UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
CS440/ECE448 Artificial Intelligence
Practice Exam 2 Spring 2024

The exam will be April 15-17, 2024

Instructions

- The exam will be held at CBTF.
- The exam will contain eight multiple-choice questions, one from each of the following eight topics: search (lectures 18-19), Markov decision process (lecture 20), minimax (lecture 21), expectiminimax (lecture 22), static game theory (lecture 23), logic (lectures 26-27), vector semantics (lectures 28-29), and robotics (lecture 30). Lectures 24 and 25 are not covered. Each of the eight topics is represented in this practice exam, but the number of questions available on each topic is somewhat variable.
- No book, notes, or calculator will be allowed.
- The following formula page, or one like it, will be available to you attached to the online exam.

Possibly Useful Formulas

Admissible: $\hat{h}(n) \leq h(n)$

Consistent: $\hat{h}(n) - \hat{h}(m) \leq h(n,m)$

Value Iteration: $u_i(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) u_{i-1}(s)$

Policy Evaluation: $u_\pi(s) = r(s) + \gamma \sum_{s'} P(s'|s,\pi(s)) u_\pi(s')$

Policy Improvement: $\pi_{i+1}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a) u_{\pi_i}(s')$

Alpha-Beta Max Node: $v = \max(v, \text{child}); \quad \alpha = \max(\alpha, \text{child})$

Alpha-Beta Min Node: $v = \min(v, \text{child}); \quad \beta = \min(\beta, \text{child})$

Expectiminimax: $u(s) = \begin{cases} \max_a \sum_{s'} P(s'|a,a) u(s') & s \in \text{max states} \\ \min_a \sum_{s'} P(s'|a,a) u(s') & s \in \text{min states} \end{cases}$

Mixed Nash Equilibrium:
$$P(A=0) r_B(0,0) + P(A=1) r_B(1,0) = P(A=0) r_B(0,1) + P(A=1) r_B(1,1)$$
$$P(B=0) r_A(0,0) + P(B=1) r_A(0,1) = P(B=0) r_A(1,0) + P(B=1) r_A(1,1)$$

Unification: $S : \{\mathscr{V}_P, \mathscr{V}_Q\} \to \{\mathscr{V}_Q, \mathscr{C}\}$ such that $S(P) = S(Q) = U$

CBOW Generative: $\mathscr{L} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{j=-c, j\neq 0}^{c} \ln \frac{\exp(\mathbf{v}_t^T \mathbf{v}_{t_j})}{\sum_{\mathbf{v} \in \mathscr{V}} \exp(\mathbf{v}^T \mathbf{v}_{t+j})}$

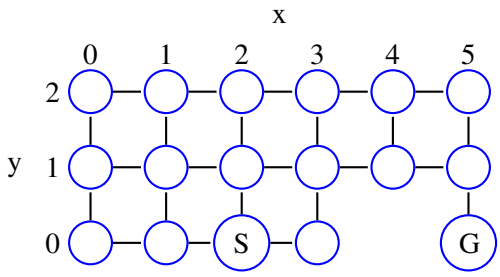
Skip-gram Contrastive: $\mathscr{L} = -\frac{1}{T} \sum_{t=1}^{T} \left(\sum_{\mathbf{v}' \in \mathscr{D}_+(w_t)} \ln \frac{1}{1+e^{-\mathbf{v}'^T \mathbf{v}_t}} + \sum_{\mathbf{v}' \in \mathscr{D}_-(w_t)} \ln \frac{1}{1+e^{\mathbf{v}'^T \mathbf{v}_t}} \right)$

Transformer: $\mathbf{c}_t = \sum_s \alpha(t,s) \mathbf{v}_s$

Attention: $\alpha(t,s) = \frac{\exp(\mathbf{q}_t^T \mathbf{k}_s)}{\sum_{s'} \exp(\mathbf{q}_t^T \mathbf{k}_{s'})}$

1 Search

Question 1 ***(0 points)***



In the maze shown above, nodes are named by their (y,x) coordinates, where y is the row number (starting from the bottom), and x is the column number (starting from the left). A robot is trying to find a path from the start node, $(0,2)$ (labeled "S"), to the goal node, $(0,5)$ (labeled "G"). It uses A* search, with Manhattan distance as a heuristic. After nodes $(0,3)$ and $(1,2)$ have been expanded, there are two copies of nodes $(1,3)$ on the frontier, one with $(0,3)$ as its parent, and one with $(1,2)$ as its parent. Which of these two copies was placed on the frontier first? Why?

Solution: Both nodes $(0,3)$ and $(1,2)$ are one step away from the start node ($g((0,3)) = g((1,2)) = 1$), but node $(0,3)$ has a lower heuristic ($h((0,3)) = 2$, while $h((1,2)) = 4$), therefore node $(0,3)$ is expanded before node $(1,2)$. The copy of $(1,3)$ with $(0,3)$ as its parent is therefore placed on the frontier before the copy that has $(1,2)$ as its parent.

Question 2 ***(0 points)***

Prove that every consistent A* search heuristic is also admissible.

Solution: A consistent heuristic is defined as

$$h(p) \leq d(p,r) + h(r)$$

for any pair of nodes p and r. An admissible heuristic is defined as

$$h(p) \leq d(p, \text{Goal})$$

which is a special case of the definition of consistent heuristic, specifically, for the case when $r = \text{Goal}$, and $h(\text{Goal}) = 0$.

Question 3 ***(0 points)***

A typical freight management problem seeks to deliver several large objects from point A to point B using a truck that can carry up to M kilograms. First, you weigh each of the objects, so that you know its mass. Then you use the following search problem to devise an optimal plan:

- State: $S =$ a list of the objects that have not yet been delivered.
- Action: Load a set of objects onto the truck, take them from A to B, then return the truck from B to A.
- Cost: Each trip from A to B has a cost of 1, regardless of the weight of the objects on the truck. Your goal is simply to minimize the number of trips.

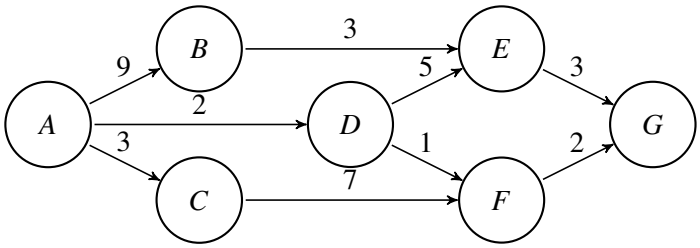
Define a nonzero heuristic for this problem, and prove that your heuristic is admissible.

Solution: A reasonable heuristic is the sum of the weights of all objects that have not yet been delivered, divided by M.

$d(S)$ is the total number of trips remaining. The truck can only carry up to M kilograms per trip, therefore $d(S) \geq \frac{1}{M}(\text{mass of remaining objets})$.

Question 4 ***(0 points)***

Consider the following search graph. The starting state is A, the goal state is G, and the cost of each possible action is shown on the corresponding edge:



You are considering trying to implement A* search over this graph. You are trying to decide whether to use $h_1(n)$ or $h_2(n)$ as the heuristic, where $h_1(n)$ and $h_2(n)$ are as given in the following table:

	A	B	C	D	E	F	G
$h_1(n)$:	4	5	8	2	2	1	0
$h_2(n)$:	4	5	8	3	2	1	0

(a) Suppose you are using an explored set to prevent repeated states, i.e., you will only expand a state if you have never expanded it before. Which of these two heuristics is better to use, and why?

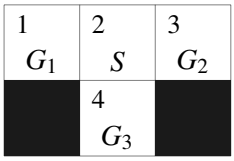
Solution: Use $h_1[n]$, because it is consistent.

(b) Your friend convinces you to use an explored dict, i.e., you will only expand a state if you have never expanded it before with a smaller $g(n)$. Which of these two heuristics is better to use, and why?

Solution: Use $h_2(n)$, because it dominates $h_1(n)$.

Question 5 ***(0 points)***

Imagine a maze with only four possible positions, numbered 1 through 4 in the following diagram. Position 2 is the start position (denoted S in the diagram below), while positions 1, 3, and 4 each contain a goal (denoted as G_1, G_2, and G_3 in the diagram below). Search terminates when the agent finds a path that reaches all three goals, using the smallest possible number of steps.



(a) Define a notation for the state of this agent. How many distinct non-terminal states are there?

Solution: The state can be defined by a pair of variables: (P,G) where $P \in \{1,\ldots,4\}$ specifies the current position, $G \in \{\varnothing, G_1, G_2, G_3, G_1G_2, G_1G_3, G_2G_3\}$ specifies which of the goals have been reached. There is only one position (2) that can be reached without touching any goal. After touching G_1, there are two positions that can be reached without touching another goal (1 and 2). After touching G_1 and G_2, there are three positions that can be reached without touching G_3 (1, 2, and 3). Generalizing, there are a total of $1+3\times 2+3\times 3 = 16$ non-terminal states.

(b) Draw a search tree out to a depth of 3 moves, including repeated states. Circle repeated states.

Solution: After the first move, possible states are $(1,G_1)$,$(3,G_2)$, and $(4,G_3)$. After the second move, possible states are $(2,G_1)$, $(2,G_2)$, and $(2,G_3)$. After the third move, possible states are $(1,G_1)$, $(1,G_1G_2)$, $(1,G_1G_3)$, $(3,G_2)$, $(3,G_1G_2)$, $(3,G_2G_3)$, $(4,G_3)$, $(4,G_1G_3)$, and $(4,G_2G_3)$. Of these, the states $(1,G_1)$, $(3,G_2)$, and $(4,G_3)$ in the last row are repeated states.

(c) For A* search, one possible heuristic, h_1, is the Manhattan distance from the agent to the nearest goal that has not yet been reached. Prove that h_1 is consistent.

Solution: From any node m, Manhattan distance to the nearest goal is always either $h_1[m] = 1$ or $h_1[m] = 2$. If $h_1[m] = 2$, then any step we take will move us to a node n such that $h_1[n] = 1$. If $h_1[m] = 1$, then it is possible to move away from the goal (to position n such that $h_1[n] = 2$), or it is possible to move toward the goal (in which case we reach the goal, and so the definition of "nearest goal that has not been reached" changes to one of the other goals, and again we have $h_1[n] = 2$). So the change in heuristic is always $h_1[m] - h_1[n] \in \{-1, 1\}$.

The distance traveled from any node m to its neighbor n is always one step. If n is closer to completing the maze than m, then the distance from m to the goal, $d[m]$, minus the distance from n to the goal, $d[n]$, is one: $d[m] - d[n] = 1$, whereas $h_1[m] - h_1[n] \in \{-1, 1\}$, so $d[m] - d[n] \geq h_1[m] - h_1[n]$.

(d) Another possible heuristic is based on the Manhattan distance $M[n, g]$ between two positions, and is given by

$$h_2[n] = M[G_1, G_2] + M[G_2, G_3] + M[G_3, G_1]$$

that is, h_2 is the sum of the Manhattan distances from goal 1 to goal 2, then to goal 3, then back to goal 1. Prove that h_2 is not admissible.

Solution: Notice that $h_2[n] = 6$ for every node n, so we only have to find a counter-example for which the total cost of the best path is $d[n] < 6$. But that's easy: the starting node S has a cost of $d[S] = 5 < h_2[S]$, so h_2 is not admissible.

(e) Prove that $h_2[n]$ is dominant to $h_1[n]$.

Solution: $h_1[n] \in \{1, 2\}$, whereas $h_2[n] = 6$ always, so $h_2[n] \geq h_1[n]$.

Question 6 ***(0 points)***

For each type of maze described below, specify the typical-case time complexity and space complexity of both breadth-first-search (BFS) and depth-first-search (DFS). Assume that both BFS and DFS return the first solution path they find.

(a) The Albuquerque maze has $b = 3$ possible directions that you can take at each intersection. No path is longer than $m = 25$ steps. There is only one solution, which is known to require exactly $d = 25$ steps.

> **Solution:** Both BFS and DFS are $O\{b^m\} = O\{b^d\}$.

(b) The Belmont maze has $b = 3$ possible directions that you can take at each intersection. No path is longer than $m = 25$ steps. About half of all available paths are considered solutions to the maze.

> **Solution:** BFS is $O\{b^m\}$. DFS has an expected computational cost of $O\{m\}$ (expected value of the path cost turns out to be exactly $2m$, if you work it out).

(c) The Crazytown maze has $b = 3$ possible directions that you can take at each intersection. The maze is infinite in size, so some paths have infinite length. There is only one solution, which is known to require $d = 25$ steps.

> **Solution:** BFS is $O\{b^d\}$. DFS is $O\{b^\infty\}$.

Question 7 ***(0 points)***

Consider the following maze. There are 11 possible positions, numbered 1 through 11. The agent starts in the position marked S (position number 3). From any position, there are from one to four possible moves, depending on position: Left, Right, Up, and/or Down. The agent's goal is to find the shortest path that will touch both of the goals (G_1 and G_2).

1	2 G_1	3 S	4
5		6 G_2	7
8	9	10	11

(a) Define a notation for the state of this agent. How many distinct non-terminal states are there?

Solution: The state can be defined by a pair of variables: (P,G) where $P \in \{1,\ldots,11\}$ specifies the current position, $G \in \{\varnothing, G_1, G_2\}$ specifies which of the goals have been reached. There are nine values of P that can be reached without touching either goal, ten that can be reached without touching G_1, and ten that can be reached without touching G_2, so the total number of non-terminal states is $9+10+10=29$.

(b) Draw a search tree out to a depth of 2 moves, including repeated states. Circle repeated states.

Solution: After the first move, possible states are $(2,G_1)$,$(6,G_2)$, and $(4,\varnothing)$. After the second move, possible states are $(1,G_1)$, $(3,G_1)$, $(3,G_2)$, $(10,G_2)$, $(7,G_2)$, $(3,\varnothing)$ (a repeated state), and $(7,\varnothing)$.

(c) For A* search, one possible heuristic, h_1, is the number of goals not yet reached. Prove that h_1 is consistent.

Solution: The heuristic difference between two neighboring states is either $h_1[n_1] - h_1[n_2] = 1$ (if they differ in the number of goals remaining) or $h_1[n_2] - h_1[n_1] = 0$ (if they have the same number of goals remaining. The distance is always $d[n_1, n_2] = 1$. So $h_1[n_1] - h_1[n_2] \leq d[n_1, n_2]$.

(d) Another possible heuristic is based on the Manhattan distance $M[n, g]$ between two positions, and is given by

$$h_2[n] = M[n, G_1] + M[G_1, G_2]$$

that is, h_2 is the sum of the Manhattan distance from the current position to G_1, plus the Manhattan distance from G_1 to G_2. Prove that h_2 is not admissible.

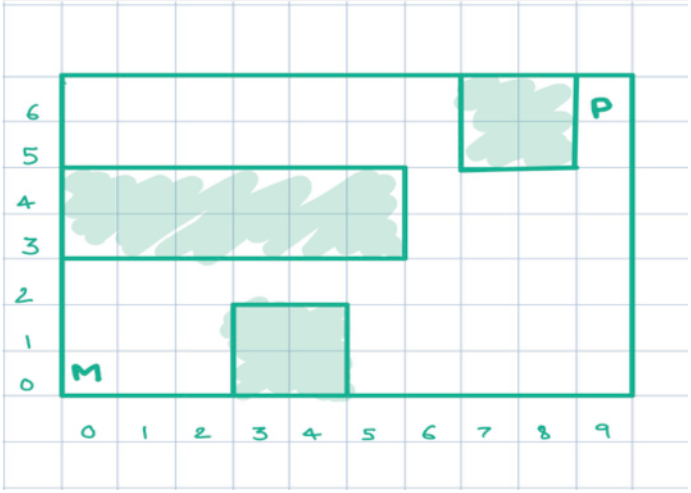
Solution: Proof by counter-example: for example, consider the state $n = (7, \varnothing)$. From this state, the shortest solution goes left, then up, then left: $d[n] = 3$ steps. The heuristic, however, is $h_2[n] = M[n, G_1] + M[G_1, G_2] = 3 + 2 = 5$, so $h[n] > d[n]$.

(e) Prove that $h_2[n]$ is dominant to $h_1[n]$.

Solution: $h_1[n] \in \{0, 1, 2\}$. The Manhattan distance $M[G_1, G_2] = 2$, therefore $h_2[n] \geq 2 \geq h_1[n]$.

Question 8 ***(0 points)***

Refer to the maze shown below. Here, 'M' represents Mario, 'P' represents Peach, and the goal of the game is to get Mario and Peach to find each other. In each move, both Mario and Peach take turns. For example, one move would consist of Peach moving a block to the bottom from her current position, and Mario moving one block to the left from his current position. Standing still is also an option.



(a) Describe state and action representations for this problem.

Solution:

- State = (x_M, y_M) position of Mario, (x_P, y_P) position of Peach.
- Action = Mario moves up,down,left,right, or stationary, Peach moves up,down,left,right, or stationary.

(b) What is the branching factor of the search tree?

Solution: 25

(c) What is the size of the state space?

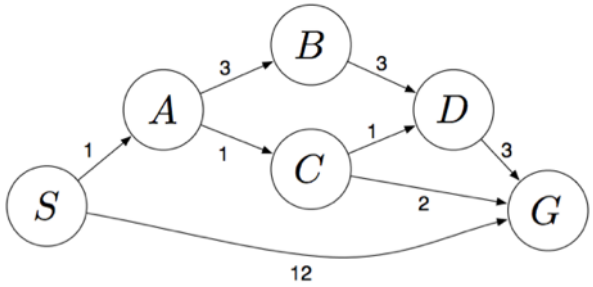
Solution: There are 50 possible locations for both Mario and Peach (if we assume that they can occupy the same square), for a total state space of 2500.

(d) Describe an admissible heuristic for this problem.

Solution: The heuristic $h(n) = 0$ is always admissible, but would receive at most a little bit of partial credit, because it's trivial. A more useful heuristic would measure the distance between Mario and Peach, e.g., $0.5 * (|x_M - x_P| + |y_M - y_P|)$.

Question 9 ***(0 points)***

Consider the search problem with the following state space:



S denotes the start state, G denotes the goal state, and step costs are written next to each arc. Assume that ties are broken alphabetically (i.e., if there are two states with equal priority on the frontier, the state that comes first alphabetically should be visited first).

(a) What path would BFS return for this problem?

Solution: SG

(b) What path would DFS return for this problem?

Solution: SABDG

(c) What path would UCS return for this problem?

Solution: SACG

(d) Consider the heuristics for this problem shown in the table below.

State	h_1	h_2
S	5	4
A	3	2
B	6	6
C	2	1
D	3	3
G	0	0

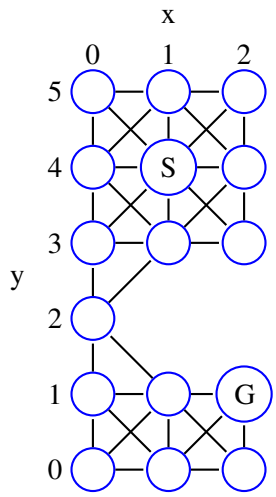
i. Is h1 admissible? Is it consistent?

Solution: Neither admissible nor consistent.

ii. Is h2 admissible? Is it consistent?

Solution: Admissible but not consistent.

Question 10 ***(0 points)***



In the maze shown above, nodes are named by their (x,y) coordinates, where x is the column number (starting from the left), and y is the row number (starting from the bottom), A robot is trying to find a path from the start node, $(1,4)$ (labeled "S"), to the goal node, $(2,1)$ (labeled "G"). Every horizontal or vertical step has a cost of 1 unit; every diagonal step has a cost of $\sqrt{2}$ units. Prove that Manhattan distance is not a consistent heuristic for this problem.

Solution: We can prove that Manhattan distance is not consistent by finding any nodes p and r such that

$$h(p) \geq d(p,r) + h(r)$$

An example would be nodes $p = (0,2)$ and $r = (1,1)$. The distance between these nodes is $d(p,r) = \sqrt{(2)}$. If Manhattan distance is the heuristic, then $h((1,1)) = 1$, while $h((0,2)) = 3$, so

$$h((0,2)) \geq d((0,2),(1,1)) + h((1,1))$$

Question 11 ***(0 points)***

Discuss the relative strengths and weaknesses of breadth-first search vs. depth-first search for AI problems.

Solution:

	BFS	DFS
Strength	Solution is guaranteed to be optimal. BFS is complete.	Can find goal faster than BFS if there are multiple solutions. Linear memory requirement $(O(bm))$.
Weakness	Exponential $(O(b^d))$ space complexity.	Solution not guaranteed to be optimal. Not complete. Computation is exponential in longest path $(O(b^m))$, rather than shortest path $(O(b^d))$.

Question 12 ***(0 points)***

In the tree search formulation, why do we restrict step costs to be non-negative?

Solution: If the cost around any loop is negative, then the lowest-cost path is to take that loop an infinite number of times.

Question 13 ***(0 points)***

What is the distinction between a world state and a search tree node?

Solution: A world state contains enough information to know (1) whether or not you've reached the goal, (2) what actions can be performed, (3) what will be the result of each action. A search tree node contains a pointer to the world state, plus a pointer to the parent node.

Question 14 ***(0 points)***

How do we avoid repeated states during tree search?

Solution: By keeping a set of "explored states." If expanding a search node results in a state that has already been explored, we don't add it to the frontier.

Question 15 ***(0 points)***

You are searching a maze in which positions are denoted by their (x,y) coordinate pairs, and in which the start node is $s = (4,4)$ and the goal node is $g = (0,0)$. Each step consists of one move in the horizontal or vertical direction; diagonal steps are not allowed. Your friend proposes implementing an A* search algorithm using the following heuristic:

$$h((x,y)) = |\sin(x)| + |\sin(y)|$$

Prove that $h((x,y))$ is an admissible heuristic for this problem.

Solution: An admissible heuristic is defined by

$$h(p) \leq d(p, \text{Goal})$$

Since diagonal steps are not allowed in this problem, and since the goal is $(0,0)$, we know that

$$|x| + |y| \leq d(p, \text{Goal})$$

But from calculus, we know that

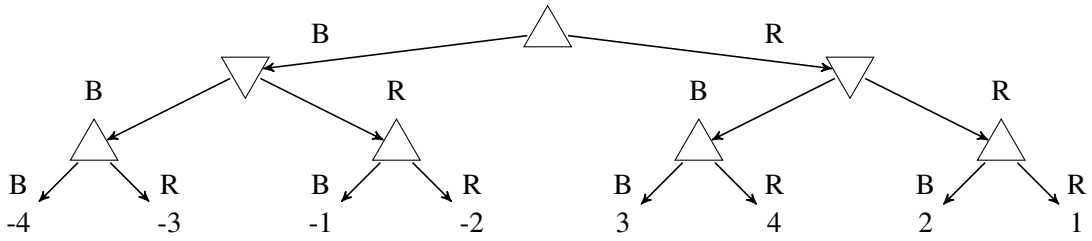
$$h((x,y)) = |\sin(x)| + |\sin(y)| \leq |x| + |y| \leq d(p, \text{Goal})$$

Therefore $h((x,y))$ is admissible.

2 Minimax

Question 16 ***(8 points)***

The following **minimax tree** shows all possible outcomes of the RED-BLUE game. In this game, Max plays first, then Min, then Max. Each player, when it's their turn, chooses either a blue stone (B) or a red stone (R); after three turns, Max wins the number of points shown (negative scores indicate a win for Min).



(a) (3 points) Max could be a Reflex Agent, following a set of predefined IF-THEN rules, and could still play optimally against Min, even if Min is not rational. To do so, Max needs just three rules of the form "If the stones already chosen are ____, then choose a ____ stone." Write those three rules in that form.

Solution:

- If no stones have been chosen, then choose a Red stone.
- If the stones already chosen are (R,B), then choose a Red stone.
- If the stones already chosen are (R,R), then choose a Blue stone.

(b) (2 points) Recall that an $\alpha - \beta$ search prunes the largest possible number of moves if there is extra information available to the players that permits them to evaluate the moves in the best possible order. IN GENERAL (not just for this game tree),

- In what order should the moves available to MAX be evaluated, in order to prune as many moves as possible?
- In what order should the moves available to MIN be evaluated, in order to prune as many moves as possible?

Solution:

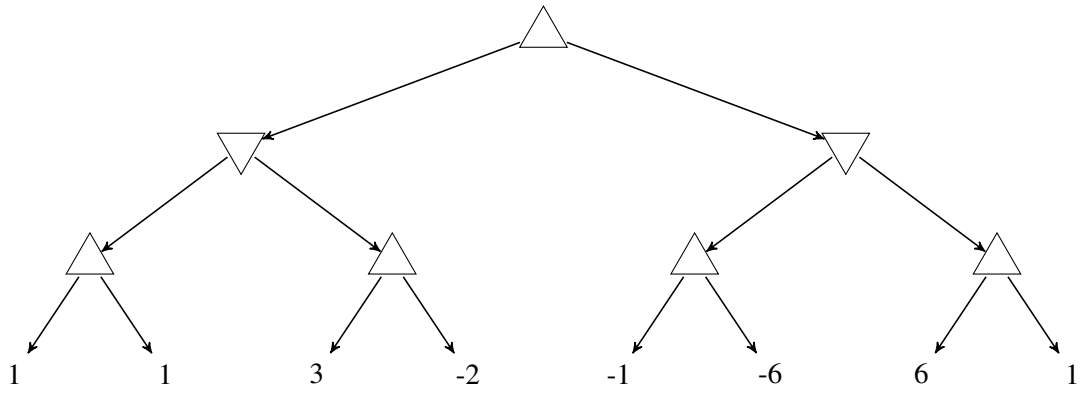
- Moves available to MAX should be evaluated in order of descending value, starting with the highest-value move.
- Moves available to MIN should be evaluated in order of ascending value, starting with the lowest-value move.

(c) (3 points) Re-draw the minimax tree for the RED-BLUE game so that, if moves are always evaluated from left to right, the $\alpha - \beta$ search only needs to evaluate 5 of the 8 terminal states.

Solution: The first two leaves should be (1,2) or (2,1). The next two leaves should be (3,4) or (4,3). The last four leaves should be (-1,2,-3,-4), in any order.

Question 17 ***(0 points)***

Two players, MAX and MIN, are playing a game. The game tree is shown below. Upward-pointing triangles denote decisions by MAX; downward-pointing triangles denote decisions by MIN. Numbers on the terminal nodes show the final score: MAX seeks to maximize the final score, MIN seeks to minimize the final score.



(a) Write the minimax value of each nonterminal node (each upward-pointing or downward-pointing triangle) next to it.

Solution: From top to bottom, left to right, the values are 1, 1, -1, 1, 3, -1, 6.

(b) Suppose that the minimax values of the nodes at each level are computed in order, from left to right. Draw an X through any edge that would be pruned (eliminated from consideration) using alpha-beta pruning.

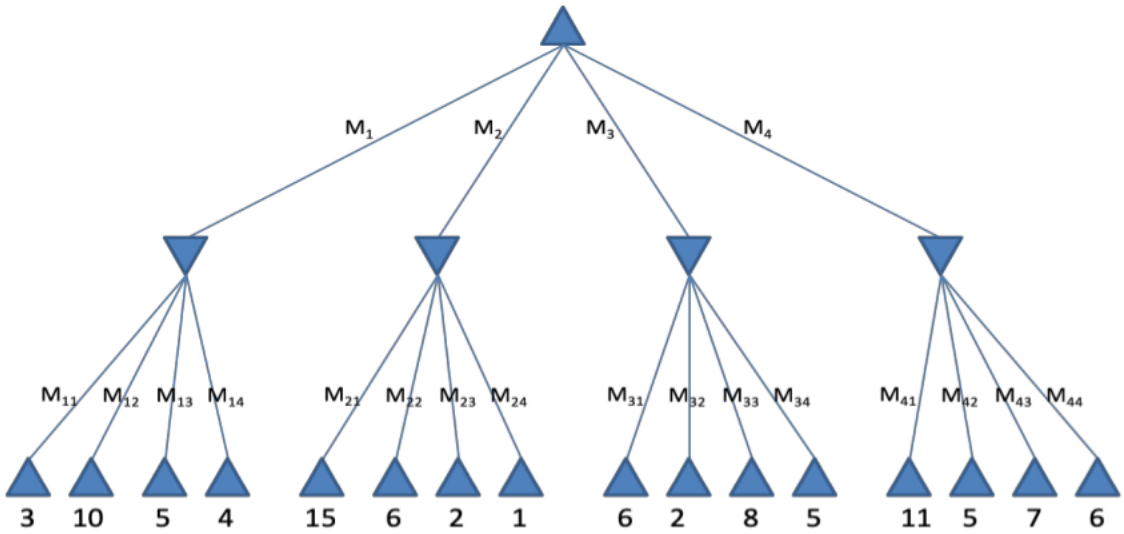
Solution: The 4th edge at the bottom level, and the 4th edge at the middle level, would both be pruned.

(c) In this game, alpha-beta pruning did not change the minimax value of the start node. Is there any deterministic two-player game tree in which alpha-beta pruning changes the minimax value of the start node? Why or why not?

Solution: No. Alpha-beta pruning only prunes branches that have no effect on the start node.

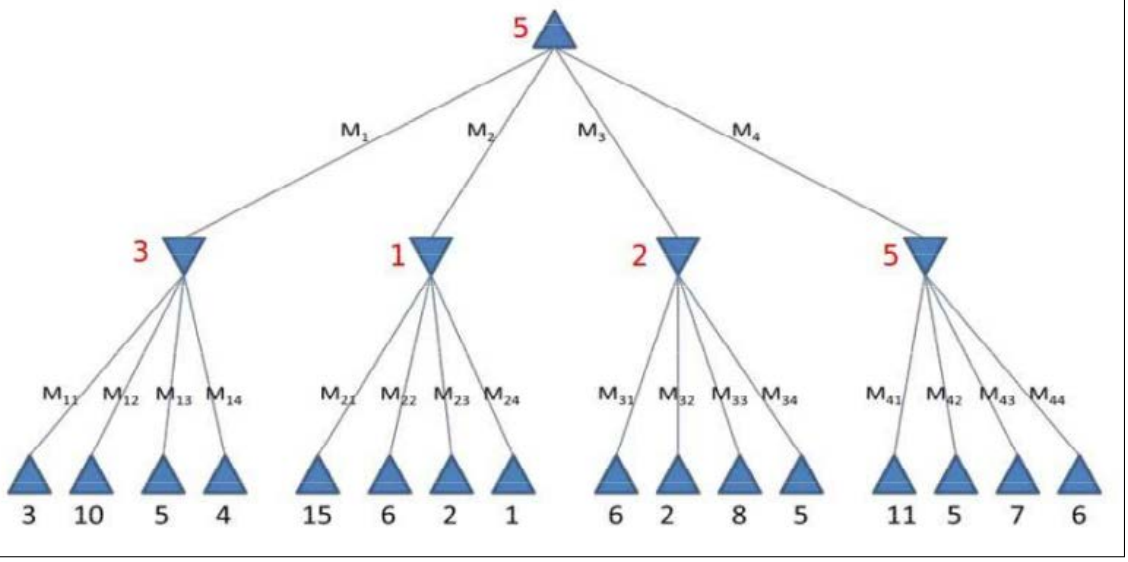
Question 18 ***(0 points)***

Consider the following game tree (MAX moves first):



(a) Write down the minimax value of every non-terminal node next to that node.

Solution:

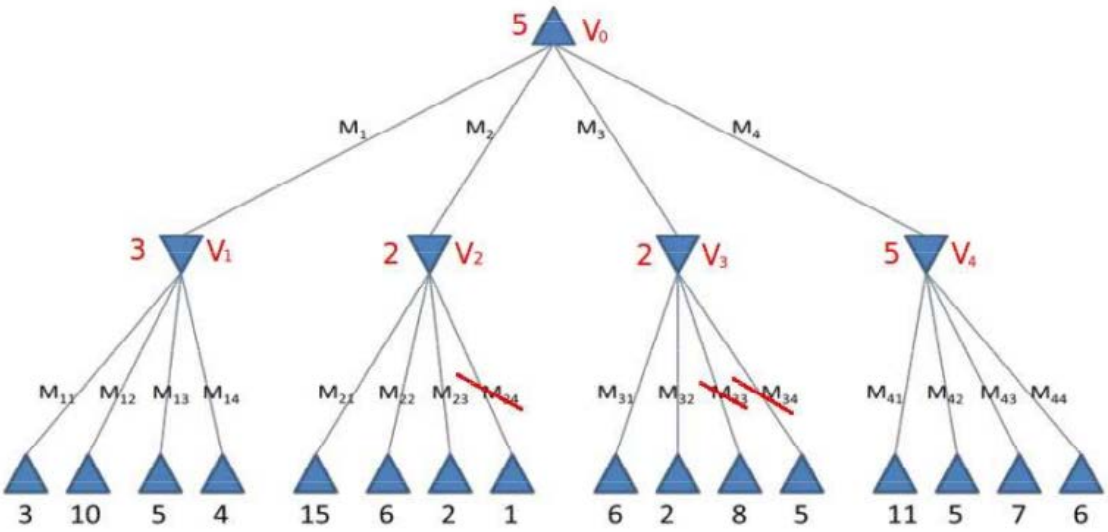


(b) How will the game proceed, assuming both players play optimally?

Solution: The game will choose the max value on depth 1, taking route M_4. It will then take the minimum value on depth 2 that is child of the chosen node, and hence take M_{42}.

(c) Cross out the branches that do not need to be examined by alpha-beta search in order to find the minimax value of the top node, assuming that moves are considered in the non-optimal order shown.

Solution:



(d) Suppose that a heuristic was available that could re-order the moves of both max (M_1, M_2, M_3, M_4) and min ($M_{11}, \ldots, M_{44}$) in order to force the alpha-beta algorithm to prune as many nodes as possible. Which max move would be considered first: M_1, M_2, M_3, or M_4? Which of the min moves ($M_{11}, \ldots, M_{44}$) would have to be considered?

Solution: The first max move to be considered would be V_4, because it allows us to set the highest α. Only 7 of the min moves would be considered: M_{41} through M_{44} would have to be considered to determine that $\alpha = 5$, and then (if the heuristic magically sorts moves in order for us), we would consider M_{32}, M_{24}, and M_{11}, find that all of them have values below α, and prune away their parents.

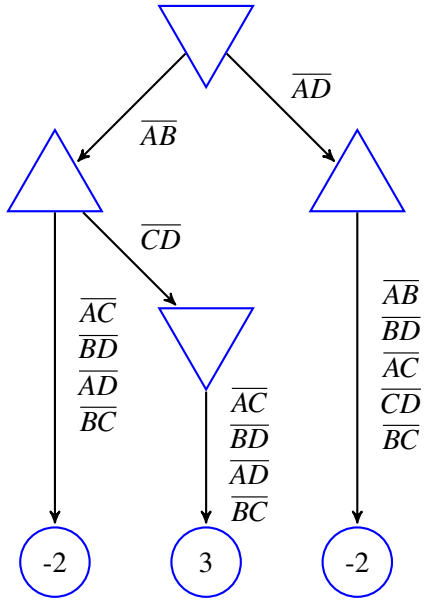
Question 19 ***(0 points)***

C D

A B

Consider a game in which Min plays first, and on the first move, Min must draw either the line segment $\overline{AB}$ or the line segment $\overline{AD}$. Thereafter players take turns; on each turn after the first one, the person playing must draw one line segment connecting any two of the four corners A, B, C, or D. The game ends when one of the players draws a line segment that touches or crosses any previously-drawn line segment; at that point, the other player wins a number of points equal to the total number of line segments that have been drawn. Draw a complete game tree, indicating each move by the letters of the endpoints of the line segment drawn by that player. If many different moves result in the same subtree, draw just one edge that lists all of those moves. Indicate a minimax sequence of moves, and specify the minimax value of the game.

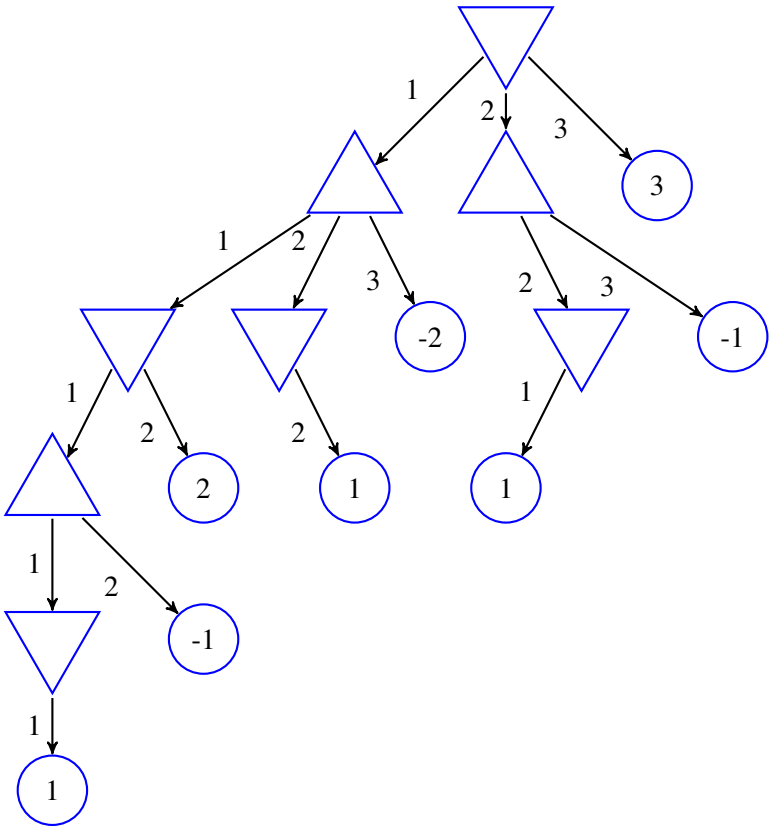
Solution:



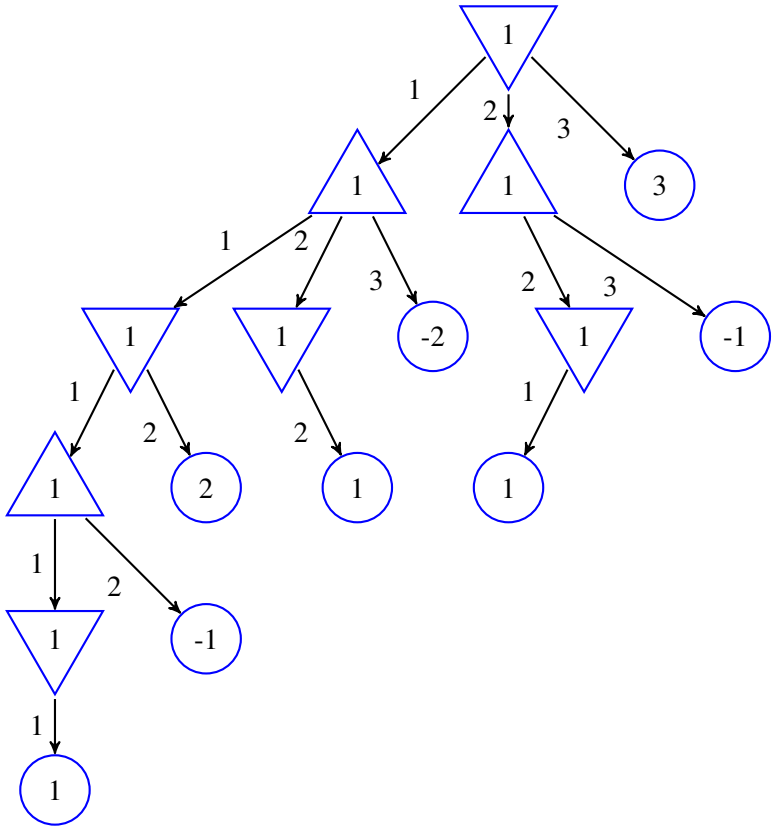
The minimax value is -2. The minimax sequence must begin with Min playing $\overline{AD}$, then any move that Max plays will result in the minimax game value.

Question 20 ***(0 points)***

Consider a game in which Max and Min each start with three stones. Min plays first. On their turn, each player must discard one, two, or three stones. The number of stones a player discards must be greater than or equal to the number of stones their opponent discarded on the immediate preceding turn (if a player does not have enough stones to satisfy this rule, they must discard all of their remaining stones). When one player loses their last stone, the other player wins a number of points equal to the number of stones they have not yet discarded. The figure below shows the game tree for this game. In each triangle (each Min or Max node), enter a number specifying the value of that node. Specify the minimax value of the game. Indicate a minimax move sequence (there are several different sequences that a pair of optimal players might play; specify one of those sequences).

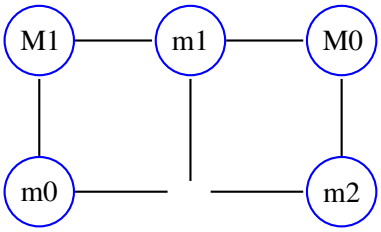


Solution:

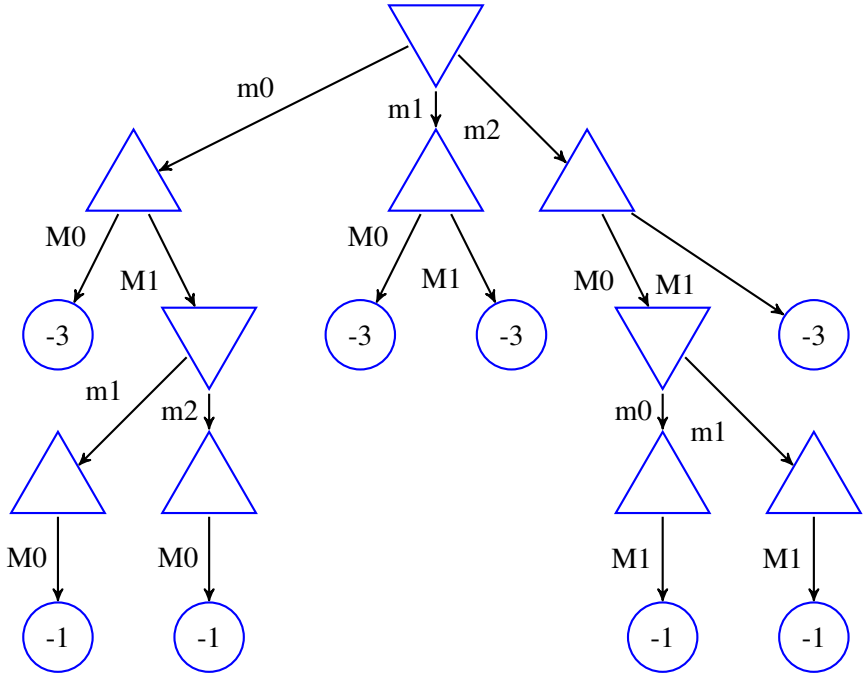


The minimax value is +1. One sequence of moves with this value: each player plays 1 stone, in turn, until Min plays her last stone. Another possible solution: Min plays 1, Max plays 2, Min plays 2. Another possible solution: Min plays 2, Max plays 2, Min plays her last stone.

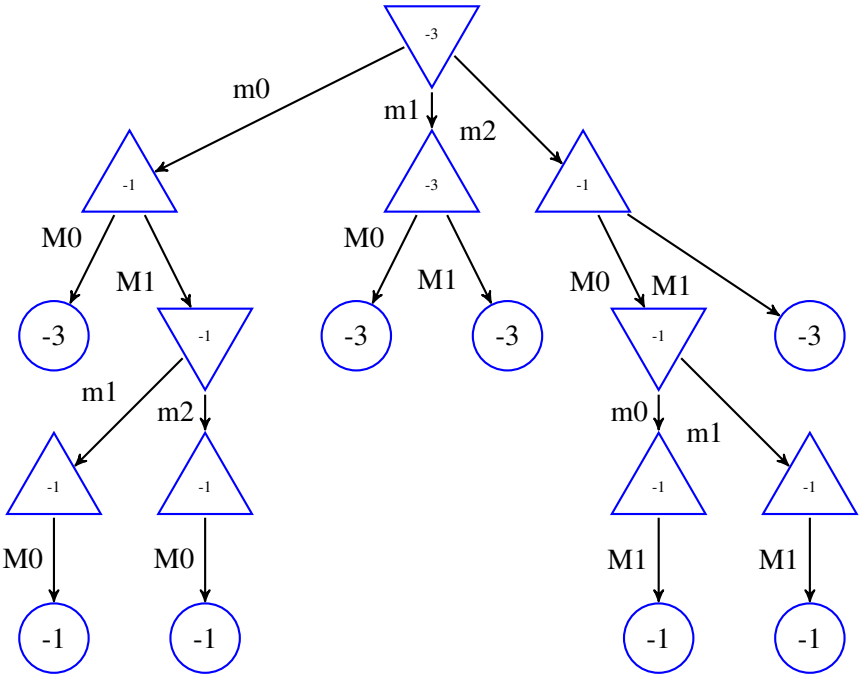
Question 21 ***(0 points)***



Consider a game in which Min starts with 3 stones on the edges of a square (shown above as "m0,m1" and "m2"), and Max starts with 2 stones on the corners of the square (shown above as "M0" and "M1"). Min plays first. On their turn, each player removes one of their own stones. The game ends when one player has any stone with no remaining neighbors; that player wins a number of points equal to the number of stones they still have on the board. The figure below shows the game tree for this game. In each triangle (each Min or Max node), enter a number specifying the value of that node. Specify the minimax value of the game. Indicate a minimax move sequence (there are several different sequences that a pair of optimal players might play; specify one of those sequences).



Solution:



The minimax value is -3. The two optimal sequences are (m1,M1) and (m1,M0); note that it is not optimal for Min to start with m0 or m2.

Question 22 ***(0 points)***

What are the main challenges of adversarial search as contrasted with single-agent search? What are some algorithmic similarities and differences?

Solution: The biggest difference is that we are unaware of how the opponent(s) will act. Because of this our search cannot simply consider my own moves, it must also figure out how my opponent will act at each level, thus effectively doubling the number of levels over which I have to search. Since the number of levels is the exponent in the computational complexity, this makes computational complexity much harder.

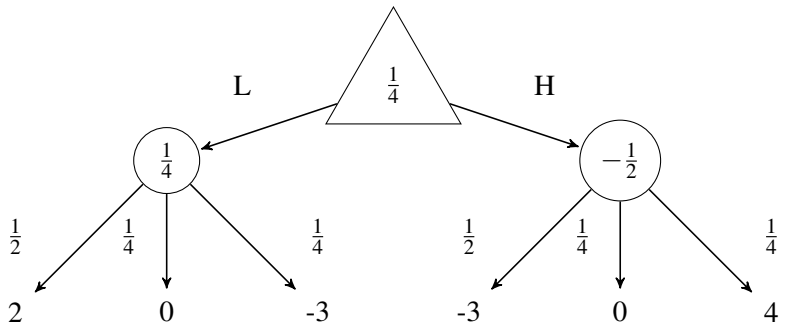
3 Expectiminimax

Question 23 ***(0 points)***

Consider the following game, called "High/Low." There is an infinite deck of cards, half of which are 2's, one quarter are 3's, and one quarter are 4's. The game starts with a 3 showing. After each card, you say "High" or "Low," and a new card is flipped. If you are correct (e.g., you say "High" and then the next card is higher than the one showing), you win the points shown on the new card. If there is a tie (the next card equals the one showing), you get zero points. If you are wrong (e.g., you say "High" and then the next card is lower than the one showing), then you lose the amount of the card that was already showing.

Draw the expectimax tree for the first round of this game and write down the expected utility of every node. What is the optimal policy assuming the game only lasts one round?

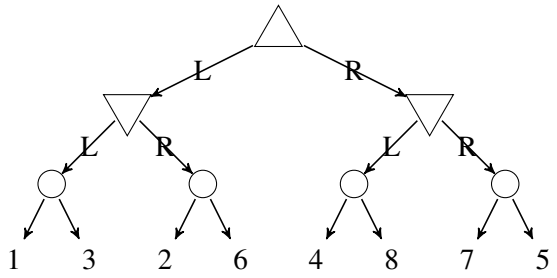
Solution:



Optimal policy: L

Question 24 ***(5 points)***

Consider a game with eight cards ($c \in \{1,2,3,4,5,6,7,8\}$), sorted onto the table in four stacks of two cards each. MAX and MIN each know the contents of each stack, but they don't know which card is on top. The game proceeds as follows. First, MAX chooses either the left or the right pair of stacks. Second, MIN chooses either the left or the right stack, within the pair that MAX chose. Finally, the top card is revealed. MAX receives the face value of the card (c), and MIN receives $9-c$. The resulting expectiminimax tree is as follows:



(a) (2 points) Assume that the two cards in each stack are equally likely. What is the value of the top MAX node?

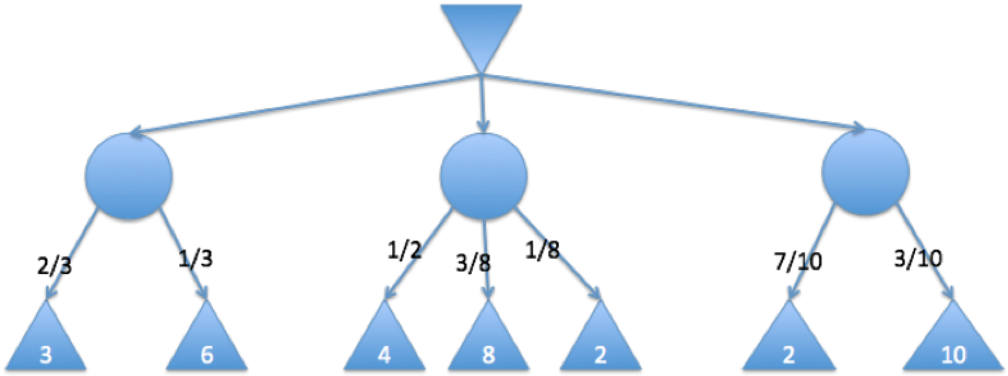
Solution: Propagating backward using expectiminimax, we find that the value of the top node is 6.

(b) (3 points) Consider the following rule change: after MAX chooses a pair of stacks, he is permitted to look at the top card in any one stack. He must show the card to MIN, then replace it, so that it remains the top card in that stack. Define the belief state, b, to be the set of all possible outcomes of the game, i.e., the starting belief state is the set $b = \{1,2,3,4,5,6,7,8\}$; the PREDICT operation modifies the belief state based on the action of a player, and the OBSERVE operation modifies the belief state based on MAX's observation. Suppose MAX chooses the action R. He then turns up the top card in the rightmost deck, revealing it to be a 7. What is the resulting belief state?

Solution: After MAX chooses the right set of stacks, the PREDICT update step results in a belief state of $b = \{4,8,7,5\}$. After he looks at the rightmost deck and finds that it contains a 7, the OBSERVE update step restricts the belief state to $b = \{4,8,7\}$.

Question 25 ***(0 points)***

Consider the following expectiminimax tree:



Circle nodes are chance nodes, the top node is a min node, and the bottom nodes are max nodes.

(a) For each circle, calculate the node values, as per expectiminimax definition.

Solution: From left to right: 4, 5.25, 4.4.

(b) Which action should the min player take?

Solution: The first action.

Question 26 ***(0 points)***

What additional difficulties does dice throwing or other sources of uncertainty introduce into a game?

Solution: Uncertainties introduce probabilities into the game. Expectiminimax is used to find solutions for these type of games. Expectiminimax doubles the number of levels as compared to minimax, because there is randomness after every play. Expectiminimax has nasty branching factor and often times defining evaluation functions and pruning algorithms are difficult.

Question 27 ***(0 points)***

How can randomness be incorporated into a game tree? How about partial observability (imperfect information)?

Solution: Randomness is incorporated using the expectiminimax algorithm, in which max tries to maximize the expected score, min tries to minimize the expected score. Partial observability is incorporated using a minimax state tree in which neither player knows for sure which state they're in; the max player chooses an action that maximizes the minimum payoff over all of the states he might be in.

4 Markov Decision Processes

Question 28 ***(7 points)***

ATARA is an Automatic Telephone-based Airplane Reservation Agent.

In order to make an airplane reservation, ATARA needs to learn the user's starting city, ending city, and date of travel (she always asks in that order). When she starts each dialog, she knows none of these things.

During each of her dialog turns, ATARA has the option of asking for 1 or 2 pieces of information. Unfortunately, her speech recognizer makes mistakes. If she asks for 1 piece of information, she always gets it. If she asks for 2 pieces of information, then she gets both pieces of information with probability $\left(\frac{1}{2}\right)$, but with probability $\left(\frac{1}{2}\right)$, she gets nothing.

ATARA receives a reward of $R(s) = 10$, and ends the dialog, when she has correctly recognized all 3 pieces of information. Otherwise, she gets a reward of $R(s) = -1$ for each dialog turn during which she has not finished the dialog.

(a) (1 point) What is the set of states for this Markov decision process?

Solution: The states are $s \in \{0,1,2,3\}$, specifying the number of pieces of information ATARA has collected.

(b) (1 point) What is the set of actions?

Solution: The actions are $a \in \{1,2\}$, representing the number of pieces of information that ATARA requests.

(c) (3 points) Write the transition probability table $P(s'|s,a)$.

Solution:

(s,a)	0	1	2	3
$(0,1)$	0	1	0	0
$(0,2)$	1/2	0	1/2	0
$(1,1)$	0	0	1	0
$(1,2)$	0	1/2	0	1/2
$(2,1)$	0	0	0	1
$(2,2)$	0	0	1/2	1/2

(d) (2 points) Use value iteration to find $U(s)$, the utility of each state, assuming a discount factor of $\gamma = 1$.

Solution: The utility estimate at each iteration is given as follows; after $t = 5$, the utility no longer changes.

t	0	1	2	3
1	0	0	0	0
2	-1	-1	-1	10
3	-2	3.5	9	10
4	2.5	8	9	10
5	7	8	9	10

Question 29 ***(0 points)***

Consider an MDP with two states ($s \in \{0,1\}$), two actions ($a \in \{0,1\}$), with rewards of $R(0) = -10$ and $R(1) = 10$, and with the following transition probabilities:

s'	$P(s'\|s=0,a=0)$	$P(s'\|s=0,a=1)$	$P(s'\|s=1,a=0)$	$P(s'\|s=1,a=1)$
0	0.4	0.7	0.9	0.2
1	0.6	0.3	0.1	0.8

Consider trying to solve for the utilities of the two states, $U(0)$ and $U(1)$, assuming $\gamma = \frac{1}{2}$, using policy iteration. Consider starting with the policy $\pi_1(0) = 0$, $\pi_1(1) = 1$. Write two equations that could be solved to find the policy-dependent utilities $U_1(0)$ and $U_1(1)$ given this policy. Write your equations in terms of the variables $R(0), R(1), P(0|0,0), P(1|0,0), P(0|0,1), P(1|0,1), P(0|1,0), P(1|1,0), P(0|1,1)$, $P(1|1,1)$, and/or γ first, then substitute in the provided numerical values.

Solution:

$$U_1(0) = R(0) + \gamma\left(\sum_{s'} P(s'|s=0,a=0)U_1(s')\right)$$

$$U_1(1) = R(1) + \gamma\left(\sum_{s'} P(s'|s=1,a=1)U_1(s')\right)$$

With the values filled in, this is:

$$U_1(0) = -10 + \frac{1}{2}\left(0.4U_1(0) + 0.6U_1(1)\right) U_1(1) \qquad = 10 + \frac{1}{2}\left(0.2U_1(0) + 0.8U_1(1)\right)$$

Question 30 ***(0 points)***

After t iterations of the "Value Iteration" algorithm, the estimated utility $U(s)$ is a summation including terms $R(s')$ for the set of states s' that can be reached from state s in at most $t-1$ steps.

- √ **True**
- ◯ False

Explain:

Solution: Value iteration starts with $U(s) = 0$. Each iteration updates $U(s)$ by adding $R(s)$, plus the maximum over all actions of the expected utility $U(s')$ of the state s' that can be reached from state s in one step. In t iterations of this algorithm, one accumulates rewards from states that are up to $t-1$ steps away.

Question 31 ***(0 points)***

Consider an MDP with two states ($s \in \{0,1\}$), two actions ($a \in \{0,1\}$), with rewards of $R(0) = -10$ and $R(1) = 10$, and with the following transition probabilities:

s'	$P(s'\|s=0,a=0)$	$P(s'\|s=0,a=1)$	$P(s'\|s=1,a=0)$	$P(s'\|s=1,a=1)$
0	0.4	0.7	0.9	0.2
1	0.6	0.3	0.1	0.8

Consider trying to solve for the utilities of the two states, $U(0)$ and $U(1)$, assuming $\gamma = \frac{1}{2}$, using value iteration. Consider starting with the values $U_1(0) = R(0) = -10$, $U_1(1) = R(1) = 10$. Write two equations that could be solved to find the second-iteration values $U_2(0)$ and $U_2(1)$. Write your equations in terms of the variables $R(0)$, $R(1)$, $U_1(0)$, $U_1(1)$, $P(0|0,0)$, $P(1|0,0)$, $P(0|0,1)$, $P(1|0,1)$, $P(0|1,0)$, $P(1|1,0)$, $P(0|1,1)$, $P(1|1,1)$ and/or γ first, then substitute in the provided numerical values.

Solution:

$$U_2(0) = R(0) + \gamma \max_a \sum_{s'} P(s'|s=0,a) U_1(s')$$
$$U_2(1) = R(1) + \gamma \max_a \sum_{s'} P(s'|s=1,a) U_1(s')$$

With the values filled in, this is:

$$U_2(0) = -10 + \frac{1}{2} \max\left((0.4(-10) + 0.6(10)), (0.7(-10) + 0.3(10))\right)$$
$$U_2(1) = 10 + \frac{1}{2} \max\left((0.9(-10) + 0.1(10)), (0.2(-10) + 0.8(10))\right)$$

	Simplified GridWorld Column Number	
	1	2
1	-0.04	-0.04
2	-1.00	1.00
3	0.00	-0.04

Question 32 ***(0 points)***

Consider a simplified version of GridWorld, shown above. The grid above shows the reward, $R(s)$, associated with each state. The robot starts in the state with $R(s) = 0.00$; if it reaches either the state with $R(s) = 1.00$ or $R(s) = -1.00$, the game ends.

The transition probabilities are simpler than the ones used in lecture. Let the action variable, a, denote the state to which the robot is trying to move. If the robot tries to move out of the maze, it always stays in the state where it started. If the robot tries to move to any state that is a neighbor of the state it currently occupies, then it either succeeds (with probability 0.8), or else it remains in the same state (with probability 0.2). To put the same transition probabilities in the form of an equation, we could write:

$$P(s'|s,a) = \begin{cases} 0.8 & s' = a, \ \ a \in \text{NEIGHBORS}(s) \\ 0.2 & s' = s \\ 0 & \text{otherwise} \end{cases}$$

After one round of value iteration, $U_1(s) = R(s)$.

(a) After the second round of value iteration, with discount factor $\gamma = 1$, what are the values of all of the states? In other words, what is $U_2(s)$ for each of the six states? List the six values, in left-to-right, top-to-bottom order.

Solution: With states indexed by (row,column), we have:

$$U_2(1,1) = -0.08$$
$$U_2(1,2) = 0.752$$
$$U_2(2,1) = -1.0$$
$$U_2(2,2) = 1.0$$
$$U_2(3,1) = 0.0$$
$$U_2(3,2) = 0.752$$

(b) After how many rounds of value iteration (at what value of t) will $U_t(\text{START})$, the value of the starting state, become positive for the first time?

Solution: $U_3(3,1) = 0.6016$, so it first becomes positive at $t = 3$.

5 Game Theory

Question 33 ***(0 points)***

The "Battle of the Species" game is defined as follows. Imagine a cat and a dog have agreed to meet for the evening, but they forgot whether they were going to meet at a frisbee field or an aquarium. The dog prefers the frisbee field and the cat prefers the aquarium. The payoff for each one's preferred activity is 4 and the payoff for the non-preferred activity is 3 – assuming the cat and the dog end up at the same place. If they end up at different places, each gets a 1 if they are at their preferred place, and 0 if they are at their non-preferred place.

(a) Give the normal form (matrix) representation of the game.

Solution:

	Dog: Frisbee	Dog: Aquarium
Cat: Frisbee	C:3,D:4	D:0,C:0
Cat: Aquarium	C:1,D:1	C:4,D:3

(b) Find dominant strategies (if any). Briefly explain your answer.

Solution: None. A dominant strategy is a strategy that maximizes the player's payoff regardless of what the other player does. In this case, if one player chooses frisbee, the other one should choose frisbee, and if one chooses aquarium, the other one should choose aquarium. Therefore, there is no dominant strategy.

(c) Find pure strategy equilibria (if any). Briefly explain your answer.

Solution: (Dog: frisbee; Cat: frisbee); (Dog: aquarium; Cat: aquarium). From either of these two states, no player can get a bigger payoff from changing actions unilaterally.

Question 34 ***(0 points)***

Suppose that both Alice and Bob want to go from one place to another. There are two routes R1 and R2. The utility of a route is inversely proportional to the number of cars on the road. For instance, if both Alice and Bob choose route R1, the utility of R1 for each of them is 1/2.

(a) Write out the payoff matrix.

Solution:

	Alice R1	Alice R2
Bob R1	A:0.5, B:0.5	A:1, B:1
Bob R2	A:1, B:1	A:0.5, B:0.5

(b) Is this a zero-sum game? Why or why not?

Solution: No. The rewards for Bob and Alice do not sum to zero.

(c) Find dominant strategies, if any. If there are no dominant strategies, explain why not.

Solution: There is no dominant strategy for either player. The best strategy for each player depends on the strategy of the other player.

(d) Find pure strategy equilibria, if any. If there are no pure strategy equilibria, explain why not.

Solution: There are two: (Alice=R1,Bob=R2) and (Alice=R2,Bob=R1).

(e) Find the mixed strategy equilibrium.

Solution: Alice chooses R1 with probability p, and R2 with probability $1-p$. p must be chosen so that Bob's reward is independent of the action he takes.

- Bob's Reward(R1)$= 0.5p + (1-p) = 1 - 0.5p$
- Bob's Reward(R2)$= p + 0.5(1-p) = 0.5 + 0.5p$

Setting the two rewards equal, we find $p = 0.5$. The equations for Alice are the same, so she should also choose R1 with probability $p = 0.5$.

Question 35 ***(0 points)***

Consider the following game:

	Player A: Action 1	Player A: Action 2
Player B: Action 1	A=3 B=2	A=0 B=0
Player B: Action 2	A=1 B=1	A=2 B=3

(a) Find dominant strategies (if any).

Solution: A dominant strategy is defined as a strategy whose outcome is better for the player regardless of the strategy chosen by the other player. Let's first look for dominant strategies for A: Suppose B chooses Action1. A gets 3 if it chooses Action1 or 0 if it chooses Action2. So it shoould choose Action1. Now suppose B chooses Action2. A gets 1 if it chooses Action1 or 2 if it chooses Action2. So it should choose Action2. Thus there is no dominant strategy for A. Let's look at B: Suppose A chooses Action1. B gets 2 if it chooses Action1 or 1 if it chooses Action2. So it should choose Action1. Now suppose A chooses Action2. B gets 0 if it chooses Action1 or 3 if it chooses Action2. So it should choose Action2. Thus there is also no dominant strategy for B.

(b) Find pure strategy equilibria (if any).

Solution: A Nash Equilibrium is a set of strategies such that no player can get a bigger payoff by switching strageties, provided the other player sticks with the same strategy. There are two: (A: Action1, B: Action1) or (A: Action2, B: Action2).

Question 36 ***(0 points)***

In each square, the first number refers to payoff for the player whose moves are shown on the row-label, the second number refers to payoff for the player shown on the column label.

	A'	B'	C'
A	0, 0	25, 40	5, 10
B	40, 25	0, 0	5, 15
C	10, 5	15, 5	10, 10

(a) Are there any dominant strategies? If so, what are they? If not, why not?

Solution: No. The best move, for each player, depends on what the other player does.

(b) Are there any pure-strategy Nash equilibria? If so, what are they? If not, why not?

Solution: A Nash equilibrium is an outcome such that, given the other player's action each player has no reason to change actions. There are three in this game: (A,B'), (B,A'), (C,C')

(c) Are there any Pareto-optimal solutions? If so, what are they? If not, why not?

Solution: A Pareto-optimal solution is one that is optimal, in the sense that no other outcome would increase the reward for player 2 unless it decreases the reward for player 1. There are two: (A,B') and (B,A').

6 Logic

Question 37 ***(0 points)***

Consider the problem of trying to prove that birds aren't real. You have the following goalset:

$$\mathscr{G} = \{\text{NOT-REAL}(\text{birds})\}$$

In the attempt to prove your goalset, you will make use of the following rule database:

$$\mathscr{D} = \left\{ \begin{array}{rl} \text{HAVE}(u, \text{wings}) & \Rightarrow \text{FLY}(u) \\ \text{HAVE}(x, \text{feathers}) & \Rightarrow \text{HAVE}(x, \text{scales}) \\ \text{FLY}(y) \wedge \text{HAVE}(y, \text{scales}) & \Rightarrow \text{NOT-REAL}(y) \end{array} \right\}$$

(a) After one step of backward chaining, what is the goalset?

Solution:

$$\mathscr{G} = \{\text{FLY}(\text{birds}) \wedge \text{HAVE}(\text{birds}, \text{scales})\}$$

(b) There are two different goalsets that might result from the second step of backward chaining. What are they?

Solution:

$$\mathscr{G} = \{\text{HAVE}(\text{birds}, \text{wings}) \wedge \text{HAVE}(\text{birds}, \text{scales})\}$$
$$\mathscr{G} = \{\text{FLY}(\text{birds}) \wedge \text{HAVE}(\text{birds}, \text{feathers})\}$$

Question 38 ***(0 points)***

u, v, w, x, y, and z are variables. You are trying to determine whether or not it's possible to perform a step of backward-chaining using the rule $T = \text{Sings}(u, \text{folkmusic}) \Rightarrow \text{Plays}(u, \text{guitar})$. Your goalset, $\mathscr{G}$, currently includes the following goals:

$$\mathscr{G} = \left\{ \begin{array}{c} P = \text{Eats}(\text{tiger}, \text{cellphone}) \\ Q = \exists v : \text{Plays}(\text{anne}, v) \\ R = \exists w : \text{Plays}(w, \text{flute}) \\ S = \exists x : \text{Zambonis}(x, \text{icerink}) \end{array} \right\}$$

Which proposition (P, Q, R, or S) can be unified with the consequent of T? What is the resulting unified proposition, what is the resulting substitution dictionary, and what new fact is added to the goalset?

Solution: Q can be unified to the consequent of T, producing $\text{Plays}(\text{anne}, \text{guitar})$, $S = \{u : \text{anne}, v : \text{guitar}\}$, and adding $\text{Sings}(\text{anne}, \text{folkmusic})$ to the goalset.

Question 39 ***(0 points)***

Consider the problem of trying to prove that Chicago is a city. Your goalset is the theorem:

$$\mathscr{G} = \{\text{CITY}(\text{chicago})\}$$

In the attempt to prove your goalset, you will use forward chaining, starting from the following fact database:

$$\mathscr{D} = \left\{ \begin{array}{ll} P_1: & \neg\text{COUNTRY}(x) \wedge \text{POPULATION}(x,y) \wedge \text{LARGER}(y,10000) \Rightarrow \text{CITY}(x) \\ P_2: & \text{POPULATION}(\text{chicago}, 7000000) \\ P_3: & \text{LARGER}(7000000, 10000) \\ P_4: & \text{PART-OF}(u,v) \wedge \text{COUNTRY}(v) \Rightarrow \neg\text{COUNTRY}(u) \\ P_5: & \text{PART-OF}(\text{chicago}, \text{usa}) \\ P_6: & \text{COUNTRY}(\text{usa}) \end{array} \right\}$$

Suppose that the first forward-chaining step determines that the consequent of P_4 can be unified with one of the antecedents of P_1. What new fact is added to the database as a result? What is the substitution dictionary that makes insertion of this new fact possible? Be sure that the variable names in your new fact are normalized so that they will not be confused with variables elsewhere in the database.

Solution: The new fact is

$$\text{PART-OF}(a,b) \wedge \text{COUNTRY}(b) \wedge \text{POPULATION}(a,c) \wedge \text{LARGER}(c,10000) \Rightarrow \text{CITY}(a)$$

The substitution dictionary is

$$\{x:a, y:c, u:a, v:b\}$$

Question 40 ***(0 points)***

Two propositions, P and Q, can be unified to create the proposition $U = \text{Flies}(\text{mary}, \text{hotairballoons})$. The substitution dictionary creating this unification is $S = \{u : \text{mary}, v : \text{hotairballoons}\}$. Is this information sufficient to specify the two propositions P and Q? Prove your answer.

Solution: No. The propositions $P = \text{Flies}(\text{mary}, \text{hotairballoons})$ and $Q = \text{Flies}(u, v)$ would produce this result. The propositions $P = \text{Flies}(u, \text{hotairballoons})$ and $Q = \text{Flies}(\text{mary}, v)$ would also produce this result.

Question 41 ***(0 points)***

u, v, w, x, y, and z are variables. You are trying to determine whether or not it's possible to perform a step of forward-chaining using the rule $T = \text{Uses}(u, \text{cellphone}) \Rightarrow \text{Human}(u)$. The facts currently available to you in the database $\mathscr{D}$ are:

$$\mathscr{D} = \left\{ \begin{array}{c} P = \text{Eats}(\text{tiger}, \text{cellphone}) \\ Q = \forall v : \text{Uses}(v, \text{landline}) \\ R = \forall w : \text{Uses}(\text{george}, w) \\ S = \exists x : \text{Zambonis}(x, \text{icerink}) \end{array} \right\}$$

Which proposition (P, Q, R, or S) can be unified with the antecedent of T? What is the resulting unified proposition, what is the resulting substitution dictionary, and what new fact is added to the database?

Solution: R can be unified with T, giving $\text{Uses}(\text{george}, \text{cellphone})$, $S = \{u : \text{george}, w : \text{cellphone}\}$, and adding $\text{Human}(\text{george})$ to the database.

7 Vector Semantics

Question 42 ***(0 points)***

The continuous bag of words (CBOW) model represents each word, w_t, using a vector, $\mathbf{v}_t$. The vectors are trained to minimize

$$\mathscr{L} = -\frac{1}{T}\sum_t \sum_j \ln P(w_t|w_{t+j}),$$

where the probability is estimated as

$$P(w_t|w_{t+j}) = \frac{\exp(\mathbf{v}_t^T \mathbf{v}_{t+j})}{\sum_{\mathbf{v}\in\mathscr{V}} \exp(\mathbf{v}^T \mathbf{v}_{t+j})}$$

It was proven in class that $\nabla_{\mathbf{v}_t} \ln P(w_t|w_{t+j}) = \mathbf{v}_{t+j}(1 - P(w_t|w_{t+j}))$. In class, however, we did not discuss the fact that $P(w_t|w_{t+j})$ depends on **every** vector in $\mathscr{V}$, not just on $\mathbf{v}_t$ and $\mathbf{v}_{t+j}$. Find the value of $\nabla_{\mathbf{v}} \ln P(w_t|w_{t+j})$ for some vector $\mathbf{v}$ that is neither $\mathbf{v}_t$ nor $\mathbf{v}_{t+j}$.

Solution:

$$\ln P(w_t|w_{t+j}) = \mathbf{v}_t^T \mathbf{v}_{t+j} - \ln \sum_{\mathbf{v}\in\mathscr{V}} \exp(\mathbf{v}^T \mathbf{v}_{t+j})$$

The derivative of the first term with respect to any vector other than $\mathbf{v}_t$ or $\mathbf{v}_{t+j}$ is zero. The derivative of the second term, however, is not zero:

$$\nabla_{\mathbf{v}} \ln P(w_t|w_{t+j}) = -\frac{\exp(\mathbf{v}^T \mathbf{v}_{t+j})}{\sum_{\mathbf{v}'\in\mathscr{V}} \exp(\mathbf{v}'^T \mathbf{v}_{t+j})} \mathbf{v}_{t+j}$$

Question 43 ***(0 points)***

Suppose the the input to a transformer is the sequence of scalar values $v_t = \cos\left(\frac{t}{1000}\right)$, where $0 \leq t \leq 999$. You are trying to find the context, c_i, for a query $\mathbf{q}_i$ whose inner product with the keys is

$$\mathbf{q}_i^T \mathbf{k}_t = \begin{cases} 0 & t \in \{250, 251, 252\} \\ -\infty & \text{otherwise} \end{cases}$$

Find the numerical value of c_i.

Solution: The transformer computes

$$\begin{aligned} c_i &= \sum_{t=0}^{999} \frac{\exp(\mathbf{q}_i^T \mathbf{k}_t)}{\sum_{t'} \exp(\mathbf{q}_i^T \mathbf{k}_{t'})} v_t \\ &= \sum_{t=250}^{252} \frac{1}{3} v_t \\ &= \frac{1}{3}\cos\left(\frac{250}{1000}\right) + \frac{1}{3}\cos\left(\frac{251}{1000}\right) + \frac{1}{3}\cos\left(\frac{252}{1000}\right) \end{aligned}$$

Question 44 ***(0 points)***

Suppose the the input to a transformer is the sequence of scalar values $v_t = \left(\frac{t}{1000}\right)$, where $0 \leq t \leq 999$. You are trying to find the context, c_i, for a query, $\mathbf{q}_i$, whose inner product with the keys is

$$\mathbf{q}_i^T \mathbf{k}_t = \begin{cases} 0 & t \in \{500, 501, 502\} \\ -\infty & \text{otherwise} \end{cases}$$

Find the numerical value of c_i.

Solution:

$$\begin{aligned} c_i &= \sum_{t=0}^{999} \frac{\exp(\mathbf{q}_i^T \mathbf{k}_t)}{\sum_{t'} \exp(\mathbf{q}_i^T \mathbf{k}_{t'})} v_t \\ &= \sum_{t=250}^{252} \frac{1}{3} v_t \\ &= \frac{1}{3}\left(\frac{500}{1000} + \frac{501}{1000} + \frac{502}{1000}\right) \end{aligned}$$

Question 45 ***(0 points)***

The position encoding for a transformer is a set of cosine-sine pairs, of the form

$$\mathbf{x}_t = \begin{bmatrix} \cos(\omega t) \\ \sin(\omega t) \end{bmatrix},$$

where ω is some constant. Consider a very simple transformer whose inputs contain only the two-dimensional position encoding shown above, for some constant value of ω. Suppose, further, that the key and value vectors are equal to the input ($\mathbf{k}_t = \mathbf{v}_t = \mathbf{x}_t$), and that only the query vector is transformed:

$$\mathbf{q}_t = \mathbf{W}\mathbf{x}_t$$

Remember that the context vector is computed as

$$\mathbf{c}_t = \sum_s \alpha(t,s)\mathbf{v}_s, \quad \alpha(t,s) = \frac{\exp(\mathbf{q}_t^T \mathbf{k}_s)}{\sum_{s'} \exp(\mathbf{q}_t^T \mathbf{k}_{s'})}$$

Find a value of the matrix $\mathbf{W}$ that will result in an attention that always looks back three time steps, i.e., $\alpha(t,t-3) > \alpha(t,s)$ for any $s \neq t-3$. Hint: you may find it useful to know that

$$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$$
$$\sin(\alpha - \beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$$

Solution: The largest value of the dot product is achieved when $\mathbf{q}_t$ and $\mathbf{k}_t$ are the same vector, so we want to rotate $\mathbf{q}_t$ backward by three time steps. Using the hint,

$$\mathbf{k}_{t-3} = \begin{bmatrix} \cos(\omega(t-3)) \\ \sin(\omega(t-3)) \end{bmatrix} = \begin{bmatrix} \cos(3\omega) & \sin(3\omega) \\ sin(3\omega) & \cos(3\omega) \end{bmatrix} \begin{bmatrix} \cos(\omega t \\ \sin(\omega t) \end{bmatrix},$$

and $\mathbf{W}$ is the matrix in this equation.

8 Robotics

Question 46 ***(0 points)***

A robot crane is trying to build a building. Suppose that the state variable is $s = \{r_1(s), \ldots, r_n(s)\}$, where $r_i(s) = (x_i(s), y_i(s), z_i(s))$ are the lattitudinal, longitudinal, and vertical positions of the i^{th} brick during the s^{th} state of partial construction. The goal of the search is to find a way to put the n available bricks into n desired positions. Order doesn't matter: it doesn't matter which particular brick ends up in each of the n desired target positions. Suppose that the cost of moving the i^{th} brick into the j^{th} target position, $r_j(g) = (x_j(g), y_j(g), z_j(g))$, is

$$\|r_i(s) - r_j(g)\| = \sqrt{(x_i(s) - x_j(g))^2 + (y_i(s) - y_j(g))^2 + (z_i(s) - z_j(g))^2}$$

Prove that the following heuristic is admissible for this problem:

$$h(s) = \sum_{i=1}^{n} \min_{j=1}^{n} \|r_i(s) - r_j(g)\|$$

Solution: Each brick needs to move into one of the goal positions. Let $j(i)$ be the position the i^{th} brick needs to move to. The total cost of moving all bricks from node s to their goal positions is

$$d(s,g) = \sum_{i=1}^{n} \|r_i(s) - r_{j(i)}(g)\|.$$

For each brick,

$$\min_{j=1}^{n} \|r_i(s) - r_j(g)\| \leq \|r_i(s) - r_{j(i)}(g)\|,$$

therefore

$$h(s) \leq d(s,g)$$

Question 47 ***(0 points)***

Suppose a robot arm has a shoulder angle of θ radians, an upper arm of length L_1, an elbow angle of ϕ radians, and a lower arm of length L_2. Define a position b on the robot arm to be the point b meters from the shoulder, thus the forward kinematics are given by:

$$\phi_b\left(\begin{bmatrix}\theta\\ \phi\end{bmatrix}\right)=\begin{bmatrix}x\\ y\end{bmatrix}=\begin{cases}\begin{bmatrix}b\cos\theta\\ b\sin\theta\end{bmatrix} & 0\le b\le L_1\\ \begin{bmatrix}L_1\cos\theta+b\cos(\theta+\phi)\\ L_1\sin\theta+b\sin(\theta+\phi)\end{bmatrix} & L_1\le b\le L_1+L_2\end{cases}$$

In the workspace $\mathscr{W}$, there is an obstacle: a wall at $y=c$, which makes it impossible for any part of the robot to exist at any point $y\ge c$, where you may assume that $c>0$. This can be written as:

$$\mathscr{W}_{\text{obs}}=\{(x,y):y\ge c\}$$

Define $\mathscr{C}_{\text{obs}}$ to be the set of robot configurations $[\theta,\phi]$ that place any part of the robot arm within $\mathscr{W}_{\text{obs}}$. This can be written as

$$\mathscr{C}_{\text{obs}}=\{(\theta,\phi):P\},$$

where P is some inequality or disjunction of inequalities in terms of θ and ϕ. Find P.

Solution: There was an error in the problem statement. It should have been

$$\phi_b\left(\begin{bmatrix}\theta\\ \phi\end{bmatrix}\right)=\begin{bmatrix}x\\ y\end{bmatrix}=\begin{cases}\begin{bmatrix}b\cos\theta\\ b\sin\theta\end{bmatrix} & 0\le b\le L_1\\ \begin{bmatrix}L_1\cos\theta+(b-L_1)\cos(\theta+\phi)\\ L_1\sin\theta+(b-L_1)\sin(\theta+\phi)\end{bmatrix} & L_1\le b\le L_1+L_2\end{cases}$$

An answer is considered correct if it uses either the correct forward kinematics, or the forward kinematics specified in the problem statement.

The robot hits the obstacle if any part of its arm is in $\mathscr{W}_{\text{obs}}$. Literally, we can write this as:

$$P=(\exists b\in[0,L_1]:b\sin\theta\ge c)\vee(\exists b\in[L_1,L_1+L_2]:L_1\sin\theta+b\sin(\theta+\phi)\ge c)$$

or

$$P=(\exists b\in[0,L_1]:b\sin\theta\ge c)\vee(\exists b\in[L_1,L_1+L_2]:L_1\sin\theta+(b-L_1)\sin(\theta+\phi)\ge c)$$

It is also acceptable to note that we only hit the obstacle if either the elbow or the wrist are inside the obstacle. These two inequalities are a little bit simpler, they are just:

$$P=(L_1\sin\theta\ge c)\vee(L_1\sin\theta+L_2\sin(\theta+\phi)\ge c)$$

or

$$P=(L_1\sin\theta\ge c)\vee(L_1\sin\theta+(L_2-L_1)\sin(\theta+\phi)\ge c)$$

Question 48 ***(0 points)***

A robot fire truck works in a workspace with a horizontal dimension (x) and a vertical dimension (y). The robot fire truck's base must remain at $y = 0$, but it can move its base back and forth to position $x = p$. The ladder can be raised to elevation angle θ ($0 \leq \theta \leq \pi$), and the ladder can be extended to length l. A point b meters along the ladder ($0 \leq b \leq l$) is thus located by the forward-kinematic function as

$$\phi_b \left(\begin{bmatrix} p \\ \theta \\ l \end{bmatrix} \right) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} p + b\cos\theta \\ b\sin\theta \end{bmatrix}$$

In the workspace $\mathscr{W}$, there is an obstacle: a wall at $x = c$, which makes it impossible for any part of the robot to exist at any point $x \geq c$, where you may assume that $c > 0$. This can be written as:

$$\mathscr{W}_{\text{obs}} = \{(x,y) : x \geq c\}$$

Define $\mathscr{C}_{\text{obs}}$ to be the set of robot configurations $[p,\theta,l]$ that place any part of the robot's ladder within $\mathscr{W}_{\text{obs}}$. This can be written as

$$\mathscr{C}_{\text{obs}} = \{(p,\theta,l) : P\},$$

where P is some inequality or disjunction of inequalities in terms of p, θ and l. Find P.

Solution: The robot hits the obstacle if any part of its ladder is in $\mathscr{W}_{\text{obs}}$. Literally, we can write this as:

$$P = (\exists b \in [0,l] : p + b\cos\theta \geq c)$$

It is also acceptable to note that we only hit the obstacle if either the lower or upper end of the ladder are inside the obstacle. These two inequalities are a little bit simpler, they are just:

$$P = (p \geq c) \vee (p + l\cos\theta \geq c)$$

Question 49 ***(0 points)***

A robot fire truck is able to manipulate its own horizontal location (D), the angle of its ladder (θ), and the length of its ladder (L). The ladder has a length of L, and an angle (relative to the x axis) of θ ($0 \leq \theta \leq \frac{\pi}{2}$ radians), so that the position of the tip of the ladder is

$$(x, z) = (D + L\cos\theta, L\sin\theta)$$

(a) What is the dimension of the configuration space of this robot?

Solution: There are three dimensions: D, L, and θ.

(b) The robot must operate between two buildings, positioned at $x = 0$ and at $x = 10$ meters. No part of the robot (neither its base, nor the tip of the ladder) may ever come closer than 1 meter to either building. What portion of configuration space is permitted? Express your answer as a set of inequalities involving only the variables D, L, and θ; the variables x and z should not appear in your answer.

Solution:

$$1 \leq D + L\cos\theta \leq 9, \quad 1 \leq D \leq 9$$

(c) The robot's objective is to save a cat from a tree. The cat is at position $(x, z) = (5, 5)$. The robot begins at position $(D = 5, L = 3, \theta = 0)$. The final position of the robot depends on how much it costs to raise the ladder by one radian, as compared to the relative cost of extending the ladder by one meter, and the relative cost of moving the truck by one meter. Why?

Solution: Minimum-cost search for the solution will explore steps, in configuration space, that are of equal cost in each direction. The resulting shortest path will depend on the number of steps required to raise the ladder by $\pi/4$ radians, versus shifting the truck by 4m and extending the ladder. Equivalently: if raising the ladder is more expensive, then the truck will move 4m away, and raise the ladder only a little, but if raising the ladder is cheap, then the truck will raise the ladder up to vertical.

Question 50 ***(0 points)***

A TBR (two-body robot) is a robot with two bodies. Each of the two bodies can move independently; they're connected by a wi-fi link, but there is no physical link. The position of the first body is (x_1, y_1), the position of the second body is (x_2, y_2).

The robots have been instructed to pick up an iron bar. The bar is 10 meters long. Until the robots pick it up, the iron bar is resting on a pair of tripods, 10 meters apart, at the locations $(1,0)$ and $(11,0)$.

(a) Define a notation for the configuration space of a TBR. What is the dimension of the configuration space?

Solution: The configuration space is a 4-dimensional vector space, (x_1, y_1, x_2, y_2).

(b) In order to lift the iron bar, the robot must reach an OBJECTIVE where one of its bodies is at position $(1,0)$ and the other is at position $(11,0)$. In terms of your notation from part (a), specify the OBJECTIVE as a set of points in configuration space. You may specify the OBJECTIVE as a set of discrete points, or as a set of equalities and inequalities.

Solution:

$$\text{OBJECTIVE} = \{(1,0,11,0),(11,0,1,0)\}$$

(c) If the TBR touches the bar (with either of its bodies) at any location other than the endpoints ($(1,0)$ and $(11,0)$), then the bar falls off its tripods. This constitutes a FAILURE. Characterize FAILURE as a set of points in configuration space. You may specify FAILURE as a set of discrete points, or as a set of equalities and inequalities.

Solution: FAILURE is the set of all vectors (x_1, y_1, x_2, y_2) such that

$$1 < x_1 < 11 \text{ and } y_1 = 0$$

or

$$1 < x_2 < 11 \text{ and } y_2 = 0$$