NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MIDTERM ASSESSMENT FOR
Semester 1 AY2024/2025

CS3243: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

SOLUTIONS

October 2, 2024 Time Allowed: 90 Minutes

INSTRUCTIONS TO CANDIDATES

1. This assessment contains THREE (3) questions. All the questions are worth a total of 30 MARKS. It is set for a total duration of 90 MINUTES. You are to complete all 3 questions.
2. This is a CLOSED BOOK assessment. However, you may reference a SINGLE DOUBLE-SIDED A4 CHEAT SHEET.
3. You are allowed to use NUS APPROVED CALCULATORS.
4. If something is unclear, solve the question under reasonable assumptions. State your assumptions clearly in the answer. If you must seek clarification, the invigilators will only answer questions with Yes/No/No Comment answers.
5. You may not communicate with anyone other than the invigilators in any way.

STUDENT NUMBER: | **A** | | | | | | | | |

EXAMINER'S USE ONLY		
Question	Mark	Score
1	10	
2	10	
3	10	
TOTAL	30	

1. Suppose that you are given a 2-dimensional grid with R rows and C columns, and with an initial position (r_s, c_s). Our objective is to find a path from this initial position to a goal position (r_g, c_g). You may assume that there is only one goal position, and that this goal position is reachable from the initial position.

Further, assume that the possible actions that the agent may take within this problem are to move in the 8 possible directions (by 1 cell). This is depicted in the figure below.

$(rr–1, cc–1)$ ↖	$(rr–1, cc)$ ↑	$(rr–1, cc+1)$ ↗
$(rr, cc–1)$ ←	(rr, cc)	→ $(rr, cc+1)$
↙ $(rr+1, cc–1)$	↓ $(rr+1, cc)$	↘ $(rr+1, cc+1)$

You are to adopt the following partial search problem formulation for the problem described above.

- ***State representation*** **(S_i)**:
  - The tuple (*pos*, *prev_act*), where:
    - *pos* = (rr, cc)
    - *prev_act* ∈ { ↖ , ↑ , ↗ , ← , → , ↙ , ↓ , ↘ }
  - ***Initial state*** **(S_0)**:
    - $S_0.pos = S_0[0] = (r_s, c_s)$
    - $S_0.prev\_act = S_0[1]$ = NULL
- ***Transition model***:
  - Given a parent state, S_i, and a selected move (i.e., action), *act* (to be taken at S_i), return the new child state ((r', c') , *act*), where (r', c') corresponds to the resultant coordinate reached based on the action taken from $S_i.pos$.
  - Assume that each action, *act*, corresponds to a tuple ((r', c') , *dir*), where *dir* corresponds to an encoding of the 8 possible directions described in the figure above. For example, we could have *act* = ((10, 10) , ↗).
- ***Goal test***:
  - Returns $S_0.pos == (r_g, c_g)$
- ***Cost function***:
  - All actions have uniform cost $k = 1$.

(i) [1.0 marks] Assuming (for now) that there are <u>*no obstacles*</u> in the grid, explain how the ***actions*** function, which takes a state, S_i, as its input, would determine <u>*all*</u> possible legal actions to output (as a tuple of all possible actions). (Note: code is unnecessary here.)

Solution: The ***actions*** function would check the 8 possible directions, and only add the action ((r', c') , *act*) to the tuple of if $r' \in [1, R] \wedge$ and $c' \in [1, C]$.

(ii) [2.0 marks] Provide an upper bound, only in terms of r_s, c_s, r_g, and c_g, for the space complexity of the Breadth-First Search (BFS) algorithm (with Tree Search), on this problem.

Solution: Given no obstacles, it would take $d = \mathbf{max}(|r_g - r_s|, |c_g - c_s|)$ actions to get to the goal position, (r_g, c_g), from the initial position, (r_s, c_s). Bounding the branching factor to $b = 8$, we would therefore have BFS space complexity: $\mathbf{O}(8^d)$.

Assume that obstacles are now introduced into the problem. However, we place a constraint on the positions of obstacles such that no two obstacles may be adjacent to each other. This is depicted in the diagram below, where OBS refers to a cell with an obstacle, and NO OBS refers to a cell without an obstacle.

NO OBS ↖	NO OBS ↑	NO OBS ↗
NO OBS ←	OBS	→ NO OBS
↙ NO OBS	↓ NO OBS	↘ NO OBS

Do note that a boundary is *not* considered an obstacle.

(iii) [4.0 marks] Redesign the ***actions*** function such that the branching factor *b* is minimised, which would in turn reduce the complexity of the resultant search tree. Do note that your redesign must still ensure that any goal that is reachable under the original set of actions must still be reachable under your redesigned set. Based on your design, also provide a rationale that explains why goals reachable under the original set are still reachable under the redesigned set.

Solution: There are two main points to note in the redesign of the ***actions*** function.

1. Going backwards is unnecessary since the maze is static.
2. Diagonal moves are a composition of two non-diagonal moves – e.g., we may perform → and ↑ in place of ↗.

Therefore, we simply need to define the ***actions*** function such that we only consider the legal actions in {↑, ↓, ←, →} (i.e., we remove the diagonal moves). Further, we will also exclude the opposite of the last action that was taken (i.e., the exclusion of the action that would bring us back to *pos* in the parent state). This can be done by simply checking *act* in the parent state.

This means that the upper bound on *b* is now 3 (except at the initial state, where it is 4).

We may further rationalise that we can still reach every goal that was reachable under the original set of moves since each removed move may be substituted with two of the current moves. For example, → and ↑ in place of ↗. The same applies for each of the other three removed diagonal moves.

Also note that the obstacles, under the given constraint, cannot block progression using {↑, ↓, ←, →} since there will always be a gap to move around any obstacle.

(iv) [2.0 marks] With the new ***actions*** function from Part **(iii)**, provide an updated upper bound, only in terms of r_s, c_s, r_g, and c_g, for the space complexity of BFS (with Tree Search).

Solution: The new upper bound on the branching factor, b, is 3 (except at the initial state, where it is 4). However, since we now cannot move diagonally, the depth at which we find a solution is no longer $\max(|r_g - r_s|, |c_g - c_s|)$, but now $|r_g - r_s| + |c_g - c_s|$.

The space complexity of BFS is thus: $\mathbf{O}(4 \cdot 3^d)$, where $d = |r_g - r_s| + |c_g - c_s| - 1$.

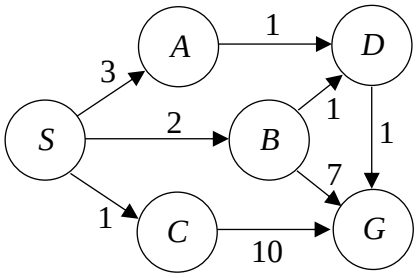
Note that the above does not consider obstacles; we require $1.5d + 1$ to weave around obstacles.

(v) [1.0 marks] Compare the two different versions of the ***actions*** function and explain when each would be (generally) preferred.

Solution: Let the distance between the initial position and goal position denote a diagonal in a rectangle. When this rectangle is a square, then we have the maximal difference between $d_1 = \max(|r_g - r_s|, |c_g - c_s|)$ and $d_2 = |r_g - r_s| + |c_g - c_s|$. Specifically, here, $d_2 = 2 \cdot d_1$. However, when the rectangle is a very thin rectangle (e.g., with width 1), then we have a minimal difference, with $d_2 = d_1$. We may observe that for any $d_1 \in [1,\infty]$, we have $8^{d_1} < 3^{2 \cdot d_1}$, but of course $8^{d_1} > 3^{d_1}$. To account for obstacles, recall that we use $1.5d + 1$ (to weave around the obstacles).

Thus, the more the diagonal forms a square, the better the original ***actions*** function, while the more the diagonal forms a thin rectangle, the better the new ***actions*** function.

2a. Consider the following state space. The initial state is S and the goal state is G. The cost of each action is reflected on each edge.



n	h(n)
S	5
A	1
B	2
C	10
D	0
G	0

(i) [1.0 marks] Is the specified heuristic, h, admissible? Provide a rationale.

Solution: No. Since $h^*(S) = 4$ (via $S - B - D - G$).

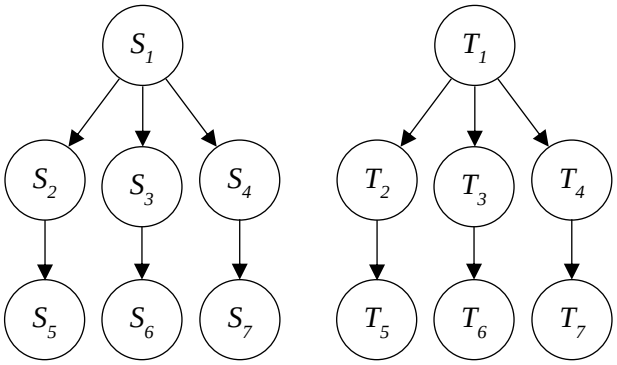
(ii) [1.0 marks] Apply the Greedy Best-First Search algorithm implemented using Graph Search Version 1, tiebreaking in ascending alphabetical order, and h, to the given graph. With this application, specify the sequence of nodes popped from the frontier.

Solution: $S - A - D - G$

(iii) [1.0 marks] Specify a change to exactly **one** h(n) value so that the Greedy Best-First Search algorithm using Graph Search Version 1 and h would output the optimal path.

Solution: Change the value of h(B) to 0 or change the value of h(A) to 3 or more.

2b. Suppose we define a certain local search problem, which results in the state space given below, where S_1 and T_1 are 2 randomly generated initial states. The heuristic values for each state are provided. Assume that we seek a global minimum. Note that vertices with no edges extending from them are terminal states (i.e., states with no successors).



n	h(n)
S_1	10
S_2	7
S_3	7
S_4	6
S_5	3
S_6	2
S_7	3

n	h(n)
T_1	12
T_2	8
T_3	6
T_4	6
T_5	3
T_6	2
T_7	1

Assume that tiebreaking is performed by first selecting nodes based on ascending subscript order and then selecting S nodes before T nodes. For example, when tiebreaking between S_2, S_3, and T_2, we would tiebreak by first selecting S_2, then T_2, then S_3 (since S_2, and T_2 both have subscripts of 2, whereas S_3 has a subscript of 3, and S_2 is chosen before T_2, given that we pick S nodes before T nodes).

(i) [1.0 marks] Trace the application of local beam search with $k = 2$ on the above local search problem. You are to assume the use of the given heuristic h and trace the progression of beam by completing the given table (i.e., define the composition of the beam in each successive iteration until the algorithm terminates).

Solution:

Iteration	Beam
1	S_1, T_1
2	T_3, S_4
3	T_6, S_7

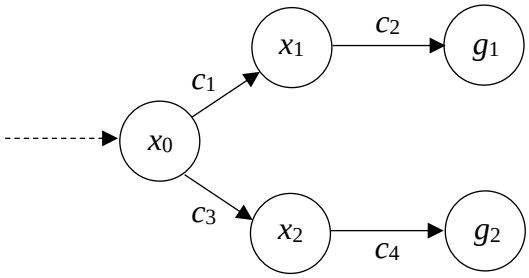
(ii) [1.0 marks] What is the state output by the beam search specified in Part **(i)**? Determine if this state corresponds to a global or local minimum.

Solution: T_6, since h(T_6) < h(S_6). While we cannot be sure what the global minimum is, we do not that h(T_7) = 1, which is lower than T_6. Therefore, T_6 must be a **local** minimum.

(iii) [1.0 marks] Explain how we may change the tiebreaker such that the local beam search specified in Part **(i)** will output the optimal solution for the given state space. Note that you may **not** reference h-values or cite specific states in your proposed tiebreaker.

Solution: Changing the tiebreaking from ascending to descending subscript order, **OR** from S preference to T preference, is sufficient for the given beam search to find the optimal state (T_7).

2c. Consider the scenario where we have a graph with k goal nodes (i.e., $g_1, \ldots, g_k$), as shown in the partial graph below.



Suppose that node x_0 has just been popped off the frontier (using A* Search), and that the goal along the optimal path is g_2, which implies that g_1 is a non-optimal goal.

(i) [2.0 marks] Given a consistent heuristic, h, determine an upper bound for h(x_1) only in terms of the action costs c_1, c_2, c_3, and c_4. Assume that node x_1 is visited before node g_2.

Solution: h(x_1) $\leq c_3 + c_4 - c_1$

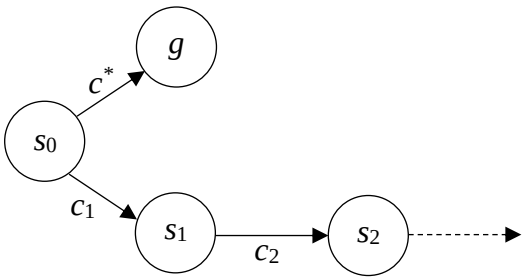
/* h(x_1) $\leq c_2$, since h(g_1) = 0 and h is consistent, but note that $c_3 + c_4 < c_1 + c_2$ (assert optimality). So, we have h(x_1) + $c_1 \leq c_3 + c_4 < c_1 + c_2$, or h($x_1$) $\leq c_3 + c_4 - c_1$, since x_1 is visited before g_2. */

(ii) [2.0 marks] Consider the following statement.

Even if we remove the assumption that action costs must be lower bounded by a positive constant, ε, the A Search algorithm implemented using Tree Search and utilising a consistent heuristic, h, is complete on a problem where there exists a solution, the branching factor, b, is finite, and the depth of the search tree, d, is infinite.*

Disprove the above statement with a counterexample.

Solution: Consider the following graph, where g is the only goal that may be reached from the initial state, s_0, via an action with cost c^*. In this graph, s_0 is also linked to an infinite series of states $s_1, s_2, \ldots, s_\infty$, such that the action cost from state s_{i-1} to s_i is denoted by c_1.



If there is no lower bound set on each action cost, $c_i \in \{c_1, \ldots, c_\infty\}$, then we may choose to define these action costs based on a convergent series such as the reciprocals of triangular numbers. In this case, since the sum of the chosen series converges to 2, we need only set c^* to 3 to find that we will never pop node g from the frontier.

/* For details on the sum of reciprocals of triangular numbers refer to the following: https://proofwiki.org/wiki/Sum_of_Sequence_of_Reciprocals_of_Triangular_Numbers */

// There are many valid answers, with the easiest relying on each c_i aside from c^* having cost 0.

3. Many students have become familiar with the NUSMODS website in recent times. One of its uses is to check course prerequisites, which allows one to plan a course schedule.

Suppose that you are a student enrolled in a lifelong-learning programme at NUS, and as such, you are interested in determining the number of semesters it would take you to complete a target set of courses $\boldsymbol{S} = \{C_{1,1}, \ldots, C_{N,1}, C_{1,2}, \ldots, C_{N,2}, \ldots, C_{N,M}\}$, where, for each $C_{i,j} \in \boldsymbol{S}$, $j \in [1, M]$ represents the type of course, and $i \in [1, N]$ represents the level of the course.

Assume that course prerequisites for each $C_{i,j} \in \boldsymbol{S}$, adhere to the following rules.

(A) To read course $C_{p,q}$, one must first complete all courses $C_{i,j}$, where $0 < i \leq p$, and $0 < j \leq q$, but excluding the case there $i = p \wedge j = q$.
(B) In each semester, you may maximally only do K courses. (Overloading is forbidden!)

As mentioned above, our objective is to perform a ***search for a course schedule*** that would allow one to complete all the courses in $\boldsymbol{S}$ (assuming that you would always pass them) in the least amount of time.

Assume that this equates to completing the course $C_{N,M}$ (i.e., our goal can be summarised to just having the course $C_{N,M}$ completed). As a corollary, do note that this means that for any given $\boldsymbol{S}$, we assume that all courses in the specification may be completed (eventually).

To perform this search, we define each state as an M-tuple to represent the highest-level course that one has taken for each course type, i.e., $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$, where each π_i represents the highest level that one has taken for course type i.

(i) [2.0 marks] Assume that at any state S_i, we have taken n_i courses. Prove or disprove the following.

$$\textit{The heuristic } h_1(S_i) = \left\lfloor \frac{MN - n_i}{K} \right\rfloor \textit{ is admissible.}$$

Solution:

The numerator $(NM - n_i)$ corresponds to the total number of remaining courses at state S_i.

Since we may, at most, take K courses in each semester, then the minimum number of semesters (i.e., ignoring prerequisites) that are required to complete the $(NM - n_i)$ courses must be $\lceil (NM - n_i) / K \rceil$.

Therefore, we have $\lfloor (NM - n_i) / K \rfloor \leq \lceil (NM - n_i) / K \rceil \leq h^*(S_i)$.

Thus, h_1 is admissible since for any state S_i, we have $h_1(S_i) = \lfloor (NM - n_i) / K \rfloor \leq h^*(S_i)$.

(ii) [4.0 marks] Specify a different heuristic, h_2, which is *consistent* and *dominates* h_1. More specifically, you must complete the following.

(1) Specify h_2.
(2) Prove that h_2 dominates h_1.
(3) Prove that h_2 is admissible.
(4) Prove that h_2 is consistent.

Your specification of h_2 must not be a function of the optimal heuristic h^*.

Solution: Let $h_2 = \lceil (NM - n_i) / K \rceil$.

Dominance:
Regardless of the given state S_i, with courses taken n_i, we must either have:

- Case 1: $(NM - n_i)$ divisible by K
- Case 2: $(NM - n_i)$ not divisible by K

For Case 1, $\lceil (NM - n_i) / K \rceil = \lfloor (NM - n_i) / K \rfloor = (NM - n_i) / K$.

For Case 2, $\lceil (NM - n_i) / K \rceil > \lfloor (NM - n_i) / K \rfloor$ as we round up with $\lceil .. \rceil$ and round down with $\lfloor .. \rfloor$.

Therefore, we have, for all S_i, $\lceil (NM - n_i) / K \rceil \geq \lfloor (NM - n_i) / K \rfloor$; i.e., h_2 dominates h_1.

Admissibility:
The argument that h_2 is admissible is similar to the argument given in Part **(i)**. The numerator, $(NM - n_i)$, corresponds to the total courses remaining. Consequently, since the student may only take K courses per semester, then, we have $\lceil (NM - n_i) / K \rceil \leq h^*(S_i)$.

Consistency:
From one semester to the next, the value of h_2 decreases by at most 1.

- For each successive state, S_{i+1}, we have $0 \leq (n_{i+1} - n_i) \leq K$.
- Therefore, $\lceil (NM - n_i) / K \rceil - \lceil (NM - n_{i+1}) / K \rceil \in \{0, 1\}$.

Consequently, as each action in this problem has a uniform cost of 1, for any parent state S_i, and a child state of S_i, S_{i+1}, we have $h_2(S_i,) \leq \textbf{\textit{cost}}(S_i, action, S_{i+1}) + h_2(S_{i+1})$ since the difference $h_2(S_{i+1}) - h_2(S_i)$ is at most 1, while $\textbf{\textit{cost}}(S_i, action, S_{i+1})$ is equal to 1.

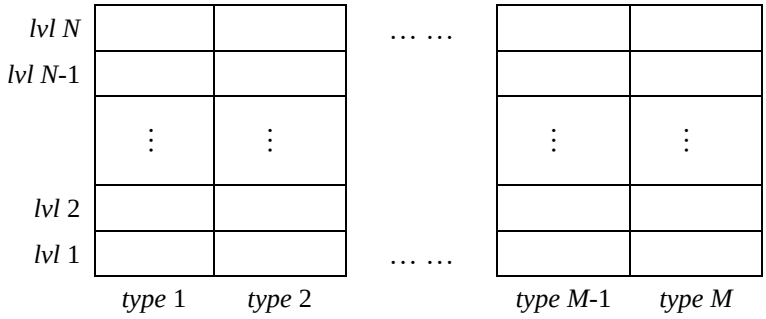
Therefore, h_2 is consistent.

(iii) [4.0 marks]. Show that when $K \geq M$, for a freshman with no courses completed, it would take minimally $(N + M - 1)$ semesters to graduate.

Solution:

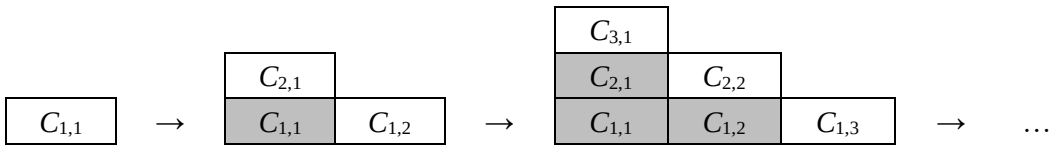
To prove the given statement, that with no courses taken yet, a freshman requires at least $(N + M - 1)$ semesters to graduate, we must explain why π is a non-increasing sequence.

Let us define a 2-dimensional grid to model the course schedule, such that the rows correspond to course levels and the columns correspond to course types. This would give us the following.



Notice that under Prerequisite Rule (A), and the goal requirement, i.e., completing course $C_{N,M}$, where there are exactly N levels and M course types, that $\boldsymbol{S}$ must contain exactly M course types, with each course type having exactly N levels, for the student to fulfil the goal of completing course $C_{N,M}$.

Further, given that the student must take $C_{1,1}$ before $C_{2,1}$, and $C_{1,1}$ before $C_{1,2}$, then we notice that in the first semester, the student may only take a single course $C_{1,1}$. In the subsequent semester, the student may only take $C_{2,1}$ and $C_{1,2}$ (having the requirement $C_{1,1}$), but no other courses (since the student would not have the requisites for any other courses. This gives us the progression shown below.



This shows that in each successive semester, at most, the student could complete the courses given by the secondary diagonal extending from the top-left to the bottom-right. Further, given that $K \geq M$, we also know that we can always complete all the courses along a diagonal over a single semester.

It follows that we will always need M semesters to complete the first level of course type M, and once we have done so, we would then still require $N - 1$ additional semesters to advance from $C_{1,M}$ to $C_{N,M}$. However, while doing so, we must have completed all the other courses following the above optimal pattern.

END OF PAPER

BLANK PAGE

BLANK PAGE

BLANK PAGE