TUFTS UNIVERSITY
Department of Computer Science

CS 131 | Artificial Intelligence | Summer 2022

Midterm Exam 01 (Practice Exam) | 08–10 July 2022

- **Please sign your name below (or write it to the top of each page of your work, if you are not solving on the exam paper). By doing so, you agree to be bound by Tufts policies on academic integrity.**
- **For this exam, you may use any notes you have taken, the Russell & Norvig text, and any materials distributed by the instructor (including lecture notes/videos and code samples). No other materials are to be used.**
- **This booklet contains 6 pages including the cover page.**
- **You have exactly 105 minutes (one hour, 45 minutes) to complete this exam *and* upload a PDF version to Gradescope. Be sure to leave yourself enough time for the latter steps.**
- **The maximum possible is 50.**

PROBLEM	SCORE
1	
2	
3	
4	
5	
6	
TOTAL	

NAME: ____________________

1. **(12 pts.)** **TRUE/FALSE & SHORT ANSWER.**

For each of the following, indicate whether the statement is true or false, and explain your answer in the space provided. Answers will not need to be longer than a few sentences. **Note**: statements about search refer to search trees with a finite branching factor and a uniform cost of operators. Don't worry about having sufficient memory to be able to run the algorithm.

(a) (3 pts.) (T ___ F ___) Let n be an arbitrary node in an *infinite* tree. If there is no solution to be found, breadth-first search will eventually reach n.

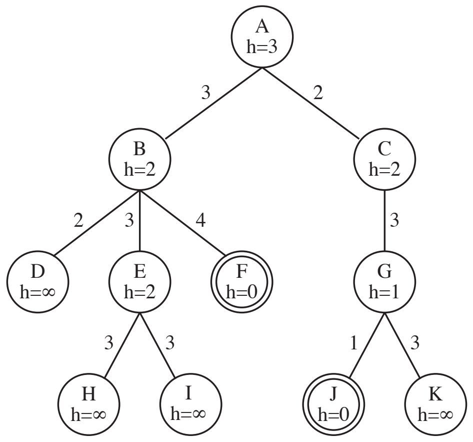
(b) (3 pts.) (T ___ F ___) If heuristic $h = 0$ for all nodes, then $A^\star$ will do the same thing as breadth-first search does, no matter what step-cost function g is used.

(c) (3 pts.) (T ___ F ___) Let $h_1(n)$ and $h_2(n)$ be two admissible heuristics for some search problem. Then $h(n) = \max(h_1(n), h_2(n))$ (i.e. the maximum of each individual heuristic) is also admissible, and will produce $A^\star$ search performance that is just as good or better than either of the originals.

(d) (3 pts.) (T ___ F ___) Suppose we have two variables, V_1 and V_2, in a constraint satisfaction problem. Furthermore, suppose that there is no constraint that features those two variables—that is, there is no constraint that restricts possible assignments to one of those two variables based upon the values assigned to the other. In this scenario, if an arc consistency algorithm like AC-3 changes the domain of V_1, this will not lead to any changes to the domain of V_2.

2. **(8 pts.)** **SEARCH ALGORITHMS.**

Consider the following search space. The initial state is A. Nodes F and J are goal states (designated by a double circle). Each node has an admissible heuristic value h associated with it (shown inside the circle). Nodes from which a goal can not be reached have an h-value of ∞. The actual edge costs for going from each node to its successors are shown.



For each one of the following search algorithms, list the order in which it will expand the nodes of this tree and the solution that it will return. Recall that *expanding* a node means generating all its successors. Assume that successors are visited in a left-to-right order when you need to break ties. Also assume that a solution is returned when the node is selected for expansion, *not* when it is generated.

(a) (3 pts.) Breadth-first search:

(b) (5 pts.) $A^\star$ search:

3. **(10 pts.)** **SEARCH ALGORITHMS II.**

We are given a state space S, a single goal state $z \in S$, and an admissible heuristic function $h(s)$. We need an algorithm that takes a set of initial states $I = \{s_1, \ldots, s_n\}$ and returns the initial state s_i that is the closest to the goal, as well as the path between s_i and z. Consider the following algorithms:

SEQ: This algorithm activates $A^\star$ sequentially from each initial state (a total of n times). It finds the shortest path from s_i to z for each i and at the end returns the path with minimal cost among the n paths.

PAR: This algorithm works like $A^\star$, but it maintains n separate open lists and n closed lists. In each iteration, the algorithm expands a node with minimal f value among all n open lists and processes the node as standard $A^\star$ (placing the node on the corresponding closed list and placing its successors on the open list from which it was removed). The algorithm stops when the next node selected for expansion is the goal.

(a) (4 pts.) Is each algorithm optimal? Explain why or why not.

(b) (6 pts.) Suppose some initial states are *much further away* from the goal z than others. Which of the two algorithms can we expect to often use the most *time*? Which can we expect to often use the most *space*? Explain your answers.

4. **(10 pts.)** **CONSTRAINT SATISFACTION.**

A magic square is an arrangement of the numbers from 1 to n^2 in an n-by-n matrix, with each number occurring exactly once, and such that the sum of the entries of any row, any column, or any main diagonal is the same. It is not hard to show that this sum must be $n(n^2+1)/2$. The simplest magic square is the 1×1 magic square whose only entry is the number 1. The next simplest is the 3×3 magic square:

$$\begin{array}{ccc} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{array}$$

and those derived from it by symmetries of the square.

Explain clearly how to define this problem as a CSP. For a given value n, clearly define what the variables would be, and explain what the constraints are. (You do not need to explicitly list all of the constraints, but you must describe them completely.)

5. **(10 pts.)** **KNOWLEDGE REPRESENTATION & REASONING.**

Consider the sentence, "Heads I win; tails you lose." We might represent this sentence and associated domain knowledge in propositional logic as follows:

1. $Heads \Rightarrow WinMe$
2. $Tails \Rightarrow LoseYou$
3. $LoseYou \Rightarrow WinMe$
4. $\neg Heads \Rightarrow Tails$

(a) (2 pts.) Convert these sentences into conjunctive normal form.

(b) (1 pts.) Is the set of converted clauses a set of Horn clauses? Why or why not?

(c) (7 pts.) Construct a resolution proof that shows I win (i.e., prove $WinMe$).