CS 188 Fall 2021 — 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	/16
Q2.	CSPs: Secret Santa	/17
Q3.	Collaborative Search	/14
Q4.	Morning Commute	/15
Q5.	Games	/19
Q6.	Reinforcement Learning	/19
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

SID: _____

Q1. [16 pts] Potpourri

(a) [3 pts] **Q1.1** Let T be the set of all possible game trees with alternating levels of maximizer and expectation nodes. Consider each of the following conditions independently. For each condition, select the condition if and only if there exists a tree in T such that knowing that condition and no others allows us to prune.

- ☐ When all the leaf node values are bounded by some lower bound
- ■ When all the leaf node values are bounded by some upper bound
- ■ When all the leaf node values are negative
- ☐ It is possible to prune a tree in T, but the necessary condition is not in the above list
- ☐ We can never prune any tree in T

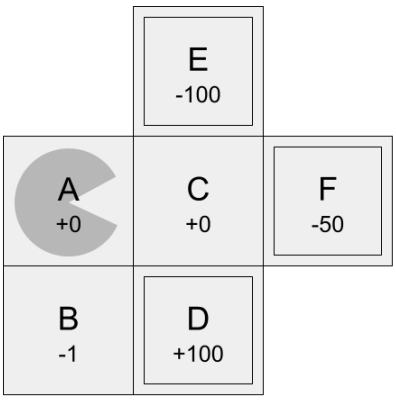
Upper bound: consider a scenario where the maximizer is choosing between two expectimax nodes that each take uniform distributions over children (10, 10, 10) and (0,0,x) respectively. If all values are upper bounded by 10, the last child x can be pruned since $\frac{1}{3}(0+0+x) < \frac{1}{3}(10+10+10)$ for any value of x less than 10.
Negative: Having all leaf values be a negative number is equivalent to applying an upper bound of 0.
Note: You cannot prune on a lower bound because this would not allow you to determine the score of the maximizer node.

(b) [3 pts] **Q1.2** Jason is on Memorial Glade, and he is searching for Soda Hall. He knows that Soda Hall exists and is reachable in some finite distance. For each action, he can only move one step in the direction North, East, South, or West. Each action incurs a cost of 1. Jason can step off campus so he represents his search problem as a graph with infinite nodes. Which of the following algorithms will allow him to eventually reach Soda?

- ■ Breadth First Graph Search
- ☐ Depth First Graph Search
- ■ Uniform Cost Search
- ■ A* tree search with an admissible heuristic
- ☐ None of the above

This is asking which algorithms are complete on an infinite graph. DFS graph search is not complete because it can go down an infinite path that doesn't bring us closer to Soda. BFS will reach the goal since it is a finite distance away, and the same is true for Iterative Deepening and A*.

(c) Consider the following grid. Here, D, E and F are exit states. Pacman starts in state A. The reward for entering each state is reflected in the grid. Assume that discount factor $\gamma = 1$.



(i) [3 pts] Write the optimal values $V^*(s)$ for $s = A$ and $s = C$ and the optimal policy $\pi^*(s)$ for $s = A$.

Q1.3 $V^*(A) =$ 100 **Q1.4** $V^*(C) =$ 100

Q1.5 $\pi^*(A) =$ ○ Up ○ Down ○ Left ● Right

Since the discount factor is 1 and actions are deterministic, all values will eventually converge to the optimal exit value of 100. The optimal action from A is to move right to state C since $Q(A, \text{Right}) = 100$ and $Q(A, \text{Down}) = 99$.

(ii) [4 pts] Now, instead of Pacman, Pacbaby is travelling in this grid. Pacbaby has a more limited set of actions than Pacman and can never go left. Hence, Pacbaby has to choose between actions: Up, Down and Right.

Pacman is rational, but Pacbaby is indecisive. If Pacbaby enters state C, Pacbaby finds the two best actions and randomly, with equal probability, chooses between the two. Let $\pi^*(s)$ represent the optimal policy for Pacman. Let $V(s)$ be the values under the policy where Pacbaby acts according to $\pi^*(s)$ for all $s \neq C$, and follows the indecisive policy when at state C. What are the values $V(s)$ for $s = A$ and $s = C$?

Q1.6 $V(A) =$ 25 **Q1.7** $V(C) =$ 25

We first calculated $V(C)$. Under the indecisive policy, Pacbaby will choose between actions down and right (exit states D and F) with equal probability so $V(C) = \frac{1}{2} \cdot 100 + \frac{1}{2} \cdot (-50) = 25$.
For $V(A)$, Pacbaby is following Pacman's optimal policy which we calculated in the previous part to be the Right action. Since the discount factor is 1 and actions are deterministic, the value of A is $V(A) = 25$.

(iii) [3 pts] Now Pacman knows that Pacbaby is going to be indecisive when at state C and he decides to recompute the optimal policy for Pacbaby at all other states, anticipating his indecisiveness at C. What is Pacbaby's new policy $\pi(s)$ and new value $V(s)$ for $s = A$?

Q1.8 $V(A) =$ 99

Q1.9 $\pi(A) =$ ○ Up ● Down ○ Left ○ Right

From the previous part, we know that $Q(A,\ \text{Right}) = 25$. We also know from part a(i) that $Q(A,\ \text{Down}) = 99$. Therefore, the value of A is the max over all Q-values which would be 99 by taking action Down.

SID: _____

Q2. [17 pts] CSPs: Secret Santa

The CS 188 staff members are playing Secret Santa! Each person brings a gift to the table, intending for a different person to receive that gift. However, past TA Albert, furious that he was not invited to participate, removes all the name cards on each gift so that none of the staff members know whose gift is whose!

The gifts are: AI textbook (A), Backgammon (B), Chess (C), Dinosaur toy (D), Easter egg (E), Flying drone (F).

The participants are: Angela, Daniel, Lexy, Ryan, Saagar, Yanlai

While the staff members have forgotten which gift is for who, Albert has left them some clues.

1. Each person should receive exactly one gift.
2. Yanlai should **not** receive the AI textbook (A).
3. Backgammon (B) should be gifted to either Lexy, Ryan, or Yanlai.
4. The Chess set (C) is **not** gifted to Daniel, Lexy or Saagar.
5. The name of the person who receives Backgammon (B) is alphabetically earlier than the name of the person who receives the AI textbook (A).

We frame this problem as a CSP, with the variables being gifts, and the domain being the six TAs who should receive gifts.

(a) **(i)** [1 pt] **Q2.1** Which of the following constraints are binary constraints?

☐ Constraint 1 ☐ Constraint 2 ☐ Constraint 3 ☐ Constraint 4 ■ Constraint 5

(ii) [2 pts] **Q2.2** Given just these clues, we use **local search** to try to find a satisfying assignment. We'll initialize the assignments alphabetically (i.e. assign the i^{th} person in alphabetical order to the i^{th} gift in alphabetical order). Which of the variables are conflicted in this assignment?

■ A ■ B ■ C ☐ D ☐ E ☐ F ○ None

A and B are conflicted because of constraint 5. C is conflicted because of constraint 4.

(iii) [2 pts] **Q2.3** Say we randomly choose to swap the value of A with the value of some other variable, and use min-conflicts to decide which to swap with. Which of the variables could be selected by the min-conflicts heuristic for A to swap with?

☐ A ☐ B ■ C ☐ D ☐ E ☐ F ○ None

Switching A with C results in only variable B being conflicted. Otherwise, at least both variable B and variable C will be conflicted.

Now Albert gives another clue:

6. The person who should get Backgammon (B) and the person who should get the Dinosaur (D) have names with the same number of letters (Angela/Daniel/Saagar/Yanlai, Lexy/Ryan).

We have provided a full list of constraints along with a table at the end of this problem. Please feel free to use it as you work through each of the subparts. Note that constraint 7 which is introduced later applies only to questions 2.7 and after.

(b) The TAs restart with all the variables unassigned, then enforce all unary constraints and perform arc consistency.

We first enforce unary constraints. Eliminate Yanlai from A. Eliminate Angela, Daniel, and Saagar from B. Eliminate Daniel, Lexy, and Saagar from C.

Next we enforce arc consistency. Eliminate Angela, Daniel, and Lexy from A, because the person who receives A must alphabetically come after the person who receives B. Eliminate Yanlai from B because the person who receives B must alphabetically come before the person who receives A. Eliminate Angela, Daniel, Saagar, and Yanlai from D, because the person who gets B must have the same number of letters as the person who receives D, and only 4-letter names are remaining in the domain of B.

Remaining values for each variable: A: Ryan, Saagar; B: Lexy, Ryan; C: Angela, Ryan, Yanlai; D: Lexy, Ryan; E: everything; F: everything.

(i) [1 pt] **Q2.4** How many values are left in the domain of C?

○ 1 ○ 2 ● 3 ○ 4 ○ 5 ○ 6

(ii) [2 pts] **Q2.5** Using the MRV heuristic, which variable(s) could be assigned next? Select multiple if there's a tie.

■ A ■ B ☐ C ■ D ☐ E ☐ F

(iii) [2 pts] **Q2.6** Say we choose to assign a value for E. Which value should we assign to E according to the LCV Heuristic? Break ties alphabetically.

Assigning E to Daniel would only remove Daniel from the domain of F. Assigning E to any other person would remove that person from more than one other domain (and potentially remove other values from other domains as well, if we perform arc-consistency). Thus, assigning E to Daniel is the Least Constraining Value.

○ Angela ● Daniel ○ Lexy ○ Ryan ○ Saagar ○ Yanlai

Albert now gives one last clue:

7. The Easter egg (E) should be given to Daniel, Ryan, or Yanlai.

(c) We want to use this information to solve our CSP. Continue where we left off from the previous part, and remove Angela, Lexy, and Saagar from the domain of E.

(i) [1 pt] **Q2.7** Given all the constraints we have so far, do we have enough information to fully deduce the value of **any** of the variables?

● Yes ○ No At first glance, it may seem like all the variables have more than one value left in their domain so we can't fully deduce the value of any variable. However, we can notice that if A is assigned to Ryan, then there's no possible way to satisfy constraint 6. So, Saagar must receive A, and so A is a variable whose value we can fully deduce.

(ii) [3 pts] Complete a full recursive backtracking search and identify a satisfying assignment. Apply the MRV heuristic when needed and break any ties alphabetically. Which gift does each person get?

Hint: Try to keep track of assignments by numbering each layer of domain pruning for easier backtracking. It may also help to keep a copy of the variables' domains after only enforcing unary constraints and arc consistency.

Following the MRV heuristic and breaking ties alphabetically, we assign A to Ryan. After enforcing arc consistency, though, we see that there are no more values in the domain of D, so we must backtrack and assign A to Saagar. We then eliminate Saagar from the domains of all the other variables

Next, we apply MRV again and assign B to Lexy. Enforcing arc consistency, we also see that D must be assigned to Ryan. We eliminate Lexy and Ryan from the domains of all the other variables.

We apply MRV one more time and assign C to Angela. Then eliminate Angela from the domains of all the other variables. Finally, apply MRV one last time and assign E to Daniel, which leaves only Yanlai in the domain of F.

Q2.8	A:	○ Angela	○ Daniel	○ Lexy	○ Ryan	● Saagar	○ Yanlai
Q2.9	B:	○ Angela	○ Daniel	● Lexy	○ Ryan	○ Saagar	○ Yanlai
Q2.10	C:	● Angela	○ Daniel	○ Lexy	○ Ryan	○ Saagar	○ Yanlai
Q2.11	D:	○ Angela	○ Daniel	○ Lexy	● Ryan	○ Saagar	○ Yanlai
Q2.12	E:	○ Angela	● Daniel	○ Lexy	○ Ryan	○ Saagar	○ Yanlai
Q2.13	F:	○ Angela	○ Daniel	○ Lexy	○ Ryan	○ Saagar	● Yanlai

(iii) [1 pt] **Q2.14** Which variable is assigned last in the backtracking search?

○ A ○ B ○ C ○ D ○ E ● F

(iv) [2 pts] **Q2.15** How many different solutions does this CSP have? 6

B/D can be Lexy/Ryan. C-E-F can be Angela-Daniel-Yanlai, Angela-Yanlai-Daniel, Yanlai-Daniel-Angela.

(d) [0 pts] **Q2.16** Optional: Which TA do you think would be most happy with their gift?

○ Angela ○ Daniel ○ Lexy ○ Ryan ○ Saagar ○ Yanlai

Q3. [14 pts] Collaborative Search

We have a grid of m by n squares where some edges between adjacent squares are blocked by a wall. An Explorer wants to move from a starting square E to a goal square G. At each time step, the Explorer can move Up (U), Down (D), Left (L) or Right (R) to the adjacent square unless there is a wall on the edge that the Explorer wants to cross, and each move incurs a cost of 1. The locations of the walls are fixed. In addition, some squares are occupied by a booster (B). If the Explorer moves to a square with a booster at time step t, then at time step $t+1$ the booster disappears from the grid, and the Explorer can choose to *teleport* in a certain direction (U, D, L, or R) for 1, 2, 3 or 4 squares (bypassing all the walls along the way), but the move incurs a cost of k the k^{th} time a booster is used.

(a) **(i)** [3 pts] **Q3.1** Assume for now that there are b boosters in the whole m by n grid, numbered $1, 2, \ldots, b$, and the locations of these boosters are fixed and known. The Explorer wants to search for a *minimum-cost* path from E to G. Which of the following quantities should be included in a **minimal but sufficient** state space for this search problem?

- [x] The current position of the Explorer
- [ ] An array of booleans indicating whether each square contains a booster
- [ ] The number of boosters used so far
- [x] An array of booleans of length b, indicating whether each of the b boosters has been used
- [ ] The number of total time steps traveled so far
- [ ] An array of booleans indicating whether each edge is occupied by a wall

Since we know the number of boosters, we only need to keep track of whether each of them has been used.

(ii) [1 pt] **Q3.2** What is the maximum branching factor? 16 Once stepped onto a square with a booster, the Explorer can choose one of the four directions (U, D, L, R), and 1, 2, 3, or 4 squares on that direction. So the maximum branching factor is $4 \times 4 = 16$. Note that "not using the booster" is not an option.

(b) Suppose that we introduce a second agent called a Transporter and we have an unknown number of boosters on the grid to start. The Transporter starts at square T and its moves follow all the same rules as the Explorer, except that when the Transporter moves to a square with a booster at time step t, then at time step $t+1$ it can choose to either move normally to an adjacent square without moving the booster, or carry the booster and move to an adjacent square along with the booster together, and put the booster down at any time. The Explorer and the Transporter can occupy the same square. For each time step, we first control the Explorer to make a move, and then we control the Transporter to make a move.

(i) [3 pts] **Q3.3** We want to search for a *minimum-cost* sequence of actions for both agents which enables the Explorer to reach the goal G. Which of the following quantities should be included in a **minimal but sufficient** state space for this search problem?

- [x] The current position of the Transporter
- [x] An array of booleans indicating whether each square contains a booster
- [x] The number of boosters used so far

Since we do not know the number of boosters in the problem, we need to have one Boolean for each square. We also need to know the number of boosters used so far to calculate the cost of using a booster.

(ii) [1 pt] **Q3.4** What is the maximum branching factor? **Represent your answer in terms of A, the solution to Q3.2.** $8A$ The explorer has a branching factor of A, and the transporter has a branching factor of 8 (it can choose to use the booster or not, and move in any direction).

(iii) [3 pts] **Q3.5** Under which of the following modifications to the state space or to the rules of moving can DFS Tree Search always find a sequence of moves which enables the Explorer to move from E to G, if one such sequence exists (in the modified problem)? These modifications are applied individually and independently.

- [ ] Include in the state space an array of numbers indicating the number of times each square has been visited by Explorer.
- [x] The Transporter is not allowed to visit a previously visited square.
- [x] The Explorer is not allowed to visit a previously visited square.
- [x] The Explorer is not allowed to visit a previously visited square, but after each usage of a booster, the set of previously visited squares is reset to empty.

DFS tree search always find a solution when there is no infinite loop and the state space is finite. Option 1 makes the state space infinite so DFS tree search may still not halt. If the transporter/explorer is not allowed to visit a previously visited square, then there is no infinite loop because the problem cannot return to a previous state.

(c) [3 pts] **Q3.6** Which of the following statements are correct regarding the existence of solutions to the search problems in parts (a) and (b)?

- ☐ When there is at least one booster, the search problem in part (a) always has a solution.
- ☐ When there is at least one booster, the search problem in part (b) always has a solution.
- ■ For a given configuration of booster locations, if the search problem in part (a) has a solution, then the search problem in part (b) also has a solution.
- ■ For a given configuration of booster locations, it is possible that the search problem in part (a) does not have a solution, but the search problem in part (b) has a solution.

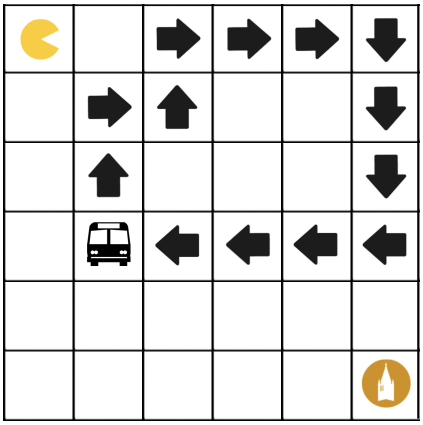
A simplest counterexample for options 1 and 2 is that the Explorer is trapped in a square with walls on all four sides, and there is no booster at that square. Option 3 is correct since the transporter is only there to help the explorer move the boosters. For option 4, consider the case where there is a vertical wall separating the grid into a left and a right region. The left region has 6 columns, and there is one booster at the left-most column. Then without the transporter, the explorer cannot use the booster to cross the wall, if the goal is in the right region.

Q4. [15 pts] Morning Commute

Pacman's apartment is really far from campus, so he decides to model his commute as a search problem for fun. He models Berkeley as an M by N square grid and starts out in the top left square; his goal is to find his way to campus C in the bottom right square in the least number of timesteps. He can move *left, right, up,* or *down*, as long as he doesn't leave the grid.

There is a single fixed bus route on the grid where a bus travels in a cycle at a constant rate of three spaces per timestep. You may assume that any square along this route is a valid bus stop. Pacman knows the route of the bus and if he is on the same square as the bus, **he can get on the bus and travel at a rate of three spaces per timestep along the bus route.** He can travel on the route for any number of timesteps and get off at any square adjacent to the path. Pacman's Clipper card is only good for one use, so he can only enter and exit the bus route at most once per commute.

Below is an example start state on a 6 by 6 grid. The arrows represent the bus's route. Note that this is just an example route, and the bus route does not necessarily have to be in this pattern. For the following questions, consider the general case, **not** the specific example below. **Also consider each part separately (i.e.: do not carry over assumptions).**



(a) [3 pts] **Q4.1** Which of the following are admissible heuristics? Select all that apply.

- ☐ The Manhattan distance between Pacman and campus.
- ■ The Manhattan distance between Pacman and campus divided by 4.
- ☐ The Manhattan distance between Pacman and the bus.
- ☐ The Manhattan distance between the closest bus stop to Pacman and campus.
- ☐ None of the above.

1. Not admissible: Pacman can reach campus in less timesteps than the actual number of spaces between if he takes the bus.
2. Admissible: At best, Pacman would be able to reach campus by moving 3 spaces per timestep, assuming he rides the bus the whole way. Therefore, this heuristic is an underestimate of the true cost of reaching campus.
3. Not admissible: Consider the case where Pacman is already next to campus; the best path does not necessarily involve Pacman's distance to the bus.
4. Not admissible: Consider the case where Pacman is already next to campus; the best path does not necessarily involve Pacman's distance to any bus stop.

(b) [3 pts] **Q4.2** Suppose Pacman knows that there is at least one bus stop adjacent to campus. Which of the following are admissible heuristics? Select all that apply.

- ☐ The Manhattan distance between Pacman and the bus.
- ☐ The Manhattan distance between the bus and the stop next to campus divided by 3.
- ☐ $\frac{1}{3}$ for all states.
- ☐ The minimum of the above heuristics.
- ■ None of the above.

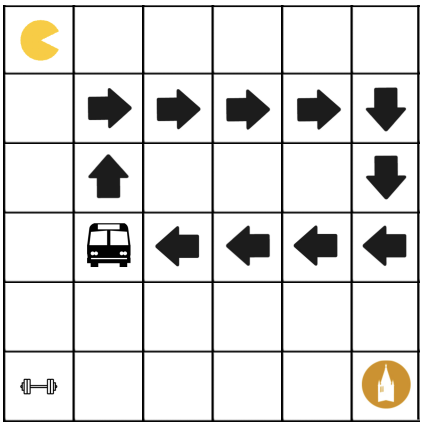
1. Not admissible: The bus could be far away from Pacman, while Pacman could be right next to campus.
2. Not admissible: The bus travels in a fixed route, and its distance from campus does not represent how close Pacman is to achieving his goal.
3. Not admissible: The heuristic's value at the goal state must be 0 for admissibility, even though we underestimate everywhere else.
4. Not admissible: The minimum would be admissible if one of the heuristics achieved 0 at the goal state, because the third heuristic underestimates at all states except the goal state; however, since none of them do, the minimum is still not admissible.

(c) [3 pts] **Q4.3** Suppose that there are B different buses with their own routes. Pacman can now ride the bus multiple times, but he must also ride a bus at least once. Which of the following are admissible heuristics? Select all that apply.

- [x] The minimum of the Manhattan distances between every bus stop and campus, and Pacman and campus.
- [ ] The number of times Pacman has ridden the bus so far.
- [ ] The Manhattan distance from Pacman to the closest bus stop plus the distance from that bus stop to campus divided by three.
- [ ] None of the above.

1. Admissible: Because Pacman must now ride a bus at least once, he must at least travel the Manhattan distance from the closest bus stop to campus at some point. After arriving at said bus stop, the Manhattan distance between Pacman and campus becomes the minimum number of steps necessary to reach the goal. The combination of these two becomes an admissible heuristic.
2. Not admissible: The number of times Pacman has ridden the bus is not representative of how close Pacman is to reaching the goal state.
3. Not admissible: Once Pacman has already ridden the bus to campus, this is an overestimate since it recalculates the distance from Pacman to the closest bus stop even when he could go straight to campus.

(d) [3 pts] Suppose Pacman now wants to work out at the RSF (campus gym), located in the bottom left square of the grid. It doesn't matter if he goes before or after he gets to campus but he must visit both locations. Below is an example start state with the RSF represented by a dumbbell in the bottom left corner:



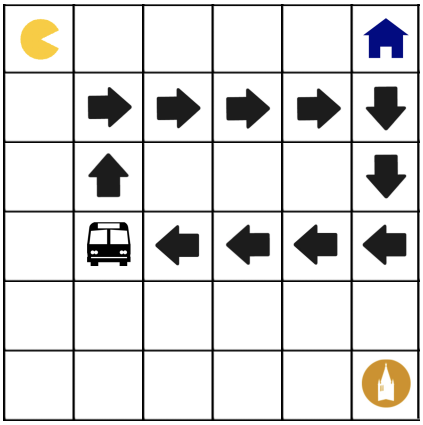
Q4.4 Which of the following are admissible heuristics? Select all that apply.

- [x] The minimum of the Manhattan distances between Pacman and the RSF, the RSF and campus, and Pacman and campus, divided by 3.
- [ ] The sum of the Manhattan distances between Pacman and the RSF, the RSF and campus, and Pacman and campus, divided by 3.
- [ ] The maximum of the above heuristics.
- [ ] None of the above.

1. Admissible: From any space, Pacman is heading to either the RSF or campus, so the minimal Manhattan distance between him and either (divided by 3 assuming he could take the bus the whole way) is less than or equal to the timesteps necessary to reach a goal state.
2. Not admissible: Since Pacman only needs to reach either the RSF or campus before going to the other, summing the distance between Pacman and the RSF with the distance between Pacman and campus overcounts by a factor.
3. Not admissible: Generally the maximum of admissible heuristics would be admissible, but since one of them is not, the maximum isn't.

(e) [3 pts] Suppose that Pacman and his friend Albert want to go to campus together today. Albert lives in an apartment in the top right corner of the grid and wants to meet up with Pacman somewhere before moving together to campus. Assume that Albert moves independently from Pacman, one space per timestep, and that they must meet in another square before either is able to go to campus. Albert cannot take the bus with or without Pacman (although Pacman can still take it alone). In this updated search problem, you can control both Pacman and Albert.

Below is an example start state with Albert's apartment in the top right corner. Albert is currently in his apartment:



Q4.5 Which of the following are admissible heuristics? Select all that apply.

- ☐ The Manhattan distance between Pacman and Albert divided by 2.
- ■ The Manhattan distance from Albert to campus.
- ■ The Manhattan distance from Pacman to campus divided by 3.
- ☐ None of the above.

1. Not admissible: Pacman travelling by bus could move toward Albert faster than Albert can move toward Pacman, making this an overestimate, since it assumes they move towards each other at the same rate.
2. Admissible: Because Albert can't take the bus, he must at minimum move exactly the number of spaces between himself and campus at a rate of 1 space per timestep in order to reach campus.
3. Admissible: Wherever Pacman is in the grid, he must at minimum reach campus to end the search, so this is a relaxation of the problem where he just goes straight to campus via bus and ignores ever meeting Albert. The solution to a relaxed problem is admissible.

SID: _____

Q5. [19 pts] Games

Alice, Eve, and Bob are playing a multiplayer game. Each game state consists of three numbers where the left value represents Alice's score, the middle value represents Eve's score, and the right value represents Bob's score. Alice makes the first move, followed by Eve, and finally Bob. All scores for a single player are **between 1 and 9 inclusive**. In all pruning scenarios, **remember that we do not prune on equality.**

Rather than trying to maximize their individual scores, Alice and Bob decide to work together to maximize their combined score, hoping that this will allow them to score higher. At each of Alice's and Bob's nodes, they will choose the option that maximizes **left value + right value**.

(a) Eve overhears their plan and decides that instead of maximizing her own score, she will try to minimize Alice and Bob's combined score. Alice and Bob are aware of Eve's strategy. Let the value of a node be the sum of the left and right scores of the node. Answer the following questions based on the game tree shown below.

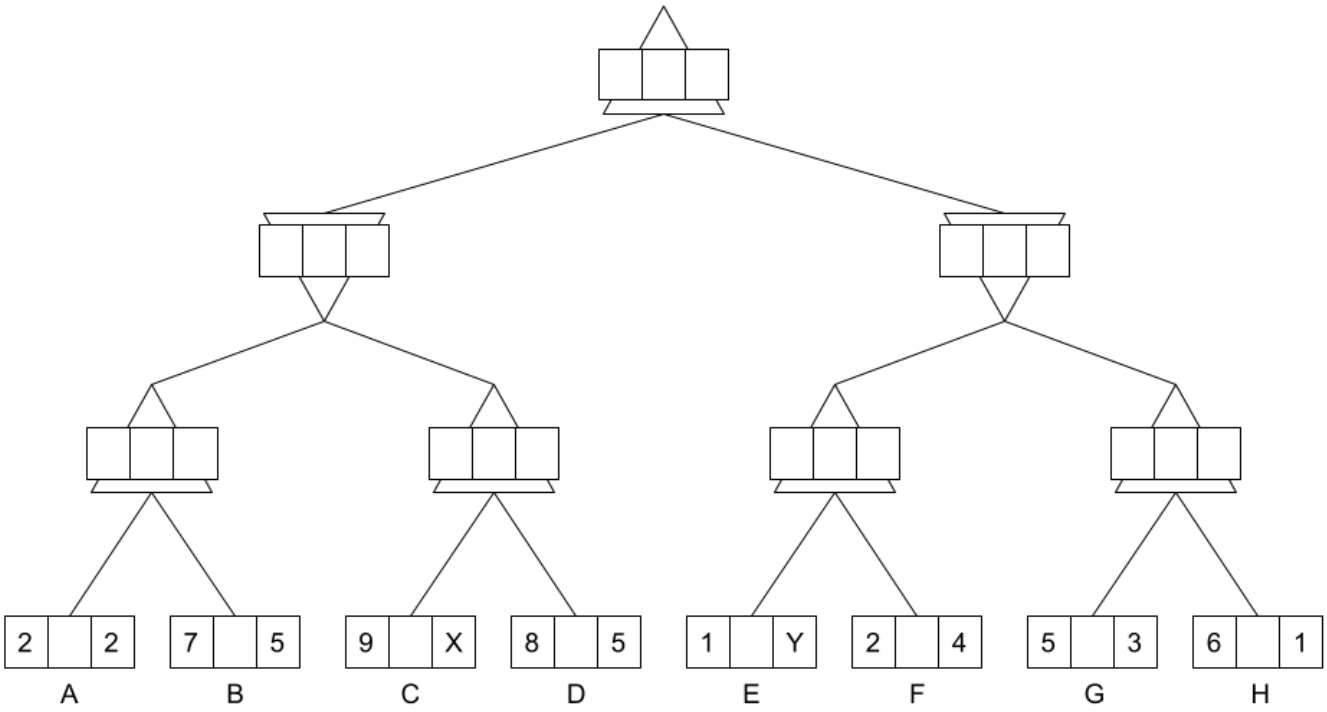


Figure 1: Game tree where Alice is the root maximizer, Eve is the minimizer, and Bob is the bottom maximizer. Eve's score at each node (center cell) is not shown for simplicity.

(i) [1 pt] **Q5.1** Solve the game tree shown in figure 1. What is the value of the root node?

- ○ 6
- ● 12
- ○ 13
- ○ Depends on the value of X only.
- ○ Depends on the value of Y only.
- ○ Depends on the values of both X and Y.
- ○ None of the above.

See diagram below.

(ii) [2 pts] **Q5.2** Without pruning, which of the following are possible values for the right minimizer?

- ☐ < 6
- ■ 6
- ■ 7
- ■ 8

☐ > 8

Consider each child of the right minimizer. The right child has a value of 8 ($\max(5+3, 6+1) = 8$). For the left child, we must consider the bounds on Y. Given that $1 \leq Y \leq 9$, we know that at minimum, the left child is $\max(1+1, 2+4) = 6$ and at most it is $\max(1+9, 2+4) = 10$. So the value of the left child is between 6 and 10. The minimizer is guaranteed to choose the lower of its two children's values so the bounds on its values are between 6 and 8.

(iii) [2 pts] **Q5.3** Which of the following nodes are **guaranteed** to be pruned when running alpha beta pruning on the game tree above? If there are nodes that may or may not be pruned depending on the values of X and Y, do not select them.

☐ A
☐ B
☐ C
☐ D
☐ E
☐ F
■ G
■ H
○ None of the above.

The root maximizer gets a value of 12 from its left subtree. When visiting the right subtree, we know that the maximum value that the maximizer of E and F can have is $\max(1+9, 2+4) = 10$ assuming Y takes on the highest value possible. Since $10 < \alpha = 12$, we can prune the entire right child of the right minimizer. Therefore, G and H are both guaranteed to be pruned regardless of the value of Y since Y is upper-bounded by 9.

(iv) [3 pts] **Q5.4** Which of the following nodes **may or may not** be pruned depending on the values of X and Y? Do not select any nodes from the previous part which are guaranteed to be pruned regardless of the values for X and Y.

☐ A
☐ B
☐ C
■ D
☐ E
☐ F
☐ G
☐ H
○ None of the above.

From the previous part, we know that the value of Y does not affect whether nodes are pruned or not, so here we consider only the possible values of X. In the left subtree, the left child of the minimizer has a value of 12 (from $\max(2+2, 7+5) = 12$). Therefore, if $C > \beta = 12$, then we can prune D. This case occurs if $9 + X > 12$ or when $X > 3$.

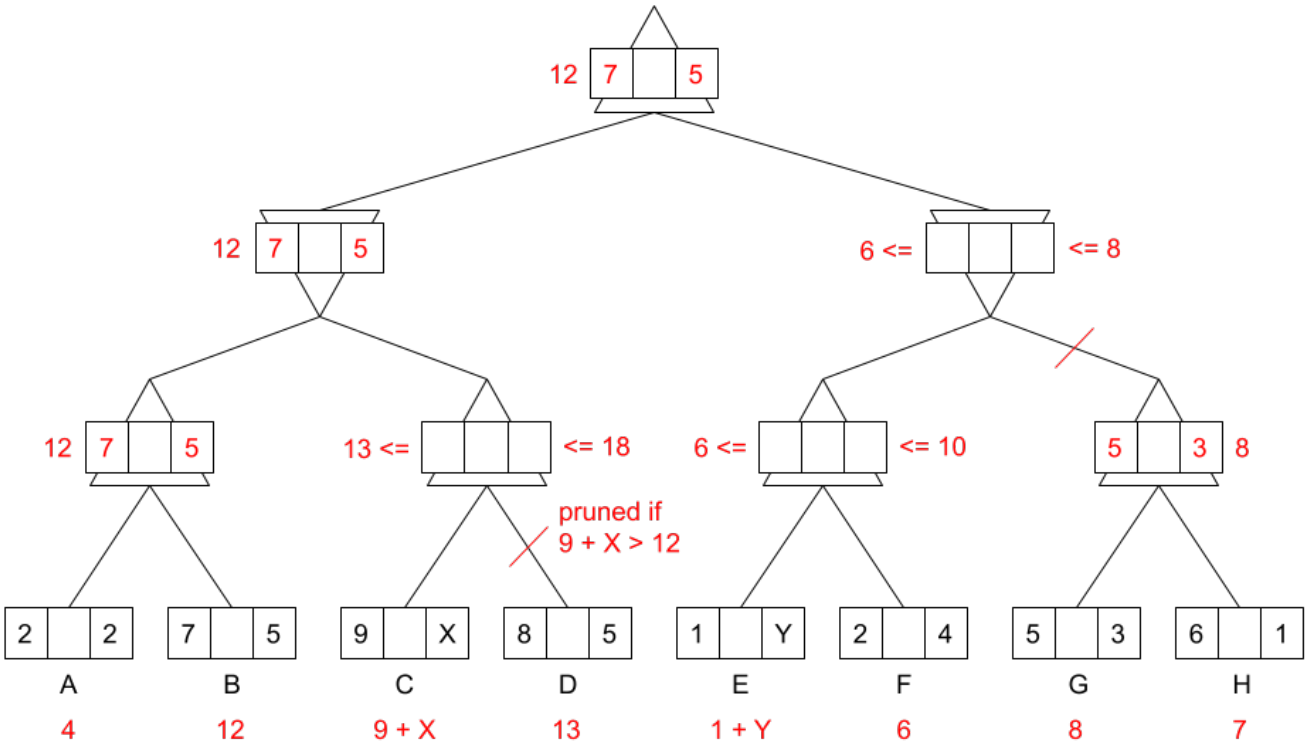


Figure 2: Part a solutions

(b) Eve now decides that in addition to minimizing Alice and Bob's scores, she also wants to maximize her own score. Her new strategy is to choose the option that maximizes her own score minus Alice and Bob's combined score. That is, at Eve's turn, she will choose the option that maximizes **middle value** − (**left value** + **right value**). Alice and Bob are aware of this new strategy. Using the same game tree shown above, assume that we can choose any number between 1 and 9 (inclusive) for X, Y, and Eve's score at each leaf node.

(i) [1 pt] **Q5.5** *True/False*: Compared to Eve's strategy in part (a), Eve's new strategy will result in an equal or higher final score **for Eve** in any leaf node configuration.

● True ○ False

Let V_1 and V_2 be the combined score for Alice and Bob in the previous strategy. In part a, Eve is trying to minimize Alice and Bob's scores, so this is equivalent to choosing the maximum between $-V_1$ and $-V_2$. In part b, Eve factors in her own score so her new strategy is to choose the maximum between $E_1 - V_1$ and $E_2 - V_2$. WLOG, assume Eve chose option 1 in the former case. Then $-V_1 > -V_2$ and she would only change her choice using strategy b if E_2 was significantly greater than E_1 such that $E_1 - V_1 < E_2 - V_2$. Therefore, Eve's score can only be higher.

(ii) [1 pt] **Q5.6** *True/False*: Compared to Eve's strategy in part (a), Eve's new strategy will result in an equal or higher final combined score **for Alice and Bob** in any leaf node configuration.

● True ○ False

Eve's strategy in part a was the optimal adversarial strategy against Alice and Bob. By considering other factors like her own score, Eve becomes an un-optimal adversary since she doesn't always choose the option that most minimizes Alice and Bob's scores. Therefore, Alice and Bob's score can only be the same or increase.

(iii) [3 pts] **Q5.7** Which of the following leaf nodes could possibly be the game outcome if all players play optimally according to their strategy?

- ☐ A
- ■ B
- ■ C
- ■ D
- ☐ E
- ☐ F
- ☐ G
- ☐ H

The bottom level of maximizers will still be the same as in part a since Bob's strategy is not changin, so we can still use the same bounds on those values that we determined in part a. Additionally, we know that the root maximizer will only choose the path that maximizes the values from the bottom maximizers. Regardless of what Eve chooses, we know based on the bounds of the depth 2 maximizers that the best value that the root can achieve from the right subtree is 10 (left maxmimizer can provide between 6 and 10 while right maximizer is 8). 10 is less than all possible options from the left subtree (12 or between 13 and 18) so none of the leaf values of the right subtree are possible outcomes since the root would never choose the right action.

Now that we have narrowed down our options to A, B, C, or D, we know that A will never be chosen because the leftmost maximizer is guaranteed to choose B. Since we don't know the value of the right maximizer, it's possible that the right maximizer could choose either C or D. From the left minimizer's perspective, it is possible to select values for Eve's score that would make it choose either left or right child. Therefore, B, C, and D are all valid possible outcomes.

(iv) [2 pts] **Q5.8** Is it possible to prune in this scenario?

- ● Yes because scores in each cell are bounded between 1 and 9.
- ○ Yes but not for the reason above.
- ○ No because Alice, Bob, and Eve are all acting as maximizers.
- ○ No but not for the reason above.

There are various possibilities here, but since the values of each leaf score are bounded between 1 and 9, it is possible to come up with a scenario where we are guaranteed to be able to prune. Consider the following example which focuses only on the left subtree:

Let $E_B = 9$ and $E_C = 1$ be Eve's value at node B and node C respectively. Also let $X = 8$. The leftmost maximizer will choose node B giving it the scores $[7, 9, 5]$. From Eve's perspective, she will see the value $9 - (7 + 5) = -3$ from her left subtree. From the right side, the maximizer will look at node C which gives $[9, 1, 8]$. We know that the maximizer will only choose D if the left and right scores in D (which we will call D_L and D_R) sum to greater than $9 + 8 = 17$. However, the maximum possible value of Eve's score at node D is 9 due to the bounds on all score values. Therefore, the value Eve gets from D must be $E_D - (D_L + D_R) < 9 - 17 = -8$ which is less than the Eve's current best β value. So after seeing C, we know that Eve would never choose the C and D so node D can be pruned.

(c) Eve is fed up with Alice and Bob teaming up and quits the game. Alice and Bob continue playing and decide to use brand new strategies that incorporate Eve's score for fun. This new game setup can be represented in the diagram below. In each of the following scenarios, Alice and Bob are aware of each other's strategies.

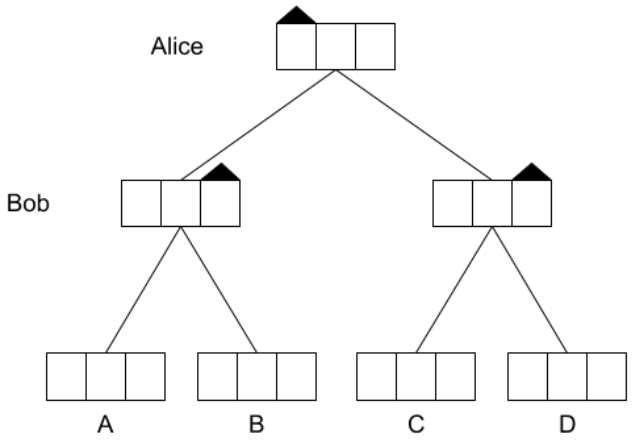


Figure 3: Game tree where Alice is the root and Bob controls the nodes in the middle level. Black triangles above a cell indicate that the cell's value contains the current player's score. (As a reminder, Bob's score is the right value and Alice's score is the left value.)

(i) [2 pts] **Q5.9** Alice and Bob agree to use the following strategy: each player maximizes their own score **plus** the average of the remaining two scores at each node. Assume that you can assign any value between 1 to 9 (inclusive) to all the leaf node scores. Is it possible to prune in this scenario?

- ○ Yes.
- ● No because Alice and Bob are both acting as maximizers.
- ○ No because Alice and Bob are both acting as expectimax nodes.
- ○ No but not for the above reasons.

Let $[S_1, S_2, S_3]$ refer to the scores in a node. Alice is trying to maximize $S_1 + \frac{1}{2}(S_2 + S_3)$ and Bob is trying to maximize $S_3 + \frac{1}{2}(S_1 + S_2)$. Jointly, they both end up maximizing $\frac{1}{2}(S_1 + S_2 + S_3)$ and the extra terms (Alice maximizing an additional $\frac{1}{2}S_1$ and Bob maximizing an additional $\frac{1}{2}S_2$) are not adversarial in any way. Therefore, Alice and Bob are both maximizing the same values so it is not possible to ever prune against the current best α value. Also, note that since we do not prune on equality, we would not prune all nodes after seeing $[9, 9, 9]$ despite it being the maximum possible score given the bounds.

(ii) [2 pts] **Q5.10** Alice and Bob decide to follow a new strategy: each player maximizes their own score **minus** the average of the remaining two scores at each node. Assume that you can assign any value between 1 to 9 (inclusive) to all the leaf node scores. Is it possible to prune in this scenario?

- ● Yes.
- ○ No because Alice and Bob are both acting as maximizers.
- ○ No because Alice and Bob are both acting as expectimax nodes.
- ○ No because Alice and Bob are maximizing different values which are not directly adversarial.
- ○ No but not for the above reasons.

Let $[S_1, S_2, S_3]$ refer to the scores in a node. Alice is trying to maximize $S_1 - \frac{1}{2}(S_2 + S_3)$ and Bob is trying to maximize $S_3 - \frac{1}{2}(S_1 + S_2)$. We can disregard the S_2 term here since it only serves as an equal shift in both players' values. What we can see then is any increase to Alice's score (either due to increase in S_1 or decrease in S_3) will always result in a decrease in Bob's score. Therefore, Alice and Bob are playing an adversarial game and it is possible to prune. The following graph provides a specific example.

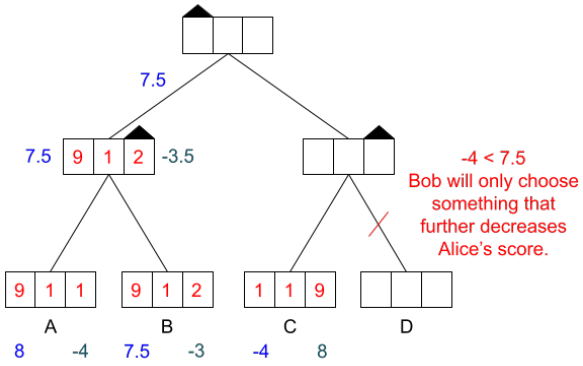


Figure 4: Part c solutions

SID: _____

Q6. [19 pts] Reinforcement Learning

(a) The first part of this problem includes a number of conceptual short questions. Unless otherwise specified, for every subpart and answer choice, assume a discount factor $\gamma < 1$ and rewards are bounded.

(i) [4 pts] **Q6.1** Select all of the following statements about MDP and RL that are true.

- [x] Let π^* be the optimal policy. Then value iteration starting from random values will converge to $V^{\pi^*}(s)$ for all states.
- [ ] Approximate Q-learning is guaranteed to return the optimal policy upon convergence.
- [ ] In environments with deterministic transitions, no exploration is required for Q-learning to converge to the optimal policy.
- [x] A large discount factor γ (approaching 1) on an MDP means that the agent emphasizes long-term rewards.

1. As covered in class both algorithms converge to the optimal policy/values.
2. The linear function approximation (or any other approximation) is not guaranteed to model the Q-values sufficiently and thus nothing can be said about optimality.
3. A deterministic transition function means that from a specific state and for a given action the next state is fixed. Exploration though is still needed to learn optimal policies.
4. $\gamma \to 1$ puts more weight on future rewards than lower values of γ.

(ii) [2 pts] **Q6.2** In which of the following scenarios is Q-learning necessarily preferable compared to vanilla Value Iteration if we want to extract an optimal policy?

- [x] Transition function known and reward function unknown.
- [x] Transition function unknown and reward function unknown.
- [x] Transition function unknown and reward function known.
- [ ] None of the choices.

To extract a policy from value iteration, we need both the reward and transition functions.

(iii) [2 pts] **Q6.3** Assume that we run ϵ-greedy Q-learning until convergence. What is the optimal policy π^* we obtain for an arbitrary state s?

- [ ] $\pi^*(s) = \arg\max_s V(s)$
- [ ] $\pi^*(s) = \begin{cases} \arg\max_a Q(s,a), & \text{w.p. } 1-\epsilon \\ \text{random action w.p. } \epsilon \end{cases}$
- [x] $\pi^*(s) = \arg\max_a Q(s,a)$
- [ ] $\pi^*(s) = \arg\max_s Q(s,a)$

The ϵ exploration is needed only during the learning process. The optimal policy we follow is the greedy one.

(iv) [2 pts] **Q6.4** Following a Boltzmann policy π at each state s an agent is selecting action a with probability $\pi(a) = \frac{e^{Q^*(s,a)/T}}{\sum_{a'} e^{Q^*(s,a')/T}}$. If we increase T the agent will:

- ○ Tend to follow a more greedy policy.
- ● Tend to randomize more among the actions.
- ○ There is no effect on the agent's policy.

In the limit $T \to \infty$ the policy randomizes among the actions while in the limit $T \to 0$ the policy becomes greedy.

(v) [3 pts] **Q6.5** We are running Q-learning with exploration on an agent to determine an optimal policy. At each iteration, the agent will take the argmax over actions of the current Q-value at each state. Which of the following exploration functions will make the agent explore more actions that are rarely sampled initially and then later act greedily as the number of iterations increases? The update rule of Q-learning using an exploration function f is $Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(R\left(s,a,s'\right) + \gamma \max_{a'} f\left(Q\left(s',a'\right), N\left(s',a'\right)\right)\right)$, with $N\left(s',a'\right)$ being the visitation count of the state-action pair s', a', α the learning rate and γ the discount factor.

- [ ] $f\left(Q\left(s',a'\right), N\left(s',a'\right)\right) = Q\left(s',a'\right) + N\left(s',a'\right)^2$
- [x] $f\left(Q\left(s',a'\right), N\left(s',a'\right)\right) = Q\left(s',a'\right) + log\left(N\left(s',a'\right)\right)/N\left(s',a'\right)$
- [x] $f\left(Q\left(s',a'\right), N\left(s',a'\right)\right) = Q\left(s',a'\right) + 1/N\left(s',a'\right)^2$

- [x] $f\left(Q\left(s',a'\right),N\left(s',a'\right)\right)=Q\left(s',a'\right)+1/N\left(s',a'\right)$
- [ ] $f\left(Q\left(s',a'\right),N\left(s',a'\right)\right)=Q\left(s',a'\right)+N\left(s',a'\right)^{1/3}$
- [x] $f\left(Q\left(s',a'\right),N\left(s',a'\right)\right)=Q\left(s',a'\right)+1/\sqrt[3]{N\left(s',a'\right)}$

The bonus function (the second term in the expressions above) must decrease as we visit a state action pair more and more often. If it increases then we will end up visiting the same state-action pairs over and over again.

(b) Now let's look more closely at a Q-learning problem. This is more of a conceptual question and should not require too many calculations. Assume we have a simple MDP with three states A, B, and C, and one action $\rightarrow$. We are given access to the following transitions:

state	action	reward	next state
A	$\rightarrow$	-1	B
B	$\rightarrow$	-1	C
C	$\rightarrow$	-1	A

In what follows, we run Q-learning using the data on the table for an infinite number of iterations.

(i) [1 pt] **Q6.6** If the discount factor γ is 1 and the learning rate $\alpha = 1$ what will the value of $Q(A, \rightarrow)$ be?

- ○ -1
- ○ -2
- ● $-\infty$

This is essentially a cycle and the Q values will keep decreasing by 1 with each iteration.

(ii) [2 pts] **Q6.7** If we decrease the learning rate α to 0.5 in the previous question what will the value of $Q(A, \rightarrow)$ now be after an infinite number of iterations?

- ○ -1
- ○ -2
- ● $-\infty$

Same logic as before, only now values will change slower towards $-\infty$.

Let's add another state D in the MDP and an extra transition $\leftarrow$ in our dataset:

state	action	reward	next state
A	$\rightarrow$	-1	B
B	$\rightarrow$	-1	C
C	$\rightarrow$	-1	A
C	$\leftarrow$	-1	D
D	$\rightarrow$	0	D

We still run Q-learning on the new dataset with a learning rate $\alpha \in (0, 1)$ for an infinite number of iterations.

(iii) [1 pt] **Q6.8** If we still keep $\gamma = 1$ then what is the value for $Q(B, \rightarrow)$ after Q-learning has converged?

- ○ -1
- ● -2
- ○ $-\infty$

From C the optimal action will be $\leftarrow$ and hence from $Q(B, \rightarrow) = -2$.

(iv) [2 pts] **Q6.9** Is there a value for the discount factor $\gamma > 0$ such that the optimal policy at state C is action $\rightarrow$ instead of $\leftarrow$?

- ○ Yes
- ● No

For any value $\gamma > 0$ the choice $\leftarrow$ is associated with a higher Q-value. Only for the extreme case of $\gamma = 0$ the two choices are equivalent.