CS 188 Spring 2021 — Introduction to Artificial Intelligence — Final

- You have approximately 170 minutes.
- The exam is open book, open calculator, and open notes.
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ○ means mark a **single choice**

First name	
Last name	
SID	

For staff use only:

Q1.	Tic-Tac-Toe	/11
Q2.	I Want a Project Partner	/20
Q3.	Pac-ML	/23
Q4.	Keyboard Navigation	/15
Q5.	Accommodating Course Robot	/17
Q6.	Holiday Planning	/14
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [11 pts] Tic-Tac-Toe

In this problem we will look at the classic game of Tic-Tac-Toe. Albert is training his intelligent agent, AlbertBot, to compete with MesutBot in a Tic-Tac-Toe competition.

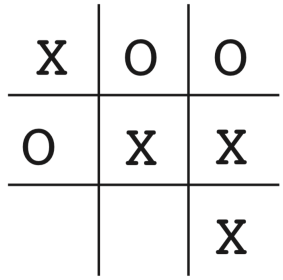


Figure 1: Example winning state for the 'X' player in a Tic-Tac-Toe game

Rules of Tic-Tac-Toe: Two players, X and O, take turns marking an empty square on a three-by-three board. A player wins by placing three marks in a row, horizontally, vertically, or diagonally, as the 'X' player has in the game shown above. If the board fills up with neither player getting three in a row, then the game is a draw.

Assumptions: AlbertBot plays 'O's, and **always plays second**. MesutBot has a *fixed*, unknown, possibly stochastic and possibly suboptimal policy. AlbertBot can train against MesutBot as many times as we want prior to the real competition. In the real competition, AlbertBot gets a reward of 1 if it wins against MesutBot, and a reward of 0 if it draws or loses.

Problem (Goal): Maximize AlbertBot's expected reward, i.e., its expected probability of winning, in the real competition.

(a) [2 pts] Among the following choices, which is a tightest upper bound on the number of leaf nodes in the complete game tree of Tic-Tac-Toe, ignoring possible symmetries?
◯ 2^9 ◯ 3^9 ◯ 9^2 ◯ 9^3 ◯ 9^8 ◯ $9!$ ◯ 9^9

(b) [2 pts] Is the minimax algorithm a suitable way of solving the problem specified above?
◯ Yes, because the state space of Tic-Tac-Toe is small enough that running Minimax on the whole game tree is feasible.
◯ Yes, but not due to the reason in the choice above.
◯ No, because the minimax only works for zero-sum games, while Tic-Tac-Toe is not zero-sum.
◯ No, but not due to the reason in the choice above.

(c) [3 pts] Let MesutBot's policy be $\pi_M(s, a)$, denoting the probability that MesutBot plays a in state s, and let $Result(s, a)$ denote the state resulting from playing a in s. Can we formulate AlbertBot's Tic-Tac-Toe problem as an MDP, and, if so, which of the following are correct formulas for the transition model $P(s' \mid a, s)$?
☐ Yes, $P(s' \mid a, s) = 1$ if $s' = Result(s, a)$, 0 otherwise.
☐ Yes, $P(s' \mid a, s) = \pi_M(s, a)$ if $s' = Result(s, a)$, 0 otherwise.
☐ Yes, $P(s' \mid a, s) = \sum_{a' \,:\, s'=Result(Result(s,a),a')} \pi_M(Result(s, a), a')$.
☐ Yes, $P(s' \mid a, s) = \pi_M(Result(s, a), a')$ if $s' = Result(Result(s, a), a')$ for some a', 0 otherwise.
◯ No, AlbertBot's problem cannot be formulated as an MDP.

We now introduce a new method for solving AlbertBot's problem that works as follows:
(1) Let $V(s)$ be a table, indexed by state, representing the value, i.e., the expected win probability starting in s. Let the initial values for $V(s)$ be 0.5, except the terminal states: for states with three 'X's in a row, or drawn terminal states, we set an initial value of 0, and for states with three 'O's in a row, we set an initial value of 1.

(2) AlbertBot plays many practice games against MesutBot. In any state s, AlbertBot plays $a^* = \arg\max_a V(Result(s, a))$, i.e., the best move according to one-step lookahead with V, breaking ties randomly.

(3) For each round (consisting of MesutBot's move followed by AlbertBot's move), let S_t denote the state before the MesutBot's move and S_{t+1} denote the state after AlbertBot's move. V is updated after each round according to the following equation:

$$V(S_t) \longleftarrow V(S_t) + \alpha(V(S_{t+1}) - V(S_t))$$

where $V(S_t)$ denotes the value of the state S_t and $0 < \alpha < 1$ is a hyperparameter.

(4) We update V for many games. Then, in the real competition, V remains fixed and AlbertBot again plays $a^* = \arg\max_a V(Result(s, a))$.

(d) [3 pts] There is at least one major problem in the method as introduced above that can be fixed to make it working better in practice. Which of the following describes the problem, and proposes a reasonable fix to the problem? Choose all correct statements. A statement is considered correct if and only if both the problem and the fix is described correctly.

☐ AlbertBot cannot reason about the opponent (MesutBot's) moves. The fix is to use self-play during training.

☐ AlbertBot cannot reason about the opponent (MesutBot's) moves. The fix is to train a neural network to predict the opponent's move given a certain state.

☐ The training process lacks exploration. The fix is to use epsilon-greedy exploration during training and update the value table for every action taken during training.

☐ The training process lacks exploration. The fix is to use epsilon-greedy exploration during training, but update the value table only for non-exploratory actions during training.

☐ The training process requires full information about the opponent's policy which we do not have access to. The fix is to use Q-learning instead.

☐ The training process requires full information about the opponent's policy which we do not have access to. The fix is to use policy iteration.

◯ None of the above (for every sentence above either the problem or the fix is incorrect).

(e) [1 pt] Given that the problem described in the previous part is successfully addressed, what is a suitable schedule for the value of α?

◯ Constant close to 1
◯ Constant close to 0
◯ Increase asymptotically towards 1
◯ Decrease asymptotically towards 0

SID: ____________________

Q2. [20 pts] I Want a Project Partner

In 2035, CS 188 will provide an AI partner for your projects. The process of acquiring a partner works as follows:

- In the *Begin* state, you draw a random AI partner whose quality x is an integer drawn from a distribution $P(X)$. $P(X)$ is non-zero over a single contiguous range of values $[a, b]$, where b may be ∞. (Throughout this question, the variables x and x' will refer to integers in the range $[a, b]$) The cost of this draw is 0.
- Afterwards, you have two choices:
  1. *Stop*: Keep the current partner, and go to the *End* state.
  2. *Draw*: Pay c tokens to draw another AI partner whose quality comes from the same distribution $P(X)$.

We denote state x to be the current quality of your partner. For example, one possible sequence of states and actions might be

$$Begin \xrightarrow{\text{Draw}} 1 \xrightarrow{\text{Draw}} 2 \xrightarrow{\text{Draw}} 5 \xrightarrow{\text{Stop}} End$$

with you paying c for each of the second and third draws and getting a quality=5 partner by stopping. The reward for stopping equals the quality of the agent that you get at the end. We assume no discounting, i.e., $\gamma = 1$.

(a) [6 pts] We will go through some steps to solve the problem as an infinite-horizon MDP.

(i) [1 pt] Assume x, x' are positive integers, which of the following are true about the reward function $R(s, a, s')$ of this MDP?

☐ For all x, $R(Begin, Draw, x) = -c$
☐ For all x, x', $R(x, Draw, x') = -c$
☐ For all x, $R(x, Stop, End) = -c + x$
☐ For all x, $R(x, Stop, End) = x$
○ None of the Above

(ii) [1 pt] What can we say about the Q-values in this MDP?

☐ $Q^*(x, Draw)$ increases with x
☐ $Q^*(x, Draw)$ is a constant D, independent of x
☐ $Q^*(x, Draw)$ is a constant D for some (possible empty) sequence $[a, ..., s]$ and thereafter increases with x
○ None of the Above

(iii) [1 pt] *True/False* If it's optimal to *Stop* at x, it's optimal to *Stop* at any $x' > x$.

○ True
○ False

(iv) [1 pt] What can we say about the U-values in this MDP?

☐ $U^*(x) = \max_{a \in \{Draw, Stop\}} Q^*(x, a)$
☐ $U^*(x)$ increases with x
☐ $U^*(x)$ is a constant D, independent of x
☐ $U^*(x)$ is a constant D for some (possible empty) sequence $[a, ..., s]$ and thereafter increases with x
○ None of the Above

(v) [1 pt] Assume at state s, the optimal policy is indifferent between *Draw* and *Stop*, i.e. $Q^*(s, Draw) = Q^*(s, Stop)$. What can we say about the optimal policy π^* and optimal Q-values Q^*?

☐ $\pi^*(x) = Draw$ for all $x < s$
☐ $\pi^*(x) = Stop$ for all $x < s$
☐ $\pi^*(x) = Draw$ for all $x > s$
☐ $\pi^*(x) = Stop$ for all $x > s$
☐ $Q^*(s, Draw) = s$
○ None of the Above

(vi) [1 pt] Let $D = Q^*(s, Draw) = Q^*(s, Stop)$. We are interested in finding D by solving a Bellman equation on U-values. Which, if any, of the following equations are correct?

- ☐ $D = -c + D \cdot \sum_{x=1}^{\infty} P(x)$
- ☐ $D = -c + \sum_{x=1}^{\infty} xP(x)$
- ☐ $D = -c + \cdot \sum_{x=1}^{D} xP(x) + D \cdot \sum_{x=D+1}^{\infty} P(x)$
- ☐ $D = -c + D \cdot \sum_{x=1}^{D} P(x) + \sum_{x=D+1}^{\infty} xP(x)$
- ○ None of the Above

(b) [4 pts] Now, suppose the $P(X)$ is given in the table below, and suppose $c = \frac{3}{4}$. Remember we define D to be the Q value of the state at which the optimal policy is indifferent between the two actions, i.e. $D = Q^*(s, Draw) = Q^*(s, Stop)$.

x	P(x)
1	1/4
2	1/4
3	1/4
4	1/4

(i) [2 pts] What is D? (hint: it's an integer) ______________

(ii) [1 pt] What is $U^*(3)$ ______________

(iii) [1 pt] What is $\pi^*(1)$ ______________

(c) [3 pts] Now suppose that $c = 1$, for the entire partner-drawing process, $x \in \{1, 2, 3, 4, 5, 6\}$, and the distribution of x, $P(X)$ can be one of $P_{uniform}(X)$ or $P_{biased}(X)$. The two distributions are shown in the table below:

x	$P_{uniform}(x)$	$P_{biased}(x)$
1	1/6	1/2
2	1/6	1/10
3	1/6	1/10
4	1/6	1/10
5	1/6	1/10
6	1/6	1/10

Define $D_{uniform} = D$ when $P(X)$ is $P_{uniform}(X)$, and define $D_{biased} = D$ when $P(X)$ is $P_{biased}(X)$. Define $Q_{uniform}$, Q_{biased}, $U_{uniform}$, U_{biased} similarly.

(i) [1 pt] Which of the following is true?

- ○ $D_{uniform} > D_{biased}$
- ○ $D_{uniform} < D_{biased}$
- ○ $D_{uniform} = D_{biased}$

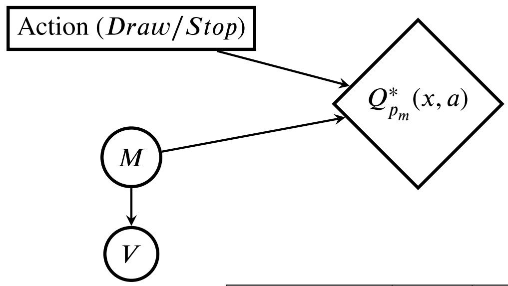
(ii) [2 pts] Given $D_{biased} = 2$, which of the following are necessarily false?

- ☐ $U^*_{uniform}(1) = U^*_{biased}(1)$
- ☐ $U^*_{uniform}(2) = U^*_{biased}(2)$
- ☐ $U^*_{uniform}(3) = U^*_{biased}(3)$
- ☐ $U^*_{uniform}(4) = U^*_{biased}(4)$
- ○ None of the Above

(d) [7 pts] Suppose that AI agents are produced by two manufacturers, m_1 and m_2. m_1 manufactures agents whose qualities come from $P_{uniform}(X)$ and m_2 manufactures agents whose qualities come from $P_{biased}(X)$.

Throughout the partner-selecting process, you will be drawing partners from exactly one of the two manufacturers, with an *a priori* probability of 0.5 for each. Moreover, agents manufactured by m_1 and m_2 might have slightly different pitches of voice V, such that agents whose qualities come from $P_{uniform}(X)$ have a 0.8 probability of having "high" pitches while agents whose qualities come from $P_{biased}(X)$ have only a 0.6 probability chance of having "high" pitches.

You are still faced with a decision between *Draw* and *Stop*. However, before you perform any actions, you can listen to the voice of your AI agent. This problem can be formulated as a decision network, as shown in Figure 2.



M	$P(M)$
m_1(uniform)	0.5
m_2(biased)	0.5

M	V	$P(V\|M)$
m_1(uniform)	high	0.8
m_1(uniform)	low	0.2
m_2(biased)	high	0.6
m_2(biased)	low	0.4

Figure 2: Decision network and associated conditional distributions for the problem.

(i) [2 pts] Define $EU_x(a|e)$ as "the expected utility of executing action a from state x given evidence e", i.e. $EU_x(a|e) = \sum_m P(m|e) \cdot Q^*_{P_m}(x, a)$. Note $Q^*_{P_m}$ represents the optimal Q-values when drawing from the quality distribution P_m associated with manufacturer m, i.e. when $m = m_1$, $Q^*_{P_m}$ is $Q^*_{uniform}$. What is $EU_2(Draw)$?

○ $0.5 \cdot Q^*_{uniform}(2, Draw) + 0.5 \cdot Q^*_{biased}(2, Draw)$
○ $Q^*_{uniform}(2, Draw)$
○ $Q^*_{biased}(2, Draw)$
○ $\frac{1}{6} \cdot Q^*_{uniform}(2, Draw) + \frac{1}{10} \cdot Q^*_{biased}(2, Draw)$

(ii) [2 pts] Define $MEU_x(e)$ as "the maximum expected utility from state x given evidence e", i.e. $MEU_x(e) = \max_a EU_x(a|e)$. What is $MEU_2(\emptyset)$?

○ $0.5 \cdot U^*_{uniform}(2) + 0.5 \cdot U^*_{biased}(2)$
○ $U^*_{uniform}(2)$
○ $U^*_{biased}(2)$
○ $\frac{1}{6} \cdot U^*_{uniform}(2) + \frac{1}{10} \cdot U^*_{biased}(2)$

(iii) [3 pts] Define $VPI_x(E'|e)$ as "the value of observing E' given our current evidence e from state x". Given that $D_{uniform} = 3$ and $D_{biased} = 2$, for which state(s) x is $VPI_x(V) > 0$ (Remember V represents the pitches of voice of the AI agents)?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ○ None of the above

Q3. [23 pts] Pac-ML

In this problem we will build classifiers to predict a binary target attribute $y \in \{0, 1\}$.

(a) [4 pts] In this part we will assume that examples are described by a single real-valued attribute x_1.

(i) [1 pt] Here are three data points:

y	0	0	1
x_1	1	2	3

Are these data points linearly separable?

◯ Yes ◯ No

(ii) [1 pt] Let's add a few more data points in our dataset.

y	0	0	1	0	0	1
x_1	1	2	3	4	5	6

Are these data points linearly separable?

◯ Yes ◯ No

(iii) [2 pts] The data can be augmented by adding a second feature $x_2 = f(x_1)$ for some function f. Which of the following candidates for f will render the data in the previous question linearly separable? For your convenience, the values of $\sin(x)$ when $x \in \{1, 2, 3, 4, 5, 6\}$ are $\{0.84, 0.91, 0.14, -0.76, -0.96, -0.28\}$.

☐ $x_2 = x_1^3$
☐ $x_2 = x_1 \bmod 2$
☐ $x_2 = \sin(x_1)$
☐ $x_2 = x_1 \bmod 3$

For the remainder of this question we consider examples described by three binary attributes. The training data are as follows:

y	0	0	0	0	0	1	1	1	1	1
x_1	1	0	0	0	0	1	0	0	1	1
x_2	0	0	1	0	0	1	1	1	1	1
x_3	0	0	0	0	0	0	1	1	0	0

(b) [4 pts]

Let us first try a naive Bayes classifier, for which we estimate the necessary probability model $\hat{P}$ from the data.

(i) [1 pt] Given a new data point x_1', x_2', x_3', which of the following are correct expressions for the predicted label y' of that data point using the naive Bayes classifier? Note that α is a normalizing constant.

☐ $\arg\max_y \alpha \hat{P}(y|x_1')\hat{P}(y|x_2')\hat{P}(y|x_3')$
☐ $\arg\max_y \alpha \hat{P}(y)\prod_{i=1}^{3} \hat{P}(x_i'|y)$
☐ $\arg\max_y \alpha \prod_{i=1}^{3} \hat{P}(x_i'|y)$
☐ $\arg\max_y \hat{P}(y)\prod_{i=1}^{3} \hat{P}(x_i'|y)$

(ii) [1 pt] How many parameters need to be estimated for $\hat{P}$? Here, "parameters" are the conditional and/or marginal probabilities needed in the naive Bayes model. Do not count parameters that can be calculated using the sum-to-1 constraint.

◯ 4
◯ 8
◯ 7
◯ 10

(iii) [1 pt] If we discard all the conditional independence assumptions of the naive Bayes model, how many parameters does $\hat{P}$ require? (Again, do not count parameters that can be calculated using the sum-to-1 constraint.)

○ 9 ○ 15
○ 12 ○ 16

(iv) [1 pt] A new, unlabeled data point arrives with an extra feature x_4. There are no previous training data to estimate the parameters associated with x_4, but it is still possible to make a prediction that incorporates x_4 using Laplace smoothing. Predicting y using x_4 and Laplace smoothing in this case will necessarily give identical results to predicting y while ignoring x_4 altogether.

○ True ○ False

(c) [3 pts] Now we consider decision trees for classifying these examples. We include the data here as well for convenience.

y	0	0	0	0	0	1	1	1	1	1
x_1	1	0	0	0	0	1	0	0	1	1
x_2	0	0	1	0	0	1	1	1	1	1
x_3	0	0	0	0	0	0	1	1	0	0

(i) [1 pt] Are all decision trees linear classifiers?

○ Yes ○ No

(ii) [1 pt] Which of x_1, x_2, and x_3 has the highest information gain?

○ x_1 ○ x_2 ○ x_3

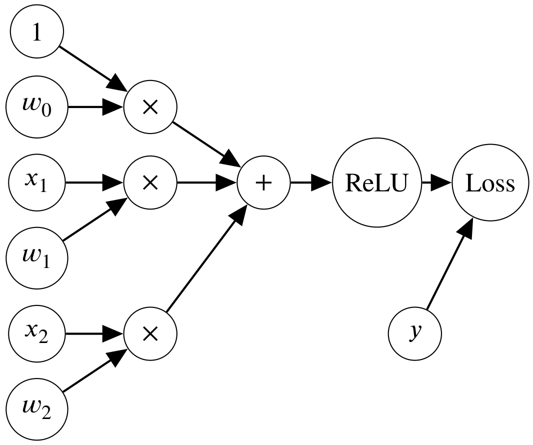
(iii) [1 pt] An adversary would like to flip one of the input bits in the first data column, so that the decision tree learning algorithm cannot fit the training data exactly. Which bit should be flipped?

○ x_1 ○ x_2 ○ x_3

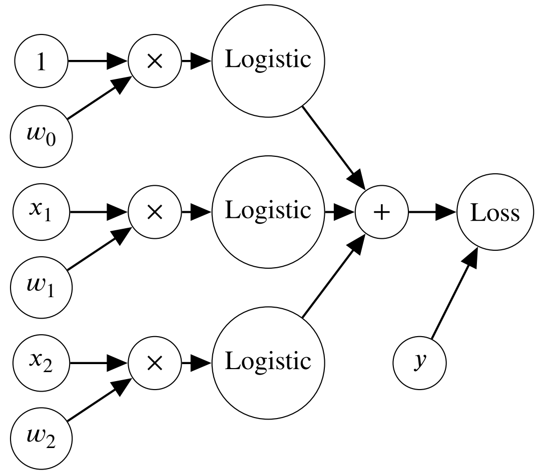
(d) [4 pts] For this part, we consider logistic regression and we ignore the attribute x_3.

(i) [2 pts] Which of the following neural networks are capable of representing a logistic regression model, assuming that the "Logistic" nodes implement the logistic function $g(z) = \frac{1}{1+e^{-z}}$?

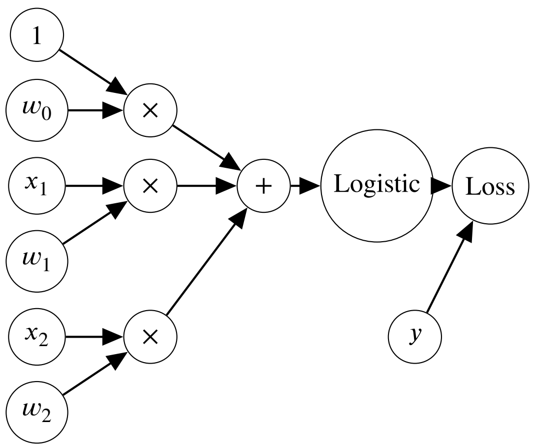
☐



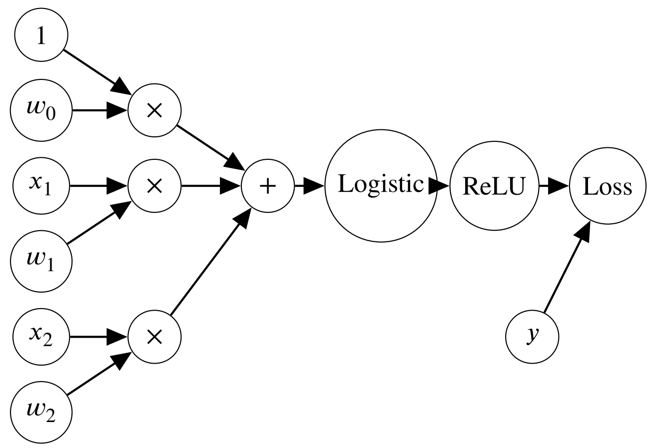
☐



☐



☐



(ii) [2 pts] Which of the following expressions is the correct gradient descent update rule for parameter w_2 in the weight vector $\mathbf{w}$? Assume that the loss function is the squared loss $\frac{1}{2}(y - h_{\mathbf{w}}(\mathbf{x}))^2$, that $h_{\mathbf{w}}(x) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$, that $\mathbf{w}^T\mathbf{x}$ denotes the inner-product and that α is the learning rate. Hint: the derivative of the logistic function $g(z) = \frac{1}{1+e^{-z}}$ is $g'(z) = g(z)(1 - g(z))$.

○ $w_2 \leftarrow w_2 - \alpha \left(y - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} \left(1 - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) x_2$

○ $w_2 \leftarrow w_2 - \alpha \left(y - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} \left(y - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) x_2$

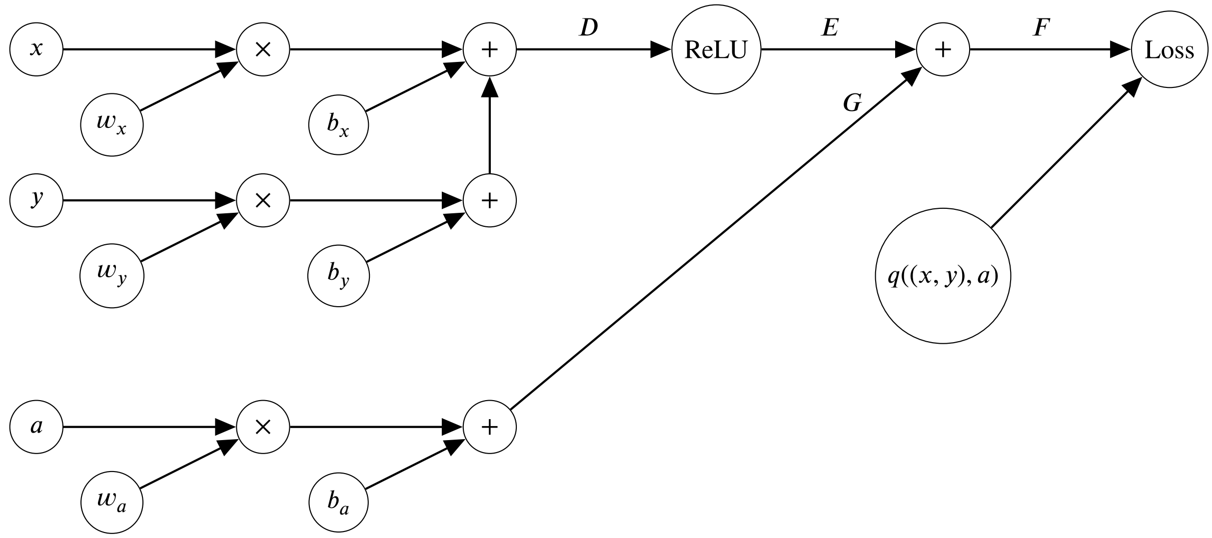
○ $w_2 \leftarrow w_2 + \alpha \left(y - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} x_2$

○ $w_2 \leftarrow w_2 + \alpha \left(y - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} \left(1 - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}\right) x_2$

(e) In this question we explore using approximate Q-learning to train a robot to complete a fixed, 2D grid maze in the fewest timesteps possible. The Q-function $Q(s, a)$ and policy $\pi(s)$ will be represented by neural networks with input vector $s = \begin{bmatrix} x & y \end{bmatrix}$, corresponding to the robot's x and y coordinates in the grid, and actions $a \in \{1, 2, 3, 4\}$ representing {North, South, East, West} respectively.

Albert trains the neural network shown below to learn his Q-values, where w_x, w_y, w_a are scalar weights and b_x, b_y, b_a are scalar biases.

$q(s = (x, y), a)$ is the target Q-value calculated using transitions (assume that this is the value we want to get close to). The labels D, E, F, G represent the output value of the preceding node:



(i) [2 pts] Which of the following are true about the above neural network? Assume that we use an arbitrary differentiable loss function for the *Loss* node.

☐ Using a sigmoid activation instead of the currently used ReLU activation will lead to minimal loss in fewer iterations of training.

☐ As long as we train on enough data, we can guarantee a validation accuracy at least as good as our training accuracy.

☐ Removing b_x will not affect the expressivity of the neural network.

☐ During backpropagation, we compute $\frac{\partial Loss}{\partial w_a}$, which we then can use to help compute $\frac{\partial Loss}{\partial w_x}$ and $\frac{\partial Loss}{\partial w_y}$.

○ None of the above

(ii) [2 pts] Which of the following are equivalent to $\frac{\partial Loss}{\partial b_x}$?

☐ $\frac{\partial Loss}{\partial F} \cdot \frac{\partial F}{\partial E} \cdot \frac{\partial E}{\partial D}$

☐ $\frac{\partial Loss}{\partial F} \cdot \frac{\partial F}{\partial E} \cdot \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial b_x}$

☐ $\frac{\partial Loss}{\partial F} \cdot \left(\frac{\partial F}{\partial E} \cdot \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial b_x} + \frac{\partial F}{\partial G} \cdot \frac{\partial G}{\partial b_a} \cdot \frac{\partial b_a}{\partial b_x} \right)$

☐ $\frac{\partial Loss}{\partial F} \cdot \left(\frac{\partial F}{\partial E} \cdot \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial b_x} + \frac{\partial F}{\partial G} \cdot \frac{\partial G}{\partial b_x} \right)$

○ None of the above

(iii) [2 pts] Which of the following loss functions should Albert use for this neural network?

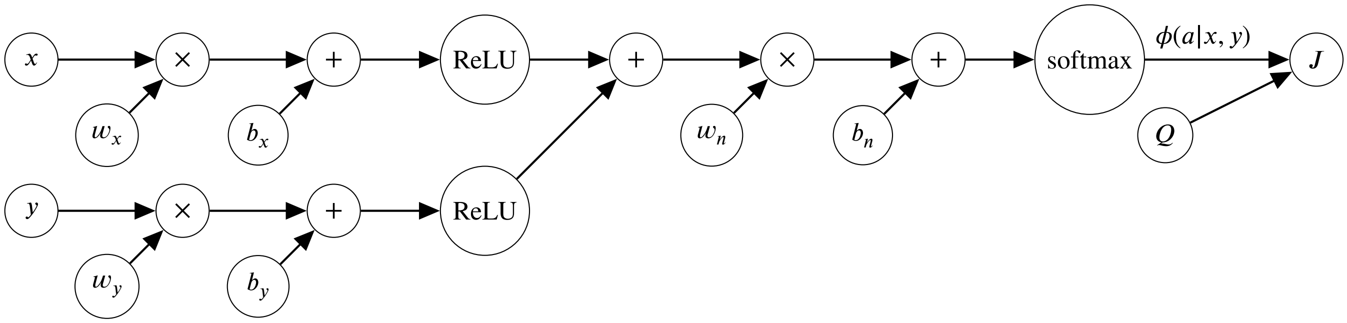
○ Mean-squared-error loss: $L(\hat{y}, y) = (y - \hat{y})^2$

○ $L(\hat{y}, y) = \begin{cases} 0 \text{ if } y = \hat{y} \\ 1 \text{ otherwise} \end{cases}$

○ Cross-entropy loss: $L(\hat{y}, y) = -[t \log y' + (1 - t)\log(1 - y')]$ where t =zero-one loss, $y' = \frac{1}{1+e^{-\hat{y}}}$

○ None of the above

(f) After training the neural network for $Q(s, a)$, Albert trains another neural network to return a distribution of actions to take at the current state $\phi(a|s)$. w_n is a vector weight in $\mathbb{R}^4$ and b_n is a vector bias in $\mathbb{R}^4$. This neural network is shown below:



The node Q in the above represents the neural network from the previous part, which takes in (x, y, a) and returns a Q-value. $\phi(a|x, y)$ is a vector containing a probability distribution over all possible actions the robot can take from a state. Albert defines J as a function that computes the expected Q-value weighted by the probabilities of the policy network output as follows:

$$J(\phi, Q, x, y) = \sum_a \phi(a|x, y)Q((x, y), a)$$

We want to train $\phi(a|x, y)$ to put higher probability on actions associated with larger Q-values.

(i) [1 pt] Which of the following can we use to appropriately update w_y at every iteration?

- ☐ Gradient descent
- ☐ Gradient ascent
- ☐ Stochastic gradient descent
- ☐ Perceptron learning rule
- ○ None of the above

(ii) [1 pt] Which of the following is / are the appropriate update rule(s) for w_x based on the above neural network and J function? We set α to be the learning rate.

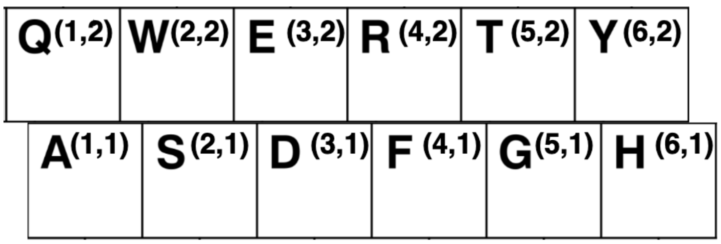
- ☐ $w_x \leftarrow w_x - \alpha \nabla_{w_x} J$
- ☐ $w_x \leftarrow w_x + \alpha \nabla_{w_x} J$
- ☐ $w_x \leftarrow w_x + \alpha \left(\nabla_{w_x} J + \nabla_{b_x} J\right)$
- ☐ $w_x \leftarrow w_x - \alpha \left(\nabla_{w_x} J - \left|\nabla_{b_x} J\right|\right)$
- ○ None of the above

SID: ____________________

Q4. [15 pts] Keyboard Navigation

Pacman is navigating in a map shown in the figure below (following the QWERTY keyboard). There are $N = 12$ grid squares in total. Each sqaure is labeled with a letter and its coordinate. Let $C_t = (x_t, y_t)$ denote Pacman's position at time step t, and it can move to any of the 6 directions to $(x_t, y_t + 1)$, $(x_t, y_t - 1)$, $(x_t - 1, y_t)$, $(x_t + 1, y_t)$, $(x_t + 1, y_t + 1)$, or $(x_t - 1, y_t - 1)$ in the next timestep. When Pacman attempts to move out of the map, i.e., $x_{t+1} \notin \{1, 2, ..., 6\}$ or $y_{t+1} \notin \{1, 2\}$, it will stay in the same square.

We assume that Pacman knows the layout of the map. However, Pacman does not know its position, and also does not know whether the move succeeds (i.e., not staying in the same square) at each step. There are two sensors to help Pacman track its position: (1) A_t is the number of neighboring squares for (x_t, y_t). For example, $A_t = 4$ when $(x_t, y_t) = (2, 2)$. (2) E_t is the number of alphabetically adjacent letters among the neighboring squares. The table below summarizes the (deterministic) value of E_t for each square.



C_t	E_t
Q, W, R, Y, A, S, T	0
E, D, F, H	1
G	2

(a) At step $t = 0$, Pacman can be in any square of the map with equal probability. In this part, at each step t, Pacman can move in any of the 6 directions with an equal probability. Note that Pacman can stay in the same square by taking an invalid move out of the map. We denote $Pr[C_t = (x_t, y_t)]$ as the probability of Pacman at position (x_t, y_t) at step t.

(i) [1 pt] Assume that Pacman does not have access to sensor signals. What is $Pr[C_1 = (2, 2)]$, i.e., the probability that Pacman is at position $(2, 2)$ when $t = 1$? Please fill in your solution as a fraction below. Please reduce the fraction into the lowest terms. For example, $2/72$ should be simplified to $1/36$. Please fill in $0/1$ if the answer is 0.

______ / ______

(ii) [1 pt] Pacman observes that $E_0 = 1$. Given this evidence, what is $Pr[C_1 = (2, 2) | E_0 = 1]$?

______ / ______.

(b) Pacman decides to track its location by particle filtering. Let $(x_t^{(i)}, y_t^{(i)})$ be the location of the i-th particle at step t. At step $t = 0$, we have 4 particles initialized in $\{(1, 1), (1, 2), (6, 1), (6, 2)\}$.

(i) [2 pts] Assume that after one forward simulation update, these particles reach states $\{(2, 1), (2, 2), (6, 2), (6, 2)\}$ respectively. What is the probability of each of these transitions?

Particle 1 from (1, 1) to (2, 1): ______ / ______

Particle 4 from (6, 2) to (6, 2): ______ / ______

(ii) [2 pts] What is the updated belief distribution based on the new particle values? Note that we have not incorporated any observations.

$Pr[C_1 = (2, 1)] =$ ______ / ______

$Pr[C_1 = (2, 2)] =$ ______ / ______

$Pr[C_1 = (6, 2)] =$ ______ / ______

(iii) [2 pts] Pacman observes that $A_1 = 4$. Please fill in the probabilities given the evidence.

$Pr[C_1 = (2, 1) | A_1 = 4] =$ ______ / ______

$Pr[C_1 = (2,2) | A_1 = 4] =$ ☐ / ☐

$Pr[C_1 = (6,2) | A_1 = 4] =$ ☐ / ☐

(c) In this part, Pacman's goal is to reach the position $(g_x, g_y) = (1,2)$ with the minimal cost. The cost of a move from (x_t, y_t) to (x_{t+1}, y_{t+1}) is $1 + E_{t+1}$. For example, a move from $(3,2)$ to $(2,2)$ has the cost of 1, and a move from $(2,2)$ to $(3,2)$ has the cost of 2. Note that even if Pacman stays in the same square after the move, Pacman still needs to pay the cost; e.g., moving left from $(1,1)$ has the cost of 1. Knowing its position at step $t = 0$, Pacman performs a search to find the optimal sequence of actions to take. Let $h(x,y)$ be the cost from (x,y) to (g_x, g_y).

(i) [3 pts] Which of the following are admissible heuristics for $h(x,y)$? Note that we know $(g_x, g_y) = (1,2)$, i.e. the letter Q.

- ☐ $|x - g_x| + |g_y - y|$
- ☐ $\min(|x - g_x|, |g_y - y|)$
- ☐ $\max(|x - g_x|, |g_y - y|)$
- ☐ $2\min(|x - g_x|, |g_y - y|)$
- ☐ $\min(|x - g_x|, |g_y - y|) + 1$
- ☐ $\max(|x - g_x|, |g_y - y|) + 1$
- ☐ $\max(|x - g_x|, |g_y - y|) + \mathbf{I}[x > 3]$, where $\mathbf{I}[x > 3] = 1$ when $x > 3$, and $\mathbf{I}[x > 3] = 0$ otherwise.
- ◯ None of the above

(ii) [2 pts] In this part and onwards, Pacman does not know the position at step $t = 0$, but can infer possible positions according to the evidence.
Pacman observes that $A_0 = 2$ and $E_0 = 1$ at step $t = 0$. What is the cost of the path found by A* Tree Search with an admissible heuristics?

☐

(iii) [2 pts] Pacman observes that $A_0 = 3$ and $E_0 = 0$ at step $t = 0$. We want to find an action sequence such that Pacman is guaranteed to be in $(g'_x, g'_y) = (1,1)$ after performing this action sequence. We also want to minimize the worst-case cost of this sequence among all possible initial positions (x_0, y_0). Pacman does not have access to A_t or E_t when $t > 0$. What is the worst-case cost of this action sequence among all possible initial positions?

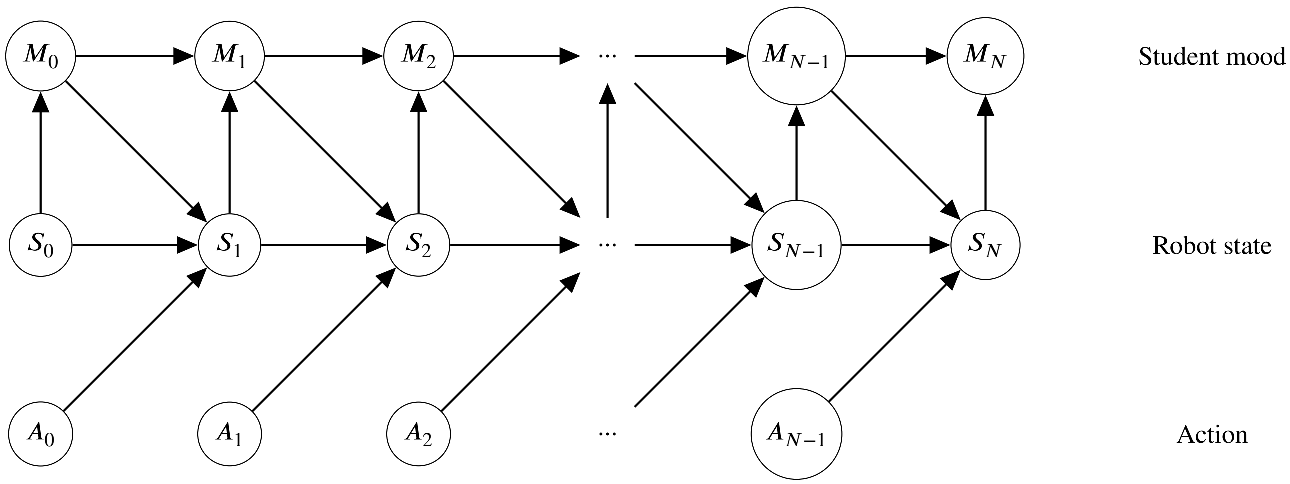
☐

Q5. [17 pts] Accommodating Course Robot

The CS188 GSIs are designing an instructional robot to improve the mood of the students.

At every timestep t, the robot stores in its state a list of ongoing CS188 assignments S_t, and the robot takes action A_t to release a list of future assignments S_{t+1}. The released assignments S_{t+1} are also improved by student mood and feedback from the previous timestep t, denoted M_t.

This gives us the Bayes net below.



(a) [1 pt] Which of the below represent the conditional probability table for the variable S_{t+1} in the Bayes net shown above? We denote this quantity $T(S_{t+1})$, as it is the transition model for S at time $t+1$.

- ☐ $P(S_{t+1}|S_t, A_t)$
- ☐ $P(S_{t+1}|S_t, A_t, M_t)$
- ☐ $P(S_{t+1}|S_t, A_t, M_{t-1})$
- ○ None of the above

(b) [1 pt] Which of the below represent the conditional probability table for the variable M_{t+1} with the Bayes net shown above? We denote this quantity $T(M_{t+1})$, as it is the transition model for M at time $t+1$.

- ☐ $P(M_{t+1}|M_t)$
- ☐ $P(M_{t+1}|M_t, A_t)$
- ☐ $P(M_{t+1}|S_t, A_t)$
- ☐ $P(M_{t+1}|S_t)$
- ☐ $P(M_{t+1}|S_{t+1})$
- ☐ $P(M_{t+1}|M_t, S_{t+1})$
- ○ None of the above

(c) [2 pts] What is the joint transition model for variables (S_t, M_t) in the Bayes net above? We denote this quantity $T(S_{t+1}, M_{t+1})$. Include all choices that can be proved to be equivalent using conditional independence assertions expressed by the Bayes net.

- ☐ $P(S_{t+1}, M_{t+1}|S_t, M_t, A_t)$
- ☐ $P(S_{t+1}, M_{t+1}|S_t, M_t)$
- ☐ $T(S_{t+1})T(M_{t+1})$
- ☐ $P(S_{t+1}, M_{t+1}|S_{1:t}, M_{1:t}, A_{1:t})$
- ☐ $P(S_{t+1}, M_{t+1}|S_{1:t})$
- ☐ $P(S_{t+1}, M_{t+1}|A_{1:t})$

☐ $P(S_{t+1}, M_{t+1} | M_{1:t})$
◯ None of the above

(d) [2 pts] Is the above Bayes net Markovian in (S_t, M_t)?
◯ Yes, always Markovian
◯ No, never Markovian
◯ It is possibly Markovian, depending on whether the values $m_{1:t}$ are given
◯ None of the above

(e) [2 pts] You decide the above Bayes net is too complex, and want to try to simplify it down to the MDP format we've learned in class, where the transition dynamics are given by $P(z_{t+1} | z_t, a_t)$ for some single state variable z_t, instead of the transition functions you computed in earlier subparts of this problem. What are valid approaches to do this?
☐ No modifications needed.
☐ Redefine states to be $z_t = [s_t, m_t]$
☐ Redefine states to be $z_t = [s_t, m_{t-1}], \forall t \geq 1$
☐ Perform variable elimination on $m_t, \forall t$
☐ Perform variable elimination on $s_t, \forall t$
◯ None of the above

For the remainder of the question, we denote the transitions in this setting as $(s_t, m_t, a_t, r_t, s_{t+1}, m_{t+1})$, where $r_t = R(m_{t+1})$, since the reward is entirely dictated by the mood of the student, and not the state of the robot. We use discount factor γ.

(f) [2 pts] What is the discounted sum of rewards if the robot observes a sequence of student moods $m_0, ..., m_N$?
☐ $\sum_{t=0}^{N-1} \gamma^t R(m_{t+1})$
☐ $R(m_1) + \gamma V([s_2, m_2])$
☐ $R(m_1) + \gamma \max_{a_1} Q([s_2, m_2], a_1)$
◯ None of the above

(g) [2 pts] Say you wish to run Q-learning on some dataset of transitions $D = \{(s_t, m_t, a_t, r_t, s_{t+1}, m_{t+1})\}_{i=0}^{N-1}$. Each transition is generated by picking a state, taking an action, and observing the result.

You initialize all $Q([s_t, m_t], a_t) = 0, \forall (s_t, m_t), a_t$, and then use the below approach, largely plugging in the Q-learning algorithm from lecture:

1. Randomly select some transition from the dataset $(s_t, m_t, a_t, r_t, s_{t+1}, m_{t+1})$.
2. Compute $sample = r_t + \gamma \max_{a_{t+1}} Q([s_{t+1}, m_{t+1}], a_{t+1})$
3. $Q([s_t, m_t], a_t) \leftarrow \alpha(sample) + (1 - \alpha) Q([s_t, m_t], a_t)$

Given the transition dynamics encoded by the Bayes net in this problem, select whether or not this algorithm needs additional modifications, and why.

◯ Yes. Since the transition dynamics differ from an MDP, we need to account for that in our Q-learning update by weighing the samples according to their likelihood of occurring.
◯ No. The samples are generated from the true dynamics and will, after a sufficient number of updates, represent the empirical probabilities of the true transition probabilities.

(h) If we were to run value iteration, what would the update formula be for the value $V([s_t, m_t])$, the expected sum of discounted rewards acting optimally from (s_t, m_t)?

For each letter **(A), (B), (C), (D), (E)** fill in a single entry for the term corresponding to the correct equation. Recall the definition of $T(s_{t+1})$, $T(m_{t+1})$, and $T(s_{t+1}, m_{t+1})$ from earlier subparts of this problem.

$$V([s_t, m_t]) \leftarrow \textbf{(A)}\ \textbf{(B)}\ \textbf{(C)}\ \left[\textbf{(D)} + \textbf{(E)}\ V([s_{t+1}, m_{t+1}])\right]$$

(i) [1 pt] **(A)** ○ $\max_{a}$ ○ $\max_{m_t}$ ○ $\max_{s_t}$ ○ $\max_{s_t,m_t}$ ○ $\sum_{a}$ ○ $\sum_{s_t}$ ○ $\sum_{m_t}$ ○ $\sum_{s_t,m_t}$ ○ 1

(ii) [1 pt] **(B)** ○ $\sum_{a}$ ○ $\sum_{m_{t+1}}$ ○ $\sum_{s_{t+1}}$ ○ γ ○ 0 ○ 1

(iii) [1 pt] **(C)** ○ $\sum_{s_{t+1}} T(s_{t+1}, m_{t+1})$ ○ $T(m_{t+1})$ ○ $T(s_{t+1})$ ○ 0 ○ 1

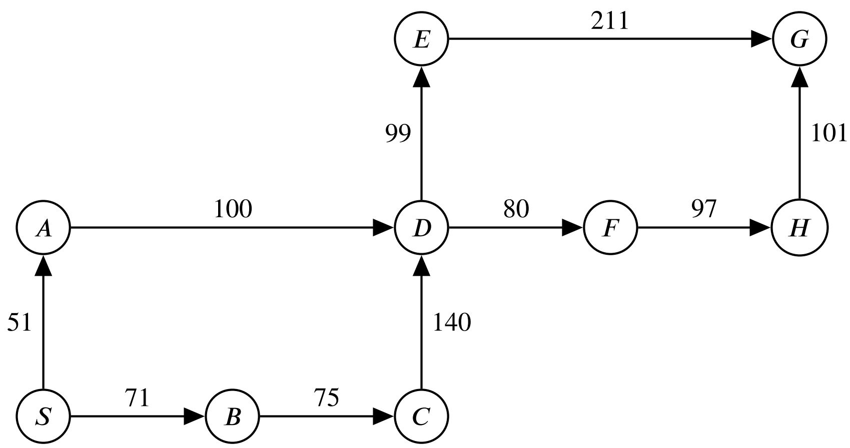
(iv) [1 pt] **(D)** ○ $R(m_{t+1})$ ○ $\sum_{t=1}^{N} R(m_{t+1})$ ○ 0 ○ 1

(v) [1 pt] **(E)** ○ γ ○ 1

Q6. [14 pts] Holiday Planning

After a stressful exam period, you plan to spend some time traveling in your new programmable self-driving car. Given two points on the map and a search procedure, the car calculates the optimal route from start to finish.

(a) Let's first frame this problem as a search problem. Consider the following directed graph:



The edges of the graph indicate the cost of the path between the two nodes connected by that edge. We start at node S and we want to reach node G. For the following sub-questions, assume ties resolve in such a way that states with earlier alphabetical order are expanded first.

(i) [1 pt] We first decide to use the graph-search version of Breadth-First Search (BFS) to find the path from S to G. What is the correct order in which states are going to be expanded?

◯ S, A, D, E, G
◯ S, A, B, D, E, G
◯ S, A, B, D, C, E, F, G
◯ S, A, D, F, H, G

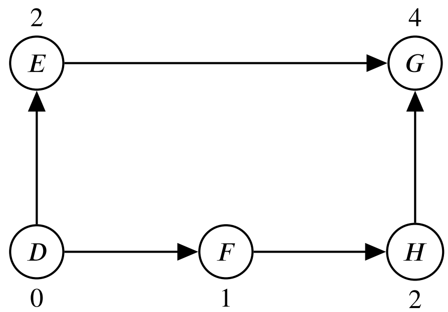
(ii) [2 pts] Now let's take into account the path costs between nodes by running Uniform Cost Search (UCS). What is the *final path* that is going to be returned by UCS?

◯ S, A, D, E, G
◯ S, A, B, D, E, G
◯ S, A, B, D, C, E, F, G
◯ S, A, D, F, H, G

(iii) [2 pts] Imagine that you are trying to implement informed search for a graph similar to the one above. Which of the following statements are true? Select all that apply.

☐ A heuristic function that is consistent must also be admissible.
☐ The graph-search version of A^* search is optimal, as long as our heuristic function is admissible.
☐ A^* search with a heuristic function $h(n) = 0$ for every node n in the graph expands fewer nodes than UCS.
☐ A non-negative heuristic that never overestimates the cost to reach the goal is admissible.
◯ None of the above

(b) To account for probabilistic events, we will now frame the problem as a Markov Decision Process (MDP). We will focus on a subset of the nodes from the graph above:



We start at node D and we want to reach node G. From D we can go either to node F or E. We denote the action of moving from D to F as *Move to F* and the action of moving from D to E as *Move to E*. For other nodes, we say that we *Move* when we go from the node to its neighbor in the direction of the edge. Specifically, we can *Move* from F to H, from H to G, and from E to G.

(i) [2 pts] For actions *Move to F*, *Move to E*, and *Move*, we arrive at the destination node with a probability of 75%, otherwise we transition to a special state called *Broke*. Which of the following values for the transition function are accurate? Select all that apply.

- ☐ $T(D, Move, s) = \frac{1}{4}$, for $s \in \{E, F\}$
- ☐ $T(s, Move, s') = \frac{3}{4}$, for $(s, s') \in \{(F, H), (H, G), (E, G)\}$
- ☐ $T(s, Move, Broke) = \frac{1}{4}$, for $s \in \{D, E, F, H\}$
- ☐ $T(D, Move, G) = 0$
- ◯ None of the above

(ii) [2 pts] At any point we can choose to take the action *Stop*, which will transition us to the special state *Done* and yield the reward indicated next to the node in the graph above. For instance, if we take the action *Stop* at node F, we obtain a reward of 1. Which of the following values for the reward function are accurate? Select all that apply.

- ☐ $R(G, Stop, Done) = 4$
- ☐ $R(s, Move, s') = 0$, for $(s, s') \in \{(F, H), (H, G), (E, G)\}$
- ☐ $R(D, Move\ to\ E, E) = 2$
- ☐ $R(s, Stop, Done) = 2$, for $s \in \{E, H\}$
- ◯ None of the above

(iii) [3 pts] Now recall the policy improvement equation:

$$\forall s,\ \pi_{i+1}(s) \leftarrow \arg\max_a \sum_{s'} T(s, a, s')\ [R(s, a, s') + \gamma V^{\pi_i}(s')]$$

We begin by performing value iteration given the initial policy π_0:

States	D	F	H	E	G
π_0	*Move to F*	*Move*	*Stop*	*Move*	*Stop*
V^{π_0}	1.125	1.5	2	3	4

What is the updated policy π_1 for each of the following states, given the value of the initial policy above? Use the discount factor $\gamma = 1$.

States	D	F	H	E	G
π_1					

(iv) [2 pts] Consider the impact of the discount factor γ on the resulting policy. How would the policy π_1 change if we set the discount factor to $\gamma = 0$?

- ◯ $\pi(D) = Move\ to\ F$, $\pi(F) = Move$, $\pi(H) = Move$, $\pi(E) = Move$, $\pi(G) = Stop$
- ◯ $\pi(D) = Stop$, $\pi(F) = Stop$, $\pi(H) = Stop$, $\pi(E) = Stop$, $\pi(G) = Stop$
- ◯ $\pi(D) = Move\ to\ F$, $\pi(F) = Move$, $\pi(H) = Stop$, $\pi(E) = Move$, $\pi(G) = Stop$
- ◯ $\pi(D) = Move\ to\ E$, $\pi(F) = Move$, $\pi(H) = Move$, $\pi(E) = Move$, $\pi(G) = Stop$