4511W, Spring-2020
ASSIGNMENT 2 :
Assigned: 02/11/20 Due: 02/17/20 at 11:55 PM (submit via Canvas, you may take a picture of handwritten solutions, but you must put them in a pdf) Submit only pdf or txt files

Written/drawn:

Problem 1. (15 points)
For each of the following scenarios, describe what the possible (1) states and (2) actions are, along with the (3) cost of each action. For the description you have given (4) determine whether it is an incremental or complete-state representation. Then (5) decide whether a tree or graph would be more appropriate.

Situation 1: You are stuck in a hedge maze and want to find the way out. Unfortunately you have a bad visual memory and sense of direction, so you have trouble telling whether you have been to a location before or not.

Situation 2: Suppose we were to implement assigned seating in csci4511. However, we wish to allow preferences, for example friends sitting next to each other.

Situation 3: You are planning the food for an event. You need to buy enough from a catering service to feed 50 people, and you want to spend as little money as possible without ordering the same item more than once.

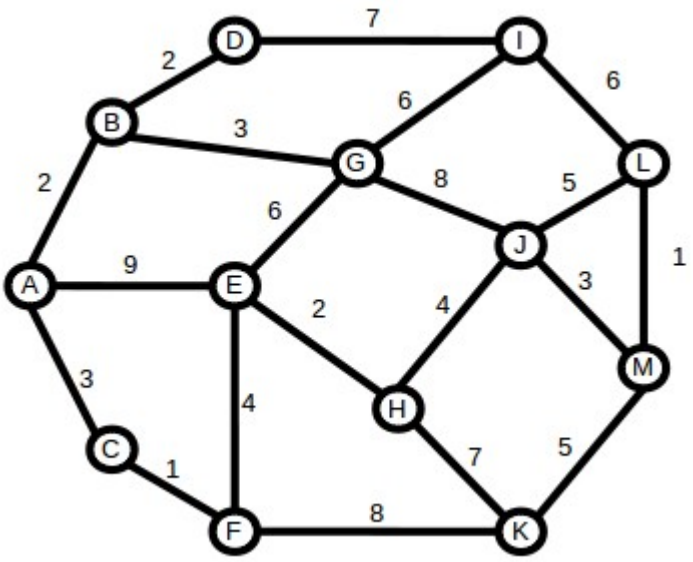
Problem 2. (20 points)
For each algorithm indicate whether it is appropriate for trees, graphs or both. If the algorithm is not appropriate, justify why.

-Breadth first search
-Depth first search
-Iterative-deepening depth-first-search
-Uniform cost search

Problem 3. (20 points)
We discussed how you can sometimes have multiple goal states for a problem. If instead you have multiple initial states, which of the following algorithms could adapt and still work well (i.e. same runtime and memory in big-O notation)? Justify your answer.

-Breadth first search
-Depth first search
-Uniform cost search
-Greedy best-first search

Problem 4. (15 points)
Run bi-directional search using uniform cost search as the “sub-search” on the graph below with the initial state “A” and goal state “M”. Like the bi-direction BFS search we discussed in-class, alternate each time choosing the most appropriate node from the initial and goal fringe set. Use the stopping condition: when there is an overlap in the nodes explored sets of the initial and goal. Clearly show the fringe set at every step.



Problem 5. (20 points)
Find an example graph where the stopping condition for bi-directional search using uniform cost search (i.e. the problem above) does not yield the optimal result. How can you modify the process to ensure the shortest path is found (while still searching from both the goal and initial state)? What impacts will this modification have on the runtime and memory?

Programming (python/lisp):

Problem 6. (10 points)
Test bi-directional, breadth first and A* search a number of different 8-puzzles (enough to answer the question) and generalize which would work best (in terms of runtime). Like last time, I suggest that you use /tests/test_search.py as a reference, despite being unable to run this. (The bi-directional search is a bit less polished than the other code... you should be able to run it, though getting things like the solution are difficult, though not necessary for this problem.)

Also answer: what sub-search is being done from both initial and goal state in the implemented bi-directional search?

(Note: some arrangement of numbers in an 8-puzzle are impossible to solve.)