4511W, Spring-2018
ASSIGNMENT 1 :
Assigned: 01/30/18 Due: ~~02/04/18~~ 02/06/18 at 11:55 PM (submit via moodle, you may scan or take a picture of your paper answers) Submit only pdf or txt files (in a zip if you have multiple files)

Problem 1. (15 points)
For each of the scenarios below, classify the environment based on the seven classifications discussed in class (i.e. fully/partially observable, single/multi-agents, etc.). Additionally for each of the seven classifications, provide a single sentence supporting your reasoning.

(1) Tic-Tac-Toe.

Solution:
Fully observable - Both players can see the whole board.
Multi-agent - Normally Tic-Tac-Toe is not played by yourself.
Deterministic - If you play in the top left, it will always appear there (and not in the center 50% of the time).
Sequential - Your next move depends on both your previous moves and the opponent's past moves. (Note: This is episodic between games... a previous game does not effect the next game, but I assume most people are talking about within a single game.)
Static - Taking turns means you can think as long as you want (assuming it is not timed like competitive Go or Chess).
Known - I assume most people know rules of Tic-Tac-Toe. (Note: you can argue unknown if you want to pretend that you don't know how the game is played. However you cannot mix this up with stochastic/deterministic.)

(2) The card game “memory” or “concentration”. See: https://en.wikipedia.org/wiki/Concentration_%28game%29

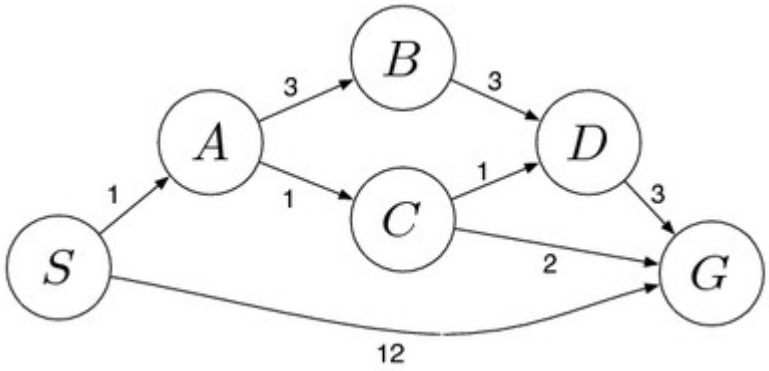
Solution:
Partially observable - You cannot see the under side of the cards.
Multi-agent - Again, not normally played by yourself.
Both deterministic and stochastic - When you first play, each card is unknown and so your picks are completely stochastic. However, after choosing a card, picking that card again is a deterministic option (it will always have the same outcome).
Sequential - You need to use past moves to match, so sequential.
Static - Again taking turns, so nothing is happening while you are thinking.
Known - Probably you both know the rules if you are playing.

(3) Suppose you are a TA for this course and assigned to grade Problem 1 on this homework for all students. (i.e. the actions are to assign a score to a student then to go to the next one.)

Solution:
Fullly observable - Students are graded by what they write down, which the TA can see. The knowledge inside a person's head is partially-observable, but this isn't graded....
Single-agent - It is only the TA and some paper (or pdfs). The paper isn't interacting, so only one “player”.

Deterministic - No ambiguity in assigning grades (assuming their handwriting/typing is not horrible).
Episodic - The past student's score has no effect on the next student's score (hopefully).
Static(ish) - Grading is not timed (unless it is past the expected return date...).
Known - I certainly hope the TA knows the problem/answer to know how they should assign the grades.

Problem 2. (20 points)
Consider the graph shown below. Assume you start at "S" and want to reach "G". Show step-by-step how you would solve this with uniform-cost search. At each step show:
(1) the "fringe"nodes
(2) the "explored" nodes
(3) which node you are taking next from the fringe set to move to the explored set



Solution:

Fringe (selected underlined)	Explored
<u>S=0</u>	
<u>A=1</u>, G=12	S
B=4, <u>C=2</u>, G=12	S, A
B=4, <u>D=3</u>, G=4	S, A, C
B=4, <u>G=4</u>	S, A, C, D

Done. (It is also acceptable to do B for one step, which would result in the line below.)

<u>G=4</u>	S, A, C, D, B

Done after this as well.

Problem 3. (25 points)
For each of the situations specify: (a) The initial state, (b) possible actions from the initial state, (c) a general description of other states, and (d) whether the approach is incremental or complete-state.

(1) Suppose you are making a salad. This requires you to chop the following; lettuce, tomatoes and carrots. You then need to mix them all together with some dressing.

Solution:
(a) Bowl empty and the lettuce, tomatoes and carrots are whole (not chopped).

(b) Three actions: chop carrots, chop tomatoes, or chop lettuce (cannot do dressing yet).
(c) The other states will the fruits/vegetables in all combinations of chopped or not. There is also a state where all the fruits/vegetables are chopped and the dressing is mixed in.
(d) Incremental (start with nothing and work towards a full salad).

(2) Suppose you are in your 3rd year of college and want to schedule classes for the next four semesters. There are a number of required classes and electives you want to take (not all classes are offered in all semesters).

Solution 1:
(a) No classes are assigned.
(b) There are multiple ways you could frame this. One way is to take some class that is required and figure out where it should go (which semester). The actions would then be: "Put CSci5432 in Fall-2018", "Put CSci5432 in Spring-2019", "Put CSci5432 in Fall-2019" or "Put CSci5432 in Spring-2020". You would have fewer options if CSci5432 was not available every semester.

Alternatively you could figure out which classes to put per semester. So the actions would be: "Take CSci5432 next semester", "Take CSci5555 next semester", "Take CSci4532 next semester", etc.. These would be the only classes that are actually offered next semester

(c) The other states would be various partial schedules for the upcoming semesters (depending on whether you did the from start to end or by class).
(d) Incremental (start with nothing and fill in classes one at a time).

Solution 2:
(a) A full schedule with random classes assigned (even if the course is not offered that semester).
(b) The actions are to swap any two class times in the schedule. If you want to consider multiple "electives" you could also add an action to substitute an elective for a different (not already scheduled) one.
(c) The other states are other combinations of classes in the schedule. (The schedule will always be full.)
(d) Complete state, as the schedule is filled from the start, but just valid. (Some classes might not be offered for the spot where they were randomly put.)

(3) You are a UPS truck and need to deliver packages to 20 different houses. (Problem 5.3 also asks how you would search this.)

Solution 1:
(a) Schedule starts blank
(b) Just like problem 2.2, there are options about whether you want to have actions as assigning houses to time slots or assigning time slots to houses. Assigning houses to time slots would have initial actions: visit house #1 first, visit house #1 second, visit house #1 third, ..., visit house #1 last.
(c) Again other states are various partial assignments, for example: visit house #1 third, visit house #2 15th, visit house #1 first. (This state is reached after taking 3 actions from the initial).
(d) Incremental (start with nothing and add a house to the schedule one at a time).

Solution 2:
(a) A full schedule with houses visited in random orders.
(b) Swap the order between any pair of houses (similar to last problem).

(c) Different permutations on the order to visit houses.
(d) Complete state, as the schedule is filled from the start. The actions of exchanging house ordering are trying to find better routes to visit the houses.

Problem 4. (25 points)
Between depth first search, breadth first search and uniform-cost search, for each part say which search is most appropriate? Support your answer with one to two sentences explaining why.

(1) While seeking when playing hide-and-go-seek.

Solution:
Depth-first-search. You keep exploring adjacent rooms until you cannot go any further. Only then do you turn around and take some other path (probably in the previous room) to explore another option.

(2) You got 5 songs for free from iTunes. You want to get the most amount of music played (i.e. longest duration of music) while also having songs from at least 3 different genres.

Solution:
Uniform-cost search. This is the only search that finds optimal solutions when there are not uniform edge-costs. Here edge costs are probably the song times (technically you should negate them as "costs" are bad and "song times" are good).

(3) Continuation of problem 3.3: What search would you use on this representation as the UPS if you wanted to minimize distance.

Solution:
Depending on how you framed the problem, either uniform-cost search or depth-first-search. If you had an incremental approach, the depth is limited to 20 (after you assign all houses, you are done), so there will be no infinite loops. This assumes you check how "good" a route is once you reach a leaf of the tree.

You could also compute the distances as you go along the tree. This would definitely require uniform-cost search as DFS and BFS don't handle variable edge costs.

If you did a complete-state, the tree is no longer a finite depth (you could swap the first and second houses visited back and forth forever). You should really use uniform-cost by evaluating how much better/worse the child is as the edge cost.

Problem 5. (15 points)
For each of the following, state whether the task is being done rationally. (Note: this is the strict artificial intelligence definition of "rational".)

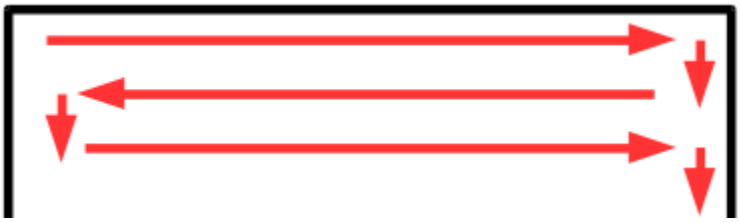
(1) Humans playing chess. (The goal is to win and the humans know the rules.)

Solution:
Not rational. The players know the rules of chess. I assume that (as humans) they will make some sort of mistake and not play "perfectly". Since this is not the "best", it is not rational.

(2) A “dumb” Roomba that just cleans and moves forward until it hits an obstacle. The Roomba then turns a random direction. The goal is to clean a single room as quickly as possible. (You can assume the Roomba can perfectly aim and sense when it hits something.)

Solution:
Not rational. There is a non-zero probability that the roomba will clean a spot twice before reaching an un-clean spot, when it does not have to “backtrack”. (The probability that it will “backtrack” gets high very quickly as more and more of the room gets cleaned.) This re-cleaning is not the “fastest”, so it is not rational.

(3) A “smart” Roomba that has figured out a map of the room and zig-zags back and forth to clean it (see picture below). Again the goal is to clean the room as fast as possible. Note: there might be furniture in the room.



Not rational. The whole “furniture” part actually does not matter. Not all rooms are going to be nice rectangles/squares, and this zig-zag approach is not optimal if the room is U shaped or something. Since the room-cleaning algorithm does not work best for all rooms, it is not rational to always use this zig-zag.

Note: I had hoped the furniture thing would help you think about routing and think “outside the box” of only considering square/rectangular rooms. You can also make the argument that there is no one way to get around furniture optimally if you have already zig-zaged up until a point.