CS 188 Fall 2019 | Introduction to Artificial Intelligence | # Practice Final

- You have approximately 170 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	
SID	
Name of person on your left	
Name of person on your right	

Your Discussion/Exam Prep* TA (fill all that apply):

- ☐ Shizhan (Tu)
- ☐ Carl (Tu)
- ☐ Emma (Tu)
- ☐ Mesut* (Tu)
- ☐ Jesse (Tu)
- ☐ Cathy (Tu)
- ☐ Peyrin* (Tu)
- ☐ Andy (Tu)
- ☐ Wilson (W)
- ☐ Ryan (W)
- ☐ Lindsay (W)
- ☐ Gokul* (W)
- ☐ Rachel (W)
- ☐ Henry* (W)
- ☐ Alan (W)
- ☐ Andreea (W)
- ☐ Chandan (W)
- ☐ Sherman* (W)
- ☐ Mike (W)
- ☐ Danny* (W)
- ☐ Jinkyu (W)
- ☐ Lawrence (W)
- ☐ Albert (W)

Q1.	Game Trees	/9
Q2.	Short Answer	/14
Q3.	Decision Networks and VPI	/9
Q4.	Bayes Net CSPs	/9
Q5.	Probability and Bayes Net Representation	/20
Q6.	Finding Waldo	/12
Q7.	Machine Learning: Potpourri	/12
Q8.	MDPs and RL	/15
	Total	/100

To earn the extra credit, one of the following has to hold true. Please circle and sign.

A I spent 170 or more minutes on the practice midterm.

B I spent fewer than 170 minutes on the practice midterm, but I believe I have solved all the questions.

Signature: ____

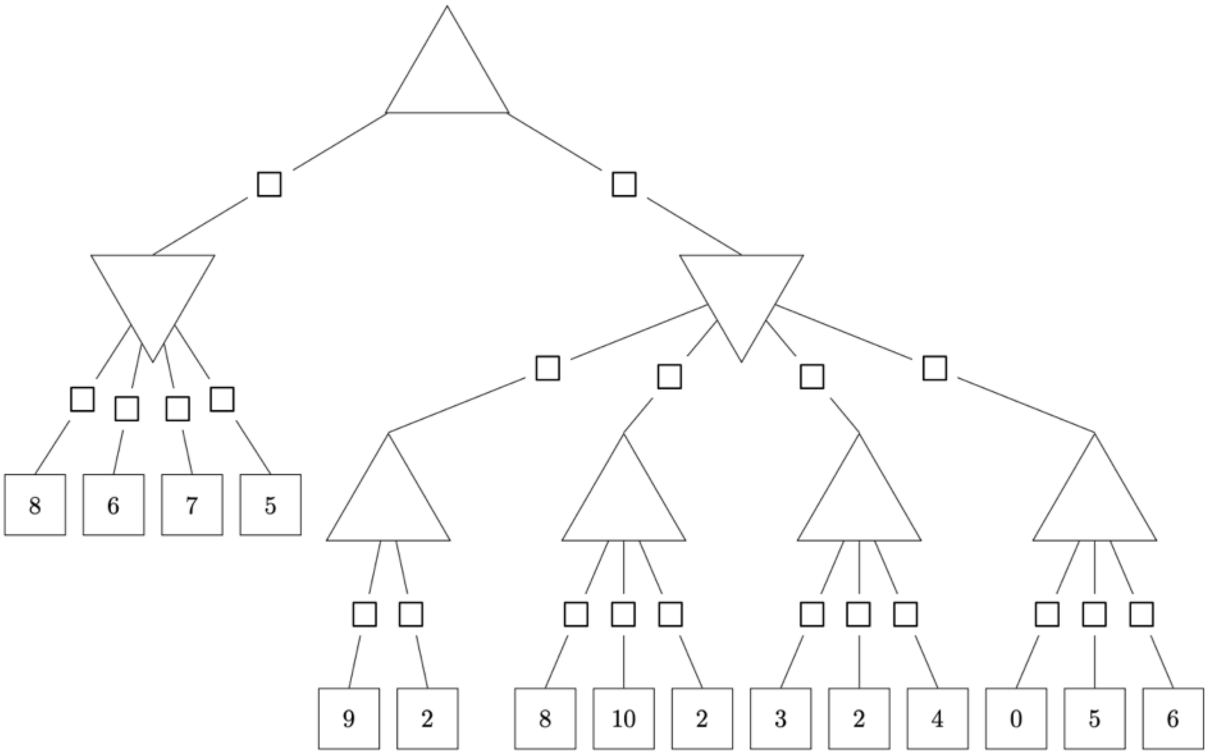
To submit the practice midterm, scan and upload the PDF to Gradescope.

Q1. [9 pts] Game Trees

The following problems are to test your knowledge of Game Trees.

(a) Minimax

The first part is based upon the following tree. Upward triangle nodes are maximizer nodes and downward are minimizers. (small squares on edges will be used to mark pruned nodes in part (ii))



(i) [1 pt] Complete the game tree shown above by filling in values on the maximizer and minimizer nodes.

(ii) [3 pts] Indicate which nodes can be pruned by marking the edge above each node that can be pruned (you do not need to mark any edges below pruned nodes). In the case of ties, please prune any nodes that could not affect the root node's value. Fill in the bubble below if no nodes can be pruned.

○ No nodes can be pruned

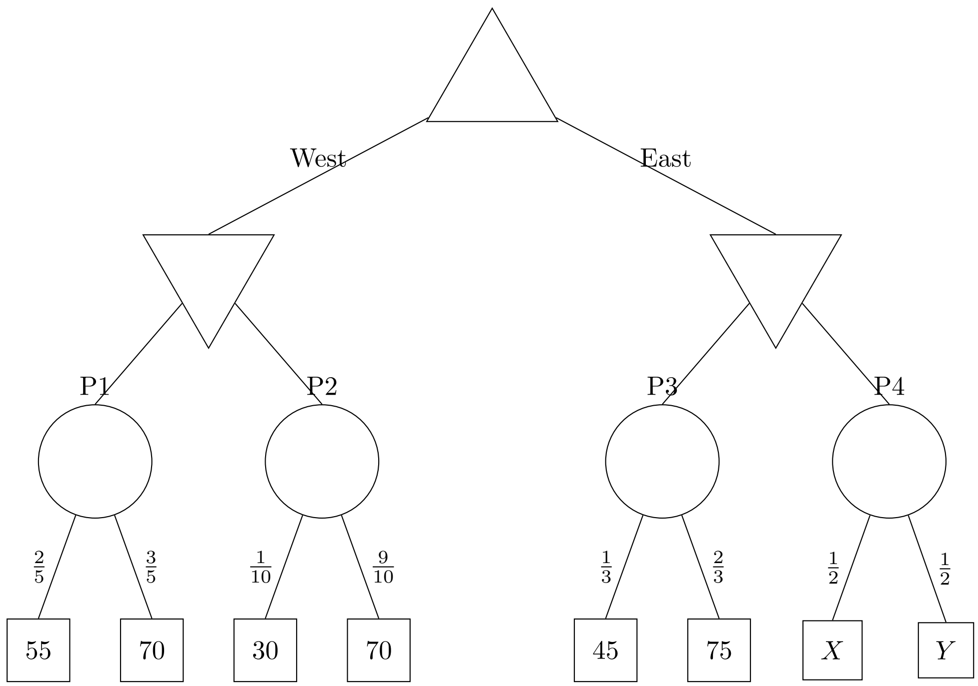
(b) Food Dimensions

The following questions are completely unrelated to the above parts.

Pacman is playing a tricky game. There are 4 portals to food dimensions. But, these portals are guarded by a ghost. Furthermore, neither Pacman nor the ghost know for sure how many pellets are behind each portal, though they know what options and probabilities there are for all but the last portal.

Pacman moves first, either moving West or East. After which, the ghost can block **1** of the portals available.

You have the following gametree. The maximizer node is Pacman. The minimizer nodes are ghosts and the portals are chance nodes with the probabilities indicated on the edges to the food. In the event of a tie, the left action is taken. Assume Pacman and the ghosts play optimally.



(i) [1 pt] Fill in values for the nodes that do not depend on X and Y.

(ii) [4 pts] What conditions must X and Y satisfy for Pacman to move East? What about to definitely reach the P4? Keep in mind that X and Y denote numbers of food pellets and must be **whole numbers**: $X, Y \in \{0, 1, 2, 3, \dots\}$.

To move East:

To reach P4:

SID: ____________________

Q2. [14 pts] Short Answer

(a) [2 pts] You have a pile of P potatoes to eat and B potato-eating bots. At any time, each bot is either a *chopper* or a *devourer*; all begin as choppers. In a given time step, a *chopper* can *chop*, *idle*, or *transform*. If it chops, it will turn 1 potato into a pile of fries. If it is idle, it will do nothing. If it transforms, it will do nothing that time step but it will be a devourer in the next time step. Devourers are hive-like and can only *devour* or *transform*. When D devourers devour, they will consume exactly D^2 piles of fries that time step – but only if at least that many piles exist. If there are fewer piles, nothing will be devoured. If a devourer transforms, it will do nothing that time step but will be a chopper in the next one. The goal is to have no potatoes or fries left. Describe a minimal state space representation for this search problem. You must write down a size expression in terms of the number of potatoes P, the number of total bots B, the number of fries F, the number of time steps elapsed T, and any other quantities you wish to name. For example, you might write $P^B + T$. You may wish to briefly explain what each factor in your answer represents.

State space size: ____________

(b) [4 pts] Consider a 3D maze, represented as an $(N+1) \times (N+1) \times (N+1)$ cube of $1 \times 1 \times 1$ cells with some cells empty and some cells blocked (i.e. walls). From every cell it is possible to move to any adjacent facing cell (no corner movement). The cells are identified by triples (i, j, k). The start state is $(0, 0, 0)$ and the goal test is satisfied only by (N, N, N). Let L_{ij} be the *loose projection* of the cube onto the first two coordinates, where the projected state (i, j) is a wall if (i, j, k) is a wall for *all* k. Let T_{ij} be the *tight projection* of the cube onto the first two coordinates, where the projected state (i, j) is a wall if (i, j, k) is a wall for *any* k. The projections are similarly defined for L_{ik} and so on.

Distance is the maze distance. If all paths to the goal are blocked, the distance is $+\infty$.

Mark each admissible heuristic below.

- ○ For (i, j, k), the value $3N - i - j - k$.
- ○ For (i, j, k), the value $N^3 - ijk$.
- ○ For (i, j, k), the distance from (i, j) to the goal in L_{ij}.
- ○ For (i, j, k), the distance from (i, j) to the goal in T_{ij}.
- ○ For (i, j, k), the distance from (i, j) to the goal in L_{ij} plus the distance from (i, k) to the goal in L_{ik} plus the distance from (j, k) to the goal in L_{jk}.
- ○ For (i, j, k), the distance from (i, j) to the goal in T_{ij} plus the distance from (i, k) to the goal in T_{ik} plus the distance from (j, k) to the goal in T_{jk}.

(c) The cube is back! Consider an $(N+1) \times (N+1) \times (N+1)$ gridworld. Luckily, all the cells are empty – there are no walls within the cube. For each cell, there is an action for each adjacent facing open cell (no corner movement), as well as an action *stay*. The actions all move into the corresponding cell with probability p but stay with probability $1 - p$. *Stay* always stays. The reward is always zero except when you enter the goal cell at (N, N, N), in which case it is 1 and the game then ends. The discount is $0 < \gamma < 1$.

(i) [1 pt] How many iterations k of value iteration will there be before $V_k(0, 0, 0)$ becomes non-zero? If this will never happen, write *never*.

(ii) [1 pt] If and when $V_k(0, 0, 0)$ first becomes non-zero, what will it become? If this will never happen, write *never*.

(iii) [1 pt] What is $V^*(0, 0, 0)$? If it is undefined, write *undefined*.

(d) The cube is still here! (It's also still empty.) Now the reward depends on the cell being entered. The goal cell is not special in any way. The reward for *staying* in a cell (either intentionally or through action failure) is always 0. Let V_k be the value function computed after k iterations of the value iteration algorithm. Recall that V_0 is defined to be 0 for all states. For each statement, circle the subset of rewards (if any) for which the statement holds.

(i) [1 pt] As the number of iterations k of value iteration increases, $V_k(s)$ cannot decrease when all cell-entry rewards:

○ are zero ○ are in the interval $[0, 1]$ ○ are in the interval $[-1, 1]$

(ii) [1 pt] The optimal policy can involve the *stay* action for some states when all cell-entry rewards:

○ are zero ○ are in the interval $[0, 1]$ ○ are in the interval $[-1, 1]$

(e) F-learning is a forgetful alternative to Q-learning. Where Q-learning tracks Q-values, F-learning tracks F-values. After experiencing an episode (s, a, r, s'), F-learning does the following update:

$F(s, a) = r + \gamma \max_{a'} F(s', a')$

As in Q-learning, All F-values are initialized to 0. Assume all states and actions are experienced infinitely often under a fixed, *non-optimal* policy π that suffices for Q-learning's convergence and optimality. Note that π will in general be stochastic in the sense that for each state s, $\pi(s)$ gives a distribution over actions that are then randomly chosen between.

For each claim, mark the classes of MDPs for which it is true:

(i) [1 pt] F-learning converges to some fixed values:

○ for deterministic state transitions ○ for stochastic state transitions
○ never ○ whenever Q-learning converges

(ii) [1 pt] F-learning converges to the optimal Q-values:

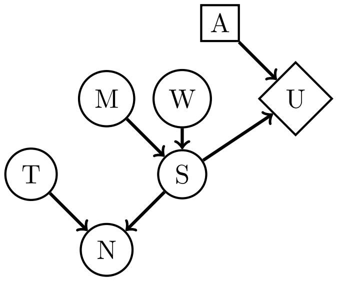
○ for deterministic state transitions ○ for stochastic state transitions
○ never ○ whenever Q-learning converges

(iii) [1 pt] F-learning converges to the Q-values of the policy π:

○ for deterministic state transitions ○ for stochastic state transitions
○ never ○ whenever Q-learning converges

Q3. [9 pts] Decision Networks and VPI

(a) Consider the decision network structure given below:



Mark all of the following statements that **could possibly be true**, for some probability distributions for $P(M), P(W), P(T), P(S|M,W)$, and $P(N|T,S)$ and some utility function $U(S,A)$:

(i) [1.5 pts]

☐ $\text{VPI}(T) < 0$ ☐ $\text{VPI}(T) = 0$ ☐ $\text{VPI}(T) > 0$ ☐ $\text{VPI}(T) = \text{VPI}(N)$

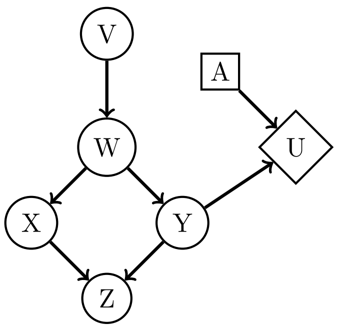
(ii) [1.5 pts]

☐ $\text{VPI}(T|N) < 0$ ☐ $\text{VPI}(T|N) = 0$ ☐ $\text{VPI}(T|N) > 0$ ☐ $\text{VPI}(T|N) = \text{VPI}(T|S)$

(iii) [1.5 pts]

☐ $\text{VPI}(M) > \text{VPI}(W)$ ☐ $\text{VPI}(M) > \text{VPI}(S)$ ☐ $\text{VPI}(M) < \text{VPI}(S)$ ☐ $\text{VPI}(M|S) > \text{VPI}(S)$

(b) Consider the decision network structure given below.



Mark all of the following statements that are **guaranteed to be true**, regardless of the probability distributions for any of the chance nodes and regardless of the utility function.

(i) [1.5 pts]

☐ $\text{VPI}(Y) = 0$

☐ $\text{VPI}(X) = 0$

☐ $\text{VPI}(Z) = \text{VPI}(W, Z)$

☐ $\text{VPI}(Y) = \text{VPI}(Y, X)$

(ii) [1.5 pts]

☐ $\text{VPI}(X) \leq \text{VPI}(W)$

☐ $\text{VPI}(V) \leq \text{VPI}(W)$

☐ $\text{VPI}(V \mid W) = \text{VPI}(V)$

☐ $\text{VPI}(W \mid V) = \text{VPI}(W)$

(iii) [1.5 pts]

☐ $\text{VPI}(X \mid W) = 0$

☐ $\text{VPI}(Z \mid W) = 0$

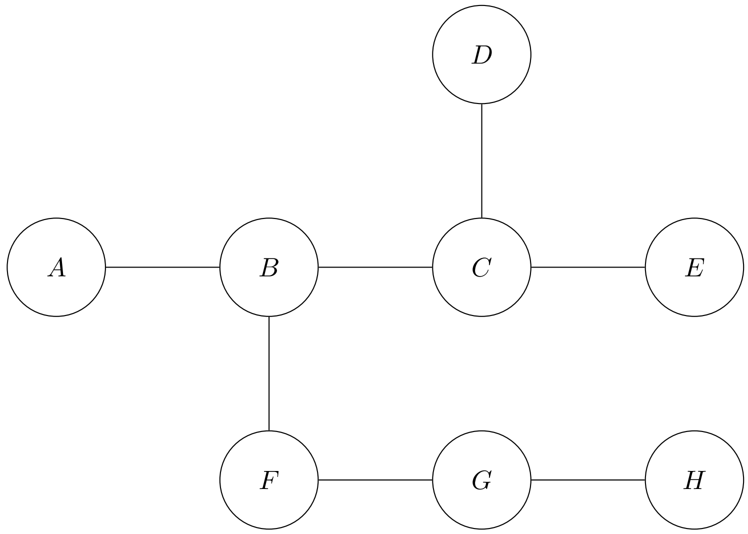
☐ $\text{VPI}(X, W) = \text{VPI}(V, W)$

☐ $\text{VPI}(W, Y) = \text{VPI}(W) + \text{VPI}(Y)$

Q4. [9 pts] Bayes Net CSPs

(a) For the following Bayes' Net structures that are missing a direction on their edges, assign a direction to each edge such that the Bayes' Net structure implies the requested conditional independences and such that the Bayes' Net structure does not imply the conditional independences requested not to be true. Keep in mind that Bayes' Nets cannot have directed cycles.

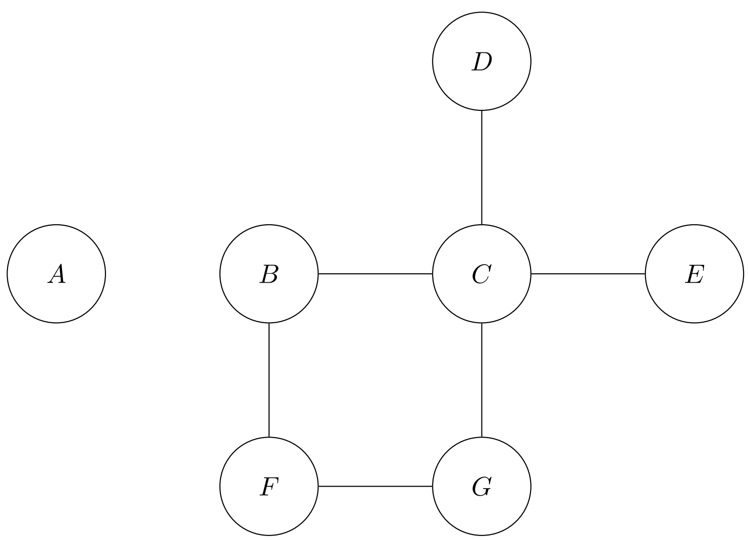
(i) [2 pts]



Constraints:

- DG
- not DA
- DE
- HF

(ii) [2 pts]

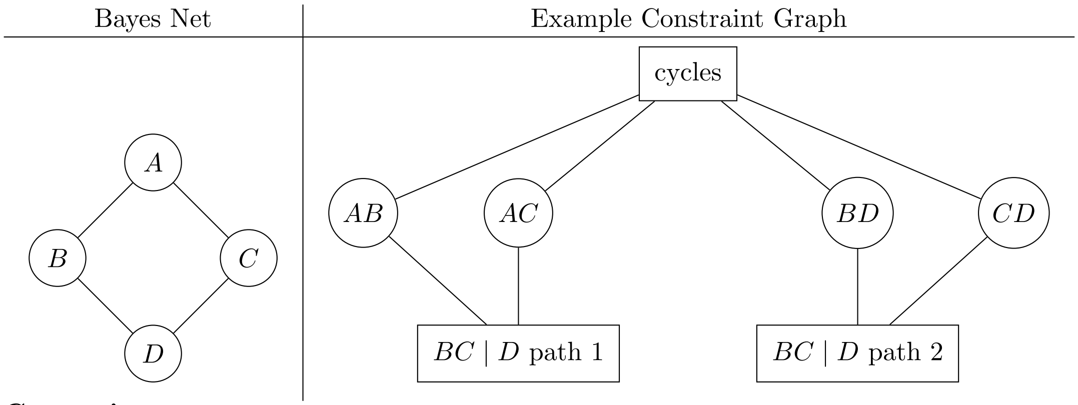


Constraints:

- DF
- not DG
- DE
- Bayes Net has no directed cycles

(b) For each of the following Bayes Nets and sets of constraints draw a constraint graph for the CSP. Remember that the constraint graph for a CSP with non-binary constraints, i.e., constraints that involve more than two variables, is drawn as a rectangle with the constraint connected to a node for each variable that participates in that constraint. A simple example is given below.

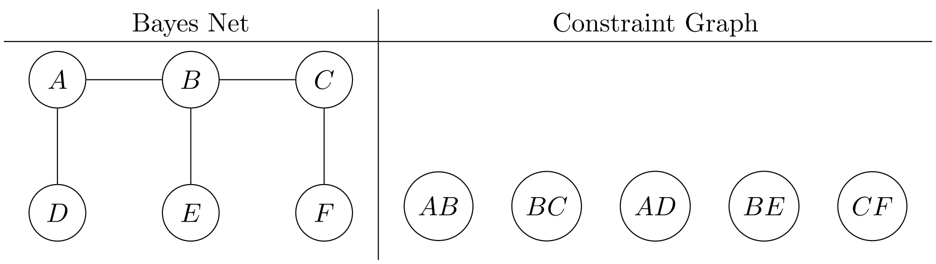
Note: As shown in the example below, if a constraint can be broken up into multiple constraints, do so.



Constraints:

- $BC \mid D$
- No directed cycles

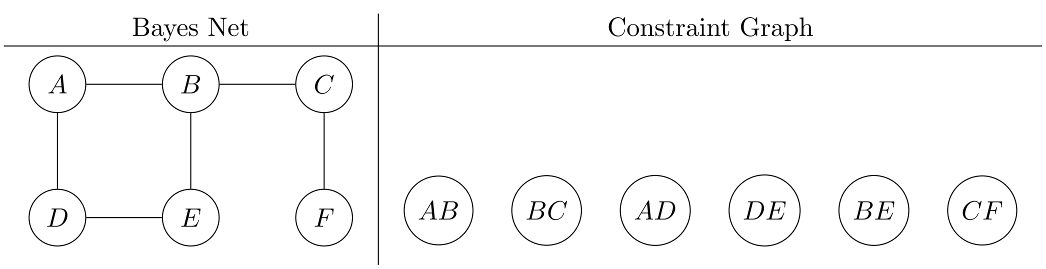
(i) [2 pts]



Constraints:

- $AF \mid E$
- not DC

(ii) [3 pts]



Constraints:

- $AE \mid F$
- CE
- No directed cycles

Q5. [20 pts] Probability and Bayes Net Representation

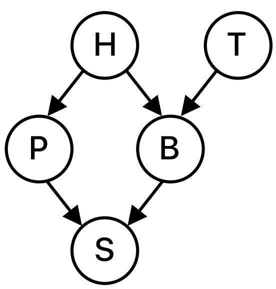
You're interested in knowing whether you would be **S**atisfied with your choice of snack(s), and so you decide to make the prediction using probabilistic inference over a model with the following variables:

- S, whether or not you will be **S**atisfied.
- H, whether or not you will be **H**ungry.
- T, whether or not you will be **T**hirsty.
- P, whether or not you will have **P**izza.
- B, whether or not you will have **B**oba.

Each of the variables may take on two values: *yes* or *no*.

(a) [1 pt] Your first idea for a probability model is a joint probability table over all of the variables. What's the **minimum** number of parameters you need to fully specify this joint probability distribution?

(b) [1 pt] You decide this is too many parameters. To fix this, you decide to model the problem with the following Bayes net instead:



$\Pr(H)$	
$+h$	0.7
$-h$	0.3

$\Pr(T)$	
$+t$	0.6
$-t$	0.4

$\Pr(P\|H)$		
$+p$	$+h$	0.8
$+p$	$-h$	0.5
$-p$	$+h$	0.2
$-p$	$-h$	0.5

$\Pr(B\|H,T)$			
$+b$	$+h$	$+t$	0.4
$+b$	$+h$	$-t$	0.2
$+b$	$-h$	$+t$	0.9
$+b$	$-h$	$-t$	0.5
$-b$	$+h$	$+t$	0.6
$-b$	$+h$	$-t$	0.8
$-b$	$-h$	$+t$	0.1
$-b$	$-h$	$-t$	0.5

$\Pr(S\|P,B)$			
$+s$	$+p$	$+b$	0.9
$+s$	$+p$	$-b$	0.4
$+s$	$-p$	$+b$	0.7
$+s$	$-p$	$-b$	0.1
$-s$	$+p$	$+b$	0.1
$-s$	$+p$	$-b$	0.6
$-s$	$-p$	$+b$	0.3
$-s$	$-p$	$-b$	0.9

You do not know which snack(s) you are going for, but you know you are both hungry, thirsty, and definitely getting Pizza. According to your model, what is the probability that you will be satisfied? (First, write out the expression *in terms of conditional probabilities from the model*; then, plug in the *values from the tables* and compute the final answer.)

(c) [3 pts] You thought the last part required too much computation so you decide to use rejection sampling, sampling variables in topological order. Write the probability of rejecting a sample for the following queries.

$P(+p \mid +h) \quad = \quad \square$

$P(-s \mid +p) \quad = \quad \square$

$P(+s \mid -h, +t) \quad = \quad \square$

(d) Given that you are satisfied with your choice of snack(s), write out the variable elimination steps you would take to compute the probability that you actually had boba, that is, $\Pr(+b \mid +s)$. (You do **not** have to plug in the values from the tables.)

(i) [2 pts] Which of the following factors do we start with?

☐ $\Pr(H)$ ☐ $\Pr(T)$ ☐ $\Pr(P)$ ☐ $\Pr(B)$ ☐ $\Pr(+s)$
☐ $\Pr(H|P)$ ☐ $\Pr(P|H)$ ☐ $\Pr(B|H)$ ☐ $\Pr(B|T)$ ☐ $\Pr(B|H,T)$
☐ $\Pr(+s|P)$ ☐ $\Pr(+s|B)$ ☐ $\Pr(+s|P,H)$ ☐ $\Pr(+s|P,H,B)$ ☐ $\Pr(+s|P,B)$

(ii) [1 pt] First, we eliminate H. What is the factor f_1 generated when we eliminate H?

○ $f_1(P)$ ○ $f_1(B)$ ○ $f_1(T)$ ○ $f_1(+s)$
○ $f_1(P,\ B)$ ○ $f_1(P,\ T)$ ○ $f_1(P,+s)$ ○ $f_1(B,\ T)$ ○ $f_1(B,+s)$ ○ $f_1(T,+s)$
○ $f_1(P,B,T)$ ○ $f_1(P,B,+s)$ ○ $f_1(B,T,+s)$

(iii) [1 pt] Write out the expression for computing f_1 in terms of the remaining factor(s) (before H is eliminated).

$f_1(\_\_\_\_\_\_\_\_) = \square$

(iv) [2 pts] Next, we eliminate T. What is the factor f_2 generated when we eliminate T?

$\square$

Write out the expression for computing f_2 in terms of the remaining factor(s) (before T is eliminated).

$f_2(\_\_\_\_\_\_\_\_) = \square$

(v) [2 pts] Finally, we eliminate P. What is the factor f_3 generated when we eliminate P?

$\square$

Write out the expression for computing f_3 in terms of the remaining factor(s) (before P is eliminated).

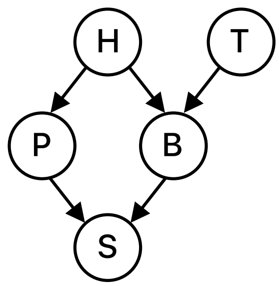
$f_3(\_\_\_\_\_\_\_\_) = \square$

(vi) [1 pt] Write out the expression for computing $\Pr(+b|+s)$ in terms of the remaining factor(s) (after P is eliminated).

$\Pr(+b \mid +s) =$

(e) **Conditional Independence**: For each of the following statements about conditional independence, mark if it is guaranteed by the Bayes Net.

The Bayes Net is reproduced below for your convenience.



(i) [1 pt] $H \perp\!\!\!\perp T$

○ Guaranteed ○ Not guaranteed

(ii) [1 pt] $P \perp\!\!\!\perp T \mid B$

○ Guaranteed ○ Not guaranteed

(iii) [1 pt] $H \perp\!\!\!\perp T \mid S$

○ Guaranteed ○ Not guaranteed

(iv) [1 pt] $S \perp\!\!\!\perp T \mid B$

○ Guaranteed ○ Not guaranteed

(v) [1 pt] $H \perp\!\!\!\perp S \mid P,\ B$

○ Guaranteed ○ Not guaranteed

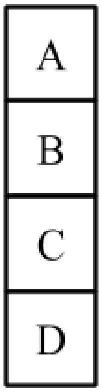
(vi) [1 pt] $P \perp\!\!\!\perp T \mid H,\ S$

○ Guaranteed ○ Not guaranteed

SID: ______________________

Q6. [12 pts] Finding Waldo

You are part of the CS 188 Search Team to find Waldo. Waldo randomly moves around floors A, B, C, and D. Waldo's location at time t is X_t. At the end of each timestep, Waldo stays on the same floor with probability 0.5, goes upstairs with probability 0.3, and goes downstairs with probability 0.2. If Waldo is on floor A, he goes down with probability 0.2 and stays put with probability 0.8. If Waldo is on floor D, he goes upstairs with probability 0.3 and stays put with probability 0.7.



X_0	$P(X_0)$
A	0.1
B	0.2
C	0.3
D	0.4

(a) [2 pts] Fill in the table below with the distribution of Waldo's location at time $t = 1$.

X_t	$P(X_1)$
A	
B	
C	
D	

(b) [2 pts] $F_T(X)$ is the fraction of timesteps Waldo spends at position X from $t = 0$ to $t = T$. The system of equations to solve for $F_\infty(A)$, $F_\infty(B)$, $F_\infty(C)$, and $F_\infty(D)$ is below. Fill in the blanks.
Note: You may or may not use all equations.

$\_\_\_\ F_\infty(A) + \_\_\_\ F_\infty(B) + \_\_\_\ F_\infty(C) + \_\_\_\ F_\infty(D) = \_\_\_$

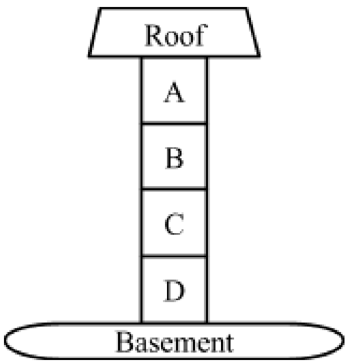
$\_\_\_\ F_\infty(A) + \_\_\_\ F_\infty(B) + \_\_\_\ F_\infty(C) + \_\_\_\ F_\infty(D) = \_\_\_$

$\_\_\_\ F_\infty(A) + \_\_\_\ F_\infty(B) + \_\_\_\ F_\infty(C) + \_\_\_\ F_\infty(D) = \_\_\_$

$\_\_\_\ F_\infty(A) + \_\_\_\ F_\infty(B) + \_\_\_\ F_\infty(C) + \_\_\_\ F_\infty(D) = \_\_\_$

$\_\_\_\ F_\infty(A) + \_\_\_\ F_\infty(B) + \_\_\_\ F_\infty(C) + \_\_\_\ F_\infty(D) = \_\_\_$

To aid the search a sensor S_r is installed on the roof and a sensor S_b is installed in the basement. Both sensors detect either sound ($+s$) or no sound ($-s$). The distribution of sensor measurements is determined by d, the number of floors between Waldo and the sensor. For example, if Waldo is on floor B, then $d_b = 2$ because there are two floors (C and D) between floor B and the basement and $d_r = 1$ because there is one floor (A) between floor B and the roof. The prior of the both sensors' outputs are identical and listed below. **Waldo will not go onto the roof or into the basement.**



X_0	$P(X_0)$
A	0.1
B	0.2
C	0.3
D	0.4

S_r	$P(S_r \mid d_r)$
$+s$	$0.3 * d_r$
$-s$	$1 - 0.3 * d_r$

S_b	$P(S_b \mid d_b)$
$+s$	$1 - 0.3 * d_b$
$-s$	$0.3 * d_b$

S	$P(S)$
$+s$	0.5
$-s$	0.5

(c) [1 pt] You decide to track Waldo by particle filtering with 3 particles. At time $t = 2$, the particles are at positions $X_1 = A$, $X_2 = B$ and $X_3 = C$. Without incorporating any sensory information, what is the probability that the particles will be resampled as $X_1 = B$, $X_2 = B$, and $X_3 = C$, after time elapse?

(d) To decouple this from the previous question, assume the particles after time elapsing are $X_1 = B$, $X_2 = C$, $X_3 = D$, and the sensors observe $S_r = +s$ and $S_b = -s$.

(i) [3 pts] What are the particle weights given these observations?

Particle	Weight
$X_1 = B$	
$X_2 = C$	
$X_3 = D$	

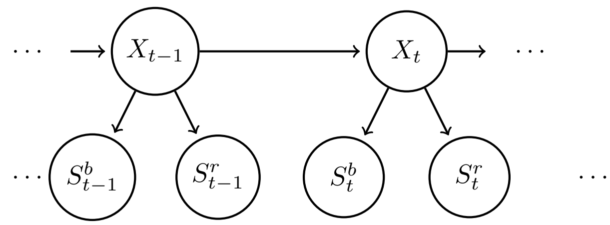
(ii) [1 pt] To decouple this from the previous question, assume the particle weights in the following table. What is the probability the particles will be resampled as $X_1 = B$, $X_2 = B$, and $X_3 = D$?

Particle	Weight
$X = B$	0.1
$X = C$	0.6
$X = D$	0.3

(e) [3 pts] **Note: the r and b subscripts from before will be written here as superscripts.**

Part of the expression for the forward algorithm update for Hidden Markov Models is given below. $s^r_{0:t}$ are all the measurements from the roof sensor $s^r_0, s^r_1, s^r_2, \ldots, s^r_t$. $s^b_{0:t}$ are all the measurements from the roof sensor $s^b_0, s^b_1, s^b_2, \ldots, s^b_t$.

Which of the following are correct completions of line (4)? Circle all that apply.



$$
\begin{aligned}
P(x_t|s^r_{0:t}, s^b_{0:t}) &\propto P(x_t, s^r_{0:t}, s^b_{0:t}) && (1)\\
&= \sum_{x_{t-1}} P(x_{t-1}, x_t, s^r_{0:t}, s^b_{0:t}) && (2)\\
&= \sum_{x_{t-1}} P(x_{t-1}, x_t, s^r_{0:t-1}, s^r_t, s^b_{0:t-1}, s^b_t) && (3)\\
&= \sum_{x_{t-1}} \underline{\hspace{4cm}} \; P(x_t|x_{t-1})P(x_{t-1}, s^r_{0:t-1}, s^b_{0:t-1}) && (4)
\end{aligned}
$$

○ $P(s^r_t, s^b_t|x_{t-1}, x_t, s^r_{0:t-1}, s^b_{0:t-1})$

○ $P(s^r_t|x_t)P(s^b_t|x_t)$

○ $P(s^r_t|x_{t-1})P(s^b_t|x_{t-1})$

○ $P(s^r_t|s^r_{t-1})P(s^b_t|s^b_{t-1})$

○ $P(s^r_t, s^b_t|x_t)$

○ $P(s^r_t, s^b_t|x_t, x_{t-1})$

○ None of the above.

Q7. [12 pts] Machine Learning: Potpourri

(a) [2 pts] What it the **minimum** number of parameters needed to fully model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label Y and n features F_i? Assume binary class where each feature can possibly take on k distinct values.

(b) [2 pts] Under the **Naive Bayes assumption**, what is the **minimum** number of parameters needed to model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label Y and n features F_i? Assume binary class where each feature can take on k distinct values.

(c) [1 pt] You suspect that you are overfitting with your Naive Bayes with Laplace Smoothing. How would you adjust the strength k in Laplace Smoothing?

○ Increase k ○ Decrease k

(d) [2 pts] While using Naive Bayes with Laplace Smoothing, increasing the strength k in Laplace Smoothing can:

☐ Increase training error ☐ Decrease training error
☐ Increase validation error ☐ Decrease validation error

(e) [1 pt] It is possible for the perceptron algorithm to never terminate on a dataset that is linearly separable in its feature space.

○ True ○ False

(f) [1 pt] If the perceptron algorithm terminates, then it is guaranteed to find a max-margin separating decision boundary.

○ True ○ False

(g) [1 pt] In multiclass perceptron, every weight w_y can be written as a linear combination of the training data feature vectors.

○ True ○ False

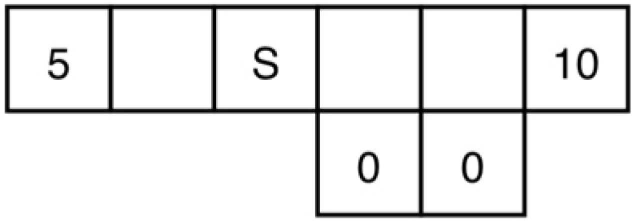
(h) [1 pt] For binary class classification, logistic regression produces a linear decision boundary.

○ True ○ False

(i) [1 pt] In the binary classification case, logistic regression is exactly equivalent to a single-layer neural network with a sigmoid activation and the cross-entropy loss function.

○ True ○ False

Q8. [15 pts] MDPs and RL

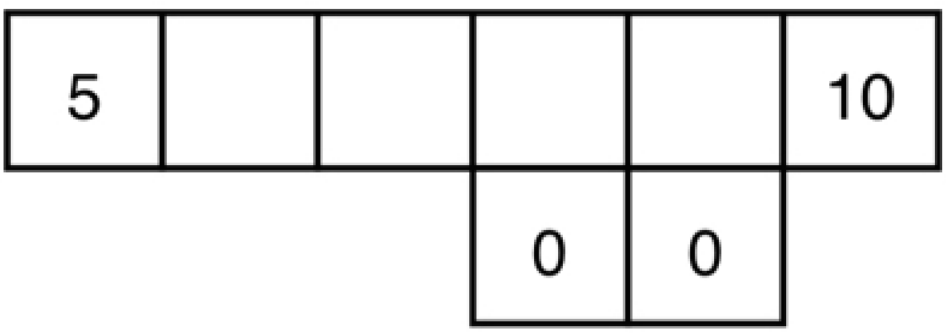


Consider the above gridworld. An agent is currently on grid cell S, and would like to collect the rewards that lie on both sides of it. If the agent is on a numbered square, its only available action is to Exit, and when it exits it gets reward equal to the number on the square. On any other (non-numbered) square, its available actions are to move East and West. Note that North and South are never available actions.

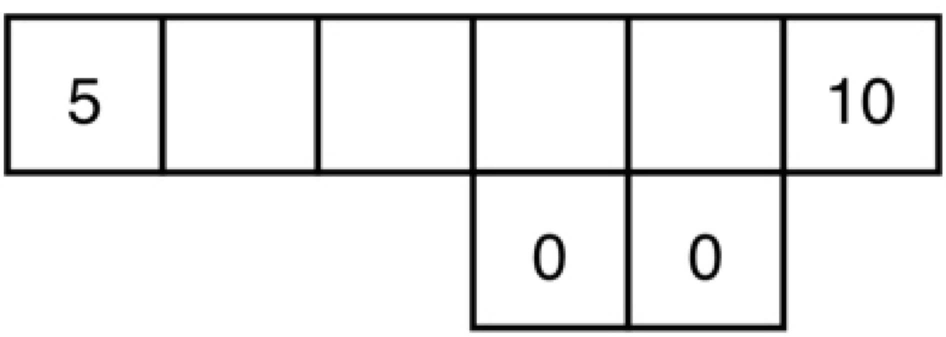
If the agent is in a square with an adjacent square downward, it does not always move successfully: when the agent is in one of these squares and takes a move action, it will only succeed with probability p. With probability $1 - p$, the move action will fail and the agent will instead move downwards. If the agent is not in a square with an adjacent space below, it will always move successfully.

For parts (a) and (b), we are using discount factor $\gamma \in [0, 1]$.

(a) [2 pts] Consider the policy π_{East}, which is to always move East (right) when possible, and to Exit when that is the only available action. For each non-numbered state x in the diagram below, fill in $V^{\pi_{\text{East}}}(x)$ in terms of γ and p.



(b) [2 pts] Consider the policy π_{West}, which is to always move West (left) when possible, and to Exit when that is the only available action. For each non-numbered state x in the diagram below, fill in $V^{\pi_{\text{West}}}(x)$ in terms of γ and p.



(c) [2 pts] For what range of values of p *in terms of* γ is it optimal for the agent to go West (left) from the start state (S)?

Range: ____________________

(d) [2 pts] For what range of values of p *in terms of* γ is π_{West} the optimal policy?

Range: ____________________

(e) [2 pts] For what range of values of p *in terms of* γ is π_{East} the optimal policy?

Range: ____________________

Recall that in approximate Q-learning, the Q-value is a weighted sum of features: $Q(s,a) = \sum_i w_i f_i(s,a)$. To derive a weight update equation, we first defined the loss function $L_2 = \frac{1}{2}(y - \sum_k w_k f_k(x))^2$ and found $dL_2/dw_m = -(y - \sum_k w_k f_k(x)) f_m(x)$. Our label y in this set up is $r + \gamma \max_a Q(s', a')$. Putting this all together, we derived the gradient descent update rule for w_m as $w_m \leftarrow w_m + \alpha \left(r + \gamma \max_a Q(s', a') - Q(s,a)\right) f_m(s,a)$.

In the following question, you will derive the gradient descent update rule for w_m using a different loss function:

$$L_1 = \left| y - \sum_k w_k f_k(x) \right|$$

(f) [4 pts] Find dL_1/dw_m. Show work to have a chance at receiving partial credit. Ignore the non-differentiable point.

(g) [1 pt] Write the gradient descent update rule for w_m, using the L_1 loss function.