

PRINT Your Name: _____

PRINT Your Student ID: _____

PRINT Student name to your left: _____

PRINT Student name to your right: _____

You have 110 minutes. There are 6 questions of varying credit. (100 points total)

Question:	1	2	3	4	5	6	Total
Points:	19	16	19	19	17	10	100

We reserve the right to deduct points for failing to follow the marking directions below:

For questions with **circular bubbles**, you may select only one choice.

☐ Unselected option (Completely unfilled)

☐ Don't do this (it will be graded as incorrect!)

☒ Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

☐ You can select

☐ multiple squares

☒ Don't do this (it will be graded as incorrect!)

Anything you write outside the answer boxes or you ~~cross-out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation.

Read the honor code below and sign your name.

By signing below, I affirm that all work on this exam is my own work. I have not referenced any disallowed materials, nor collaborated with anyone else on this exam. I understand that if I cheat on the exam, I may face the penalty of an "F" grade and a referral to the Center for Student Conduct.
--

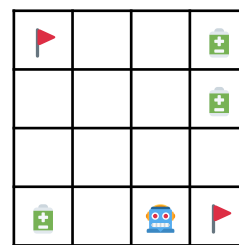
SIGN your name: _____

Q1 Search: Delivery Drone Dilemma

(19 points)

Consider a delivery drone moving on a $N \times N$ grid.

- At each time step, the drone must move to an adjacent square (up, down, left, right). All actions cost 1.
- There are k fixed destination squares on the grid. The drone needs to visit all k destinations at least once.
- The drone starts with a battery charge of C units, and must always have at least 1 unit of charge at every time step. The drone loses 1 unit of charge per action.
- There are p power cells on the grid (with fixed locations). When the drone visits a power cell, its battery charge resets to C units, and the power cell cannot be used again.



Sample grid with $N = 4$,
 $k = 2$ destinations 🚩,
 $p = 3$ power cells 🔋,
 and drone 🤖.

Q1.1 (2 points) What is the maximum branching factor for this problem? Write an integer in the box.

Q1.2 (3 points) What is the size of the smallest state space representation for this problem?

Your answer is the product of all selected choices, e.g. if you select N^2 and C , your answer is N^2C .

☐ A N

☐ C p

☐ E k

☐ G C

☐ B N^2

☐ D 2^p

☐ F 2^k

☐ H C^2

Q1.3 (2 points) Select all algorithms that are guaranteed to find the lowest-cost solution to this problem (assuming one exists).

Note: "Lowest-cost" refers to each action costing 1, which is separate from the battery rules.

☐ A DFS tree search

☐ E UCS tree search

☐ B DFS graph search

☐ F UCS graph search

☐ C BFS tree search

☐ G None of the above

☐ D BFS graph search

Q1.4 (3 points) Suppose we are running BFS tree search, and there are 7 search tree nodes on the fringe.

After we dequeue one node, and enqueue all of its successors, how many nodes could possibly be on the fringe? Select all that apply.

☐ A 5

☐ B 6

☐ C 7

☐ D 8

☐ E 9

☐ F 10

☐ G 11

☐ H 12

(Question 1 continued...)

Q1.5 (5 points) Select all admissible heuristics.

- ☐ A Manhattan distance to the closest unvisited destination, or 0 if all destinations are visited.
- ☐ B Euclidean distance to the furthest unvisited destination, or 0 if all destinations are visited.
- ☐ C Number of unused power cells.
- ☐ D Number of unvisited destinations.
- ☐ E Current battery charge.
- ☐ F None of the above

Q1.6 (2 points) For this subpart, we want to find an *energy-maximizing solution*, which is a solution (not necessarily lowest-cost) that maximizes the battery charge when the drone reaches a goal state.

Select all algorithms that are guaranteed to output an energy-maximizing solution.

- ☐ A BFS graph search, with no modifications.
- ☐ B BFS graph search, modified to continue outputting solutions until the entire fringe is empty.
- ☐ C None of the above

Q1.7 (2 points) For this subpart only, we modify the search problem: The drone must finish visiting all destination squares within T actions.

Do we need to change the state space representation to solve this modified problem?

- ☐ A Yes
- ☐ B No

Q2 CSPs: Confused Subway Planners

(16 points)

Pacman is designing the new BART train schedule for the Oski Line:

- There are 4 variables, corresponding to 4 trains (numbered 101 through 104).
- The values are the possible arrival times at the Berkeley Station: 3:00pm, 3:05pm, ..., 3:55pm, 4:00pm.

The Oski Line's train schedule must satisfy these rules:

1. There must be at least 10 minutes between two trains arriving, e.g. if a train arrives at 3:10pm, then no train can arrive at 3:05pm, 3:10pm, or 3:15pm.
2. Train 103 must arrive earlier than Train 102.
3. A train must arrive at the Berkeley Station at 3:30pm.
4. Train 101 must arrive at the Berkeley Station between 3:00pm and 3:20pm (inclusive).

Q2.1 (2 points) If each rule is represented using a single constraint, which rules must be represented with a **higher-order** constraint? Select all that apply.

- ☐ A 1 ☐ B 2 ☐ C 3 ☐ D 4

Q2.2 (2 points) If each rule is represented using a single constraint, which rules can be represented with a **unary** constraint? Select all that apply.

- ☐ A 1 ☐ B 2 ☐ C 3 ☐ D 4

Consider the partial assignment (with only Train 104 assigned) and domains below:

Train 101: {3:15pm, 3:20pm} **Train 102:** {3:00pm, 3:20pm, 3:45pm, 3:50pm, 3:55pm}
Train 103: {3:20pm, 3:30pm, 3:40pm} **Train 104:** 3:30pm

Subparts Q2.3 and Q2.4 are independent.

Q2.3 (2 points) Which of these arcs is already consistent, without removing values from any domains?

- ☐ A (Train 102 \rightarrow Train 103) only. ☐ C Both are consistent.
☐ B (Train 103 \rightarrow Train 102) only. ☐ D Neither are consistent.

Q2.4 (2 points) Suppose we assign Train 103 to 3:40pm and run forward checking.

Select all values that **remain** in the domain of Train 102.

- ☐ A 3:00pm ☐ B 3:20pm ☐ C 3:45pm ☐ D 3:50pm ☐ E 3:55pm

(Question 2 continued...)

Now, suppose we modify the CSP, such that it outputs train arrival times at both the Berkeley Station and the Ashby Station (the next station down the line).

- All four trains are traveling down the line, from Berkeley to Ashby.
- Every train takes exactly 10 minutes to travel from Berkeley to Ashby. 4:05pm and 4:10pm are added to the domain of every variable.
- The four rules from earlier still apply to Berkeley arrivals.

Q2.5 (2 points) How many total variables are needed to model this new CSP, such that it outputs train arrival times at both stations?

- ☐ (A) 2 ☐ (B) 6 ☐ (C) 8 ☐ (D) 12 ☐ (E) 16

We want to add as few unary and binary constraints as possible to correctly model this new CSP.

Q2.6 (2 points) How many additional unary constraints do we need to add?

- ☐ (A) 0 ☐ (B) 1 ☐ (C) 2 ☐ (D) 3 ☐ (E) 4

Q2.7 (2 points) How many additional binary constraints do we need to add?

- ☐ (A) 0 ☐ (B) 1 ☐ (C) 2 ☐ (D) 3 ☐ (E) 4

Q2.8 (2 points) Does the original CSP (Berkeley only) or the modified CSP (Berkeley and Ashby) have more possible solutions?

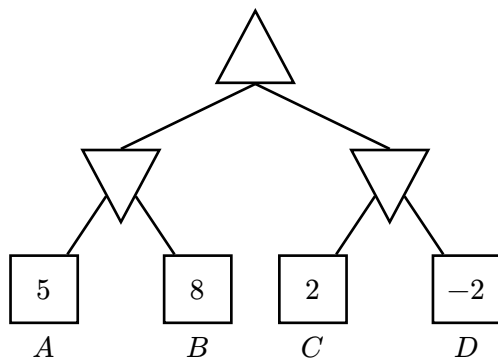
- ☐ (A) Original, because it has fewer constraints. ☐ (C) Both have the same number of solutions.
☐ (B) Modified, because it has more constraints. ☐ (D) Not enough information.

Q3 Games: The Battle of Pac-Man

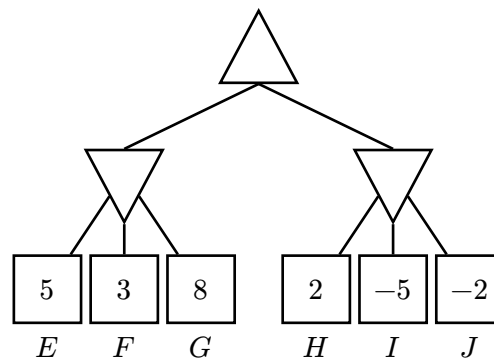
(19 points)

Pacman (maximizer) and Blinky (minimizer) are pirates on rival ships, and they are engaged in a battle!

Pacman first solves a minimax game tree, assuming Blinky has 2 possible actions. Then, Pacman learns that Blinky actually has 3 possible actions, and solves an updated minimax game tree.



Tree 1: before Blinky's extra action



Tree 2: after Blinky's extra action

Q3.1 (1 point) What is the root value for Tree 1?

- ☐ A 5
 ☐ B 8
 ☐ C 2
 ☐ D -2

Q3.2 (1 point) What is the root value for Tree 2?

- ☐ A 5
 ☐ B 3
 ☐ C 8
 ☐ D 2
 ☐ E -5
 ☐ F -2

Consider running alpha-beta pruning for Tree 2, visiting nodes from left to right.

Q3.3 (3 points) Which leaf nodes are **not visited** due to pruning? Select all that apply.

- ☐ E
 ☐ F
 ☐ G
 ☐ H
 ☐ I
 ☐ J

Q3.4 (2 points) For this subpart, suppose the terminal nodes in Tree 2 can take on any values.

Can we ever prune node *H*?

- ☐ A Yes
 ☐ B No

(Question 3 continued...)

For the rest of the question, consider an arbitrary game tree, i.e. not necessarily the trees above.

- The root node is a maximizer node.
- The tree can be any depth, with any values at the terminal nodes.
- When we add an extra action, the original parts of the tree are unchanged.

For each subpart, we want to compare the value at the root *after* adding an extra action, compared to the value at the root *before* adding the extra action.

For each subpart, select all that apply.

In Q3.5 and Q3.6, consider a **minimax** game tree.

Q3.5 (3 points) If we add an extra action to every **minimizer** node, what could happen to the root value?

- | | |
|--|---|
| <input type="checkbox"/> A The value could increase. | <input type="checkbox"/> C The value could stay the same. |
| <input type="checkbox"/> B The value could decrease. | <input type="checkbox"/> D None of the above |

Q3.6 (3 points) If we add an extra action to every **maximizer** node, what could happen to the root value?

- | | |
|--|---|
| <input type="checkbox"/> A The value could increase. | <input type="checkbox"/> C The value could stay the same. |
| <input type="checkbox"/> B The value could decrease. | <input type="checkbox"/> D None of the above |

In Q3.7 and Q3.8, consider a **expectimax** game tree (with only maximizer, chance, and terminal nodes).

Q3.7 (3 points) If we add an extra action to every **chance** node, what could happen to the root value?

- | | |
|--|---|
| <input type="checkbox"/> A The value could increase. | <input type="checkbox"/> C The value could stay the same. |
| <input type="checkbox"/> B The value could decrease. | <input type="checkbox"/> D None of the above |


Q3.8 (3 points) If we add an extra action to every **maximizer** node, what could happen to the root value?

- | | |
|--|---|
| <input type="checkbox"/> A The value could increase. | <input type="checkbox"/> C The value could stay the same. |
| <input type="checkbox"/> B The value could decrease. | <input type="checkbox"/> D None of the above |

Q4 MDPs: Most Drivable Prius

(19 points)

Consider a car agent in the Gridworld below.

+1						+10
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>

Remember that in Gridworld, when you are in the +1 or +10 square, the only action available is “Exit,” and you must take the “Exit” action to earn the reward. In all other squares, the available actions are “Left” and “Right”.

All actions succeed with 100% probability, $\gamma = 0.5$, and there is 0 living reward.

Q4.1 (2 points) When running value iteration, what is the first iteration k where $V_k(C)$ is nonzero?

- Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ 4 Ⓕ 5 Ⓖ 6

Q4.2 (5 points) After running value iteration to convergence, what are the optimal values $V^*(s)$ and optimal policies $\pi^*(s)$ at each of the states? Some blanks are filled in for you.

Write a number in each of the top 5 boxes. Write “Left” or “Right” in each of the bottom 5 boxes.


State s :	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>
$V^*(s)$	1						10
$\pi^*(s)$	Exit						Exit

Consider a modification called *potential-based reward shaping*, where we replace the reward function R with a modified reward function R' :

$$R'(s, a, s') = R(s, a, s') + [\gamma f(s') - f(s)]$$

where f is a *potential function* that maps states to values.

For the Gridworld above, suppose our potential function $f(s)$ is the negative distance between state s and the +10 square, as shown below:

	+1						+10
State s :	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>
$f(s)$:	−6	−5	−4	−3	−2	−1	0

Note: For the terminal state after taking the “Exit” action, $f(\text{terminal state}) = 0$.

Given the $f(s)$ values above and 0 living reward, we can derive these values for $R'(s, a, s')$:

$$\begin{aligned} R'(B, \text{Left}, A) &= 2 & R'(C, \text{Left}, B) &= 1.5 & R'(D, \text{Left}, C) &= 1 \\ R'(B, \text{Right}, C) &= 3 & R'(C, \text{Right}, D) &= 2.5 & R'(D, \text{Right}, E) &= 2 \end{aligned}$$

(Question 4 continued...)

We can now run value iteration with this modified reward function:

$$V'_{k+1}(s) = \max_a \sum_{s'} T(s, a, s') \left[\underbrace{R(s, a, s') + [\gamma f(s') - f(s)]}_{R'(s, a, s')} + \gamma V'_k(s') \right]$$

Q4.3 (3 points) Let $V'_0(s) = 0$ for all states. Fill in $V'_1(s)$ for the states below. Write a number in each box.

State s :	B	C	D
$V'_1(s)$			

Q4.4 (4 points) What is the relationship between $V'^*(s)$ and $V^*(s)$?

Write an expression using any of $V^*(s)$, γ , $f(s)$, $T(s, a, s')$, $R(s, a, s')$, and any mathematical operators (you may not need all of these).

Hint: Value iteration with the modified reward function converges to these values:

State s :	A	B	C	D	E	F	G
$V'^*(s)$	7	5.5	4.625	4.25	4.5	6	10

$V'^*(s) =$

Q4.5 (3 points) We extract a policy using the following equation:

$$\pi'^*(s) = \operatorname{argmax}_a \sum_{s'} T(s, a, s') \left[\underbrace{R(s, a, s') + [\gamma f(s') - f(s)]}_{R'(s, a, s')} + \gamma V'^*(s') \right]$$

What is the policy extracted from the optimal values $V'^*(s)$? Some blanks are filled in for you. Write “Left” or “Right” in each box.

Math hints: $0.5(4.625) = 2.3125$, and $0.5(4.25) = 2.125$.

State s :	A	B	C	D	E	F	G
$\pi'^*(s)$	Exit				Right	Right	Exit

Q4.6 (2 points) Consider any arbitrary MDP and potential function (not necessarily the ones above). We compute the optimal policy once using R , then again using R' .

The optimal policy computed using R is _____ the same as the optimal policy computed using R' .

Hint: Take your expression from Q4.4, and plug it into the equation from Q4.5.

Ⓐ Always

Ⓑ Sometimes

Ⓒ Never

Q5 RL: Biking in Berkeley**(17 points)**

Pacman is biking around campus, trying to find his classes. He models the campus as the Gridworld MDP shown to the right, where the transition probabilities and rewards are all unknown.

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>
<i>I</i>	<i>J</i>	<i>K</i>	<i>L</i>

Remember that in Gridworld, actions don't always succeed. For example, if you move Right from *E*, you sometimes end up in *A* or *I*.

For Q5.1 to Q5.3, Pacman's policy π is to always move Right (except for states where he can only Exit and reach the terminal state End).

Q5.1 (3 points) For this subpart only, Pacman collects many samples, starting in state *E* each time and following π . In other words, in Pacman's samples (s, a, s', r) , state *s* is always *E*.

If you apply model-based learning, which values from the MDP can you estimate from the samples? Select all that apply.

- ☐ $\hat{T}(E, \text{Right}, A)$
☐ $\hat{T}(E, \text{Right}, I)$
☐ $\hat{T}(E, \text{Up}, I)$
- ☐ $\hat{T}(E, \text{Right}, F)$
☐ $\hat{T}(E, \text{Up}, A)$
☐ None of the above

Q5.2 (3 points) For this subpart only, Pacman records one episode, and collects these four samples:

$(E, \text{Right}, F, 3)$ $(F, \text{Right}, J, 0)$ $(J, \text{Right}, K, 2)$ $(K, \text{Exit}, \text{End}, 100)$

If you apply direct evaluation, which values from the MDP can you estimate from the samples? Select all that apply.

- ☐ $V^\pi(E)$
☐ $V^\pi(F)$
☐ $V^\pi(G)$
☐ $V^\pi(J)$
☐ $V^\pi(K)$
☐ $V^\pi(L)$

Q5.3 (2 points) For this subpart only, Pacman records three episodes, and collects these 10 samples:

Episode (i):

- 1: $(E, \text{Right}, F, 3)$
- 2: $(F, \text{Right}, J, 0)$
- 3: $(J, \text{Right}, K, 2)$
- 4: $(K, \text{Exit}, \text{End}, 100)$

Episode (ii):

- 5: $(C, \text{Right}, G, 5)$
- 6: $(G, \text{Right}, K, 10)$
- 7: $(K, \text{Exit}, \text{End}, 100)$

Episode (iii):

- 8: $(F, \text{Right}, G, 0)$
- 9: $(G, \text{Right}, K, 10)$
- 10: $(K, \text{Exit}, \text{End}, 100)$

You apply TD learning (with $\alpha > 0, \gamma > 0$), processing the samples one at a time (from 1 to 10).

Reminder: In TD learning, $V^\pi(s)$ for all states starts at 0 before processing any samples.

Which sample causes $V^\pi(F)$ to become nonzero when that sample is processed?

- ☐ Sample 1
 ☐ Sample 8
- ☐ Sample 2
 ☐ Sample 10
- ☐ Sample 4
 ☐ $V^\pi(F)$ is 0 after processing all samples.

(Question 5 continued...)

Subparts Q5.4 to Q5.6 are each independent.

Q5.4 (4 points) Pacman follows policy π (move Right), and collects a finite number of samples. He runs Q-learning on these samples to compute a table Q_{init} (which has some nonzero entries).

Then, Pacman switches to a policy of picking a random action at each state. He collects many new samples, visiting each state-action pair infinitely often.

He tries processing the new samples in two different ways, using the same α for both runs:

- **Run A:** Initialize Q-values to Q_{init} , then run Q-learning with the new samples.
- **Run B:** Initialize Q-values to all zeroes, then run Q-learning with the new samples.

Select all true statements.

- ☐ **A** If α is decreased to 0 over time, both runs can converge to the same optimal Q^* .
- ☐ **B** With a constant and nonzero α , both runs are guaranteed to converge to fixed Q-values.
- ☐ **C** Since Run A begins with nonzero Q-values, Run A will always converge faster than Run B.
- ☐ **D** If α is decreased to 0 over time, then the learned policies in the runs can differ.
- ☐ **E** None of the above

Q5.5 (2 points) Pacman runs approximate Q-learning, with two features:

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a)$$

- $f_1(s, a)$ represents *hilliness*, a number between 0 (flat) and 1 (steep).
- $f_2(s, a)$ represents *crowd size*, a number between 0 (empty) and 1 (crowded).

After several iterations, Pacman has learned weights $w_1 = 1$ and $w_2 = -5$.

Pacman acts exploitatively (maximizing expected reward) based on these weights.

Select all true statements.

- ☐ **A** If choosing between two Q-states with equal crowd sizes, Pacman prefers the flatter Q-state.
- ☐ **B** If choosing between two Q-states with equal hilliness, Pacman prefers the emptier Q-state.
- ☐ **C** None of the above

Q5.6 (3 points) In general (not necessarily for any specific MDP), when is following an ε -greedy strategy with $\varepsilon > 0$ useful, compared to $\varepsilon = 0$? Select all that apply.

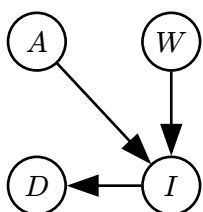
- ☐ **A** When not all states have been visited.
- ☐ **B** When all states have been visited exactly once.
- ☐ **C** When the Q-values have converged to the optimal Q-values.
- ☐ **D** None of the above

Q6 Bayes Nets: Probabilistic Planes

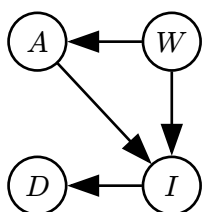
(10 points)

We want to model airplane departures based on several factors. Air traffic (A) and weather (W) affect inbound flight arrivals (I), and I affects the departure time (D).

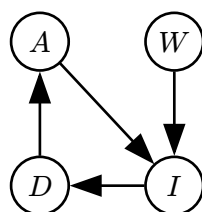
Q6.1 (2 points) Select all of the well-formed Bayes Nets that *could* represent a scenario consistent with the assertions given above (even if the Bayes Net is not the most efficient representation).



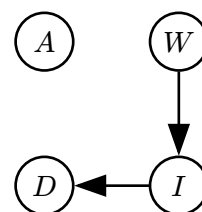
☐ Bayes Net (i)



☐ Bayes Net (ii)



☐ Bayes Net (iii)

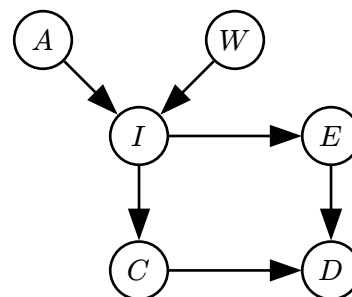


☐ Bayes Net (iv)

For the rest of the question, consider the Bayes Net to the right, with added variables E and C . Assume that all variables are ternary (i.e. each variable has a domain size of 3).

Q6.2 (2 points) How many entries are in I 's CPT (conditional probability table)?

- ☐ 3 ☐ 9 ☐ 27 ☐ 81



For Q6.3 to Q6.6, use d -separation to determine if the independence assumption is true or false.

Q6.3 (1 point) $E \perp\!\!\!\perp C$ ☐ True ☐ False

Q6.4 (1 point) $W \perp\!\!\!\perp D \mid I$ ☐ True ☐ False

Q6.5 (1 point) $A \perp\!\!\!\perp W \mid D$ ☐ True ☐ False

Q6.6 (1 point) $A \perp\!\!\!\perp C \mid E$ ☐ True ☐ False

Q6.7 (2 points) Suppose we combine E and C into one variable EC , as shown below.

The values that EC takes on are tuples of (E, C) , representing all combinations of E 's values and C 's values from the original Bayes Net.

How does the total number of entries across all CPTs change, compared to the original Bayes Net?

- ☐ Increases by 9 entries ☐ Decreases by 9 entries
☐ Increases by 27 entries ☐ Decreases by 27 entries

