CS 4100 artificial intelligence / CS 5100 foundations of AI

Khoury College, Northeastern University

Problem Set 1

Instructions

- All submissions must be typed. No exceptions are made to this rule.
- Hand-drawn figures are acceptable only where specified in the question, and provided all labels are clear and legible. If we can't read your text, we can't assign points. Please scan and insert any such figures in the final PDF document.
- You may not, as a general rule, use generative AI for problem sets. Any use of GenAI that is **solely** intended to improve writing clarity or sentence structure is acceptable, but must be accompanied by an appendix containing your original, unedited answers. If you use generative AI in this manner, start a new page titled 'Appendix' at the end of the written submission, and paste your original answers here with the corresponding question numbers clearly indicated.
- If you discuss this problem set with one or more classmates, please ensure that all parties must declare collaborators on their individual submissions. Such discussions must be kept at a conceptual level, and no sharing of written answers is permitted.

Deadlines

- Written submissions should be uploaded to Gradescope by 6:00 PM on 9/21/2024.
- Gradescope will show a 'late' deadline of 9/24. This is intended solely for any students who may wish to invoke the freebie, outlined in the course policies.
- Any submissions received after 6:00 PM on 9/21 will be considered late, and will automatically invoke the use of your freebie.
- Regrade requests must be submitted on Gradescope within 1 week of receiving your grade, after which no further requests will be entertained.

Reach Out!

If at any point you feel stuck with the assignment, please reach out to the TAs or the instructor, and do so early on! This lets us guide you in the right direction in a timely fashion and will help you make the most of your assignment.

Categorizing AI Environments

Q1 Under what conditions may maze-solving be considered a **fully observable** environment? What are the agent's percepts, and which algorithms are best-suited to solving this problem in this setting? (2)

Q2 Under what conditions may maze-solving be considered a **partially observable** environment? What are the agent's percepts, and how would you approach this problem in terms of implementing the agent's logic? (2)

Q3 Is a known environment always fully observable? Explain with an example. (2)

Q4 Is a partially observable environment always an unknown environment? Explain with an example. (2)

Q5 What is the difference between an unknown environment and a non-deterministic environment? Explain with an example. (2)

Search[1]

Q6 For a search problem with average branching factor b, the version of BFS from our class notes explores $\mathcal{O}(b^d)$ nodes. Explain how the number of nodes explored changes when the `if w == goal: terminate` condition block is removed; i.e., when expanding a node, v, each neighbor w is not checked for whether it is the goal state. (2)

Q7 Derive the number of nodes (asymptotic analysis, $\mathcal{O}$-notation) iterative deepening search expands, assuming the depth limit is increased by 1 at every iteration? (2)

[LINK] Big-O notation - notes, MITx

In the rest of this section on Search, we will be using Rook Jumping Mazes (RJMs) - a game based on Rook moves in Chess - to understand various search techniques. In an RJM, you are given a square $n \times n$ grid, with each cell containing a number between 1 to $n-1$. From any cell, the player may move either horizontally or vertically, but must take exactly as many steps in the chosen direction as the number on the player's current cell. For instance, in the example on the next page, the agent starts at D1, and the cell has the number 3 on it; therefore the agent can move to either D4 or A1, since they are the only cells that are 3 squares away from D1, and so on. Regardless of the length of the jump (i.e. number of cells passed), this action is considered a single jump (i.e., having a cost of 1) for purposes of our search algorithms. The objective is to get from an assigned start state to a pre-specified goal state.

[1] The RJM questions in the Search section are inspired by Todd Neller's work presented at EAAI 2010. Original publication: **T. Neller**, "Model AI Assignments", AAAI, vol. 24, no. 3, pp. 1919-1921, Jul. 2010.

Consider this example:

	1	2	3	4	5
A	4	G	3	3	2
B	3	3	4	4	4
C	1	1	3	4	4
D	3	3	3	1	2
E	3	1	1	3	3

The cell D1 is the start state and the cell A2 is the goal state. The shortest solution to this RJM is: [D1, A1, A5, C5, C1, C2, D2, A2]. Such a game lends itself well to the various search algorithms we covered in class.

Q8 Can both Breadth-First Search and Depth-First Search be used to solve RJMs? Will both yield the shortest path? Explain your answer. (2)

Q9 Assuming that each jump has a cost of 1, irrespective of the number of cells the rook passes while making that jump, we can use A* search to find the shortest path. Recall from class that A* search requires a consistent heuristic to guarantee an optimal solution. Show that for this problem, the Manhattan distance from any cell to the goal cell divided by $(n-1)$ is a consistent heuristic, where n is the number of cells in a row/column. In other words, show that

$$H(node, goal, n) = \frac{|node_x - goal_x| + |node_y - goal_y|}{n-1}$$

is a consistent heuristic. Remember that a proof may not rely on a single example, but instead must hold true in general. (5)

[LINK] Here is a good resource on concise and clear proof-writing, which closely mirrors the expectations for most written answers in the course. It is a skill that will serve you well, especially if you are interested in taking more advanced courses or are considering getting involved in research.

Q10 Consider the following RJM:

	1	2	3	4
A	2	3	2	3
B	1	2	1	1
C	3	2	2	3
D	1	1	3	G

On this maze, using the heuristic from Q9, complete the A* search process shown below, starting with Step 2. Terminate your search when the Goal node (cell D4) is popped from the priority queue. Double check your calculations! (8)

Initialization:

- Priority Queue, PQ = {}
- cost = 0
- v = `NULL`
- path = []

Priority for B2 = cost + Heuristic(B2) = 0 + 4/3

Push start state (`v`=B2, `priority`=4/3, `cost`=0, `path`=[]) to PQ

PQ = {(B2, 4/3, 0, [])}

Step 1

Since PQ $\neq \emptyset$, pop by priority (lower is better in this setting, since priority is based on total heuristic cost).

- Priority Queue, PQ = {}
- cost = 0
- v = B2
- path = [B2]

Since B2 is not the goal state, continue.
Neighbors of B2 = {D2, B4}

Priority for D2 = cost + wt(B2, D2) + H(D2) = 0 + 1 + 2/3 = 5/3
Priority for B4 = cost + wt(B2, B4) + H(B4) = 0 + 1 + 2/3 = 5/3

New cost for D2 = cost at B2 + wt(B2, D2) = 0 + 1 = 1
New cost for B4 = cost at B2 + wt(B2, B4) = 0 + 1 = 1

Push (`v`=D2, `priority`=5/3, `cost`=1, `path`=[B2])
Push (`v`=B4, `priority`=5/3, `cost`=1, `path`=[B2])

PQ = {(D2, 5/3, 1, [B2]), (B4, 5/3, 1, [B2])}

Step 2, . . . : `TODO`

Local Search

Your professor is a coffee snob, and since he knows a thing or two about AI, he tries to build an espresso machine that will learn a user's preferences over time to pull the perfect shot. He lets the machine control the following values:

- Weight of coffee beans (15-21 grams)
- Grind size (250-400 microns)
- Water Pressure (7-11 bars)
- Brew time (27-33 seconds)

`Sidenote: There was a kickstarter project in 2016 that proposed something similar and failed spectacularly. Link to story.`

Q11 Based on our discussions in class, how you would implement such a program using local search? What would be a reasonable objective function? (This question is somewhat open-ended, so don't worry too much about the 'right' answer. I am really looking for whether your thought process reflects a solid understanding of local search.) (2)

Q12 Explain how the three variants of local search discussed in class (random restarts, varying step size, and simulated annealing) may be applicable to this problem. (2)

Q13 An important design choice in the implementation of a simulated-annealing-based coffee machine would be the initial temperature T and the decay constant d. Given the objective function you proposed in Q11, what values of T and d should you choose such that the probability of accepting **any** worse neighboring configuration starts at above 0.995, and decays to less than 0.005 over 35 iterations. Explain how you arrived at your values for T and d. (2)

[HINT: if your objective function in Q11 is designed to be maximized, then you should convert it to an energy function that needs to be minimized for simulated annealing.]

Academic Integrity

Q14 Review, and copy/paste the following academic integrity acknowledgement in your final submission as the answer to Q14:

I have read and understood the academic integrity policy as outlined in the course syllabus for CS4100/CS5100. By pasting this acknowledgement in my submission, I declare that all work presented here is my own, and any conceptual discussions I may have had with classmates have been fully disclosed. I declare that generative AI was not used to answer any questions in this assignment. Any use of generative AI to improve writing clarity alone is accompanied by an appendix with my original, unedited answers.