CS 662 Midterm answers
Total: 150 pts

Instructions: You will have 105 minutes to complete this midterm. The test is closed book; no notes, computers, or calculators are allowed. Please show all relevant work and calculations in order to receive partial credit (when appropriate). Many of the questions on this test are short answer, and are designed to get you to think about or explain the topics we have covered. You should answer these at an appropriate length and explain your reasoning when asked to do so. A one-word answer is usually too short; a one-page answer is usually too long. You may use the backs of the pages if you need extra space.

1 True/False, plus corrections
35 pts, 5 pts each
Each of the following statements is either true or false. If it is true, mark it true. If it is false, correct the statement so that it is true. Note: Adding "not" or otherwise negating the sentence is not acceptable. You must change the facts in the sentence if it is false. For example:
Question: The Turing Test is a test of whether a computer program is rational.
Bad answer, no credit: The Turing Test is not a test of whether a computer program is rational.
Good answer: The Turing Test is a test of whether a computer program is indistinguishable from a human.

• A stochastic environment is one in which an action may have more than one result.
  True.

• The Schema Theorem says that more fit bitstrings will be selected with exponentially increasing likelihood.
  False. The Schema Theorem says that more fit, low order schema will be selected with exponentially increasing likelihood.

• A rational agent is an agent that does what a human would do in any situation.
  False. A rational agent is an agent that selects actions that maximize its performance measure (or utility).

• A toy problem is a problem used to illustrate or study the performance of an algorithm.
  True

• A complete algorithm is one that always finds all possible solutions to a problem.
  False. A complete algorithm is one that will always find a solution to a problem if one exists.
• Mutation in genetic algorithms and temperature in simulated annealing both improve performance by eliminating suboptimal solutions.

False. Mutation in GAs and temperature in simulated annealing both improve solution quality by helping to escape from local optima.

• A Goal-based agent selects actions that are likely to maximize its utility.

False. A goal-based agent selects actions that are most likely to achieve its goal.

2 Agents and environments
a. 24 pts, 6 pts each
For each of the following agents and environments, characterize the environment according to the six properties used in R & N (static vs dynamic, discrete vs. continuous, fully vs. partially observable, deterministic vs stochastic, episodic vs. sequential, single-agent vs. multi-agent). If necessary, you may want to give a brief justification of your answers.

• The game of checkers.
  Static, discrete, fully observable, deterministic, sequential, multi-agent.

• An autopilot that successfully lands a plane.
  Dynamic, continuous, partially observable, stochastic, sequential, single-agent

• An handwriting recognition agent that reads the addresses on envelopes and places each of them in the appropriate mailbox.
  Static, discrete, fully observable, deterministic, episodic, single-agent

• A music-playing accompanist agent that can play along with a human piano player.
  Dynamic, continuous, partially observable, deterministic, sequential, multi-agent.

b. 6 pts
Consider the following problem: We want to build an agent that can play word games.

For our first task, we would like to build an agent that can recognize palindromes. (A palindrome is a word or phrase that reads the same backwards as forwards excepting punctuation, such as “Anna” or “Madam, I’m Adam”, or “A man, a plan, a canal - Panama!”)

Assume that our agent receives its input one letter at a time. That is, at time t=0 it sees ’a’, and at time t=1 it sees ’n’, and so on. Once the string is seen completely, the agent must output ’yes’ or ’no’.

Can a reflex agent solve this problem? If so, provide pseudocode for how such an agent might work. (You may use the back of the page if needed.) If not, explain precisely why it cannot solve this problem.

No, a reflex agent cannot solve this problem. The agent must be able to remember a sequence of characters and say whether a condition (namely that the sequence is reversible)
is true, but reflex agents do not keep any memory; they select actions based only on their current percepts.

For our second task, we want to build an agent that can find anagrams. That is, rearrangements of a phrase that are also a phrase. For example, “scab” can be rearranged to “cabs”, and “artificial intelligence” can be rearranged to make (among other things): “Intergalactic lie? Fie! Nif!” (punctuation added).

Let’s assume that we want to build a goal-based agent that will use search to find an anagram for an arbitrary input phrase. Your task in this problem will be to formulate the problem precisely enough so that it can be implemented. In grading this, I’ll ask two questions. 1) Is the answer correct? 2) Could I write a program from this answer without any additional information?

a) **4 pts** Give an initial state for this search.

Our state will be an ordering of the letters in the input string. The initial state is simply the ordering provided as input.

b) **4 pts** Give a description of how a goal test could be implemented.

We’ve found a goal when the string in a state can be separated into substrings that are English words, which can be verified by checking an English dictionary. One way to do this is by a separate backtracking search; scan the string from left to right; when a valid word is found, insert a space and continue from this point. If a dead end is reached, back up and undo the last assigned space.

c) **4 pts** Describe what the successor function does. In other words, for a given input, what is produced?

We want our successor function to generate new permutations of the current state’s string. Two incorrect approaches would be to generate all permutations at once (this will be a huge number for long strings) or to randomly shuffle the string (randomness will just slow things down here). One correct approach is to exchange the first letter in the string with every other letter, generating \( n - 1 \) new strings. We can also keep a closed list to eliminate duplicate strings.

3 Search

a) **(4 pts each)** Give the time and space requirements in big-O terms for each of the following algorithms. Assume that the search tree has a branching factor of \( b \), the solution is at depth \( d \), and the tree has a maximum depth of \( n \).

- **Breadth-first search** **Time:** \( O(b^{d+1}) \), **Space:** \( O(b^{d+1}) \)

- **Depth-first search** **Time:** \( O(b^n) \), **Space:** \( O(bn) \)

- **Depth-limited search** (depth = \( d \)) **Time:** \( O(b^d) \), **Space:** \( O(bd) \)

- **Iterative deepening search** **Time:** \( O(b^d) \), **Space:** \( O(bd) \)
b) 18 points - 6 points for each search

Trace BFS, DFS, and A* on the following graph. Uninformed search methods should expand nodes from left-to-right. (B should be visited before C.) Both g1 and g2 are goal states; your algorithm may stop as soon as either is expanded. Show the order in which nodes are visited. For A*, also show the f, g, and h costs for each node visited. For A*, the costs for each edge are indicated next to the edge, and the heuristic costs for each state are in parentheses next to the state.

BFS: A, B, C, D, E, F, H, G2
DFS: A, B, D, H, G1
A*: A [B (g = 3, h = 6, f = 9), C (g = 5, h = 7, f = 12)]
B [D (g = 5, h = 3, f = 8), E (g = 7, h = 4, f = 11), C (g = 5, h = 7, f = 12)]
D [*H (g = 10, h = 1, f = 11), *E (g = 7, h = 4, f = 11), C (g = 5, h = 7, f = 12)]
H [E (g = 7, h = 4, f = 11), G1 (g = 12, h = 0, f = 12), C (g = 5, h = 7, f = 12)]
E [H (g = 9, h = 1, f = 10), G1 (g = 12, h = 0, f = 12), C (g = 5, h = 7, f = 12)]
H [G1 (g = 11, h = 0, f = 11), C (g = 5, h = 7, f = 12)]
G1. Goal. Optimal Path: A, B, E, H, G1
(*) Note: E and H can be enqueued in either order since they have the same f-value. Both answers are correct.
4 Heuristics

a. 4 pts
Consider the Tower of Hanoi problem. There are three towers, with N disks of increasing size stacked on
the leftmost tower. The problem is to move the disks from the left-hand to the right-hand tower, with the
constraint that a larger disk can never be placed on top of a smaller disk.

To begin, let $h$ be the number of disks that still need to be moved. Is this heuristic admissible? If so,
prove it. If not, provide a counterexample.

This heuristic is admissible. Since we can move one disk at a time, it will take at least
as many moves as there are misplaced disks to solve the problem (more, due to the problem
constraints). Therefore, the heuristic will never overestimate the number of moves to the goal,
and is admissible.

b. 6 pts
Next, consider the following heuristic:

Number the disks according to size, where 1 is the smallest disk and $n$ is the number of the largest disk.
$h$ is the sum of the numbers of each misplaced disk.

Which heuristic should be used? Justify your answer with a precise technical argument, using the
appropriate terminology.

The second heuristic provides a better estimate of the number of moves needed. To place
the disk numbered $n$ will require $2^n - 1$ moves. Since this number is larger than the number
of misplaced disks, we say that this heuristic dominates the first one.

A few people noted that the second heuristic is inadmissible, due to its overestimate in the
case where disk 2 is on the first post, disk 1 is on the second post, and all other disks are in
place. This requires two moves to solve, and the heuristic gives an estimate of 3. Full credit
was given for noting this and not using h2 because of it. We could correct this heuristic to be
admissible by numbering the disks from 0 to $n - 1$.

5 pts

Recall that $A^*$ search is optimal when an admissible heuristic is used. An admissible
heuristic is one that always underestimates the cost to the goal. Uniform cost search expands
nodes only according to their cost so far, so it uses $h = 0$. By definition, this will always be
less than or equal to the cost to the goal (we assume step costs are positive), so therefore, the
heuristic is admissible, and uniform cost search is optimal.
a) **10 pts**

Consider the following fitness function:

\[
\text{fitness} = 5a + 3b + c + -d + 2e
\]

where a-e are indices of the bitstring. Compute the fitness of each of the members of the initial population below, as well as the probability that each string will be selected using roulette selection.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>fitness</th>
<th>selection probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6/27</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>5/27</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5/27</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6/27</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5/27</td>
</tr>
</tbody>
</table>

b) **4 pts**

Assuming the first two members of the population are selected for reproduction, and the cross-over point is that between the b and the c, show the resulting children:

11101 and 01011
6 Adversarial Search

10 pts
For the following tree, give the backed-up minimax value for each node, assuming that the Max player chooses first at node A, and can choose B or C. What is the optimal sequence of plays?

Node values:
A: 3  B: 3  C: 1  D: 8  E: 7  G: 1  H: 5
Optimal sequence of plays: A, B, D, J