• You have approximately 110 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,
  – □ means mark all options that apply
  – ○ means mark a single choice
  – When selecting an answer, please fill in the bubble or square completely (● and ■)

<table>
<thead>
<tr>
<th>First name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Last name</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>SID</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Student to your right</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Student to your left</td>
</tr>
</tbody>
</table>

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

□ Jasmine (Tu) □ Abhishek (Tu) □ Katie (Tu) □ Allan (W)
□ Brijen (Tu) □ Jaime (Tu) □ Peter* (Tu) □ Kelvin* (W)
□ Gary (Tu) □ Nader (Tu) □ Nick (Tu) □ Anwar (W)
□ Justin (Tu) □ Hong (Tu) □ Ellis* (W) □ Steven (W)

For staff use only:

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1. State Space Sizes</td>
<td>14</td>
</tr>
<tr>
<td>Q2. CSP Potluck</td>
<td>12</td>
</tr>
<tr>
<td>Q3. Bayesics</td>
<td>9</td>
</tr>
<tr>
<td>Q4. Zig-Zag Traversal</td>
<td>18</td>
</tr>
<tr>
<td>Q5. Informed Search</td>
<td>12</td>
</tr>
<tr>
<td>Q6. Analysis of a MDP</td>
<td>14</td>
</tr>
<tr>
<td>Q7. Value Iteration</td>
<td>13</td>
</tr>
<tr>
<td>Q8. Q-Learning</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
</tr>
</tbody>
</table>
Some Useful Formulas

\[ P(A \cap B) = P(A)P(B|A) = P(B)P(A|B) \]

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} \]

\[ S_n = \sum_{k=0}^{n} ar^k = \frac{a(1 - r^{n+1})}{1 - r} \]

\[ |r| < 1 \implies \lim_{n \to \infty} S_n = \sum_{k=0}^{\infty} ar^k = \frac{a}{1 - r} \]
Q1. [14 pts] State Space Sizes

Pacman lives in an $N \times M$ board, but this game differs from a traditional game in a few ways:

- Each food pellet comes in one of three colors (red, blue, or gold) that are worth 1, 2, or 3 points respectively.
- There are no capsules (consuming a capsule increments all ghost scared timers by $S$).
- There are three distinguishable ghosts - Inky, Blinky, and Pinky.
- Pacman’s actions are {turn left, turn right, move forward}.

Let $T > 0$ be the maximum possible value of a ghost’s scared timer. Ghosts can share a space with each other and with Pacman; if Pacman and a ghost overlap, Pacman dies and the game ends. Any food pellet in the space occupied by Pacman is consumed.

Pacman’s goal is to score as many points as possible before the game ends (either by dying or finishing all the pellets). Assume Pacman always has access to the initial configuration of the board.

(a) [9 pts] Select the factors which, when multiplied together, yield the size of a minimal state space representation. If you don’t think a feature belongs in the minimal state space, select “1”. Select one choice per row.

<table>
<thead>
<tr>
<th>Pacman Alive Status</th>
<th>○ 4</th>
<th>○ 3</th>
<th>○ 2</th>
<th>● 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacman Direction</td>
<td>● 4</td>
<td>○ 3</td>
<td>○ 2</td>
<td>● 1</td>
</tr>
<tr>
<td>Pacman Actions</td>
<td>○ 4</td>
<td>○ 3</td>
<td>○ 2</td>
<td>● 1</td>
</tr>
<tr>
<td>Pacman Position</td>
<td>○ $2^{NM}$</td>
<td>○ $N + M$</td>
<td>● $NM$</td>
<td>○ 1</td>
</tr>
<tr>
<td>Food Pellet Configurations</td>
<td>● $2^{NM}$</td>
<td>○ $3^{NM}$</td>
<td>○ $4^{NM}$</td>
<td>○ 1</td>
</tr>
<tr>
<td>Number of Pellets Eaten</td>
<td>○ $NM$</td>
<td>○ $NM - 1$</td>
<td>○ $NM + 1$</td>
<td>● 1</td>
</tr>
<tr>
<td>Pacman Score</td>
<td>○ $3 NM$</td>
<td>○ $3(NM - 1)$</td>
<td>○ $3(NM - 1) + 1$</td>
<td>● 1</td>
</tr>
<tr>
<td>Ghost Scared Timers</td>
<td>○ $3^{T+1}$</td>
<td>○ $(T + 1)^3$</td>
<td>○ $3(T + 1)$</td>
<td>● 1</td>
</tr>
<tr>
<td>Ghost Positions</td>
<td>● $(NM)^3$</td>
<td>○ $(NM - 1)^3$</td>
<td>○ $3(NM - 1)$</td>
<td>○ 1</td>
</tr>
</tbody>
</table>

The size of the state space is:

$$4 \cdot NM \cdot (NM)^3 \cdot 2^{NM}$$

The factors are broken down as follows:

- 4 directions of Pacman
- $NM$ positions of Pacman
- $(NM)^3$ positions of the ghosts. Ghosts can share spaces with Pacman and each other so each ghost has $NM$ possible positions.
- $2^{NM}$ pellet configurations. There are $NM$ spaces that could contain a food pellet. Note that pellet color does not matter since we can determine pellet types from the initial configuration.

Pacman’s actions are an invariant set, so they should be excluded. Ghosts are never scared since there are no capsules, so we don’t track their scared timers. We can tell if Pacman is alive by comparing his position with the positions of the ghosts.

Given the initial food pellet configuration and the current food pellet configuration, we can compute the number of pellets eaten and score.
Now you are told that every time Pacman consumes $P$ pellets, one of the three ghosts at random will become scared for $S$ timer ticks. If all of the ghosts are scared, then one ghost will be chosen at random and its scared timer will be incremented by $S$. Assume that Pacman can eat scared ghosts (thus eliminating them from the board permanently and setting their position to NULL).

(b) [4 pts] Select the features that should be included in a minimal state representation for this game. **Your answer should be independent from part (a).**

- □ Pacman alive status
- ■ Pacman direction
- □ Pacman actions
- ■ Pacman position
- ■ Food pellet configurations
- □ Number of pellets eaten
- ■ Ghost scared timers
- ■ Ghost positions
- ■ Pacman score

The same reasoning from part (a) holds here, but this time we need to track the ghost scared timers since there is now a mechanism to scare ghosts.

(c) [1 pt] Is your state space size larger, smaller, or equal to part (a)?

- • Larger
- ○ Smaller
- ○ Equal

It’s larger since there are more features to track (ghost scared times)
Q2. [12 pts] CSP Potluck

John and Jane want to organize a potluck, so they give their friends a sign-up sheet. Their friends write the number of dishes they are going to bring to the potluck on the sheet. However, once all their friends go home, John and Jane have difficulty reading their friends’ handwriting and also realize they forgot to tell their friends to write their own names on the sheet. Their friends are Alice, Bob, Carl, and Doug.

John and Jane are able to read the sheet enough to gather the following constraints:

1. Each friend is bringing at minimum 1 dish and at maximum 6 dishes. (That is, the dish amount is between the range of 1 and 6, inclusive.)
2. Nobody is bringing more dishes than Alice is.
3. Bob is bringing an odd number of dishes.
4. Carl is bringing twice as many dishes as Bob is.
5. Doug is bringing exactly two more dishes than Carl is.

We will model this as a CSP where the variables are Alice, Bob, Carl, and Doug. The constraints are defined above, and the domains of the variables, after enforcing unary constraints but before we execute any algorithms, are below.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
<th>Carl</th>
<th>Doug</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6</td>
<td>1 3 5</td>
<td>1 2 3 4 5 6</td>
<td>1 2 3 4 5 6</td>
</tr>
</tbody>
</table>

(a) [2 pts] Before we run arc consistency or assign any variables, how many arcs are on the queue?

[10]

In the constraint graph, there is an edge between Alice and Bob, Alice and Carl, Alice and Doug, Bob and Carl, and Carl and Doug. There is no edge between Bob and Doug. As we have 5 edges in the constraint graph, we have 10 arcs initially on the queue.

(b) [4 pts] Enforce the arc from Carl to Doug. What values are left in Carl’s domain? Select all that apply.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

What values are left in Doug’s domain?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Because we enforce Carl to Doug, we remove values from Carl’s domain that violates the constraint “Doug is bringing exactly two more dishes than Carl is.” Therefore, 5 and 6 are removed from Carl’s domain, and Doug’s domain remains unchanged.

(c) [2 pts] Assume that we have the variable domains as shown in the figure at the top of the page (that is, the initial domains after we enforce Bob’s unary constraint). Assuming that we just enforced the arc from Carl to Bob and that there are no arcs on the queue, how many arcs are re-added to the queue?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

After we enforce the arc from Carl to Bob, Carl’s domain is changed. Therefore, we need to re-add to the queue the arcs Alice to Carl, Bob to Carl, and Doug to Carl.
(d) [4 pts] Suppose we run arc consistency to completion. What values are in each person's domain once the algorithm has finished?

Alice: 4 5 6
Bob: 1
Carl: 2
Doug: 4

After we enforce Carl to Doug (taken from part (b)), we have the domains
A 1 2 3 4 5 6
B 1 3 5
C 1 2 3 4
D 1 2 3 4 5 6

After enforcing Bob to Carl and Carl to Bob:
A 1 2 3 4 5 6
B 1
C 2
D 1 2 3 4 5 6

Then, we enforce Doug to Carl, Bob to Carl, and Alice to Carl:
A 2 3 4 5 6
B 1
C 2
D 4

Then, we enforce Alice to Bob, Alice to Doug, and Carl to Doug, which yields us the final solution (changing Alice's domain with enqueue more arcs, but we can see that the solutions will not change despite that):
A 4 5 6
B 1
C 2
D 4
Q3. [9 pts] Bayesics

(a) [6 pts]

(i) [3 pts] Draw edges on the above Bayes net such that the following probability tables are directly reflected on its nodes:

- \( P(A) \)
- \( P(B|A) \)
- \( P(C|A,B) \)
- \( P(D|C) \)
- \( P(E|C,F) \)
- \( P(F) \)

(ii) [3 pts] Each node in a Bayes net represents a table. Given that every variable is binary, how many entries are contained in the tables denoted by the following nodes? Assume that every event that could occur involving a table’s entries is stored as a row in the table.

C: \( 2^3 \)  
E: \( 2^3 \)  
F: \( 2 \)

(b) [3 pts] Remove the minimum possible number of edges such that the Bayes net below is valid and the following independence relationships are guaranteed to hold (to indicate an edge’s removal, please very clearly X it out):

- \( A \perp \perp E|B,C \)
- \( C \perp \perp B|A,D \)
- \( E \perp \perp B|A,D \)

Remove D-B from this graph (remember that our network must be acyclic)
Let’s go over the rest of the edges and see why they don’t suffice:

- A-C, A-D, C-D, and D-E all do not do the job if removed alone because they leave a cycle in the bayes net, so that bayes net is still invalid
• if B-A is removed, then all cycles are broken, but now not all the independence assumptions are met. Specifically, the first one fails. There is an active triple from A to E since, while it is a common effect triple, the descendant of that triple is conditioned on and so it is in fact active.
In the following parts, we will run minimax / expectimax in a slightly different way called Zig-Zag Traversal. At the root level, we will check each child from right to left. At the next level, from left to right and so on. For example, in the following tree, we would visit the nodes in order: a, e, f, g, b, c, d.

(a) [14 pts]
Fill in the resulting values of the GameTree below and mark which edges can be pruned by filling in the corresponding box (Fill in the box None if none of the edges can be pruned). As always, upward triangles are maximizers and downward triangles are minimizers.

(i) [4 pts]

(ii) [6 pts] For the above tree, select the optimal ordering(s) of utility nodes (from left to right) such that the most nodes are pruned in alpha-beta pruning. Notice that you can move the minimizer nodes around and change the order of their respective children, but you cannot trade children between different minimizer nodes because this would result in a different game!

Any solution with (i) 2 or 4 appearing first in the \{2, 9, 4\} subtree, (ii) 3 appearing first in the \{7, 8, 3\} subtree, and (iii) \{6, 7, 8\} being the rightmost subtree (with any internal ordering) is correct.

(iii) [2 pts] In general, using zig-zag traversal, do we get the same solution as regular (left to right) traversal?

- Always
- Sometimes
- Never
(iv) [2 pts] In general, using zig-zag traversal, do we prune the same edges as regular (left to right) traversal?

- Always
- Sometimes
- Never

(b) Consider the following tree and answer the questions that follow. As usual, circular nodes are chance nodes (assume equal probabilities for each child), upward triangles are maximizers and downward triangles are minimizers. As a reminder, the arrow in the root node points to the first child to be visited.

(i) [2 pts] While following the Zig-Zag traversal, which of the following nodes can never be pruned?

- h
- m
- c
- d
- None of the above

The entire subtree containing h, i and j is prunable.

m is prunable if \( \frac{l+n}{2} \geq k \)

c is never prunable, since it is required to assign a value for the chance node that is the first child of the left subtree.

d is prunable since the subtree consisting of d and e is prunable.

(ii) [2 pts] While following the Zig-Zag traversal, which of the following nodes can possibly be pruned?

- e
- g
- j
- i
- None of the above

The entire subtree containing d and e is prunable.

g is never prunable, since it is the first leaf node visited in the middle subtree

The entire subtree containing h, i and j is prunable.
Q5. [12 pts] Informed Search

Pacman is trying to speed run pellet collecting! In this game, Pacman is trying to score 10 points while minimizing
the cost, \(d\), the distance traveled. The goal state is achieved by collecting 10 points, and each step Pacman takes has
a cost of 1. There are 2 types of pellets that he can collect:

1. Constant Pellets, which are each worth 1 point.
2. Linear Pellets, which are each worth \(\max(0, 10 - t)\) points, where \(t\) is the current time step.

(a) [4 pts] Given this problem which of the following are admissible heuristics? Select all that apply.

- [ ] Distance to the closest pellet, 0 if no pellet exists
- [x] Distance to the closest pellet, unless you have reached the goal state, in which you would return 0
- [ ] Number of pellets Pacman has eaten + Distance to the closest pellet
- [ ] 10 - [points eaten already]
- [x] Minimum distance needed to win, assuming all pellets maintain current point value (i.e. Linear Pellets stop decreasing in value)
- [ ] None of the above

For an admissible heuristic, you are required to not have an overestimate cost to the goal state. The above two
are underestimates of the cost to obtaining 10 points.

(b) [4 pts] Which of the following are consistent heuristics? Select all that apply.

- [ ] Distance to the closest pellet, 0 if no pellet exists
- [x] Distance to the closest pellet, unless you have reached the goal state, in which you would return 0
- [ ] Number of pellets Pacman has eaten + Distance to the closest pellet
- [ ] 10 - [points eaten already]
- [x] Minimum distance needed to win, assuming all pellets maintain current point value (i.e. Linear Pellets stop decreasing in value)
- [ ] None of the above

For a consistent heuristic, it must be at least admissible so none of the inadmissible heuristics can be consistent.
However, a consistent heuristic must also underestimate the cost between states. Recall that a consistent
heuristic must hold that \(h(N) - h(P) \leq \text{cost}(N, P) = 1\). In both the second and fifth option, you cannot
decrease the distance, and thus the heuristic, by more then one because you can only move one square. So the
difference is always upper bounded by 1. Both options are therefore consistent.

(c) [4 pts] Assume that Pacman now would like to minimize a new cost function, where he wishes to minimize a
linear combination of Constant Pellets eaten, \(d\) is the total distance traveled, and \(a\) and \(b\) are positive constants whose sum is equal
to 1.

\[
\begin{align*}
C_d &= d \\
C_n &= n \\
C_{\text{sum}} &= a \times d + b \times n
\end{align*}
\]

Given that we already have a heuristic \(h_d\) that is admissible for cost function \(C_d\) and \(h_n\) that is admissible for
cost function \(C_n\), which of the heuristics below are admissible? Select all that apply.

- [x] \(\min(h_d, h_n)\)
- [ ] \(\max(h_d, h_n)\)
The minimum of two admissible heuristics is admissible, as it can never overestimate it. The maximum of two, however, cannot guarantee that it does not overestimate the new cost, $C_{sum}$. The linear combination of admissible heuristics in proportion to the cost is admissible as well. The mean of the two is not admissible for the new cost unless $a = b = 0.5$, thus is not generally admissible.
Q6. [14 pts] Analysis of a MDP

Consider the following three-state MDP, with states A, B and C. Each edge denotes an action between states, and all transitions are deterministic. The rewards for each transition are labeled on the edges of the graph.

The formula for the convergence of an infinite geometric series is \( \sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \) for some \( r \in (0,1) \). You may or may not find this formula useful.

(a) [4 pts] Suppose we ran value iteration for the above MDP. Assume \( \gamma = \frac{1}{2} \). Recall that \( V_0(A) = V_0(B) = V_0(C) = 0 \). After how many iterations will value iteration converge?

○ 0  ○ 1  ○ 2  ○ 3  ○ 4  ○ 5  ● None of the provided

B has a self-loop. Convergence only occurs at the limit.

(b) [6 pts] Now let \( 0 < \gamma < 1 \) (don’t fix \( \gamma = 1/2 \) anymore). Mark all correct expressions for \( V(A) \). If none of the expressions are correct, mark “None of the provided.”

□ \( \max\{ \frac{1+\gamma}{1-\gamma}, \frac{1}{2} \} \)

■ \( \max\{ \gamma V(B), \frac{1}{2} \} \)

□ \( \max\{ \frac{1+\gamma}{1-\gamma}, \frac{1}{2} \} \)

□ \( \gamma \cdot \max\{ V(B), \frac{1}{2} + V(C) \} \)

□ None of the provided

Applying the Bellman equation, \( \max\{0 + \gamma V(b), \frac{1}{2} + \gamma V(c)\} = \max\{ \gamma V(b), \frac{1}{2} \} = \max\{ \frac{\gamma}{1-\gamma}, \frac{1}{2} \} \)

(c) [4 pts] For which \( \gamma \in [0,1) \) is the optimal action from A, to go to B.

● \( \frac{1}{3} \leq \gamma < 1 \)  ○ 0 \( \leq \gamma \leq \frac{1}{3} \)  ○ \( \frac{1}{4} \leq \gamma < 1 \)  ○ \( \frac{1}{3} \leq \gamma \leq \frac{2}{3} \)  ○ None of the provided

We care about when \( \frac{\gamma}{1-\gamma} \geq \frac{1}{2} \). \( 2\gamma \geq 1 - \gamma \Rightarrow 3\gamma \geq 1 \Rightarrow \gamma \geq \frac{1}{3} \)
Q7. [13 pts] Value Iteration

Consider the following finite-state MDP, with states A, B, C, D, and E. Each edge indicates an action between states, and all transitions are deterministic. The edge weights denote the transition rewards. Assume $\gamma = 1$ for all parts.

(a) [5 pts] Let $x = 2$ and $y = 5$. For each of the states, let $k$ be the first iteration in Value Iteration where the values have converged (in other words, the smallest $k$ such that $V_k(s) = V(s)$). Find $k$ for each state.

<table>
<thead>
<tr>
<th>State</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
</tbody>
</table>

Here is the value iteration table, we bold the first iteration where a value converges.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>V(A)</th>
<th>V(B)</th>
<th>V(C)</th>
<th>V(D)</th>
<th>V(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) [4 pts] For which $x$ and $y$ will Value Iteration require 4 iterations before the values converge on every state? (Write this as an inequality in terms of $x$ and $y$).

$x > \max\{0, y - 1\}$

There are three paths from A to E:

- $A \rightarrow C \rightarrow D \rightarrow E$, reward $1 + 1 + 1 = 3$
- $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$, reward $1 + x + 1 + 1 = 3 + x$
- $A \rightarrow B \rightarrow D \rightarrow E$, reward $1 + y + 1 = 2 + y$

To ensure Value Iteration requires 4 iterations, the longest path ($A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$) must have highest reward:

$3 + x > 0$ and $3 + x > 2 + y \implies x > 0$ and $x > y - 1 \implies x > \max\{0, y - 1\}$

(c) [4 pts] Now let $x = 1$ and $y = 1$. In value iteration, at each iteration we compute $V_{k+1}(s)$ for all states $s$, using $V_k(s)$. In async value iteration, at each iteration we choose a single state $s$ and update $\hat{V}(s)$ for that single...
state, according to:

$$\hat{V}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[ R(s,a,s') + \gamma \hat{V}(s') \right]$$

We need to choose an order in which we iterate through and update the values of states.

(i) [2 pts] We initialize $\hat{V}(s) = 0$ for all states, then run async value iteration, iterating through each state exactly once in the order $A,C,B,D,E$. What is $\hat{V}(B)$?

- $-1$
- $0$
- $\frac{1}{2}$
- $1$
- $\frac{3}{2}$
- None of the provided

$\hat{V}(A) = \hat{V}(C) = \hat{V}(D) = 1$, $\hat{V}(B) = 2$

(ii) [2 pts] For each of the orderings below, suppose we initialize $\hat{V}(s) = 0$ for all states, then run async value iteration by iterating through each state exactly once in that order. Mark the orderings for which the resulting $\hat{V}(s)$ have converged to the true value (for all states).

- $E, D, B, C, A$
- $A, B, C, D, E$
- $E, D, C, B, A$
- $A, E, B, C, D$
- $D, C, B, A, E$
- None of the provided

Reverse topological sort. Also, $E$ can be updated whenever because it is a terminal state (so $V(E) = 0$ is correct at the start).
Q8. [8 pts] Q-Learning

Consider the following MDP with 5 states. The agent starts in state c and can move left and right as depicted. The agent receives -1 reward for transitioning from b to a and +1 reward for transition from d to e. The episode ends when the agent reaches state a or e.

(a) [5 pts]
Suppose the agent experiences an episode E1.

If we perform Q learning once on this episode with $\gamma = 1$ and $\alpha = 0.5$ what are $Q(c, R)$ and $Q(d, R)$? All Q values are initialized to zero in the beginning.

- $0, 0$
- $0.25, 0.5$
- $0, 0.5$
- $0, 1$
- $0.5, 1$
- $0.25, 1$
- $0.5, 0.5$
- None of the above

$Q(c, R) = 0.5 \times 0 + 0.5 \times 0 = 0$. $Q(d, R) = 0.5 \times 0 + 0.5 \times 1 = 0.5$

(b) [3 pts] Now disregard the previous experience. Suppose the agent experiences two episodes E2 and E3. If we perform Q learning on each episode once, what are $Q(c, L)$ and $Q(b, L)$?

- $0, 0$
- $-0.25, -0.5$
- $0, -0.5$
- $0, -1$
- $-0.5, -1$
- $-0.25, -1$
- $-0.25, -0.75$
0.5, -0.5

- None of the above

On E2 first iteration does $Q(c, L) = 0.5 \times 0 + 0.5 \times 0 = 0$. $Q(b, L) = 0.5 \times 0 + 0.5 \times -1 = -0.5$. On E3 the second iteration does $Q(c, L) = 0.5 \times 0 + 0.5 \times 0 = 0$. $Q(b, L) = 0.5 \times -0.5 + 0.5 \times -1 = -0.75$