CSE 486/586, Assignment 1

Due: Tuesday, September 14, by 11:59 pm.

Note 1: The total mark for this assignment is 30. You should *NOT* directly copy anything from slides or other resources. You may get the ideas from slides but what you submit *must be in your own words*. Any help must be acknowledged.

Note 2: Question 1 is optional but you are strongly encouraged to work on this kind of problems (extra credit!). Question 7 is only for CSE 586 students (0 point for correct answer, and -5 for wrong answer)

1. Online Code Repository. Codes for all the algorithms in the book (in Java, Lisp, and Python) have been provided here:

   http://aima.cs.berkeley.edu/code.html

   All groups are strongly encouraged to implement several of the algorithms/problems discussed in class and compare their codes with the codes provided in the above link. Groups who implement some of these algorithms/problems and show me their codes (in person) by November 10, 2021 will get 1 extra credit per code (up to 5 extra credits).

2. Pacman and Ms. Pacman are lost in an *N*x*N* maze and would like to meet; *they don't care where*. In each time step, *both* simultaneously move in one of the following directions: {NORTH, SOUTH, EAST, WEST, STOP}. They do *not* alternate turns. We must devise a plan which positions them together, somewhere, in as few time steps as possible. Passing each other does not count as meeting; they must occupy the same square at the same time. We can formally state this problem as a *single-agent* state-space search problem as follows:

   States: The set of pairs of positions for Pacman and Ms. Pacman, that is, $\{((x_1, y_1), (x_2, y_2)) \mid x_1, x_2, y_1, y_2 \in \{1, 2, \ldots, N\}\}$
   Size of state space: N^2 for both Pacmen, hence N^4 total
   Goal test: $\text{Goal}((x_1, y_1), (x_2, y_2)) := (x_1 = x_2)$ and $(y_1 = y_2)$

   a) Determine the branching factor. (2 points) 25 (Pacman has 5 choices, Ms. Pacman has 5 choices)
   b) For each of the following graph search methods determine and explain whether it is guaranteed to output optimal solutions to *this problem*? (6 points)
      (i) DFS No, this will return the first solution found regardless of if the solution is optimal
      (ii) BFS Yes as the arcs/operators have the same constant cost as such BFS will be optimal
      (iii) UCS No, as the cost we can assume will be the same at every step as such since all costs are the same UCS will not be guaranteed to output an optimal solution

3. Consider a state space where the start state is number 1 and each state *k* has two successors: 2*k* and 2*k* + 1. Suppose you want to navigate your robot from state 1 to state 2020. Can you find an

algorithm that outputs the solution to this problem without any search at all? Output the solution for 2020. (6 points) You can write the goal number in binary and discarding the initial number (left most) you can use 0 to go left and 1 to right. So we can take 11111100100 and remove the first 1 to get 1111100100 which will give us the path to the 2020 state from state 1. Right, right, right, right, right, left, left, right, left, left.

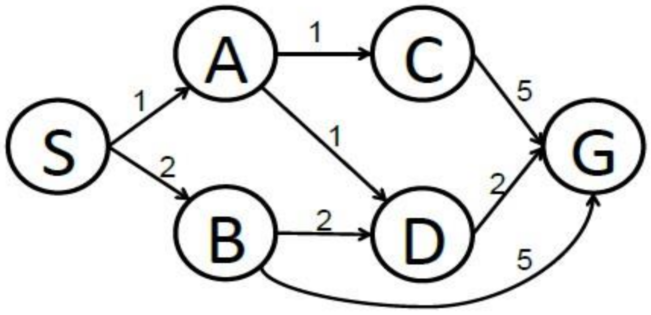
4. Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall. We can formulate this problem as follows. We'll define the coordinate system so that the center of the maze is at (0, 0), and the maze itself is a square from (−1, −1) to (1, 1).

   Initial state: Robot at coordinate (0, 0), facing North.
   Successor function: Move forwards any distance *d*; change direction robot is facing.
   Cost function: Total distance moved.

   a) Let (*x, y*) be the current location. What is the Goal test? (2 points) When the x coordinate is not between the range (-1,1) or the y coordinate is not between the range (-1,1).
   b) How large is the state space? (2 points) N^2, because the maze is itself a square and the robot will move a distance d, but that distance can be variable, and the robot can stop and change direction.

5. For each of the following assertions, say whether it is true or false. Justify your answers. (6 points:
   2 points each)
   a) Imagine the next Mars rover malfunctions upon arrival on Mars. From this we can deduce that Mars rover is not a rational agent. (Note that a rational agent is not necessarily perfect, it's only expected to maximize goal achievement, given the available information.) True, because if it was a rational agent it should have been able to take the information provided when it landed on mars and be able to fulfill the task that it had been assigned based on the information it had gained from its surroundings
   b) Every optimal search strategy is necessarily complete. True, because for a search strategy to be optimal it must have the ability to find a solution which is what being complete is.
   c) Breadth-first search is optimal if the step cost is positive. False, because BFS can be optimal if the step cost is positive but only when all step costs are the same

6. For the search problem shown below, S is the start state, and G is the (only) goal state. Break any ties alphabetically. What paths would the following search algorithms return for this search problem? Give your answers in the form 'S - A - D - G'. (6 points: 2 points each)



a) BFS: S- B- G
b) UCS: S- A- D-G
c) DFS: S- A- C- G

7. Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, O(n^2) vs. O(n)). (Only for CSE 586 students) A tree that has a depth of n and on the left the tree is only nodes with a single child node where the last node is goal and, on the right, each node has m children. With this depth first will keep going left and eventually reach our goal because that is the first depth it will check, but for iterative deepening search it will have to constantly keep going down and checking the right side of the tree even though we know the goal is not on the right side of the tree.