CS 188 Spring 2009 — Introduction to Artificial Intelligence — Midterm Solutions

1. (16 points) True/False

For the following questions, a correct answer is worth 2 points, no answer is worth 1 point, and an incorrect answer is worth 0 points. Circle *true* or *false* to indicate your answer.

a) (*true* or *false*) If $g(s)$ and $h(s)$ are two admissible A* heuristics, then their average $f(s) = \frac{1}{2}g(s) + \frac{1}{2}h(s)$ must also be admissible.

True. Let $h^*(s)$ be the true distance from s. We know that $g(s) \leq h^*(s)$ and $h(s) \leq h^*(s)$, thus $f(s) = \frac{1}{2}g(s) + \frac{1}{2}h(s) \leq \frac{1}{2}h^*(s) + \frac{1}{2}h^*(s) = h^*(s)$

b) (*true* or *false*) For a search problem, the path returned by uniform cost search may change if we add a positive constant C to every step cost.

True. Consider that there are two paths from the start state (S) to the goal (G), $S \rightarrow A \rightarrow G$ and $S \rightarrow G$. $cost(S, A) = 1$, $cost(A, G) = 1$, and $cost(S, G) = 3$. So the optimal path is through A. Now, if we add 2 to each of the costs, the optimal path is directly from S to G. Since uniform cost search finds the optimal path, its path will change.

c) (*true* or *false*) The running-time of an efficient solver for tree-structured constraint satisfaction problems is linear in the number of variables.

True. The running time of the algorithm for tree-structured CSPs is $O(n \cdot d^2)$, where n is the number of variables and d is the maximum size of any variable's domain.

d) (*true* or *false*) If $h_1(s)$ is a consistent heuristic and $h_2(s)$ is an admissible heuristic, then $\min(h_1(s), h_2(s))$ must be consistent.

False. For instance, if $h_2(s)$ be admissible but inconsistent, and $h_1(s)$ dominate $h_2(s)$, then $\min(h_1(s), h_2(s)) = h_1(s)$, which is inconsistent.

e) (*true* or *false*) The amount of memory required to run minimax with alpha-beta pruning is $O(b^d)$ for branching factor b and depth limit d.

True and **False** (everyone wins). The memory required is only $O(bd)$, so we accepted **False**. However, by definition an algorithm that is $O(bd)$ is also $O(b^d)$, because O denotes upper bounds that may or may not be tight, so technically this statement is *True* (but not very useful).

f) (*true* or *false*) In a Markov decision process with discount $\gamma = 1$, the difference in values for two adjacent states is bounded by the reward between them: $|V(s) - V(s')| \leq \max_a R(s, a, s')$.

False. Let $V(s') = 0$, and $R(s, a, s') = 0 \ \forall a$, but there is an action a' which takes s to the terminal state T and gives a reward 100. Thus $V(s) \geq 100$, but the inequality above says that $|V(s) - 0| \leq 0$.

g) (*true* or *false*) Value iteration and policy iteration must always converge to the same policy.

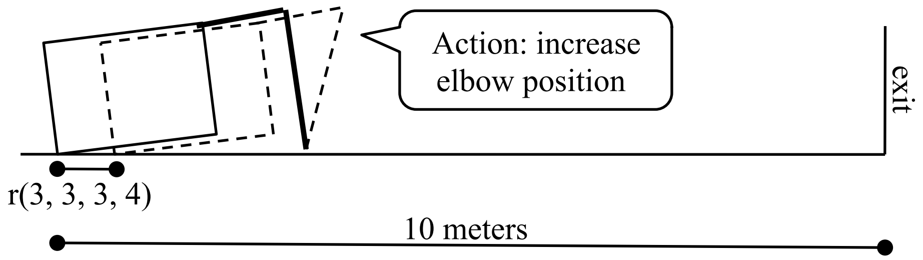
True and **False** (everyone wins). Both algorithms are guaranteed to converge to the optimal policy, so we accepted **True**. If there are multiple policies that are optimal (meaning they yield the same maximal values for every state), then the algorithms might diverge. Both value iteration and policy iteration will always lead to the same optimal values.

h) (*true* or *false*) In a Bayes' net, if $A \perp\!\!\!\perp B$, then $A \perp\!\!\!\perp B \mid C$ for some variable C other than A or B.

False. Consider the Bayes' net $A \rightarrow C \leftarrow B$. Clearly, $A \perp\!\!\!\perp B$, but $A \not\perp\!\!\!\perp B \mid C$.

2. (15 points) Search: Crawler's Escape

Whilst Pacman was Q-learning, Crawler snuck into `mediumClassic` and stole all the dots. Now, it's trying to escape as quickly as possible. At each time step, Crawler can *either* move its *s*houlder or its *e*lbow one position up or down. Both joints s and e have five total positions each (1 through 5) and both begin in position 3. Upon changing arm positions from (s, e) to (s', e'), Crawler's body moves some distance $r(s, e, s', e')$, where $|r(s, e, s', e')| \leq 2$ meters (negative distance means the crawler moves backwards). Crawler must travel 10 meters to reach the exit.



In this problem, you will design a search problem for which the optimal solution will allow Crawler to escape in as few time steps as possible.

(a) **(3 pt)** Define a state space, start state and goal test to represent this problem.

A state is a 3-tuple consisting of distance to exit, shoulder position, and elbow position.

The start state is $(10, 3, 3)$.

Goal test: distance to exit is less than or equal to zero.

(b) **(3 pt)** Define the successor function for the start state by listing its (successor state, action, step cost) triples. Use the actions s_+ and s_- for increasing and decreasing the shoulder position and e_+ and e_- for the elbow.

$$\begin{aligned}\{ \quad & ((10 - r(3,3,4,3), 4, 3), s_+, 1), \\ & ((10 - r(3,3,2,3), 2, 3), s_-, 1), \\ & ((10 - r(3,3,3,4), 3, 4), e_+, 1), \\ & ((10 - r(3,3,3,2), 3, 2), e_-, 1) \ \}\end{aligned}$$

(c) (2 pt) Would depth-first graph search be complete for the search problem you defined? Explain.

No, depth-first search would not be complete because the state space is infinite. Depth-first search could forever keep expanding states with increasing distance to the goal.

(d) (3 pt) Design a non-trivial, admissible heuristic for this problem.

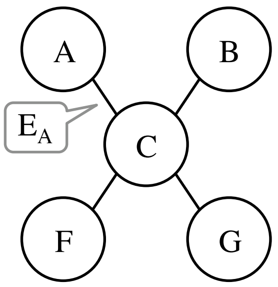
Distance to exit divided by 2 is admissible because each step can move Crawler at most 2 meters closer to the goal. A slightly better heuristic would be to round this heuristic up to the nearest integer.

(e) (4 pt) Crawler's shoulder overheats if it switches direction more than once in any three consecutive time steps, making progress impossible. Describe how you would change your search problem so that an optimal search algorithm would only return solutions that avoid overheating.

Consider a reversal to be s_+ followed by s_- or s_- followed by s_+. To avoid these, we can extend the state space to include the last two actions. We then remove from the output of the successor function any triple that includes the action sequences (s_+, s_-, s_+) or (s_-, s_+, s_-). We also accepted solutions that gave a very high step cost to these successors.

3. (20 points) CSPs: Constraining Graph Structure

Hired as a consultant, you built a Bayes' net model of Cheeseboard Pizza customers. Last night, a corporate spy from Zachary's Pizza deleted the *directions* from all your arcs. You still have the undirected graph of your model and a list of dependencies. You now have to recover the direction of the arrows to build a graph that allows for these dependencies. *Note:* $X \not\perp\!\!\!\perp Y$ *means* X *is not independent of* Y.



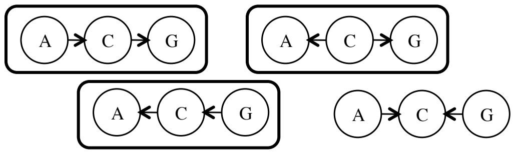
Distribution properties (dependencies):

$A \not\perp\!\!\!\perp G$

$B \not\perp\!\!\!\perp F$

$B \not\perp\!\!\!\perp G \mid C$

a) (1 pt) Given the first constraint only, circle all the topologies that are allowed for the triple (A, C, G).



b) (3 pt) Formulate direction-recovery for this graph as a CSP with *explicit binary constraints* only. The variables and values are provided for you. The variable E_A stands for the direction of the arc between A and the center C, where *out* means the arrow points outward (toward A), and *in* means the arrow points inward (toward C).

Variables:	E_A, E_B, E_F, E_G
Values:	*out*, *in*
Constraints:	

An explicit constraint lists all legal tuples of values are listed for a tuple of variables.

$$\begin{aligned}(E_A, E_G) &\in \{(in, out), (out, out), (out, in)\}\\ (E_B, E_F) &\in \{(in, out), (out, out), (out, in)\}\\ (E_B, E_G) &\in \{(in, in)\}\end{aligned}$$

c) (1 pt) Draw the constraint graph for this CSP.
$E_A - E_G - E_B - E_F$

d) (2 pt) After selecting $E_A = in$, cross out all values eliminated from E_B, E_F and E_G by forward checking.

E_A	E_B	E_F	E_G
in	*out in*	*out in*	*out* ~~*in*~~

Forward checking removes the values of variables that directly contradict the assignment $E_A = in$.

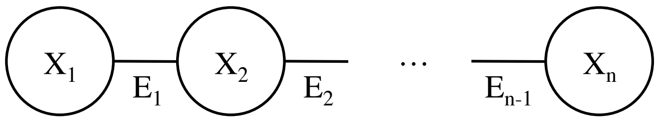
e) (2 pt) Cross out all values eliminated by arc consistency applied before any backtracking search.

E_A	E_B	E_F	E_G
out ~~*in*~~	~~*out*~~ *in*	*out* ~~*in*~~	~~*out*~~ *in*

We first remove $E_B = out$ *because it is incompatible with any value for* E_G. *Likewise, we remove* $E_G = out$. *Now, we can remove* $E_A = in$ *based on* E_G *and* $E_F = in$ *based on* E_B.

f) (1 pt) Solve this CSP, then add the correct directions to the arcs of the graph at the top of the page.
$E_A = out$, $E_B = in$, $E_F = out$, and $E_G = in$.

Now consider a chain structured graph over variables $X_1, \dots, X_n$. Edges $E_1, \dots, E_{n-1}$ connect adjacent variables, but their directions are again unknown.



g) **(4 pt)** Using only *binary* constraints and *two-valued* variables, formulate a CSP that is satisfied by *only and all* networks that can represent a distribution where $X_1 \not\perp\!\!\!\perp X_n$. Describe what your variables mean in terms of direction of the arrows in the network.

Variables:

Variables $E_1, \dots, E_{n-1}$ correspond to the directions of these edges and take values $\{left, right\}$.

Constraints:

$(E_i, E_{i+1}) \neq (right, left)$, for $i \in \{0, \dots, n-2\}$.

h) **(6 pt)** Using only *unary, binary and ternary* (3 variable) constraints and *two-valued* variables, formulate a CSP that is satisfied by *only and all* networks that enforce $X_1 \perp\!\!\!\perp X_n$. Describe what your variables mean in terms of direction of the arrows in the network. *Hint: You will have to introduce additional variables.*

Variables:

Variables $E_1, \dots, E_{n-1}$ correspond to the directions of these edges and take values $\{left, right\}$.
$I_1, \dots, I_{n-2}$ can take values $\{T, F\}$.
$O_1, \dots, O_{n-2}$ can take values $\{T, F\}$.

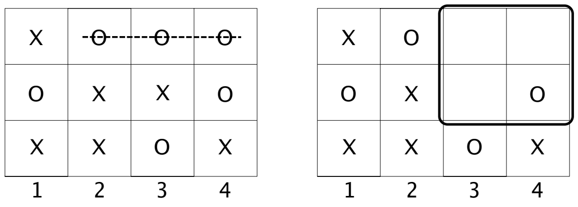
Constraints:

Intuitively, we want I_i to be T if $X_i \rightarrow X_{i+1} \leftarrow X_{i+2}$, i.e. the triple X_i, X_{i+1}, X_{i+2} is inactive. Also, we want O_i to be T if either I_i or O_{i-1} is T, i.e. if the current triple is inactive or if any of the prior triples was inactive. Finally, we want O_{n-2} to be T which would imply that there is some inactive triple.

$(I_i, E_i, E_{i+1}) \in \{(T, right, left), (F, right, right), (F, left, right), (F, left, left)\}$ for $i \in \{1, \dots, n-2\}$.
$(I_1, O_1) \in \{(T, T), (F, F)\}$
$(O_{i-1}, I_i, O_i) \in \{(F, F, F), (F, T, T), (T, F, T), (T, T, T)\}$ for $i \in \{2, \dots, n-2\}$
$O_{n-2} = true$

4. (10 points) Multi-agent Search: Connect-3

In the game of Connect-3, players X and O alternate moves, dropping their symbols into columns 1, 2, 3, or 4. Three-in-a-row wins the game, horizontally, vertically or diagonally. X plays first.



(a) (1 pt) What is the maximum branching factor of minimax search for Connect-3?
4.

(b) (1 pt) What is the maximum tree depth in plies?
6. According to our terminology, a search ply in a multi-agent search problem includes one move for every agent. Because the textbook differs in its definition, we also accepted 12.

(c) (1 pt) Give a reasonably tight upper bound on the number of terminal states. Several answers were accepted:

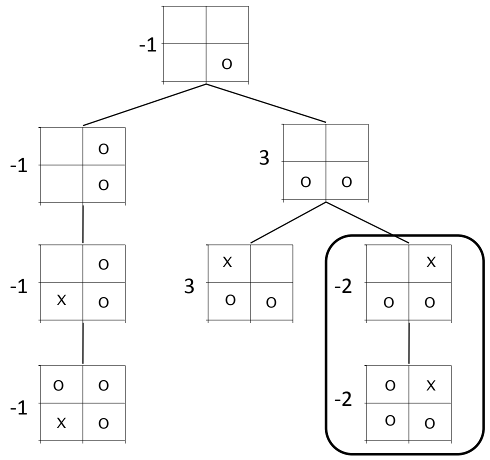
4^{12} is a reasonable bound on the number of leaves of the search tree.

3^{12} is a reasonable bound on the number of total states in the game, because each square can be empty, contain an X or contain an O.

A tighter bound is 15^4, which follows from the observation that there are only 15 ways to legally fill a column from the bottom up with X's and O's.

Bounds based on the assumption that the entire board would be filled with X's and O's were not accepted, even if they in fact overestimated the true number of terminal states.

(d) (2 pt) Draw the game tree starting from the board shown above right (with O to play next). You may abbreviate states by drawing only the upper-right region circled. The root node is drawn for you.

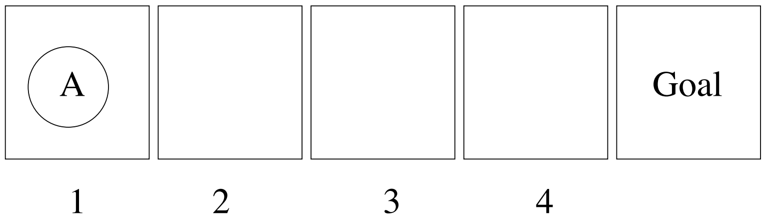


(e) (2 pt) X is the maximizer, while O is the minimizer. X's utility for terminal states is k when X wins and $-k$ when O wins, where k is 1 for a horizontal 3-in-a-row win (as in above left), 2 for a vertical win, and 3 for a diagonal win. A tie has value 0. Label each node of your tree with its minimax value.

(f) (3 pt) Circle all nodes of your tree that will *not* be explored when using alpha-beta pruning and a move ordering that maximizes the number of nodes pruned.

5. (21 points) MDPs: Robot Soccer

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.



In this MDP, the states are 1, 2, 3, 4, G and M, where G and M are terminal states. The transition model depends on the parameter y, which is the probability of dribbling success. Assume a discount of $\gamma = 1$.

$T(k, S, G) = \frac{k}{6}$	$T(k, S, M) = 1 - \frac{k}{6}$	for $k \in \{1, 2, 3, 4\}$
$T(k, D, k+1) = y$	$T(k, D, M) = 1 - y$	for $k \in \{1, 2, 3\}$
$R(k, S, G) = 1$	for $k \in \{1, 2, 3, 4\}$,	and rewards are 0 for all other transitions

(a) (2 pt) What is $V^\pi(1)$ for the policy π that always shoots?
$V^\pi(1) = T(1, S, G)R(1, S, G) + T(1, S, M)R(1, S, M) = \frac{1}{6}$

(b) (2 pt) What is $Q^*(3, D)$ in terms of y?

$$\begin{aligned}
Q^*(3, D) &= T(3, D, 4)(R(3, D, 4) + V^*(4)) + T(3, D, M)R(3, D, M) \\
&= T(3, D, 4)V^*(4) \\
&= T(3, D, 4)Q^*(4, S) \\
&= T(3, D, 4)(T(4, S, G)R(4, S, G) + T(4, S, M)R(4, S, m)) \\
&= T(3, D, 4)T(4, S, G)R(4, S, G) \\
&= \frac{2}{3}y
\end{aligned}$$

(c) (2 pt) Using $y = \frac{3}{4}$, complete the first two iterations of value iteration.

i	$V_i^*(1)$	$V_i^*(2)$	$V_i^*(3)$	$V_i^*(4)$
0	0	0	0	0
1	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{3}$
2	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{2}$	$\frac{2}{3}$

(d) (2 pt) After how many iterations will value iteration compute the optimal values for all states?
After 3 iterations, the values will have converged when $y = \frac{3}{4}$. Above, only $V^*(1)$ has not yet converged. We note that for $y > \frac{3}{4}$, a fourth iteration would be required because a fast break has up to four transitions.

(e) (2 pt) For what range of values of y is $Q^*(3, S) \geq Q^*(3, D)$?

$$\begin{aligned}
Q^*(3, S) &\geq Q^*(3, D) \\
T(3, S, G) \cdot 1 &\geq T(3, D, 4) \cdot T(4, S, G) \cdot 1 \\
\frac{1}{2} &\geq y \cdot \frac{2}{3} \\
\frac{3}{4} &\geq y \geq 0
\end{aligned}$$

The dribble success probability y in fact depend on the presence or absence of a defending robot, **D**. **A** has no way of detecting whether **D** is present, but does know some statistical properties of its environment. **D** is present $\frac{2}{3}$ of the time. When **D** is absent, $y = \frac{3}{4}$. When **D** is present, $y = \frac{1}{4}$.

(f) (4 pt) What is the posterior probability that **D** is present, given that **A** Dribbles twice successfully from 1 to 3, then Shoots from state 3 and scores.

We can use Bayes' rule, where D is a random variable denoting the presence of **D**, and e is the evidence that **A** dribbled twice and scored.

$$P(d|e) = \frac{P(e|d) \cdot P(d)}{P(e)}$$

$$\begin{aligned} P(e) &= P(e|d) \cdot P(d) + P(e|\neg d) \cdot P(\neg d) \\ P(e|d) &= \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} \\ P(e|\neg d) &= \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{2} \\ P(e) &= \frac{1}{32} \cdot \frac{2}{3} + \frac{9}{32} \cdot \frac{1}{3} = \frac{11}{96} \\ P(d|e) &= \frac{2}{96} / \frac{11}{96} = \frac{2}{11} \end{aligned}$$

(g) (3 pt) What transition model should **A** use in order to correctly compute its maximum expected reward when it doesn't know whether or not **D** is present?

To maximize expected total reward, the agent should model the situation as accurately as possible.

We accepted two answers for $T(k, D, k+1)$. One answer is to claim that the agent should use its marginal belief that each dribble will be successful, summing over the two possibilities of **D**'s presence or absense. In this case, $T(k, D, k+1) = \frac{2}{3} \cdot \frac{1}{4} + \frac{1}{3} \cdot \frac{3}{4} = \frac{5}{12}$, and $T(k, D, M) = \frac{7}{12}$.

Another acceptable answer would be to update the beliefs of the agent after every successful dribble. Hence, while $T(1, D, 2) = \frac{5}{12}$ as above, reaching state 2 implies a successful previous dribble, and so another successful dribble is more likely. In particular, let X_1, X_2 and X_3 indicate successful first, second and third dribbles, while D is the presence of the defender. Since $X_1 \perp\!\!\!\perp X_2 \mid D$, we have $P(x_2|x_1) = \sum_d P(d|x_1)P(x_2|d)$. Computing similarly to part (f), $P(d|x_1) = \frac{2}{5}$, so $T(2, D, 3) = \frac{2}{5} \cdot \frac{1}{4} + \frac{3}{5} \cdot \frac{3}{4} = \frac{11}{20}$.

We can compute $T(3, D, 4)$ similarly. X_3 is conditionally independent of both X_1 and X_2 given D, so $P(x_3|x_1, x_2) = \sum_d P(d|x_1, x_2)P(x_3|d)$. Using our result from (f), $P(d|x_1, x_2) = \frac{2}{11}$, and so $T(3, D, 4) = \frac{2}{11} \cdot \frac{1}{4} + \frac{9}{11} \cdot \frac{3}{4} = \frac{29}{44}$.

Since a dribble either succeeds or fails, $T(k, D, M) = 1 - T(k, D, k+1)$ for all k. Shooting probabilities are unchanged by the presence of the defender.

$$\begin{array}{lll} T(k, S, G) = \frac{k}{6} & T(k, S, M) = 1 - \frac{k}{6} & \text{for } k \in \{1, 2, 3, 4\} \\ T(k, D, k+1) = \frac{5}{12} & T(k, D, M) = \frac{7}{12} & \text{for } k \in \{1, 2, 3\} \end{array}$$

OR

$$\begin{array}{lll} T(k, S, G) = \frac{k}{6} & T(k, S, M) = 1 - \frac{k}{6} & \text{for } k \in \{1, 2, 3, 4\} \\ T(1, D, 2) = \frac{5}{12} & T(2, D, 3) = \frac{11}{20} & T(3, D, 4) = \frac{29}{44} \\ T(1, D, M) = \frac{7}{12} & T(2, D, M) = \frac{9}{20} & T(3, D, M) = \frac{15}{44} \end{array}$$

(h) **(4 pt)** What is the optimal policy π^* when **A** doesn't know whether or not **D** is present?

Using $T(k, D, k+1) = \frac{5}{12}$, we find that $\pi^* = \{1 : S, 2 : S, 3 : S, 4 : S\}$. We showed from part (e) that for any $y < \frac{3}{4}$, shooting is preferable to dribbling from state 3. Therefore, we know that $V^*(3) = Q^*(3, S)$. We can perform similar computations for states 1 and 2:

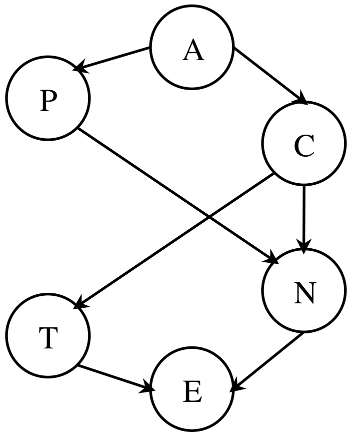
$$\begin{aligned}
V^*(3) &= Q^*(3, S) = \frac{1}{2} \\
\pi^*(3) &= S \\
Q^*(2, S) &= \frac{1}{3} \\
Q^*(2, D) &= \frac{5}{12} \cdot V^*(3) + \frac{7}{12} \cdot 0 = \frac{5}{24} \\
V^*(2) &= \max(Q^*(2, S), Q^*(2, D)) = \frac{1}{3} \\
\pi^*(2) &= S \\
Q^*(1, S) &= \frac{1}{6} \\
Q^*(1, D) &= \frac{5}{12} \cdot V^*(2) + \frac{7}{12} \cdot 0 = \frac{5}{36} \\
V^*(1) &= \max(Q^*(1, S), Q^*(1, D)) = \frac{1}{6} \\
\pi^*(1) &= S
\end{aligned}$$

Under the second answer for part (g), similar computations give $\pi^* = \{1 : S, 2 : S, 3 : S, 4 : S\}$.

$$\begin{aligned}
V^*(4) &= \frac{2}{3} \\
Q^*(3, S) &= \frac{1}{2} \\
Q^*(3, D) &= \frac{29}{44} \cdot V^*(4) + \frac{15}{44} \cdot 0 = \frac{58}{132} \approx 0.44 \\
V^*(3) &= \max(Q^*(3, S), Q^*(3, D)) = \frac{1}{2} \\
\pi^*(3) &= S \\
Q^*(2, S) &= \frac{1}{3} \\
Q^*(2, D) &= \frac{11}{20} \cdot V^*(3) + \frac{9}{20} \cdot 0 = \frac{11}{40} \approx 0.28 \\
V^*(2) &= \max(Q^*(2, S), Q^*(2, D)) = \frac{1}{3} \\
\pi^*(2) &= S \\
Q^*(1, S) &= \frac{1}{6} \\
Q^*(1, D) &= \frac{5}{12} \cdot V^*(2) + \frac{7}{12} \cdot 0 = \frac{5}{36} \\
V^*(1) &= \max(Q^*(1, S), Q^*(1, D)) = \frac{1}{6} \\
\pi^*(1) &= S
\end{aligned}$$

6. (18 points) Bayes' Nets: The Mind of Pacman

Pacman doesn't just eat dots. He also enjoys pears, apples, carrots, nuts, eggs and toast. Each morning, he chooses to eat some subset of these foods. His preferences are modeled by a Bayes' net with this structure.



a) (2 pt) Factor the probability that Pacman chooses to eat only apples and nuts in terms of conditional probabilities from this Bayes' net.

$P(\neg p, a, \neg c, n, \neg e, \neg t) = P(a) \cdot P(\neg p|a) \cdot P(\neg c|a) \cdot P(n|\neg p, \neg c) \cdot P(\neg t|\neg c) \cdot P(\neg e|n, \neg t)$

b) (4 pt) For each of the following properties, circle whether they are *true*, *false* or *unknown* of the distribution $P(P, A, C, N, E, T)$ for this Bayes' net.

Without the conditional probability tables, we do not know if any two variables are dependent. From the network topology, we can only conclude true *or* unknown *for the questions below.*

$N \perp\!\!\!\perp T$	(*true* , *false* , *unknown*)	**unknown**: $T - C - N$ is an active path.
$P \perp\!\!\!\perp E \mid N$	(*true* , *false* , *unknown*)	**unknown**: $P - A - C - T - E$ is an active path.
$P \perp\!\!\!\perp N \mid A, C$	(*true* , *false* , *unknown*)	**unknown**: P and N are directly connected.
$E \perp\!\!\!\perp A \mid C, N$	(*true* , *false* , *unknown*)	**true**: All three paths from E to A are inactive.

c) (2 pt) If the arrow from T to E were reversed, list two conditional independence properties that would be true under the new graph that were not guaranteed under the original graph.

The triple $C - T - E$ is no longer active when T is unknown, therefore
$P \perp\!\!\!\perp E \mid N$ and $A \perp\!\!\!\perp E \mid N$ and $C \perp\!\!\!\perp E \mid N$
The triple $N - E - T$ is no longer active when E is known, therefore
$T \perp\!\!\!\perp N \mid C, E$ and $T \perp\!\!\!\perp P \mid C, E$ and $T \perp\!\!\!\perp A \mid C, E$
There are further independence assumptions based on these 6 that condition on more variables, such as $A \perp\!\!\!\perp E \mid N, P$.

d) (2 pt) You discover that $P(a|\neg c, \neg t) > P(a|\neg c)$. Suggest how you might change the network in response.

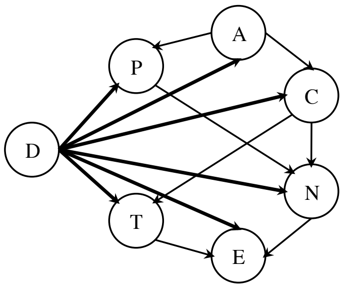
This inequality implies that $A \not\perp\!\!\!\perp T \mid C$ by definition, but in the network above, A and T are conditionally independent given C. So, the network must be changed to allow for this dependence. Adding an arc from A to T would fix the problem. *Note: adding an arc from T to A creates a cycle, which is not allowed in a Bayes' net.* Other acceptable solutions exist, like reversing the direction of the arc from T to C.

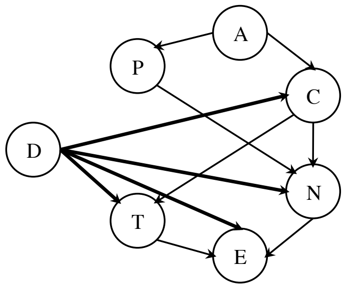
e) (4 pt) You now want to compute the distribution $P(A)$ using variable elimination. List the factors that remain before and after eliminating the variable N.

Before: *The initial factors are just the conditional probability tables of the Bayes' net.*
$P(A)$, $P(P|A)$, $P(C|A)$, $P(T|C)$, $P(N|P,C)$, $P(E|T,N)$

After: *First, all factors that include the variable N are joined together, yielding $P(E,N|P,C,T)$*
Next, the variable N is summed out of this new factor, yielding $P(E|P,C,T)$
The remaining factors include this new factor and the unused original factors.
$P(A)$, $P(P|A)$, $P(C|A)$, $P(T|C)$, $P(E|P,C,T)$
Referring to the final factor as $m(E,P,C,T)$ (like in the textbook) was also accepted.

f) (2 pt) Pacman's new diet allows only fruit (P and A) to be eaten, but Pacman only follows the diet occasionally. Add the new variable D (for whether he follows the diet) to the network below by adding arcs. Briefly justify your answer.





The best answer is the one on the left, which allows us to express changes to all food preferences based on the diet. Other answers were accepted *with appropriate justification*, such as the network on the right, which specifies that the diet specifically disallows C, N, E and T. Connecting D only to P and/or A is problematic: for instance, observing P and A would make D independent of C, but the diet should still affect whether carrots (C) are allowed even when A and P are known.

g) (2 pt) Given the Bayes' net and the factors below, fill in the table for $P(D|\neg a)$ or state that there is not enough information.

From the tables $P(D)$ and $P(A|D)$, the entries of the factor $P(D|A=\neg a)$ can be computed from Bayes' rule (which always holds): no additional information about the properties of the distributions are required. The trick is to observe that while $P(\neg a)$ is not explicitly given, it can be computed from $P(D)$ and $P(A|D)$ using the product rule: $\forall_{a,d} : P(a,d) = P(d)P(a|d)$. Computing $P(\neg a)$ in this way is equivalent to applying the normalization trick.

D	$P(D)$
d	0.4
$\neg d$	0.6

D	A	$P(A\|D)$
d	a	0.8
d	$\neg a$	0.2
$\neg d$	a	0.5
$\neg d$	$\neg a$	0.5

D	A	$P(D\|A=\neg a)$
d	$\neg a$	$\frac{P(\neg a\|d)\cdot P(d)}{P(\neg a\|\neg d)\cdot P(\neg d)+P(\neg a\|d)\cdot P(d)} = \frac{0.2\cdot 0.4}{0.5\cdot 0.6+0.2\cdot 0.4} = \frac{4}{19}$
$\neg d$	$\neg a$	$\frac{P(\neg a\|\neg d)\cdot P(\neg d)}{P(\neg a\|\neg d)\cdot P(\neg d)+P(\neg a\|d)\cdot P(d)} = \frac{0.5\cdot 0.6}{0.5\cdot 0.6+0.2\cdot 0.4} = \frac{15}{19}$