United International University
Department of Computer Science and Engineering
CSI 341: Artificial Intelligence, Mid-Term Exam, Spring 2018
Total Marks: 30, Time: 1:45 Hours

Answer all the questions. Marks are indicated in the right side of each questions.

1. You are tasked with designing a robot navigation system where the robot can move up, down, left, or right, if there are no walls blocking its way. The goal is to reach the bottom right corner of the room. The robot can only see its immediate next squares, but not anything else. Blocks can appear randomly anywhere at the room, at any time. There is also an opposing robot whose objective is to damage your robot. Determine whether this agent's environment is single- or multi-agent, fully or partially observable, episodic or sequential, static or dynamic, and deterministic or stochastic. [2.5]

2. Consider the **16-queens** problem. In a game of chess a queen can attack another one if they are placed on the same row, column, or diagonal. You have to place 16 queens on an empty 16x16 chessboard so that no queen can attack each other.

   Suppose at the initial state, the queens are already placed on the chessboard so that each column has exactly one queen. As action, a queen can be moved anywhere in its same column. Answer the following: [3]

   (a) How many successors does a state have? Explain your calculation.

   (b) What is the size of the state-space? Explain your calculation.

   (c) Why should you use **local search** algorithms to solve this problem, instead of **uninformed** and **informed** searches?

3. (a) Consider the state-space graph for a problem with branching factor b, depth of the shallowest (nearest) goal node being d, and maximum path length being m. On this state-space – [2 + 2]

      i. What will be the height of the **Depth-First search-tree** in the worst-case? What about the **Breadth-First search-tree**?

      ii. The **Depth-First search** will require $\mathcal{O}(bm)$ memory in the worst-case. Why so? Explain briefly.

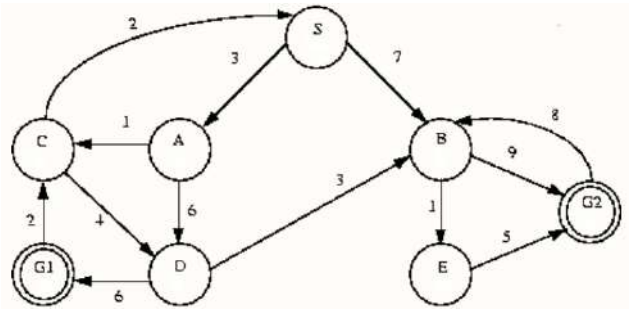

Figure 1: State-space graph for questions Q. 3b and 4b

   (b) Consider the state-space graph at Fig. 1. Find out the **solution paths** (if any) and **costs** returned by the following **tree-search** algorithms. In case of ties while picking nodes from the fringe, follow the alphabetical order. S is the initial state, while G_1 and G_2 are goal states. [4.5]

      i. Depth-First search;

ii. Iterative-deepening search;

iii. A* search with the following heuristic function h.

$$h(x) = \begin{cases} 0, & \text{if } x \text{ is a goal state.} \\ \text{minimum cost among the outgoing actions from } x, & x \text{ is not a goal state.} \end{cases}$$

For example, $h(S) = 3$, $h(G_1) = 0$, and $h(C) = 2$.

Which of these searches have provided **non-optimal solutions**?

(c) Suppose that you have to design a vacuum-cleaner robot, which will operate in an apartment with 3 rooms in a row. It has the following allowed actions with the associated costs: move left ($L : 5$), move right ($R : 20$), and clean ($C : 25$). Initially, only the middle room is clean, the robot is there, and the other rooms are dirty. The goal of the agent is to make all the rooms dirt-free. Representing the states of the problem with required variables, determine the **optimal plan** (solution path) for the robot using the **Uniform-cost search** algorithm. Use **graph-search** (avoid generating repeated nodes). [3]

4. (a) Consider the following board configurations for the **8 puzzle** problem. Using the **A* search** with the **misplaced tiles** heuristic, determine an optimal solution path. [3]

3	1	2
4	7	5
6		8

Initial state

	1	2
3	4	5
6	7	8

Goal state

(b) Consider the state-space graph of a problem at Fig. 1. Provide heuristic values for each state for three heuristic functions h_A, h_B, and h_C, such that in an **A* search**, using - i) h_A produces the **maximum** number of nodes; ii) h_B produces the **minimum** number of nodes; and iii) h_C may return **non-optimal solution(s)**. Using h_A and h_B should provide optimal solutions. [3]

5. (a) Consider the problem at Q. 4a. Can we use any **local search** algorithm to solve it? If yes, how? If no, why not? [2]

(b) What is the **Random-restart hill climbing search**? Explain briefly how this strategy can help to escape being stuck at local maxima, for maximization problems. [2]

6. Consider the game-tree at Fig. 2. Using the **minimax algorithm** with **alpha-beta pruning**, determine the utility of the root node. You must clearly show which branches have been pruned away by the algorithm. [3]

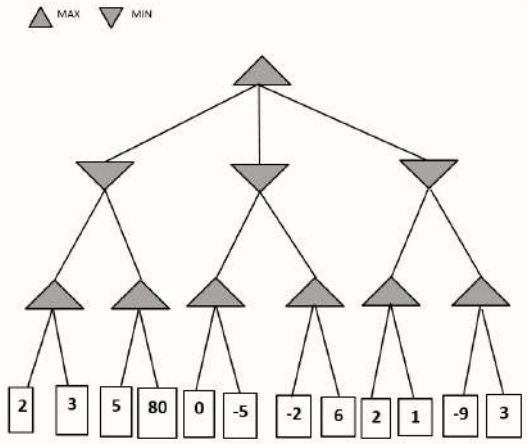

Figure 2: Game-tree for Q. 6