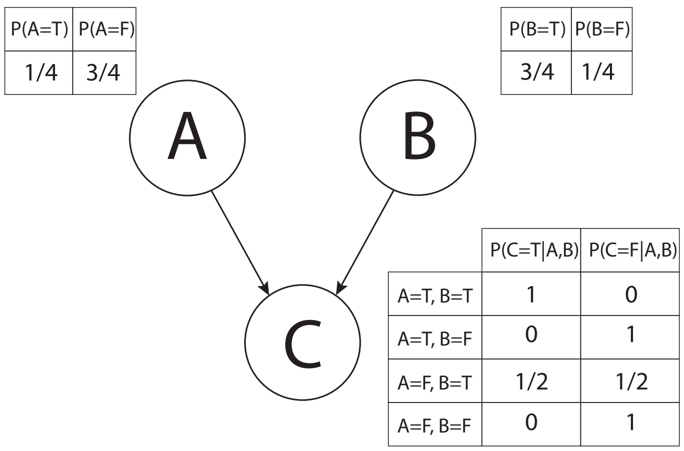
Many people got lost trying to directly apply Bayes' rule. The simplest way to solve this is to realize that

$$\mathbf{P}(B = \text{true}) = \mathbf{P}(B = \text{true} \mid A = \text{true})\mathbf{P}(A = \text{true}) + \mathbf{P}(B = \text{true} \mid A = \text{false})\mathbf{P}(A = \text{false}).$$

Using this fact, you can solve for $\mathbf{P}(B = \text{true} \mid A = \text{false})$:

$$\begin{aligned} (1)\left(\frac{1}{2}\right) + \mathbf{P}(B = \text{true} \mid A = \text{false})\left(\frac{1}{2}\right) &= \frac{3}{4} \\ \implies \mathbf{P}(B = \text{true} \mid A = \text{false})\left(\frac{1}{2}\right) &= \frac{1}{4} \\ \implies \mathbf{P}(B = \text{true} \mid A = \text{false}) &= \frac{1}{2} \end{aligned}$$

Therefore $\mathbf{P}(B = \text{true} \mid A = \text{false}) = \frac{1}{2}$.



P(A=T)	P(A=F)
1/4	3/4

P(B=T)	P(B=F)
3/4	1/4

	P(C=T|A,B)	P(C=F|A,B)
A=T, B=T	1	0
A=T, B=F	0	1
A=F, B=T	1/2	1/2
A=F, B=F	0	1

(c) [2 pts] Give the formula for the joint probability distribution induced by the above Bayes Net:

$$\mathbf{P}(A,B,C) = P(A)P(B)P(C|A,B)$$

Compute the values of the following probabilities:

(d) [2 pts]

$$\mathbf{P}(C=T) = \sum_{A,B} P(A)P(B)P(C=T|A,B) = \frac{1}{4}\frac{3}{4}1 + \frac{1}{4}\frac{1}{4}0 + \frac{3}{4}\frac{3}{4}\frac{1}{2} + \frac{3}{4}\frac{1}{4}0 = \frac{15}{32}$$

(e) [2 pts]

$$\mathbf{P}(A=T,B=T) = \sum_{C} P(A=T)P(B=T)P(C|A=T,B=T) = \frac{1}{4}\frac{3}{4}(1+0) = \frac{3}{16}$$

(f) [2 pts]

$$\mathbf{P}(A=T,B=T|C=T) = \frac{P(A=T,B=T,C=T)}{P(C=T)} = \frac{P(A=T)P(B=T)P(C=T|A=T,B=T)}{P(C=T)} = (\tfrac{1}{4}\tfrac{3}{4}1)/\tfrac{15}{32} = \tfrac{2}{5}$$

Q6. [13 pts] Crossword Puzzles as CSPs

You are developing a program to automatically solve crossword puzzles, because you think a good income source for you might be to submit them to the New York Times ($200 for a weekday puzzle, $1000 for a Sunday).[1] For those unfamiliar with crossword puzzles, a crossword puzzle is a game in which one is given a grid of squares that must be filled in with intersecting words going from left to right and top to bottom. There are a given set of starting positions for words (in the grid below, the positions $1, 2, 3, 4,$ and 5), where words must be placed going across (left to right) or down (top to bottom). At any position where words intersect, the letters in the intersecting words must match. Further, no two words in the puzzle can be identical. An example is the grid below, in which the down words (1, 2, and 3) are `DEN`, `ARE`, and `MAT`, while the across words (1, 4, and 5) are `DAM`, `ERA`, and `NET`.

Example Crossword Grid and Solution

^{1}D	^{2}A	^{3}M
^{4}E	R	A
^{5}N	E	T

A part of your plan to make crosswords, you decide you will create a program that uses the CSP solving techniques you have learned in CS 188, since you want to make yourself obsolete at your own job from the get-go. Your first task is to choose the representation of your problem. You start with a dictionary of all the words you could put in the crossword puzzle, where the dictionary is of size K and consists of the words $\{d_1, d_2, \ldots, d_K\}$. Assume that you are given a grid with N empty squares and M different entries for words (and there are 26 letters in the English language). In the example above, $N = 9$ and $M = 6$ (three words across and three words down).

You initially decide to use words as the variables in your CSP. Let D_1 denote the first down word, D_2 the second, D_3 the third, etc., and similarly let A_k denote the kth across word. For example, in the crossword above, $A_1 =$ `DAM`, $D_1 =$ `DEN`, $D_2 =$ `ARE`, and so on. Let $D_1[i]$ denote the letter in the ith position of the word D_1.

(a) [1 pt] What is the size of the state space for this CSP?

Several answers are acceptable for this problem. The simplest is that the dictionary has size K and there are M words, giving state space size K^M. A slightly tighter bound is achieved by noting that once one word is placed, the next words must all be different, giving $K(K-1)(K-2)\cdots(K-M+1) = \frac{K!}{(K-M)!}$. Noticing that we are choosing M distinct words out of a possible K gives the state space bound $\binom{K}{M}$.

Several students tried to include N in their answers; since the letters have nothing to do with this formulation of the problem, this was incorrect. Many students also incorrectly had M^K.

(b) [3 pts] Precisely (i.e. use mathematical notation to) describe the constraints of the CSP when we use words as variables.

For every pair of across and down words D_k and A_l that intersect, we have the constraint that their letters are equal. Specifically, if they intersect in positions i and j, we have $D_k[i] = A_l[j]$.

We also have the pairwise constraints that none of the words are the same: for $k \neq k'$, $D_k \neq D_{k'}$ and $A_k \neq A_{k'}$, and for all k, k', we have $A_k \neq D_{k'}$.

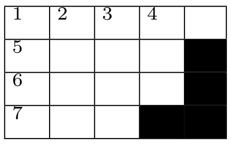
In addition, each word must have the correct length. One possible formulation is that for all $L \in \mathbb{N}$, for all words D_k and A_l with length L in the puzzle, we have $\text{length}(D_k) = L$ and $\text{length}(A_l) = L$.

The biggest problem that students had was assuming that all crossword puzzles were contiguous squares (or rectangles) like the example. While that works for the above example, it will not work generally. Several students missed one or two of the above constraints, and all three were necessary for full credit. Minor mistakes included missing a few of the inequality constraints.

[1] `http://www.nytimes.com/2009/07/19/business/media/19askthetimes.html`

After defining your CSP, you decide to go ahead and make a small crossword using the grid below. Assume that you use the words on the right as your dictionary.

Crossword Grid



Dictionary Words

ARCS, BLAM, BEAR, BLOGS, LARD, LARP,
GAME, GAMUT, GRAMS, GPS, MDS, ORCS, WARBLER

(c) [1 pt] Enforce all *unary* constraints by crossing out values in the table below.

D_1	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_2	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_3	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_4	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	BLOGS	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	GAMUT	GRAMS	~~ORCS~~	~~WARBLER~~
A_5	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_6	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_7	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~

Here's an extra table in case you make a mistake:

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

(d) [1 pt] Assume that in backtracking search, we assign A_1 to be GRAMS. Enforce unary constraints, and in addition, cross out all the values eliminated by forward checking against A_1 as a result of this assignment.

D_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_2	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_3	ARCS	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_4	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	BLOGS	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	GAMUT	$\boxed{\texttt{GRAMS}}$	~~ORCS~~	~~WARBLER~~
A_5	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_6	ARCS	BLAM	BEAR	~~BLOGS~~	LARD	LARP	~~GPS~~	~~MDS~~	GAME	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
A_7	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~

Here's an extra table in case you make a mistake:

D_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_2	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_3	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
D_4	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_1	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_5	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_6	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER
A_7	ARCS	BLAM	BEAR	BLOGS	LARD	LARP	GPS	MDS	GAME	GAMUT	GRAMS	ORCS	WARBLER

(e) [3 pts] Now let's consider how much arc consistency can prune the domains for this problem, even when no assignments have been made yet. I.e., assume no variables have been assigned yet, enforce unary constraints first, and then enforce arc consistency by crossing out values in the table below.

D_1	~~ARCS~~	BLAM	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_2	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	LARD	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
D_3	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	ORCS	~~WARBLER~~
D_4	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	GPS	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_1	~~ARCS~~	~~BLAM~~	~~BEAR~~	BLOGS	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_5	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	LARP	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_6	ARCS	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	~~MDS~~	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~
A_7	~~ARCS~~	~~BLAM~~	~~BEAR~~	~~BLOGS~~	~~LARD~~	~~LARP~~	~~GPS~~	MDS	~~GAME~~	~~GAMUT~~	~~GRAMS~~	~~ORCS~~	~~WARBLER~~

The common mistake in this question was to leave a few blocks of words that students thought could not be eliminated. Probably the most common was to allow both LARD and LARP for D_2 and A_5. This is incorrect; for D_2, no assignment of A_7 is consistent with LARP, and for A_5, no assignment of D_4 is consistent with LARD.

(f) [1 pt] How many solutions to the crossword puzzle are there? Fill them (or the single solution if there is only one) in below.

1 B	2 L	3 O	4 G	S
5 L	A	R	P	■
6 A	R	C	S	■
7 M	D	S	■	■

1	2	3	4	
5				■
6				■
7			■	■

1	2	3	4	
5				■
6				■
7			■	■

There is one solution (above)

Your friend suggests using letters as variables instead of words, thinking that sabotaging you will be funny. Starting from the top-left corner and going left-to-right then top-to-bottom, let X_1 be the first letter, X_2 be the second, X_3 the third, etc. In the very first example, $X_1 = \texttt{D}$, $X_2 = \texttt{A}$, and so on.

(g) [1 pt] What is the size of the state space for this formulation of the CSP?

26^N. There are 26 letters and N possible positions.

(h) [2 pts] Assume that in your implementation of backtracking search, you use the least constraining value heuristic. Assume that X_1 is the first variable you choose to instantiate. For the crossword puzzle used in parts (c)-(f), what letter(s) might your search assign to X_1?

We realized that this question was too vague to be answered correctly, so we gave everyone 2 points for the problem. The least constraining value heuristic, once a variable has been chosen, assigns the value that according to some metric (chosen by the implementer of the heuristic) leaves the domains of the remaining variables most open. How one eliminates values from the domains of other variables upon an assignment can impact the choice of the value as well (whether one uses arc consistency or forward checking).

We now sketch a solution to the problem assuming we use forward checking. Let $X_1, X_2, \ldots, X_5$ be the letters in the top row of the crossword and X_1, X_6, X_7, X_8 be the first column down. Upon assigning $X_1 = \texttt{G}$, the possible domains for the remaining letters are

$$X_2 \in \{\texttt{A},\texttt{R}\}, X_3 \in \{\texttt{M},\texttt{A}\}, X_4 \in \{\texttt{U},\texttt{M}\}, X_5 \in \{\texttt{T},\texttt{S}\}, X_6 \in \{\texttt{A}\}, X_7 \in \{\texttt{M}\}, X_8 \in \{\texttt{E}\}.$$

Upon assigning $X_1 = \texttt{B}$, the possible domains remaining are

$$X_2 \in \{\texttt{L}\}, X_3 \in \{\texttt{O}\}, X_4 \in \{\texttt{G}\}, X_5 \in \{\texttt{S}\}, X_6 \in \{\texttt{L},\texttt{E}\}, X_7 \in \{\texttt{A}\}, X_8 \in \{\texttt{M},\texttt{R}\}.$$

The remaining variables are unaffected since we are using only forward checking. Now, we see that with the assignment $X_1 = \texttt{G}$, the minimum size remaining for any domain is 1, while the sum of the sizes remaining domains is 11; for $X_1 = \texttt{B}$, the minimum size is 1, while the sum of the sizes remaining is 9. So depending on whether we use minimum domain or the sum of the sizes of the remaining domains, the correct solutions are G and B or only G, respectively.

Any choice but $X_1 = \texttt{B}$ or $X_1 = \texttt{G}$ will eliminate all values for one of the other variables after forward checking.

Q7. [33 pts] Short Answer

Each true/false question is worth 1 point. Leaving a question blank is worth 0 points. **Answering incorrectly is worth -1 point.**

(a) Assume we are running A^* graph search with a consistent heuristic h. Assume the optimal cost path to reach a goal has a cost c^*. Then we have that

- **(i)** [*true* or [✓ *false*]] All nodes n reachable from the start state satisfying $g(n) < c^*$ will be expanded during the search.
- **(ii)** [[✓ *true*] or *false*] All nodes n reachable from the start state satisfying $f(n) = g(n) + h(n) < c^*$ will be expanded during the search.
- **(iii)** [*true* or [✓ *false*]] All nodes n reachable from the start state satisfying $h(n) < c^*$ will be expanded during the search.

(b) Running A^* *graph search* with an inconsistent heuristic can lead to suboptimal solutions. Consider the following modification to A^* graph search: replace the closed list with a cost-sensitive closed list, which stores the f-cost of the node along with the state ($f(n) = g(n) + h(n)$).

Whenever the search considers expanding a node, it first verifies whether the node's state is in the cost-senstive closed list and only expands it if either (a) the node's state is not in the cost-sensitive closed list, or (b) the node's state is in the cost-sensitive closed list with a higher f-cost than the f-cost for the node currently considered for expansion.

If a node is expanded because it meets criterion (a), its state and f-cost get added to the cost-sensitive closed list; if it gets expanded because it meets criterion (b), the cost associated with the node's state gets replaced by the current node's f-cost. Which of the following statements are true about the proposed search procedure?

- **(i)** [[✓ *true*] or *false*] The described search procedure finds an optimal solution if h is admissible.
- **(ii)** [[✓ *true*] or *false*] The described search procedure finds an optimal solution if h is consistent.
- **(iii)** [[✓ *true*] or *false*] Assuming h is admissible (but possibly inconsistent), the described search procedure will expand no more nodes than A^* tree search.
- **(iv)** [[✓ *true*] or *false*] Assuming h is consistent, the described search procedure will expand no more nodes than A^* graph search.

(c) Let H_1 and H_2 both be admissible heuristics.

- **(i)** [[✓ *true*] or *false*] $\max(H_1, H_2)$ is necessarily admissible
- **(ii)** [[✓ *true*] or *false*] $\min(H_1, H_2)$ is necessarily admissible
- **(iii)** [[✓ *true*] or *false*] $(H_1 + H_2)/2$ is necessarily admissible
- **(iv)** [*true* or [✓ *false*]] $\max(H_1, H_2)$ is necessarily consistent

(d) Let H_1 be an admissible heuristic, and let H_2 be an inadmissible heuristic.

- **(i)** [*true* or [✓ *false*]] $\max(H_1, H_2)$ is necessarily admissible
- **(ii)** [[✓ *true*] or *false*] $\min(H_1, H_2)$ is necessarily admissible
- **(iii)** [*true* or [✓ *false*]] $(H_1 + H_2)/2$ is necessarily admissible
- **(iv)** [*true* or [✓ *false*]] $\max(H_1, H_2)$ is necessarily consistent

(e) For Markov Decisions Processes (MDPs), we have that:

- **(i)** [[✓ *true*] or *false*] A small discount (close to 0) encourages shortsighted, greedy behavior.
- **(ii)** [[✓ *true*] or *false*] A large, negative living reward ($\ll 0$) encourages shortsighted, greedy behavior.
- **(iii)** [*true* or [✓ *false*]] A negative living reward can always expressed using a discount < 1.
- **(iv)** [*true* or [✓ *false*]] A discount < 1 can always be expressed as a negative living reward.

(f) You are given a game-tree for which you are the maximizer, and in the nodes in which you don't get to make a decision an action is chosen uniformly at random amongst the available options. Your objective is to maximize the probability you win \$10 or more (rather than the usual objective to maximize your expected value). Then:

- **(i)** [*true* or [✓ *false*]] Running expectimax will result in finding the optimal strategy to maximize the probability of winning \$10 or more.
- **(ii)** [*true* or [✓ *false*]] Running minimax, where chance nodes are considered minimizers, will result in finding the optimal strategy to maximize the probability of winning \$10 or more.
- **(iii)** [[✓ *true*] or *false*] Running expectimax in a modified game tree where every pay-off of \$10 or more is given a value of 1, and every pay-off lower than \$10 is given a value of 0 will result in finding the optimal strategy to maximize the probability of winning \$10 or more.
- **(iv)** [*true* or [✓ *false*]] Running minimax in a modified game tree where every pay-off of \$10 or more is given a value of 1, and every pay-off lower than \$10 is given a value of 0 will result in finding the optimal strategy to maximize the probability of winning \$10 or more.

(g) Assume we run $\alpha - \beta$ pruning expanding successors from left to right on a game with tree as shown in Figure 1 (a). Then we have that:

- **(i)** [[✓ *true*] or *false*] For some choice of pay-off values, no pruning will be achieved (shown in Figure 1 (a)).
- **(ii)** [[✓ *true*] or *false*] For some choice of pay-off values, the pruning shown in Figure 1 (b) will be achieved.
- **(iii)** [*true* or [✓ *false*]] For some choice of pay-off values, the pruning shown in Figure 1 (c) will be achieved.
- **(iv)** [*true* or [✓ *false*]] For some choice of pay-off values, the pruning shown in Figure 1 (d) will be achieved.
- **(v)** [*true* or [✓ *false*]] For some choice of pay-off values, the pruning shown in Figure 1 (e) will be achieved.
- **(vi)** [*true* or [✓ *false*]] For some choice of pay-off values, the pruning shown in Figure 1 (f) will be achieved.

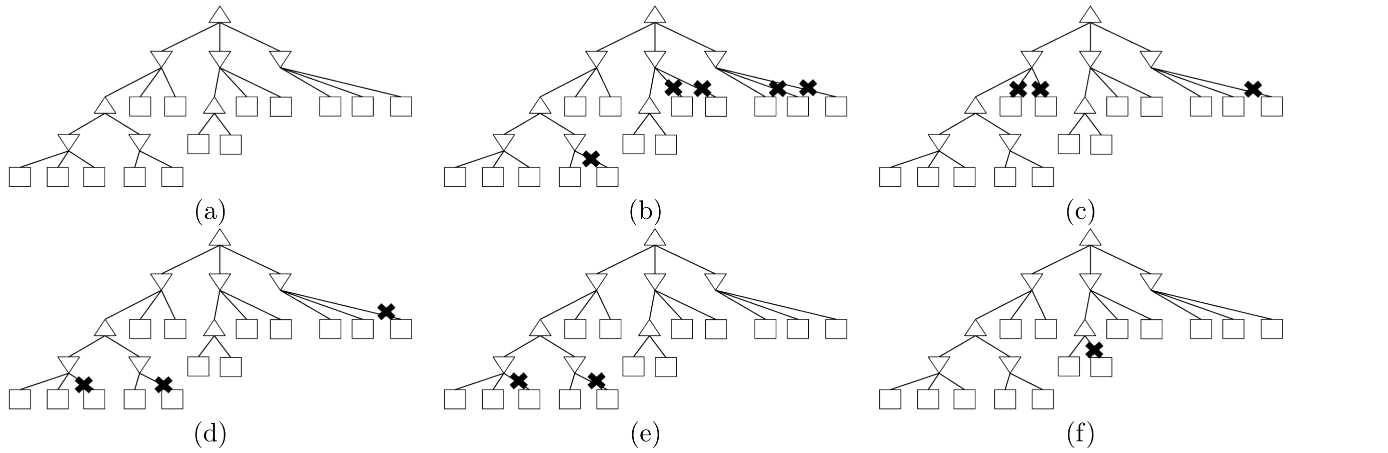


Figure 1: Game trees.

(h) Assume a probability distribution P over binary random variables A, B, C is given. Assume also a probability distribution Q is given which is defined by the Bayes net shown in Figure , with conditional probability tables such that $Q(A) = P(A)$, $Q(B|A) = P(B|A)$, and $Q(C|A) = P(C|A)$. Then we have that:

- **(i)** [[✓ *true*] or *false*] $\forall a, \quad Q(A = a) = P(A = a)$
- **(ii)** [[✓ *true*] or *false*] $\forall b, \quad Q(B = b) = P(B = b)$
- **(iii)** [[✓ *true*] or *false*] $\forall c, \quad Q(C = c) = P(C = c)$
- **(iv)** [*true* or [✓ *false*]] $\forall a, b, c \;\; Q(A = a, B = b, C = c) = P(A = a, B = b, C = c)$

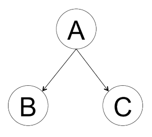


Figure 2: Bayes net