MAI5100: Fundamentals of Artificial Intelligence

Midterm Examination - 2025

University of Guyana
Computer Science
Instructor: Dr. Christopher Clarke

Exam Information

Date:	Saturday, May 3, 2025
Time:	10:15 AM - 1:15 PM (Guyana Time)
Duration:	2 hours
Total Points:	100 points
Exam Format:	Online

Instructions

1. This examination covers all material from Weeks 1-7 of the course.
2. Answer all questions in the exam. Do **not** worry about the total points being **132**.
3. Each section indicates the points allocated per question.
4. Read each question carefully before attempting to answer.
5. Time management is crucial - allocate your time according to point values.

Policies

- **Resources Allowed:**
  - You're free to consult your notes and any materials provided during the course.
  - No other resources (books, websites, AI tools, etc.) are permitted.
- **Academic Integrity:**
  - This is an individual assessment. Collaboration with others is prohibited.
  - All solutions must be your own work.
- **Technical Issues:**
  - If you experience technical difficulties, contact the instructor immediately.
  - Document any issues with screenshots or recordings if possible.

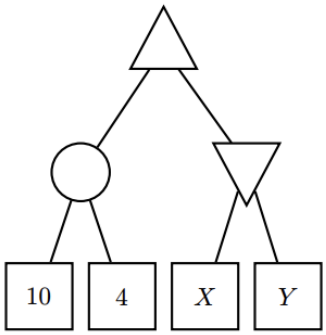
- **Submission:**
  - Submit your answers via the Moodle portal by the deadline.
  - Ensure your submission is complete and includes all required components.

Good luck!

---

1 Question 1

1. (17 points) In the next 3 subparts, consider the following game tree, representing a two-player game where the players take turns. Assume that the children of the expectation node have equal probability.



Assume nodes are pruned from left to right, and we prune on equality.

(a) (3 points) Can you ever prune X?

If you select "Yes", write an inequality that describes when X will be pruned. For example, you could write "$X \leq 5$" or "$X + Y > 30$". If you select "No", leave the box blank.

◯ Yes
◯ No

Inequality: _____

(b) (3 points) Can you ever prune Y?

If you select "Yes", write an inequality that describes when Y will be pruned. For example, you could write "$X \leq 5$" or "$X + Y > 30$". If you select "No", leave the box blank.

◯ Yes
◯ No

Inequality: _____

(c) (2 points) Suppose the maximizer node represents Pacman, and both the minimizer node and the chance node represent Blinky. What kind of situation does this specific game tree represent?

◯ Pacman acts randomly in some states, but adversarially in others.
◯ Blinky acts randomly in some states, but adversarially in others.
◯ Pacman and Blinky use different utility functions.

◯ Pacman and Blinky act adversarially in all states.

(d) (4 points) Select all true statements about game trees.

◯ Increasing the depth of a minimax game tree may result in a smaller value at the root node.

◯ Decreasing the depth of a minimax game tree may result in a smaller value at the root node.

◯ Increasing the depth of a minimax game tree may result in a larger value at the root node.

◯ Decreasing the depth of a minimax game tree may result in a larger value at the root node.

◯ None of the above

(e) (3 points) Select all true statements about policy iteration.

◯ Policy iteration always converges in strictly fewer iterations than value iteration.

◯ During policy evaluation, we can compute V^{π_i} using a system of linear equations.

◯ If V^{π_i} is optimal, then the policy extraction step will not change the policy in the next iteration.

◯ None of the above

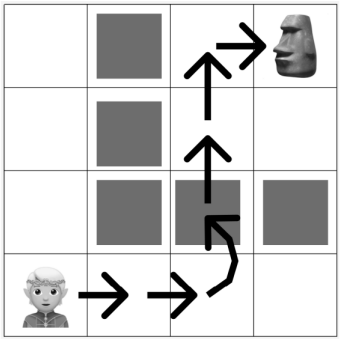
(f) (2 points) Consider policies π_1, π_2 for the same MDP. Select all true statements about their value functions.

◯ If $\pi_1 = \pi_2$, then $Q^{\pi_1}(s, a) = Q^{\pi_2}(s, a)$ for all (s, a).

◯ If $\pi_1 = \pi_2$, then $V^{\pi_1}(s) = V^{\pi_2}(s)$ for all s.

◯ None of the above

2 Question 2



2. (15 points)

In this example, straight arrows are regular moves and the curved arrow is a jump.

Edgar is an elf on Easter Island, seeking Pietru the Paradise Dweller, an ancient Moai statue, for midterm wisdom. Easter Island is represented as an $N \times N$ grid, with k randomly placed obstacles. At each step, Edgar can either move up, down, left, or right (if unobstructed). He also has 3 jumps to use during the journey.

A jump is an action that puts him on top of an obstacle adjacent to him in the direction he chooses. For example, in the figure to the right, Edgar jumps north from (3,0) to the top of

the obstacle at (3, 1). He may continue to walk on top of adjacent obstacles until he descends. Descending does not require him to use a jump.

Edgar knows the location of all obstacles, as well as his starting position and the position of the Moai statue on the grid. Edgar also knows what the legal moves are, at every position. If there are no legal moves—for example, if he's surrounded by obstacles with no jumps remaining—his mission ends. All actions have a cost of 1, and he cannot remain still.

(a) (2 points) Edgar is comparing DFS, BFS, and A^* Search using the trivial heuristic $h(n) = 0$. Assume the branching factor is 4, and the nearest solution is 4 moves away. Select all true statements. *Hint: You expand a state when you call its successor function.*

- ◯ A* Search will find the solution by expanding fewer states than BFS.
- ◯ A* Search will always expand $4^0 + 4^1 + 4^2 = 21$ states or fewer.
- ◯ BFS will always find a solution at an equal or shallower depth than DFS.
- ◯ None of the above.

(b) (2 points) Edgar uses A^* Search with an admissible heuristic. What is the worst-case time complexity in terms of the branching factor b and the depth of the shallowest solution d?

- ◯ $O(n)$ where n is the number of states
- ◯ $O(b^d)$
- ◯ $O(d^b)$
- ◯ $O(b)$

(c) (3 points) Edgar now uses greedy graph search with heuristic $h(n) = E(n)/100$, where $E(n)$ is his Euclidean distance to the statue at state n. Select all true statements.

- ◯ Greedy graph search is complete, because it only evaluates legal moves.
- ◯ Greedy graph search will always find a solution with $h(n)$ because $h(n)$ is an admissible heuristic.
- ◯ Greedy graph search will always find a shorter path than A^* search with $h(n)$ as its heuristic.
- ◯ None of the above.

Edgar wants to design an admissible heuristic function $h(n)$. He has the following functions:

1. $M(n)$ = the Manhattan distance from Edgar to the goal for state n.
2. $E(n)$ = the Euclidean distance from Edgar to the goal for state n.
3. $O(n)$ = the smallest number of obstacles that lie along any path of distance $M(n)$ between Edgar and the goal for state n.
4. $J(n)$ = the number of jumps remaining.

(d) (2 points) Which of the following are admissible heuristics?

- ◯ $M(n)$
- ◯ $E(n)$
- ◯ $O(n)$
- ◯ $J(n)$
- ◯ None of the above

(e) (3 points) Which of the following are admissible heuristics?

- ◯ $\min(O(n), J(n))$

- ◯ $E(n)/3$
- ◯ $O(n) + 1$
- ◯ $M(n) + O(n)$
- ◯ None of the above

(f) (1 point) Edgar decides to use the simulated annealing algorithm with an initial temperature of 100 and a cooling schedule of $T_i = \frac{T_0}{1+i}$ where i is the number of iterations. Approximately what is the algorithm's temperature after 50 iterations?

- ◯ 0
- ◯ 2
- ◯ 10
- ◯ 50

3 Question 3

3. (19 points) Consider the following word game. You start with 4 letters in a sequence, and you want to transform the sequence into a goal sequence by swapping adjacent letters. On each turn, you can choose one of the following actions, each with cost 1:

   - Swap the 1^{st} and 2^{nd} letters.
   - Swap the 2^{nd} and 3^{rd} letters.
   - Swap the 3^{rd} and 4^{th} letters.

   The objective of the game is to get to the goal sequence with the minimum number of swaps. Here is an example:

Goal: *HATS*			
H	*S*	*A*	*T*
Swap *A* and *S*			
H	*A*	*S*	*T*
Swap *T* and *S*			
H	*A*	*T*	*S*

   When we call the successor function on a given state, we add the successor states onto the fringe in the following order:

   1. The state after swapping the 1st and 2nd letters.
   2. The state after swapping the 2nd and 3rd letters.
   3. The state after swapping the 3rd and 4th letters.

For example, when we call the successor function on state "ABCD", we add [BACD, ACBD, ABDC] on the fringe, in that order.

For parts Q3.1 and Q3.2, consider the following start sequence and goal sequence:

Start: KOTO Goal: TOOK

(a) (4 points) What is the resulting solution from running BFS graph search?

- ◯ $KOTO \rightarrow OKTO \rightarrow OTKO \rightarrow TOKO \rightarrow TOOK$
- ◯ $KOTO \rightarrow OKTO \rightarrow OTKO \rightarrow OTOK \rightarrow TOOK$
- ◯ $KOTO \rightarrow KTOO \rightarrow TKOO \rightarrow TOKO \rightarrow TOOK$
- ◯ BFS fails to find a solution

(b) (4 points) Running DFS results in this solution: KOTO → OKTO → OKOT → KOOT → KTO → TKOO → TOKO → TOOK. How many states were expanded to find this solution during DFS graph search? *Hint: You expand a state when you call the successor function on that state.*

- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10

(c) (4 points) Select all admissible heuristics.

- ◯ The number of letters not in the right place.
- ◯ The sum of distances for each letter between its current position and the closest identical letter in the goal sequence.
- ◯ For each adjacent pair (i.e., 1st and 2nd letters, 2nd and 3rd letters, 3rd and 4th letters), count the number of pairs that do not match the corresponding pair in the goal sequence.
- ◯ The minimum distance from any letter to the closest identical letter in the goal sequence.
- ◯ None of the above.

For the rest of this question, we change the cost of each action as follows:

- Swap 1^{st} and 2^{nd} letters: Cost 1.
- Swap 2nd and 3rd letters: Cost 2.
- Swap 3^{rd} and 4^{th} letters: Cost 3.

If two states on the fringe have the same priority, break ties alphabetically. For example, if ABCD and BACD have the same priority, pop ABCD off the fringe first.

(d) (3 points) Which statements are true about running UCS graph search on this game, assuming a solution exists?

- ◯ UCS will always find a solution.
- ◯ UCS will always return the lowest-cost solution.
- ◯ UCS will always expand fewer states than BFS.

◯ None of the above.

Consider the following heuristic: The maximum distance from any letter to its position in the goal sequence. For example, for the goal "ABCD," the state "CDBA" has a heuristic of 3, because A has distance 3, B has distance 1, C has distance 2, and D has distance 2.

For part Q3.5, consider the following start sequence and goal sequence:

Start: OKRW Goal: WORK

(e) (4 points) Which solution is returned by greedy graph search?

◯ OKRW → ORKW → ORWK → OWRK → WORK
◯ OKRW → OKWR → OWKR → OWRK → WORK
◯ OKRW → OKWR → OWKR → WOKR → WORK

4 Question 4

4. (19 points) Xavier and Matei are playing a new online game called Pacman.io.

They each move their own Pacman around a 20×20 grid, taking turns, with actions (UP, DOWN, LEFT, RIGHT, STOP).

To start, 100 food dots are placed around the grid in **random locations**. After a food is eaten by either Pacman, another food is **immediately** placed in a random empty grid location.

Both Pacmen start at power level 1 and gain 1 power level for each food eaten. A player wins if their Pacman reaches power level 100 or "eats" the other player's Pacman.

"Eating" occurs as follows: When both Pacmen occupy the same grid location, the Pacman with higher power level eats the other one. If they have the same power level, nothing occurs, and both Pacmen occupy the same position.

Xavier represents this game with a game tree.

(a) (4 points) Which of the following are valid state representations for this problem?

◯ 100 booleans for food, and each Pacman's (x, y) coordinates and power level.
◯ A list of visited food locations, and each Pacman's (x, y) coordinates and power level.
◯ A list of current food locations, and each Pacman's (x, y) coordinates and power level.
◯ Each Pacman's (x, y) coordinates and power level only.
◯ None of the above.

(b) (4 points) Xavier chooses to represent the state with 400 booleans for the food locations (one for each grid location), each Pacman's (x, y) coordinates and their power levels. What is the size of the state space using this state representation? Represent the answer in the form: $A^B \times B^C$. Fill in the boxes for A, B, and C using only positive integers.

A: _____ B: _____ C: _____

(c) (4 points) Xavier considers running depth-limited minimax. In general, which of the following are valid reasons for running depth-limited minimax instead of a full minimax search?

◯ Searching the full tree is often computationally infeasible due to a large branching factor.

- ◯ Using an evaluation function on the leaf nodes in depth-limited minimax is more accurate than computing the utilities for the entire minimax tree.
- ◯ A suboptimal agent is better modeled by a depth-limited minimax tree.
- ◯ Depth-limited minimax does alpha-beta pruning for us.
- ◯ None of the above.

(d) (5 points) Xavier needs a function to compute the utility at each leaf node in the minimax tree.

- **Case 1:** Xavier's Pacman has a higher power level than Matei's. For two states with the same power levels, the function should prefer states where the Pacmen are closer.
- **Case 2:** Matei's Pacman has a higher power level than Xavier's. For two states with the same power levels, the function should prefer states where the Pacmen are farther apart.

Write an evaluation function that behaves as above, using these terms:

- Manhattan Distance between Xavier and Matei's Pacmen, $MH(p_x, p_m)$.
- Power level of Xavier's Pacman, l_x.
- Power level of Matei's Pacman, l_m.

Your function cannot be piecewise—write one expression for all states. You may assume that $l_x \neq l_m$ and $MH(p_x, p_m) \neq 0$.

Note: An example of "two states with the same power levels" is a pair of states where $l_x = 10$ and $l_m = 5$ in both states.

Evaluation Function: _____

(e) (2 points) Suppose there are now a known, arbitrary number of Pacmen on a grid of known, arbitrary size. After a Pacman is eaten, the eater will gain power level equal to the power level of the eaten Pacman. The win conditions are to reach power level P_{max} first or eat all other Pacmen. Which of the following setups represents this new scenario the best for Xavier's Pacman?

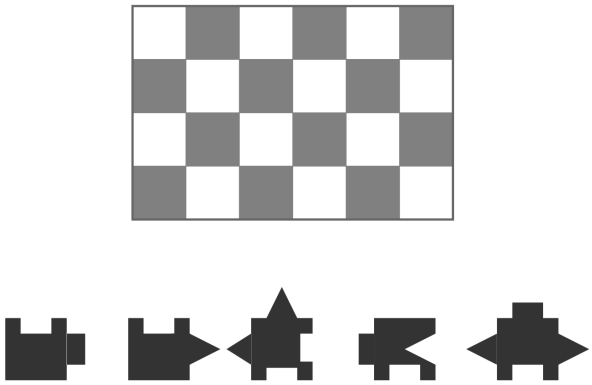
- ◯ Run minimax with all other Pacmen as minimizer nodes. The evaluation function is the power level of Xavier's Pacman.
- ◯ Run expectimax with all other Pacmen as expectation nodes. The evaluation function is power level of Xavier's Pacman.
- ◯ Construct a game tree with each Pacman maximizing their own utility. The evaluation function for each Pacman is their own power level.

5 Question 5

5. (14 points) Pacman is a big fan of jigsaw puzzles and designs a hard puzzle:

- There is an $M \times N$ grid, where each element in the grid fits exactly one puzzle piece.
- There are 5 different pieces. Each piece may be used more than once.
- Each piece may be rotated by $0°, 90°, 180°$, or $270°$.
- If a piece has a flat edge, that edge must be touching the edge of the grid.
- The goal is completely fill the grid by placing pieces, forming a $M \times N$ rectangle.

An example grid and pieces are shown below:



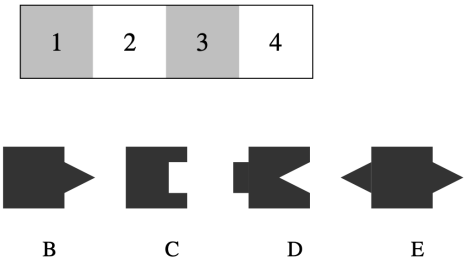
(a) (2 points) Suppose we define the CSP variables to be each grid position. What is the size of the domain for each variable before filtering?

- ◯ 5
- ◯ 10
- ◯ 15
- ◯ 20

(b) (2 points) If we have d pieces, rather than 5, what is the time complexity of running backtracking search on this puzzle without using any optimizations?

- ◯ $O(dMN)$
- ◯ $O(d^{MN})$
- ◯ $O((4d)^{MN})$
- ◯ $O((MN)^{4d})$

For the rest of the question, we modify the puzzle so that pieces can only be rotated 0° or 180°. Consider the following puzzle, with a 4×1 grid, and the following 5 pieces:



Reminders: Pieces may be reused. If a piece has a flat edge, that edge must be touching the edge of the grid. For example, piece A cannot go in position 2 or 3.

Pacman solves the puzzle using backtracking search. First, he assigns puzzle piece C to position 1.

Note: If there is any orientation of a piece that would fit in a position, include that piece in the domain of that position.

(c) (4 points) After applying unary constraints, apply arc consistency to $(2 \to 1)$ and then $(4 \to 2)$. What pieces are in the domains of positions 2 and 4?

Domain of position 2:

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ E
- ◯ None of the above

Domain of position 4:

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ E
- ◯ None of the above

(d) (4 points) Continuing from the previous subpart, Pacman applies arc consistency on $(2 \to 3)$ and then $(3 \to 2)$. What pieces are in the domains of positions 2 and 3?

Domain of position 2:

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ E
- ◯ None of the above

Domain of position 3:

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ E
- ◯ None of the above

(e) (2 points) Consider the domains you computed in the previous subparts. According to the MRV heuristic, which variable should we assign next? Break ties by picking the lower number.

- ◯ 2
- ◯ 3
- ◯ 4

6 Question 6

6. (21 points) We want to assign 100 students $(s_1, s_2, ..., s_{100})$ to either Dwinelle (D) or VLSB (V). We'd like to model this as a CSP. For the first 4 subparts, suppose each room is infinitely large. Assume that only the first 10 students that are assigned $(s_1, ..., s_{10})$ are left-handed. VLSB has at least 10 left-handed desks and Dwinelle has no left-handed desks. Left-handed students must be assigned to left-handed desks.

   (a) (2 points) Is this CSP easy or hard to solve, and why?

   ◯ Easy, because it is under-constrained.
   ◯ Easy, because it is over-constrained.
   ◯ Hard, because it is under-constrained.
   ◯ Hard, because it is over-constrained.

   (b) (2 points) How many possible assignments exist (including invalid assignments)?

   ◯ 10
   ◯ 100
   ◯ 100^2
   ◯ 2^{100}
   ◯ 100!

   Recall the naive DFS-based algorithm from lecture. In particular, notice that we only check constraints when all variables have been assigned values, and we do not pre-process by applying unary constraints.

```
1 dfs(assignment, csp):
2     if assignment is complete and satisfies all constraints:
3         print(assignment) # found solution
4         exit # terminate program
5     else:
6         for each value in domain of the next unassigned variable:
7             new_assignment ← assignment with (variable, value) assigned
8             dfs(new_assignment, csp)
```

   (c) (2 points) If you always assign students to Dwinelle first on line 6, how many complete assignments does DFS examine?

   ◯ 1
   ◯ Between 2 and 100
   ◯ More than 100

   (d) (2 points) If you always assign students to VLSB first on line 6, how many complete assignments does DFS examine?

   ◯ 1
   ◯ Between 2 and 100
   ◯ More than 100

   For the rest of the question, we add a new constraint: Each room can hold at most 50 students.

   (e) (3 points) Suppose we enforce Dwinelle's room capacity using only one higher-order constraint. In the worst case, how many variables do you need to check to determine if this constraint is satisfied?

   Answer: _____

(f) (2 points) If you always assign students to VLSB first on line 6 with this new room constraint, how many complete assignments does DFS examine?

- ◯ 1
- ◯ Between 2 and 100
- ◯ More than 100

Recall backtracking search from lecture. In particular, notice that unlike DFS, we now check constraints immediately following each variable assignment. We do not pre-process by applying unary constraints.

```
1 backtracking(assignment, csp):
2     if assignment is complete:
3         print(assignment) # found solution
4         exit # terminate program
5     else:
6         for each value in domain of the next unassigned variable:
7             new_assignment ← assignment with (variable, value) assigned
8             if new_assignment does not violate any constraints:
9                 backtracking(new_assignment, csp)
```

(g) (2 points) Suppose you always assign students to VLSB first. During backtracking search, how many 'new_assignments' violate a constraint on line 8?

- ◯ 1
- ◯ Between 2 and 100
- ◯ More than 100

(h) (2 points) Consider two unassigned students s_1 (left-handed) and s_{88} (non-left-handed). If you enforce arc consistency between s_1 and s_{88}, what happens?

- ◯ At least one value gets removed from s_1 domain.
- ◯ At least one value gets removed from s_{88} domain.
- ◯ At least one value gets removed from both variables' domains.
- ◯ No values get removed from either variable's domain.

(i) (2 points) Suppose for this subpart only, we pre-process by applying unary constraints. What happens if you use the MRV (minimum remaining values) heuristic on this CSP?

- ◯ Students are assigned to VLSB first.
- ◯ Students are assigned to Dwinelle first.
- ◯ Left-handed students are assigned first.
- ◯ Non-left-handed students are assigned first.

(j) (2 points) Pacman suggests using local search to solve this CSP. Local search (1) _____ guaranteed to find a solution, and when local search finds a solution, the runtime is (2) _____ than exponential time on average.

- ◯ (1) is, (2) slower
- ◯ (1) is NOT, (2) slower
- ◯ (1) is, (2) faster
- ◯ (1) is NOT, (2) faster

7 Question 7

7. (19 points) Recall the standard Bellman equation discussed in lecture:

$$Q(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

In this equation, $\max_{a'} Q(s',a')$ is calculated by computing $Q(s',a')$ for every a', and then taking the maximum of all the resulting Q-values.

Consider a modified Bellman equation, where we replace the max function with an average function:

$$Q_\mu(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \text{ avg}_{a'} Q_\mu(s',a')]$$

In this equation, $\text{avg}_{a'} Q_\mu(s',a')$ is calculated by computing $Q_\mu(s',a')$ for every a', and then taking the average of all the resulting Q_μ values.

(a) (1 point) It is always possible to solve the standard Bellman equation (shown above) using a system of linear equations.

- ◯ True
- ◯ False

(b) (1 point) It is always possible to solve the modified Bellman equation (shown above) using a system of linear equations.

- ◯ True
- ◯ False

(c) (2 points) The optimal policy extracted from the Q_μ values (in the modified equation) is the same as the optimal policy extracted from the Q-values (in the standard equation).

- ◯ never
- ◯ sometimes
- ◯ always

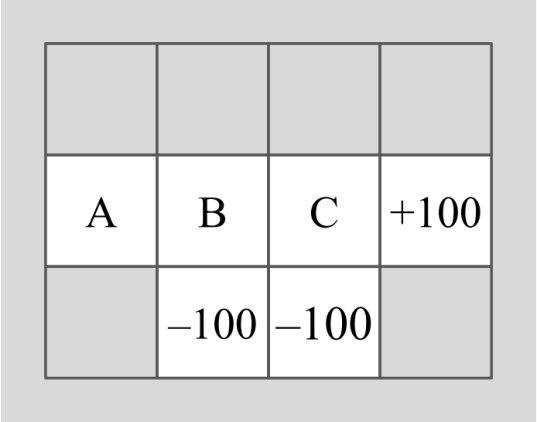
(d) (2 points) For this subpart only, consider a state s with many successor states, all of which are terminal states. Recall that once you enter a terminal state, no future rewards are available. Which of these statements is always true?

- ◯ $Q_\mu(s,a) < Q(s,a)$ for all actions a
- ◯ $Q_\mu(s,a) > Q(s,a)$ for all actions a
- ◯ $Q_\mu(s,a) = Q(s,a)$ for all actions a
- ◯ None of the above

(e) (2 points) Given a set of Q_μ values for all state-action pairs, which of these equations extracts a policy from the Q_μ values?

- ◯ For each state s, compute arg $\max_a Q_\mu(s,a)$
- ◯ For each state s, compute $\text{avg}_a Q_\mu(s,a)$
- ◯ For each action a, compute $\max_s Q_\mu(s,a)$.
- ◯ For each action a, compute arg $\max_s Q_\mu(s,a)$.

For the rest of this question, consider the Gridworld shown. Some states are labeled with letters (A, B, C). Assume all actions succeed with 100% probability, $\gamma = 1$, and there is 0 living reward.



Reminder: In a Gridworld, there is only one action, "Exit", available from the squares labeled +100 and -100, that gives the rewards of +100 and -100, respectively.

(f) (6 points) For this Gridworld, fill in the tables for $Q(s, a)$, the Q-values from the standard equation. Write one number per box. The last row has been filled in for you as an example.

(s, a)	Q(s,a)
$(A, \rightarrow)$	
$(B, \leftarrow)$	
$(B, \downarrow)$	
$(B, \rightarrow)$	

(s, a)	Q(s,a)
$(C, \leftarrow)$	
$(C, \downarrow)$	
$(C, \rightarrow)$	100

For the remaining subparts, select the correct expression for $Q_\mu(s, a)$, the Q_μ values from the modified equation.

(g) (1 point) $Q_\mu(C, \rightarrow)$

- ◯ -100
- ◯ 100
- ◯ $\frac{1}{3}[Q_\mu(A, \downarrow) + Q_\mu(B, \downarrow) + Q_\mu(C, \downarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(C, \rightarrow) + Q_\mu(C, \downarrow) + Q_\mu(C, \leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(B, \rightarrow) + Q_\mu(B, \downarrow) + Q_\mu(B, \leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(A, \rightarrow) + Q_\mu(B, \rightarrow) + Q_\mu(C, \rightarrow)]$

(h) (1 point) $Q_\mu(B, \downarrow)$

- ◯ -100
- ◯ 100
- ◯ $\frac{1}{3}[Q_\mu(A, \downarrow) + Q_\mu(B, \downarrow) + Q_\mu(C, \downarrow)]$

- ◯ $\frac{1}{3}[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)]$

(i) (1 point) $Q_\mu(A,\rightarrow)$

- ◯ -100
- ◯ 100
- ◯ $\frac{1}{3}[Q_\mu(A,\downarrow) + Q_\mu(B,\downarrow) + Q_\mu(C,\downarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)]$

(j) (1 point) $Q_\mu(B,\rightarrow)$

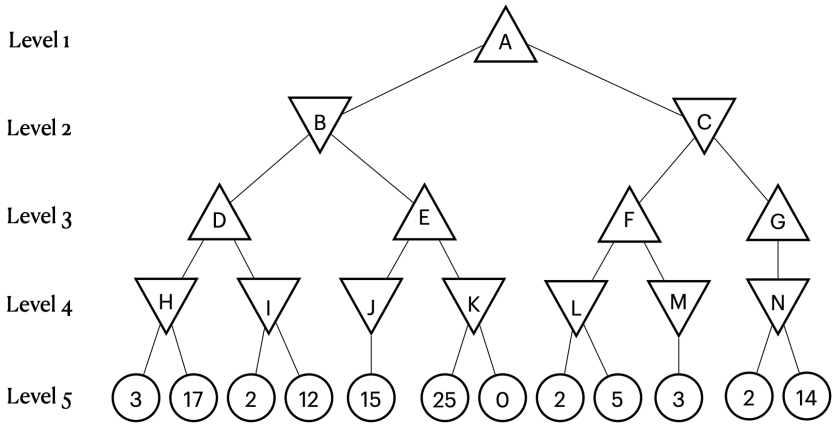
- ◯ -100
- ◯ 100
- ◯ $\frac{1}{3}[Q_\mu(A,\downarrow) + Q_\mu(B,\downarrow) + Q_\mu(C,\downarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)]$

(k) (1 point) $Q_\mu(C,\leftarrow)$

- ◯ -100
- ◯ 100
- ◯ $\frac{1}{3}[Q_\mu(A,\downarrow) + Q_\mu(B,\downarrow) + Q_\mu(C,\downarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(C,\rightarrow) + Q_\mu(C,\downarrow) + Q_\mu(C,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(B,\rightarrow) + Q_\mu(B,\downarrow) + Q_\mu(B,\leftarrow)]$
- ◯ $\frac{1}{3}[Q_\mu(A,\rightarrow) + Q_\mu(B,\rightarrow) + Q_\mu(C,\rightarrow)]$

8 Question 8

8. (8 points) Consider the following game tree.



(a) (1 point) What is the minimax value at node A?

Answer: _____

(b) (3 points) Which branches will be pruned after running minimax search with alpha-beta pruning? For instance, if the edge between node A and node B is pruned, write 'A-B'. If the edge between H and 3 is pruned, write 'H-3'. List the pruned branches from left to right. If a branch from an upper level is pruned, you don't have to list the branches below that.

Pruned Branches: _____

(c) (2 points) Suppose all the min nodes in layer 4 are changed to max nodes and all the max nodes in layer 3 are changed to min nodes. In other words, we have max, min, min, max as levels 1, 2, 3, 4 in the game tree. We then run minimax search with alpha-beta pruning. Which of the following statements is true?

- ◯ The same set of leaf nodes will be pruned, because this is still a minimax problem and running alpha-beta pruning will result in the same set of leaf nodes to be pruned.
- ◯ A different set of leaf nodes will be pruned.
- ◯ No leaf nodes can be pruned, because pruning is only possible when the minimizer and maximizer alternate at each level.
- ◯ None of the above

(d) (2 points) Now, consider another game tree which has the same structure as the original game tree shown above. This modified game tree can take on any values at the leaf nodes. What is the minimum and the maximum number of leaf nodes that can be pruned after running alpha-beta pruning?

Minimum leaf nodes pruned: _____

Maximum leaf nodes pruned: _____