UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Department of Electrical and Computer Engineering

CS 440/ECE 448 Artificial Intelligence
Spring 2020

PRACTICE EXAM 1

Monday, February 24, 2020

- This is will be a CLOSED BOOK exam. Book and notes are not allowed.
- No calculators are permitted. You need not simplify explicit numerical expressions.
- This exam is more than twice as long as the actual exam will be.
- You must SHOW YOUR WORK to get full credit.

Name: ____

netid: ____

Problem 1 (4 points)

Give one reason why a test of intelligence should evaluate action rather than thought. Give one reason why such a test should evaluate thought rather than action.

Solution:

- One reason to evaluate action: it is difficult to evaluate the thought processes of a human being, so a test of thought can't really evaluate the intelligence of human beings.
- One reason to evaluate thought: in a closed environment, a simple reflex (rule-based) software agent can easily be programmed to act with perfect rationality (i.e., in a way that optimally maximizes its own reward, even if it has no internal model of "reward" or any kind of self-awareness).

Problem 2 (4 points)

How can an agent think like a human without thinking rationally?

Solution: To think like a human means that your thoughts (perceptions and plans) are similar to those that a human being would have in the same circumstance. To think rationally is to use logic in order to infer the sequence of actions that maximizes your own benefit. Humans sometimes think rationally, and sometimes not, e.g., humans sometimes make decisions on the basis of intuition, or emotion, with no explicit consideration of future reward. An agent that feels emotions, or feels human-like impulses, could be said to think humanly, even though it is not thinking rationally.

Problem 3 (4 points)

How can an agent think rationally without acting rationally?

Solution: To think rationally is to logically infer, from known and observed facts, the sequence of actions that would maximize your own benefit. To act rationally is to act in a manner that maximizes your own benefit. An agent might be designed to compute the best actions without acting; for example, it might be designed to simply advise humans on the best course of action, without acting for itself.

Problem 4 (4 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 ($O(b^d)$).

Problem 5 (2 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.

Problem 6 (4 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.

Problem 7 (3 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.

Problem 8 (4 points)

Explain why it is a good heuristic to choose the variable that is *most* constrained but the value that is *least* constraining in a CSP search.

Solution:

- When we choose a variable, we are not eliminating any possible solutions, we are only deciding on the order in which variables will be considered. It makes sense to choose the order that minimizes complexity.
- When we choose an assignment, we are eliminating possible solutions; if we're wrong, we'll have to back-track. Therefore it makes sense to choose the assignment that eliminates as few solutions as possible, to minimize the chance of back-tracking.

Problem 9 (2 points)

Can an environment be both known and unobservable? Give an example.

Solution: An environment can be both known and unobservable. An example of this is Russian roulette. Russian roulette is a potentially lethal game of chance in which a player places a single round in a revolver, spins the cylinder, places the muzzle against his or her head, and pulls the trigger. The environment is known beforehand (i.e., each agent continues on until the other agent is terminated from the game) but the revolver is unobservable from agents (they cannot observe where the bullet is located).

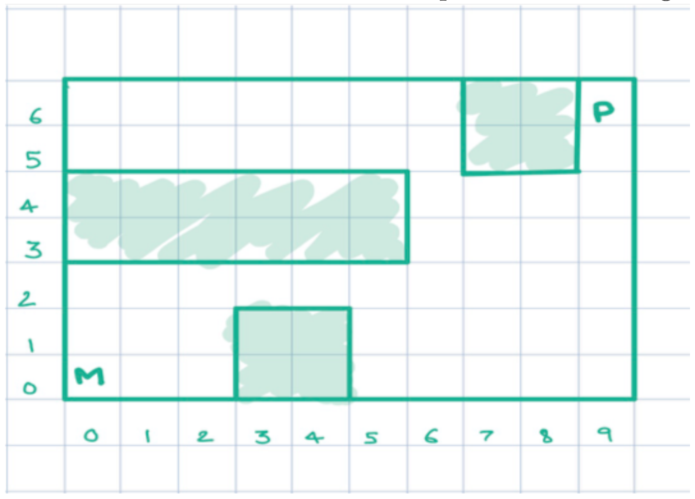
Problem 10 (2 points)

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.

Problem 11 (8 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?

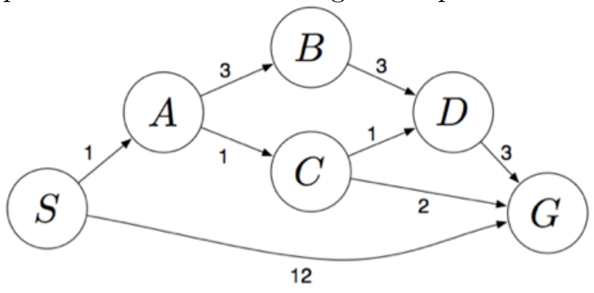
Solution: There are 50 possible locations for both Mario and Peach (if we assume that they can occupy the same square), for a total state space of 2500.

(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., $0.5 * (|x_M - x_P| + |y_M - y_P|)$.

Problem 12 (10 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?

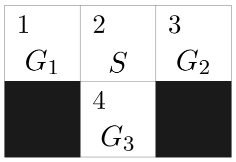
Solution: Neither admissible nor consistent.

(ii) Is h2 admissible? Is it consistent?

Solution: Admissible but not consistent.

Problem 13 (10 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 $d[m] - d[n] \geq h_1[m] - h_1[n]$.

The only way to move backward (away from maze completion) is by moving away from the nearest goal. So in that case, moving from m to n would cause $d[m] - d[n] = -1$, but it would also cause $h_1[m] - h_1[n] = -1$, so again we have $d[m] - d[n] \geq h_1[m] - h_1[n]$, so h_1 is consistent.

(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 $h_2[n] \geq h_1[n]$.

Problem 14 (10 points)

For each of the following problems, determine whether an algorithm to optimally solve the problem requires worst-case computation time that is polynomial or exponential in the parameters d and m (assuming that P $\neq$ NP).

(a) A map has d regions. Colors have been applied to all d regions, drawing from a set of m possible colors. Your algorithm needs to decide whether or not any two adjacent regions have the same color.

Solution: Polynomial in d and m.

(b) b. A map has d regions. Your algorithm needs to assign colors to all d regions, drawing colors from a set of m possible colors, in order to guarantee that no two adjacent regions have the same color.

Solution: Exponential in d.

(c) Your algorithm needs to find its way out of a maze drawn on a d-by-d grid.

Solution: Polynomial in d.

(d) Your algorithm needs to find the shortest path in a d-by-d maze while hitting m waypoints (equivalent to dots in MP1 part 1.2).

Solution: Exponential in m.

(e) Your algorithm needs to solve a planning problem in a blocks world consisting of d blocks.

Solution: Exponential in d

Problem 15 (6 points)

The four-queens problem is a constraint satisfaction problem in which one must place four queens on a 4×4 chess board so that none of the queens is on the same row, column, or diagonal with any other queen. One way to formulate the problem is by specifying four variables, A, B, C, and D (one per row), each of which must take a different column from the set $\{1, 2, 3, 4\}$ so that no two (row,column) coordinates are on the same diagonal:

	1	2	3	4
A				
B				
C				
D				

Remember that, in choosing an evaluation sequence for the depth-first search in a constraint satisfaction problem, three heuristics are often useful: LRV (least remaining values), MCV (most constraining variable), and LCV (least constraining value).

(a) According to the LRV, MCV, and LCV heuristics, does it make more sense to choose a value for row A first, or for row B? Why?

Solution: Row B is the MCV. Starting with the assignments $B = 2$ or $B = 3$ would eliminate three possibilities from each of rows A and C, two possibilities from row D, for a total of 8 constraints; conversely, starting with the assignments $A = 2$ or $A = 3$ would eliminate three possibilities from row B, two from row C, and only one from row D, for a total of only 6 constraints.

(b) Suppose we've decided to assign row B first. According to the LRV, MCV, and LCV heuristics, should we try the value $B = 1$ first, or the value $B = 2$? Why?

Solution:$B = 1$ rules out two possibilities from every other row (a total of 6 constraints), whereas $B = 2$ imposes a total of 8 constraints. Thus $B = 1$ is the LCV.

(c) Suppose we have assigned $A = 2$, $B = 4$. According to the LRV, MCV, and LCV heuristics, should we try to assign row C next, or row D? Why?

Solution: Row C, because it has only 1 remaining possible value (LRV), whereas row D still has 2 remaining values.

Problem 16 (6 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

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

Problem 17 (5 points)

A particular planning problem is defined by a set of three variables (x, y and z) and a set of two possible values (A and B). At any given state of the planning process, each of the three variables may be set to either value, or to the value null ($\varnothing$). There are only two possible actions, called SETX and MOVE:

Action	Preconditions	Results
SETX(val)	$x = \varnothing$	$x =$ val
for val $\in \{A, B\}$		
MOVE(val,var1,var2)	var1 = val, var2 = $\varnothing$	var1 = $\varnothing$, var2 = val
for val $\in \{A, B\}$, var1,var2 $\in \{x, y, z\}$, var1 $\neq$ var2		

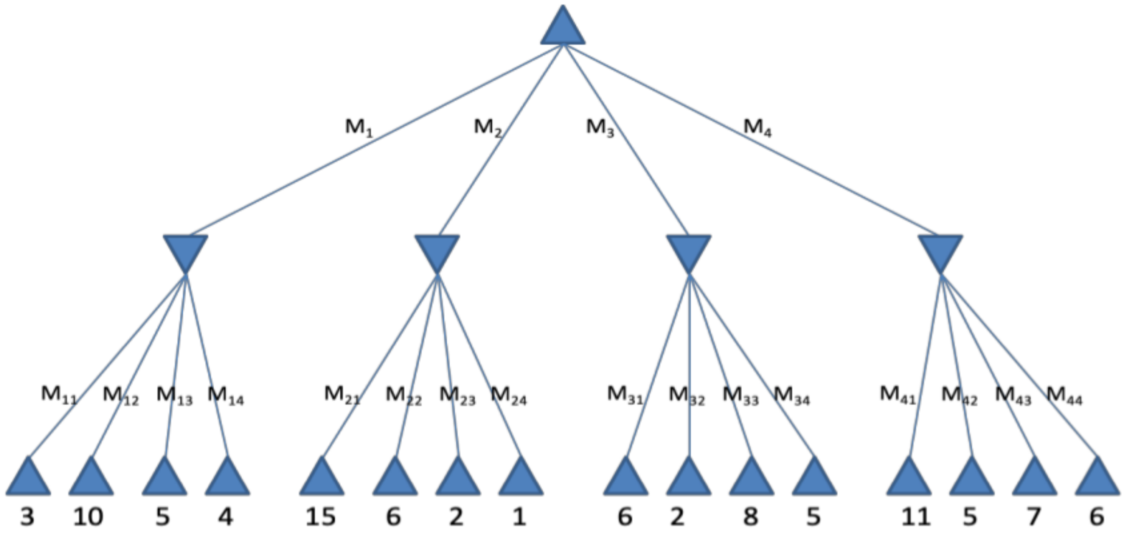
In the starting condition, all variables are set to null. In the desired ending condition, $y = A$ and $z = B$. Draw a breadth-first 2-level backward-chaining tree, showing all of the possible n-step predecessors of the desired ending condition for $n \leq 2$.

Solution:

$$\begin{tikzcd}
 & & (y=A, z=B) & & & \\
 & (x=A, z=B) \arrow[ur] & & & (x=B, y=A) \arrow[ull] & \\
(z=B) \arrow[ur] & (x=A, y=B) \arrow[u] & (y=A, z=B) \arrow[ul] & (y=A) \arrow[ur] & (x=B, z=A) \arrow[u] & (y=A, z=B) \arrow[ul]
\end{tikzcd}$$

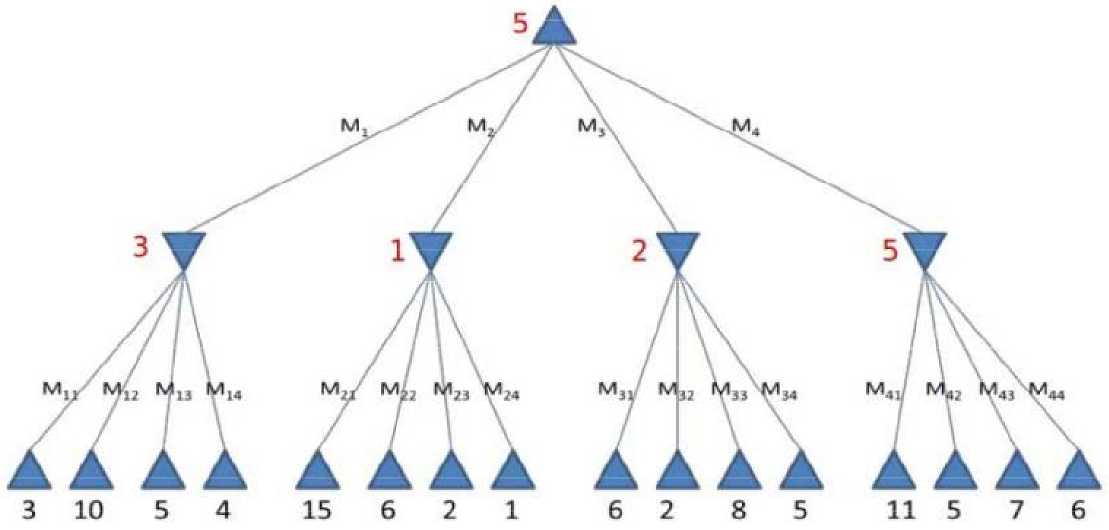
Problem 18 (12 points)

Consider the following game tree (MAX moves first):



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

Solution:

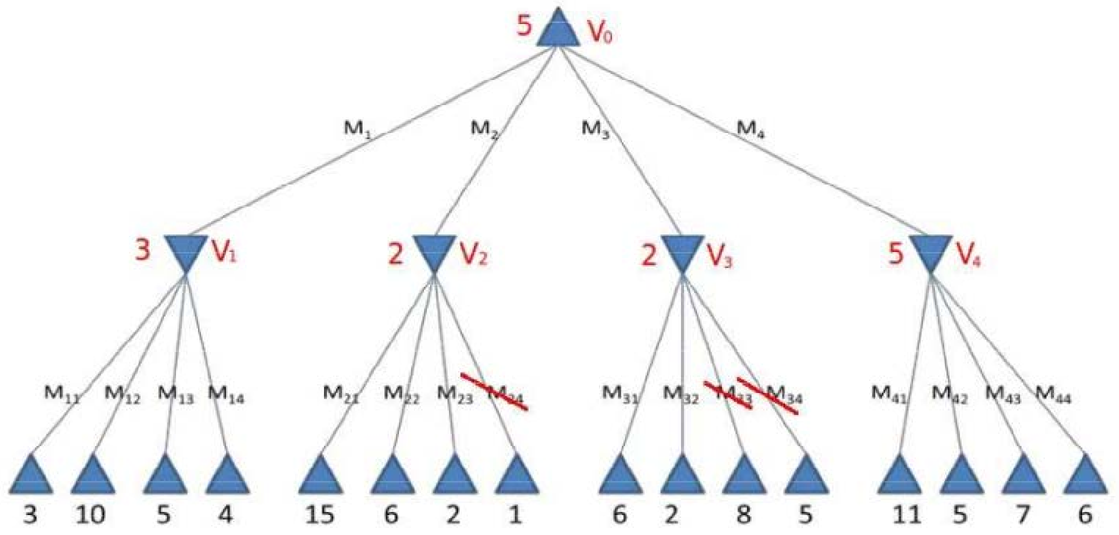


(b) How will the game proceed, assuming both players play optimally?

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.