CS 188 Fall 2020 — Introduction to Artificial Intelligence — Midterm

- You have approximately 110 minutes.
- The exam is open book, open calculator, and open notes.
- In the interest of fairness, we want everyone to have access to the same information. To that end, we will not be answering questions about the content or making clarifications.
- 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.	Potpourri	/15
Q2.	Demolition Pacman	/17
Q3.	Pacman Marching Band	/10
Q4.	CSP: Cramming for the Exam	/15
Q5.	Learning without Actions	/15
Q6.	BYOR - Bring Your Own Reward	/12
Q7.	Tom and Jerry	/16
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

SID: _____

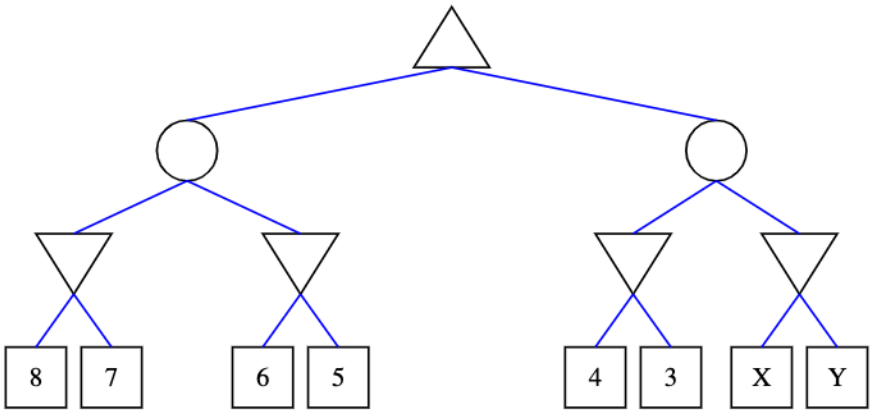
Q1. [15 pts] Potpourri

(a) [2 pts] Suppose we define a modified A* graph search (M-A*GS) that takes a constant $c \geq 1$ and multiplies it by our heuristic before calculating $f(n)$. Essentially in M-A* GS, $f(n) = g(n) + c \cdot h(n)$.

Which of the following statements are true? Select all that apply. Assume that ties of the same priority are broken alphabetically and that $h(n) \geq 0$.

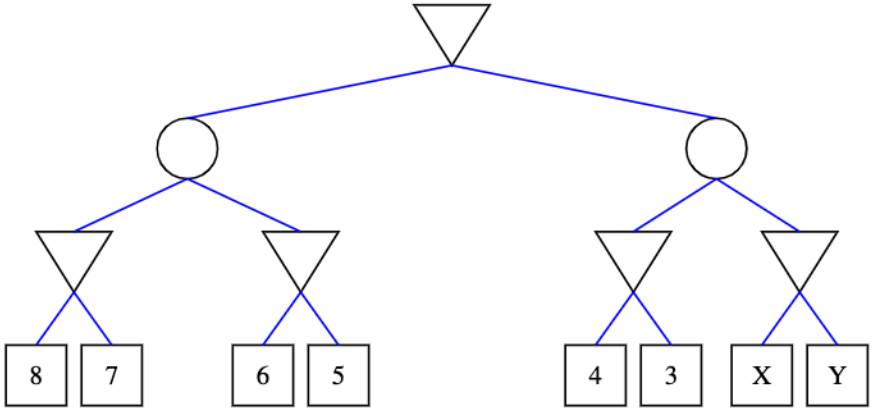
- ☐ M-A*GS will return the same path as unmodified A* graph search for the same graph and heuristic.
- ☐ Given an admissible heuristic, M-A*GS is guaranteed to return the optimal path to the goal.
- ☐ Given a consistent heuristic, M-A*GS is guaranteed to return the optimal path to the goal.
- ☐ The number of nodes expanded in M-A*GS is less than or equal to the number of nodes expanded in unmodified A* graph search for the same graph and heuristic.
- ○ None of the Above

(b) Consider the following game tree, where the upward-facing triangle (the root node) is a maximizer, the upside down triangles are minimizers, the circles are chance nodes which select values uniformly at random, and the squares are leaf nodes.



(i) [2 pts] **Assume that $X \geq 0$, $Y \geq 0$, and that we prune based on equality.** For example, for a maximizer node, if a subtree is guaranteed to propagate a value to its parent that is **less than or equal** to the parent's current value, we prune the remaining nodes in the subtree. What is the maximum possible integer value of X such that we are guaranteed to prune Y? If no such value exists, write 'Not Possible'.

(ii) [2 pts] Suppose we change our root node to a minimizer so the tree looks like the following:



Assume that $X \geq 0$, $Y \geq 0$, and that we prune based on equality. For example, for a minimizer node, if a subtree is guaranteed to propagate a value to its parent that is **greater than or equal** to the parent's current value, we prune

the remaining nodes in the subtree. What is the maximum possible value of X such that we are guaranteed to prune Y? If no such value exists, write 'Not Possible'.

(c) Sick of fighting with ghosts all the time, Pacman decides to start fresh, finding his new passion as a factory worker delivering packages.

We can model his job as an MDP: assume that the factory is a 10 by 10 chessboard, Pacman starts in the lower left corner $(1, 1)$ holding a package, and the goal state G is the upper right corner of the grid $(10, 10)$. Pacman can move up, down, left, or right one square at a time, unless doing so would cause him to leave the board.

Additionally, all of Pacman's actions are deterministic, and when he is at the goal state G, he can only take the *exit* action, which will move Pacman to a terminal state X and end the game.

(i) [1 pt] Denote Pacman's starting location $S = (1, 1)$. Which of the following values will be equal to $V^*(S)$?

Assume that $R(G, exit, X) = 100$ and all other state-action pairs have a reward of 0. Also assume that $\gamma = 1$ and that the values of all states are initialized to value 0 at iteration 0. Select all that apply.

- ☐ $V_{100}(S)$
- ☐ $V_{22}(S)$
- ☐ $V_{20}(S)$
- ☐ $V_{18}(S)$
- ☐ $V_{10}(S)$
- ☐ $V_0(S)$
- ○ None of the Above

(ii) [2 pts] Pacman wants his optimal action at every state to be to head in the direction of the goal state G. For which of the following reward functions will running value iteration **guarantee** that the optimal action will be to head toward G for every state (except at G, where the optimal action should be to *exit*)?

Recall that $G = (10, 10)$. Assume that $\gamma = 1$ and that the values of all states are initialized to value 0 at iteration 0. Select all that apply.

- ☐ $R(s, a, s') = 1$ for non-goal states s and $R(G, exit, X) = 100$
- ☐ $R(s, a, s') = 0$ for non-goal states s and $R(G, exit, X) = 100$
- ☐ $R(s, a, s') = -1$ for non-goal states s and $R(G, exit, X) = 100$
- ○ None of the Above

(iii) [2 pts] Now suppose that Pacman decided to use samples to figure out his Q-values instead, and after several episodes, he arrives at the following Q-values for his starting state $S = (1, 1)$:

- $Q(S, up) = 4.5$
- $Q(S, right) = 10$
- $Q(S, left) = 0$
- $Q(S, down) = 0$

Suppose that Pacman is now back at state S and wants to decide what action he should take using an $\epsilon-$greedy exploration method. What is the probability that the action he chooses is *right*? Your answer may contain ϵ. Assume that any random movements are chosen uniformly from all possible actions.

○ 0	○ $\frac{1-\epsilon}{4}$	○ $\frac{1}{2}$	○ $\frac{3}{4}$
○ $\frac{1}{4}$	○ $\frac{1+3\epsilon}{4}$	○ $1 - \frac{\epsilon}{2}$	○ 1
○ $\frac{\epsilon}{4}$	○ $1 - \frac{3\epsilon}{4}$	○ $1 - \frac{\epsilon}{4}$	○ None of these

(d) You decide to use approximate Q-learning to figure out the optimal policy in a generic MDP. You have n feature functions, $f_1(s, a)$, $f_2(s, a)$, ... , $f_n(s, a)$ and n corresponding weights to learn, w_1 , w_2, ... , w_n that are all initialized to zero. Assume that discount factor $\gamma = 1$ and learning rate $\alpha = 1$.

(i) [1 pt] You observe your first transition from state, s_1 to state s_2, having taken action a_1, and earning a reward of 10. After updating the weights according to the weight update from lecture you noticed w_1 increased. Select the most accurate choice below.

- ◯ $f_1(s_1, a_1)$ must be positive
- ◯ $f_1(s_1, a_1)$ must be negative
- ◯ $f_1(s_1, a_1)$ must be equal to 0
- ◯ There is not enough information to select any of the above choices

(ii) [1 pt] You let the agent run for a while, updating your weights as you go. Then, you observe a transition from state s_3 to state s_4, having taken action a_3, and earning a reward of 10. After updating the weights according to the weight update from lecture you noticed w_1 increased. Select the most accurate choice below.

- ◯ $f_1(s_3, a_3)$ must be positive
- ◯ $f_1(s_3, a_3)$ must be negative
- ◯ $f_1(s_3, a_3)$ must be equal to 0
- ◯ There is not enough information to select any of the above choices

(iii) [1 pt] You finish training your weights and you are about to use them to calculate the optimal policy for each state. However, you accidentally increment one of your weights, w_2, by an unknown amount. You notice the corresponding feature function, $f_2(s, a)$, is actually only dependent on s (i.e.: $f_2(s, a) = g(s)$ for some function g). Could your mistake change the optimal policy you calculate?

- ◯ Your change to w_2 can affect your calculated policy
- ◯ Your change to w_2 cannot affect your calculated policy

(iv) [1 pt] Your friend argues that the true value of w_2 is not very important. They claim that if you continued to train your weights infinitely, on data with sufficient random exploration, w_2 will approach zero. Is your friend correct?

- ◯ Yes, w_2 will approach zero as you continue to train
- ◯ No, w_2 may not approach zero as you continue to train

Q2. [17 pts] Demolition Pacman

Pacman just bought a new square (S) in an M by N grid world that contains some rough terrain. He wants to clear a path from his new home to the closest grocery store, which is another square (G) on the grid world. However, there are R squares on the grid that are impassable because there are rocks in the way. Fortunately, Pacman is carrying $D-1$ sticks of dynamite with him. On any turn, Pacman can throw a stick of dynamite at one of eight neighboring squares (diagonal throws are allowed) if it contains rocks, consuming a stick and removing the rocks in that location. If he chooses not to throw dynamite, Pacman can also drive his truck onto one of eight neighboring squares(diagonal moves are allowed) if there is nothing blocking him. However, his truck only has $F-1$ units of fuel and whenever Pacman moves to a square, one unit of fuel is consumed. Once Pacman runs out of fuel and dynamite he has no more actions available. Assume R, D, and F are all greater than 3 and $2 * D < R$.

(a) [4 pts]

(i) [3 pts] Complete the expression below so that X evaluates to the size of the minimal state space. You can use integers as well as any of the variables mentioned in the problem.

$$X = M^a * N^b * R^c * D^d * F^e * 2^f$$

$a =$ ____ $b =$ ____ $c =$ ____

$d =$ ____ $e =$ ____ $f =$ ____

(ii) [1 pt] What is the maximum possible branching factor?

____

(b) [4 pts] For each subpart below, select all algorithms that can be used to find the specified sequence of actions. The algorithm should be able to find such a sequence on any grid world that contains that sequence without failure. Assume you are allowed to define costs between states however you want.

(i) [1 pt] Any sequence of actions that results in Pacman reaching the Grocery store.

☐ DFS Tree Search ☐ DFS Graph Search ☐ BFS ☐ UCS ○ None of these

(ii) [1 pt] The sequence of actions that results in Pacman reaching the Grocery store while using the least amount of dynamite possible.

☐ DFS Tree Search ☐ DFS Graph Search ☐ BFS ☐ UCS ○ None of these

(iii) [1 pt] The sequence of actions that results in Pacman reaching the Grocery store while using the most amount of dynamite possible.

☐ DFS Tree Search ☐ DFS Graph Search ☐ BFS ☐ UCS ○ None of these

(iv) [1 pt] The sequence of actions that results in Pacman reaching the Grocery store while having the lowest sum of fuel consumed plus dynamite consumed.

☐ DFS Tree Search ☐ DFS Graph Search ☐ BFS ☐ UCS ○ None of these

(c) [9 pts] Each subpart below describes a change to the above problem. **All subparts are independent** of each other, so answer each question as if it was the only change to the above description. **Note, X is equal to the size of the state space from the original problem.** You are essentially finding a scaling factor that should be multiplied by the original state space size.

(i) [3 pts] Pacman upgraded his truck so that he can now throw dynamite towards the north for free. It still costs him dynamite to throw in any of the other seven directions.
Complete the expression below so that it evaluates to the size of the new minimal state space. You can use integers

as well as any of the variables mentioned in the problem.

$$X * R^a * D^b * F^c * 2^d * 3^e$$

$a =$ $b =$ $c =$

$d =$ $e =$

Select all that could possibly be true when comparing to the original problem statement.
- ☐ The new ruleset will cause BFS to expand less nodes
- ☐ The new ruleset will cause BFS to expand the same amount of nodes
- ☐ The new ruleset will cause BFS to expand more nodes

(ii) [3 pts] Y of the R rocks are larger than others ($1 < Y < R$). They require two sticks of dynamite to be removed (Pacman can still only throw one stick of dynamite per turn).
Complete the expression below so that it evaluates to the size of the new minimal state space. You can use integers as well as any of the variables mentioned in the problem.

$$X * R^a * D^b * F^c * 2^d * 3^e * Y^f$$

$a =$ $b =$ $c =$

$d =$ $e =$ $f =$

Select all that could possibly be true when comparing to the original problem statement.
- ☐ The new ruleset will cause BFS to expand less nodes
- ☐ The new ruleset will cause BFS to expand the same amount of nodes
- ☐ The new ruleset will cause BFS to expand more nodes

(iii) [3 pts] Y of the R rocks are now too large for Pacman to destroy ($1 < Y < R$). Dynamite does nothing on them so Pacman will have to move around them.
Complete the expression below so that it evaluates to the size of the new minimal state space. You can use integers as well as any of the variables mentioned in the problem.

$$X * R^a * D^b * F^c * 2^d * 3^e * Y^f$$

$a =$ $b =$ $c =$

$d =$ $e =$ $f =$

Select all that could possibly be true when comparing to the original problem statement.
- ☐ The new ruleset will cause BFS to expand less nodes
- ☐ The new ruleset will cause BFS to expand the same amount of nodes
- ☐ The new ruleset will cause BFS to expand more nodes

Q3. [10 pts] Pacman Marching Band

The University of Blockley has a Pacman marching band, which consists of n members. It is the morning before the Big Game, and all band members need to get together from their homes for one last rehearsal. Suppose you can control all n Pacmen simultaneously. Multiple Pacmen can be on the same square, and every Pacman can pause, or move left, right, up, or down. Suppose there are M squares on the map that are not walls, where the Pacmen can go. Your goal is to move all n Pacmen to the same square in the minimum number of time steps. Let's formulate the problem as a search problem.

(a) [8 pts] Suppose you use A^* search with heuristics to solve the search problem. At any time step, let $p_i = (x_i, y_i)$ be the coordinate of the ith Pacman. For each heuristic below, indicate (1) whether the heuristic is admissible, and (2) whether the heuristic is consistent.

(i) [2 pts] The number of pairs of Pacmen that are not on the same square.

◯ Admissible ◯ Not Admissible
◯ Consistent ◯ Not Consistent

(ii) [2 pts] 2 if the closest pair of Pacmen is 4 squares away, 0 otherwise.

◯ Admissible ◯ Not Admissible
◯ Consistent ◯ Not Consistent

(iii) [2 pts] $\frac{1}{2}\max_{i,j}|x_i - x_j| + \frac{1}{2}\max_{i,j}|y_i - y_j|$

◯ Admissible ◯ Not Admissible
◯ Consistent ◯ Not Consistent

(iv) [2 pts] $\frac{1}{2}\max(\max_{i,j}|x_i - x_j|, \max_{i,j}|y_i - y_j|)$

◯ Admissible ◯ Not Admissible
◯ Consistent ◯ Not Consistent

(b) [2 pts] Suppose you come up with an admissible and consistent heuristic h_a for the previous part based on the positions p_i of all Pacmen. Now the problem changes, and you have $m \leq n$ Pacmen carrying tubas. Because the tuba is heavier than other instruments, the tuba players need to rest for 1 time step after every time step that they move. The rest of the Pacmen remain the same as before. Is h_a still admissible for the new problem? Is it consistent?

◯ Admissible ◯ Not Admissible
◯ Consistent ◯ Not Consistent

SID: _____

Q4. [15 pts] CSP: Cramming for the Exam

Pacman is taking CS 881: Intro to Ghost Intelligence at University of Blockley this semester. It is 7 days before the midterm, but Pacman is still procrastinating! Pacman still has 1 Electronic Homework (E), 1 Written Homework (W), and 1 Project (P) to finish before the exam. Each of them takes 1 day to complete, and Pacman can only work on at most one task every day. Also, Pacman needs 2 days to review the course material before the exam (R_1 and R_2). Pacman needs your help to assign the dates to complete theses tasks!

Pacman formulates the problem as a CSP, where the tasks (E, W, P, R_1, R_2) are variables, each with domain $\{1, ..., 7\}$, representing the seven days from now until the exam.

Pacman wants the assignments of tasks to meet the following constraints:

1. Each task (E, W, P, R_1, R_2) must be assigned to a different day.
2. Both the Electronic Homework (E) and Project (P) are due in 4 days, so they must be finished in days $1, 2, 3$, or 4.
3. Since we use R_1 and R_2 to represent the first and the second day of reviewing for the exam, we assume $R_1 < R_2$, and the two days for reviewing (R_1, R_2) must also **not** be consecutive.
4. Pacman must finish all the assignments (E, W, P) before starting to review for the exam (R_1).
5. The Written Homework (W) can only be completed on even-numbered days.

We recommend summarizing and noting down these constraints for your later reference.

For all the questions below, treat the higher order constraints (1) and (4) as an equivalent series of binary constraints, and treat constraint (3) as a single binary constraint. *E.g., statement (4) represents three binary constraints: $E < R_1$, $W < R_1$, and $P < R_1$. We count them as three distinct constraints.*

(a) Pacman tries to solve the problem with local search, using the min-conflicts heuristic. Pacman starts with an arbitrary assignment: $E = 1, W = 3, P = 6, R_1 = 4, R_2 = 2$. Recall that in min-conflicts local search, Pacman will choose a conflicted variable and reassign its value to the one that violates the fewest constraints in every iteration.

(i) [1 pt] Assume that Pacman decides to reassign R_1. Which value will Pacman choose to reassign the variable to? Break value selection ties from smallest (1) to largest (7).

◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7

(b) **(i)** [3 pts] Pacman notices that local search is not guaranteed to find a solution, so Pacman decides to use backtracking search with arc-consistency instead.

Please select the elements in the domains that will **remain** after enforcing unary constraints and arc consistency. Note that we have not assigned any value to any variable yet.

E	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5	☐ 6	☐ 7
W	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5	☐ 6	☐ 7
P	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5	☐ 6	☐ 7
R_1	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5	☐ 6	☐ 7
R_2	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5	☐ 6	☐ 7

(ii) [1 pt] Pacman decides to do backtracking search using the remaining domains. Recall that in backtracking search we first need to pick a variable and assign a value to it. According to the MRV Heuristic, which variable(s) should Pacman prioritize assigning to? If there are multiple variables tying, mark **all** of them.

☐ E ☐ W ☐ P ☐ R_1 ☐ R_2

(iii) [2 pts] After picking a value with the MRV heuristic, Pacman is not sure whether he should use the LCV heuristic. Select **all** of the following statements about the LCV heuristic that are correct. Note that for a statement to be correct **both the conclusion and the reasoning need to be correct**.

☐ LCV prioritizes assigning a variable to a value that results in the least number of choices in the remaining variables.

☐ LCV makes backtracking search much more efficient because it improves the asymptotic runtime of naive backtracking search.

☐ LCV empirically speeds up backtracking search (compared to selecting a random value to assign) because it often reduces the number of backtracks in the search process.

☐ Depending on the specific setup of a CSP problem, one may not want to use the LCV heuristic because it introduces additional computation.

(iv) [2 pts] Pacman feels tired after formulating and trying to solve the CSP. He adds two new constraints: he wants to take an extra day off, which means no tasks can be assigned to day 1. Also, he wants the Electronic Homework (E) to be completed before the written homework (W). Solve for all the possible solutions (if any) to this CSP after enforcing all the constraints above and these two extra constraints. Which day(s) can the project (P) possibly get assigned to? If there is no possible solution, choose "None".

○ None ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

(c) Pacman survived the midterm! He decides to make a plan for the second half of the semester.

The second half of the semester has N_e electronic homeworks, N_w written homeworks, and N_p projects, and we know $N_e > N_w > N_p$. Each electronic homework, written homework and project needs to be finished in a window of D_e, D_w, and D_p days, respectively, with $D_e < D_w < D_p$. For example, the domain of each variable that represents an electronic homework is $\{1, 2, ..., D_e\}$.

(i) [2 pts] With only the information given above, what is the tightest upper bound for the runtime for finding a satisfying assignment?

○ $O(D_e^{N_e} + D_w^{N_w} + D_p^{N_p})$ ○ $O(D_e^{N_e} D_w^{N_w} D_p^{N_p})$ ○ $O((D_e + D_w + D_p)^{N_e + N_w + N_p})$ ○ $O(\max\{D_e^{N_e}, D_w^{N_w}, D_p^{N_p}\})$

○ $O(D_e^{N_e + N_w + N_p})$ ○ $O(D_p^{N_e + N_w + N_p})$ ○ $O((D_e + D_w + D_p)^{N_e})$ ○ $O((D_e + D_w + D_p)^{N_p})$

Pacman realizes that he also has homework drops! The CS 881 course policy allows at most one electronic homework drop and at most one written homework drop. An additional option, 0, is added to the domain of each variable representing an electronic homework or a written homework. For example, the domain of each variable that represents an electronic homework is now $\{0, 1, 2, ..., D_e\}$, but only one of these variables representing electronic homeworks can be assigned the value "zero".

(ii) [2 pts] The statement "There can be at most one electronic homework drop and at most one written homework drop" involves higher order constraints. Can you write a series of binary constraints that is equivalent to the higher order constraints posed in this statement? If so, what is the size of the smallest set of binary constraints that achieves this?

○ $(N_e(N_e - 1) + N_w(N_w - 1))$ ○ $(N_e^2 + N_w^2)$ ○ $\frac{N_e^2 + N_w^2}{2}$

○ $\frac{N_e(N_e-1) + N_w(N_w-1)}{2}$ ○ $\frac{(N_e-1) + (N_w-1)}{2}$ ○ $(N_e - 1) + (N_w - 1)$ ○ Not Possible

(iii) [2 pts] Suppose that a power outage is expected to take place and the instructor decides to compensate with an additional homework drop which can be used for either written homeworks or electronic homeworks. Now Pacman has a total of three homework drops, while at most two can be used for either electronic homeworks or written homeworks. Can you write a series of binary constraints that is equivalent to the higher order constraints posed in this new statement? If so, what is the size of the smallest set of binary constraints that achieves this?

○ $(N_e(N_e - 1) + N_w(N_w - 1))$ ○ $N_e N_w$ ○ $\frac{N_e N_w}{2}$

○ $\frac{N_e(N_e-1) + N_w(N_w-1)}{2}$ ○ $\frac{(N_e-1)(N_w-1)}{2}$ ○ $(N_e - 1)(N_w - 1)$ ○ Not Possible

SID: _____

Q5. [15 pts] Learning without Actions

We are looking for ways to change the vanilla Bellman equations presented many times in this course. To start, consider a grid world Markov Decision Process (MDP) shown in Figure 1 with **deterministic** dynamics. The agent will start in state A and the only terminal state is I, where it can take any action to exit for a reward of 10. The action space is defined as $\mathcal{A}$: {right, down, left, up, stay}. We define the following values for the MDP:

1. Discount factor $\gamma = 0.5$.
2. Living reward $r(s, a, s') = 0$ for all states $s \neq$ I.
3. Deterministic dynamics: *The next-states in this problem follow directly from the action. E.g.* if the agent selects right, it will move east, unless it is blocked by a wall, in which case the agent will stay in the current grid location, or in a terminal state, where the agent will always exit and get the reward.

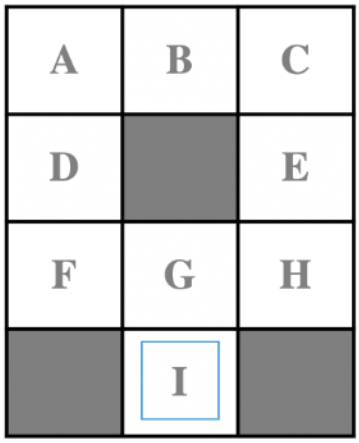


Figure 1: The MDP that we are learning.

With this information, we remember we can do value iteration to learn values of states. Let's start here.

(a) [1 pt] What is the value after one iteration of value iteration for the following states? **To start, all non-terminal states have a value of 0**, $V_0(s) = 0$ and $V_0(I) = 10$.

- $V_1(A) =$ _____
- $V_1(F) =$ _____
- $V_1(G) =$ _____

(b) [3 pts] (1 pt each) What is the optimal value function at each state?

- $V^*(A) =$ _____
- $V^*(F) =$ _____
- $V^*(G) =$ _____

Now, we are working with a different formulation. Given the Markov Decision Process above, we are wondering if we can create a formulation to learn the value of states when we no longer need to consider actions: in a deterministic MDP, the final state encodes the action implicitly; how does this come into play numerically? This can give us a new value function, $Q(s, s')$.

Consider a new model, called an **inverse dynamics model**, $\mathcal{I}(s, s')$, that returns an action given the current state and the next state – this could be useful. Reminder, the action space is $\mathcal{A}$: {**right, down, left, up, stay**}.

$$\mathcal{I}(s, s') : \mathcal{S} \times \mathcal{S} \mapsto \mathcal{A}$$

(c) [3 pts] (1 pt each) As a check for understanding, return the evaluation of the inverse dynamics at the following state-pairs: (If no such action exists, the inverse dynamics model returns *None*.)

- $\mathcal{I}(A, B) =$ _____
- $\mathcal{I}(C, C) =$ _____
- $\mathcal{I}(D, E) =$ _____

(d) [5 pts] Alice wishes to derive an algorithmic procedure that is analogous to Q-value iteration in this new formulation. For each letter **(A), (B), (C)** and **(D)**, fill in a single entry for the term corresponding to the correct equation to implement Q-value iteration. $N(s)$ refers to the *neighboring states* of state s (i.e.: the set of states s' that can possibly be reached by taking some action from state s). Select one bubble per row to form the whole equation.

$$\text{Regular MDP:} \quad Q(s, a) \leftarrow \sum_{s' \in N(s)} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q(s', a') \right] \tag{1}$$

$$\text{New formulation:} \quad Q(s, s') \leftarrow \textbf{(A)} \quad \Big[\textbf{(B)} \quad + \quad \textbf{(C)} \quad \textbf{(D)} \Big] \tag{2}$$

- **(A)** : ○ 1 ○ $\sum_{s'' \in N(s)}$ ○ $\sum_{s'' \in N(s')}$

- **(B)** : ◯ $R\big(s, \mathcal{I}(s, s'), s'\big)$ ◯ 0 ◯ 1
- **(C)** : ◯ γ ◯ 0 ◯ 1
- **(D)** : ◯ $\max_{s'' \in N(s)} Q(s, s'')$ ◯ $\max_{s'' \in N(s)} V(s'')$ ◯ $\max_{s'' \in N(s')} Q(s, s'')$ ◯ $\max_{s'' \in N(s')} V(s'')$ ◯ $\max_{s'' \in N(s')} Q(s', s'')$ ◯ $V(s'')$ ◯ $Q(s', s'')$ ◯ $Q(s, s'')$

(e) Conceptual questions about the new formulation:

(i) [1 pt] In a deterministic MDP like above, can Q-learning with the new value function $Q(s, s')$ learn the optimal policy?

◯ Yes ◯ No

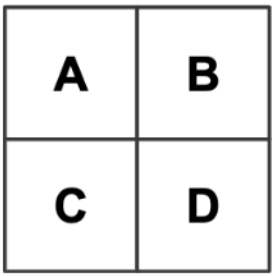
(ii) [2 pts] We can also extend this formulation of Q-learning without actions, $Q(s, s')$, to non-deterministic dynamics $\mathcal{T}(s, a, s')$ as follows. For each transition $(s, a, s', R(s, a, s'))$ we take, we update the corresponding $Q(s, s')$ value according to equation (2), where we replace all occurrences of $\mathcal{I}(s, s')$ by the action we actually take, a. After the Q-values converge, we extract the policy with $\pi(s) = \arg\max_a \sum_{s' \in N(s)} T(s, a, s') Q(s, s')$. Will this new Q-value iteration process always converge to the same policy as vanilla Q-learning in environments with non-deterministic dynamics in $\mathcal{T}(s, a, s')$?

◯ Yes ◯ No

SID: _____

Q6. [12 pts] BYOR - Bring Your Own Reward

Consider the following gridworld as an MDP. You assign your agent Mesut-Bot to traverse it:



From state A, the possible actions are *right* ($\rightarrow$) and *down* ($\downarrow$). From state B, the possible actions are *left* ($\leftarrow$) and *down* ($\downarrow$). For states C and D, the only valid action is to *exit*, after which Mesut-Bot will enter the $done$ state and the rollout will end. We also know that in this MDP all actions are deterministic and always succeed.

Consider the following two episodes. You want to use Q-learning to learn the Mesut-Bot's optimal behavior; however, we don't know the reward value of each sample, and denote them as r_1, r_2, r_3, r_4, and r_5 accordingly (assume $r_1, r_2, r_3, r_4, r_5 \geq 0$).

s	a	s'	r
A	$\rightarrow$	B	r_1
B	$\downarrow$	D	r_3
D	exit	$done$	r_5

s	a	s'	r
A	$\downarrow$	C	r_2
C	exit	$done$	r_4

(a) [3 pts] Suppose that $\gamma = 0.5, r_1 = r_2 = r_3 = 0, r_4 = 1, r_5 = 10$. Calculate the following values after repeatedly processing the episodes one at a time using Q-Learning until convergence. Assume that all Q-values are initialized to 0 and that $\alpha = 1$.

- $Q(A, \rightarrow) =$ _____
- $Q(A, \downarrow) =$ _____
- $Q(B, \leftarrow) =$ _____

(b) Now assume that $\gamma = 1$, $\alpha = 1$ and Mesut-Bot starts from state A. You don't know the rewards, but you know that you want the optimal behavior to be $(A - B - D - done)$. You have the following reward designs at your disposal:

- $S_1 : (r_1 = 0, r_2 = 0, r_3 = 0, r_4 = 0, r_5 = 0)$
- $S_2 : (r_1 = 0, r_2 = 0, r_3 = 0, r_4 = 1, r_5 = 10)$
- $S_3 : (r_1 = 5, r_2 = 5, r_3 = 5, r_4 = 5, r_5 = 5)$
- $S_4 : (r_1 = 1, r_2 = 0, r_3 = 0, r_4 = 2, r_5 = 1)$

(i) [2 pts] Using the same episodes, which of the above reward setup(s) **could result in**, but **do not guarantee**, the optimal behavior after Q-learning converges?

☐ S_1 ☐ S_2 ☐ S_3 ☐ S_4 ○ None

(ii) [2 pts] Using the same episodes, which of the above reward setup(s) will **guarantee** the optimal behavior after Q-learning converges?

☐ S_1 ☐ S_2 ☐ S_3 ☐ S_4 ○ None

(iii) [1 pt] For $0 < \gamma \leq 1$ and $\alpha = 1$, which of the following is the condition for Q-learning to **guarantee** the optimal behavior upon convergence? If none of these are the correct condition, select "None".

○ $r_2 + \gamma r_4 < r_1 + \gamma r_3 + \gamma^2 r_5$ ○ $r_2 + \gamma r_4 = r_1 + \gamma r_3 + \gamma^2 r_5$ ○ $r_2 + \gamma r_4 > r_1 + \gamma r_3 + \gamma^2 r_5$ ○ None

(c) Now we add one more transition to Mesut-Bot's dataset before $(A, \downarrow, C, r_2)$ in the second episode: $(B, \leftarrow, A, r)$, where $r = max(r_1, r_2, r_3, r_4, r_5)$. Assume that $\gamma = 1$ and $\alpha = 1$. The modified episodes and reward designs are repeated below for your convenience:

s	a	s'	r
A	$\rightarrow$	B	r_1
B	$\downarrow$	D	r_3
D	exit	$done$	r_5

s	a	s'	r
B	$\leftarrow$	A	r
A	$\downarrow$	C	r_2
C	exit	$done$	r_4

- $S_1 : (r_1 = 0, r_2 = 0, r_3 = 0, r_4 = 0, r_5 = 0)$
- $S_2 : (r_1 = 0, r_2 = 0, r_3 = 0, r_4 = 1, r_5 = 10)$
- $S_3 : (r_1 = 5, r_2 = 5, r_3 = 5, r_4 = 5, r_5 = 5)$
- $S_4 : (r_1 = 1, r_2 = 0, r_3 = 0, r_4 = 2, r_5 = 1)$

(i) [2 pts] Using the modified episodes, which of the above reward setup(s) **could result in**, but **do not guarantee**, the optimal behavior after Q-learning converges?

☐ S_1 ☐ S_2 ☐ S_3 ☐ S_4 ○ None

(ii) [2 pts] Which of the above reward setup(s) will **guarantee** the optimal behavior after Q-learning converges?

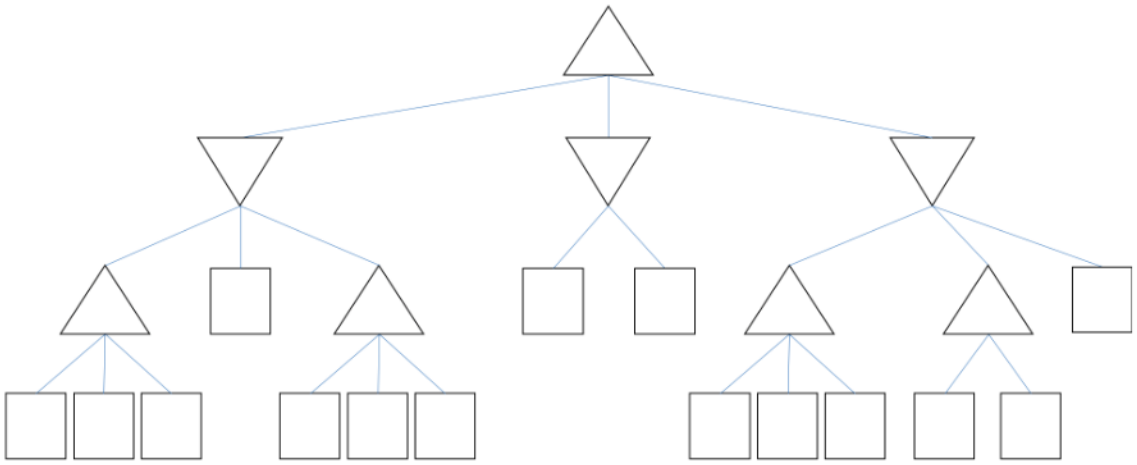
☐ S_1 ☐ S_2 ☐ S_3 ☐ S_4 ○ None

SID: _____

Q7. [16 pts] Tom and Jerry

Bored at home, Tom and Jerry decided to play a game.
For this question, assume that branches are visited in left to right order.

(a) [3 pts] To analyze the game, Tom drew a game tree. However, Jerry quickly erased all the leaf node values in the tree, as shown below, and asked Tom to think about running alpha-beta pruning on game trees with the same structure.



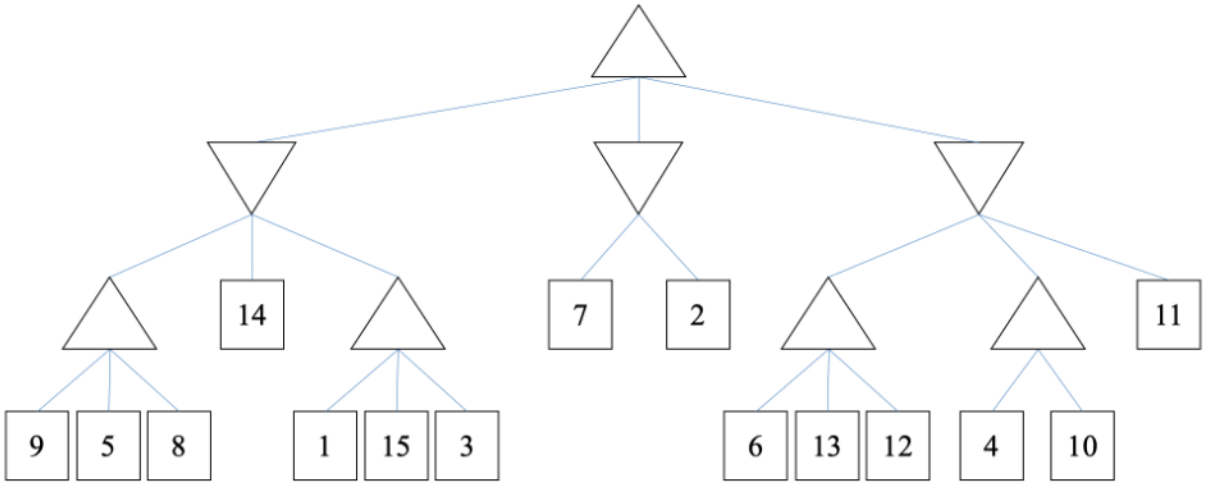
(i) [2 pts]

The maximum possible number of leaf nodes pruned = ☐

(ii) [1 pt]

The minimum possible number of leaf nodes pruned = ☐.

Tom answered the questions correctly, so Jerry filled out the leaf node values.

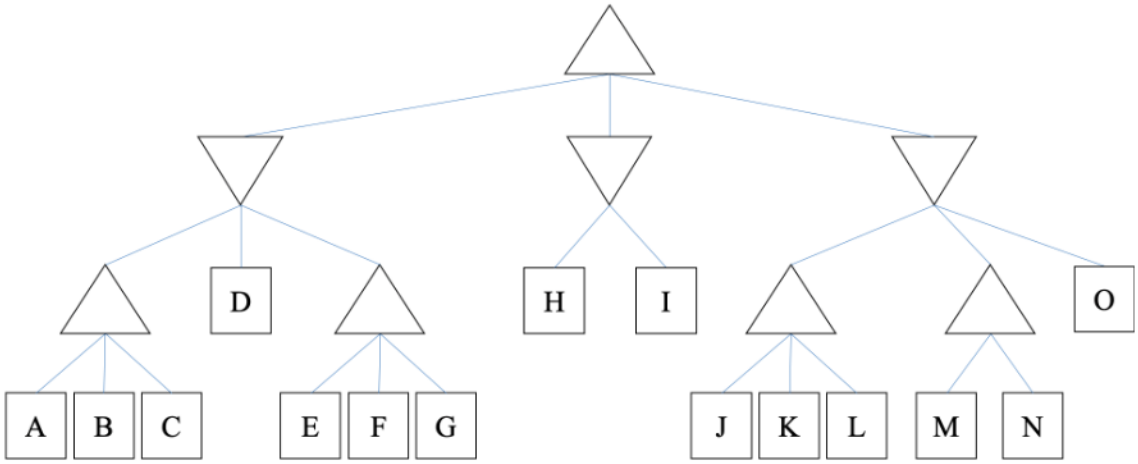


(b) [2 pts] Using alpha beta pruning, how many leaf nodes can be pruned?

☐.

(c) [11 pts] Tom rearranges the **leaf node values from the tree above** such that alpha beta pruning prunes the maximum amount of leaf nodes. Each leaf node value could have been moved to any of the other leaf nodes (or not moved at all).

Assume we are talking about such a tree for all subquestions in this part. **Note, there are 15 leaf nodes and 15 possible values from 1-15 so each value is used exactly once.**
Hint: Tom has a few options in how to rearrange leaves to get max pruning. Try to derive constraints on the values that need to hold for max pruning to happen.
Use this tree with place holder letters to answer the questions below.



(i) [2 pts] If the value of the root is 8, which of the following leaf nodes are guaranteed to have value < 8?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J
☐ K ☐ L ☐ M ☐ N ☐ O ○ None of the above

(ii) [2 pts] If the value of the root is 8, which of the following leaf nodes are guaranteed to have value ≥ 8?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J
☐ K ☐ L ☐ M ☐ N ☐ O ○ None of the above

(iii) [2 pts] If the value of the root is 8, which of the following leaf nodes are guaranteed to have value 8?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J
☐ K ☐ L ☐ M ☐ N ☐ O ○ None of the above

(iv) [2 pts] If the value of the root is 6, which of the following leaf nodes are guaranteed to have value 6?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J
☐ K ☐ L ☐ M ☐ N ☐ O ○ None of the above

(v) [3 pts] Which of the results (root values) are possible for Tom's rearranged game tree?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10
☐ 11 ☐ 12 ☐ 13 ☐ 14 ☐ 15