National University of Singapore
School of Computing
CS3243 Introduction to AI

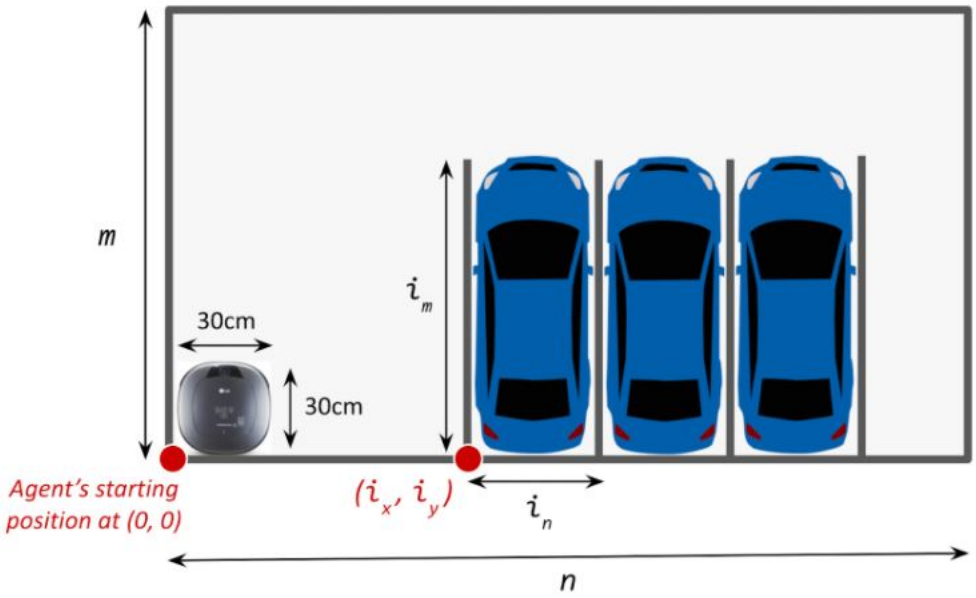
Midterm Examination Review

Issued: March 7, 2022 Discussion in: Week 8

1. You are tasked to design a floor-cleaning agent for a carpark.

   Assume that the problem environment is a rectangular carpark that is enclosed, i.e., all entrances/exits are sealed and the agent cannot leave the carpark. The size of the carpark is $m \times n$, where length m refers to the distance, in centimetres (cm), in the North direction starting from the agent's starting position, and width n refers to the distance, in centimetres (cm), in the East direction.

   Assume further that the floor-cleaning agent is of size $30cm$ by $30cm$ and that it starts at the bottom-left-most corner of the carpark. The figure below shows an example of one such carpark with the floor-cleaning agent in its starting position (image may not be drawn to scale).



   The agent can only move in four directions: Up, Down, Left and Right, with a specified (non-zero) distance. At each position, the agent can also choose to clean the area directly beneath it, i.e., the $30cm$ by $30cm$ area that it currently occupies.

Assume further that the carpark contains k rectangular parking spaces that are orthogonal to the perimeter of the carpark (as shown in the figure above, the lines defining each rectangular parking space are parallel to the lines defining the perimeter of the rectangular carpark).

For each parking space i (where $1 \leq i \leq k$), the size ($i_m \times i_n$) and position (i_x, i_y) of the space are known. Length i_m refers to the distance, in centimetres (cm), in the North direction starting from the bottom-left-most point of the parking space, and width i_n refers to the distance, in centimetres (cm), in the East direction. Further, i_y refers to the displacement, in centimeters (cm), of the bottom-left-most point of the parking space in the North direction relative to the agent's starting position, and i_x refers to the displacement, in centimetres (cm), in the East direction. The starting position of the agent is $(0, 0)$.

Assume that all parking spaces in the carpark are occupied by vehicles and that the agent cannot access these spaces (occupied areas). You may assume that the environment is static, i.e., the vehicles will not move. Also assume that either the front, back, or both front and back, of all parked vehicles are not obstructed by other vehicles.

The goal of the agent is to clean all the accessible areas in the carpark, i.e., all areas not occupied by the parking spaces (unoccupied areas). You may assume that all areas not occupied by the parking spaces are accessible by the agent.

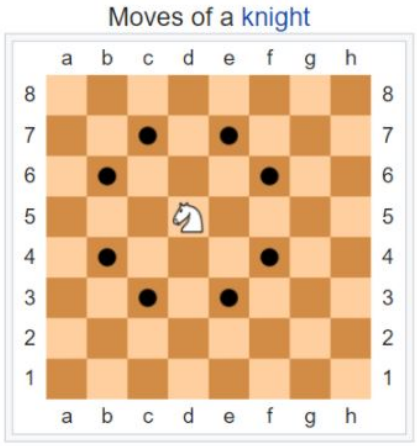
Formulate the above as a general search problem. More specifically, define the following:

- State representation
- Initial state
- Actions
- Transition model
- Step cost
- Goal test

If necessary, you may identify any other assumptions you have made. However, assumptions that are contradictory to any instruction in the question, or that are unreasonable, will be invalid.

2. Consider a chess-variant game, where we are given a Knight on a 3 board, with its initial position labelled as S and its goal position labelled as G.

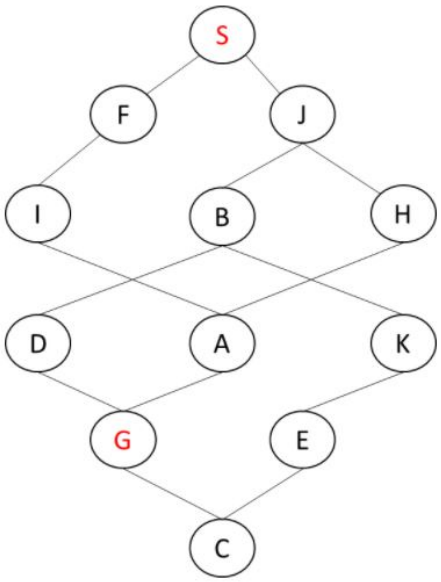
   Refer to the picture below on the moves that a Knight can make (which you should be familiar with by now given Project 1 and 2).



   In the table below, you are given the (3) board configuration, with letters labelling each grid in the board. As mentioned above, the Knight is placed at the starting position S and the goal position is labelled as G.

S	A	B	C
D	E	F	H
I	J	G	K

The undirected search graph for the Knight is as follows.



(a) Trace the following algorithms. With each trace, specify the order of the nodes in terms of when the goal test is performed (i.e., first node tested, then second, then third, etc.). Assume that when pushing to the frontier, ties (i.e., when pushing multiple successors) are broken based on alphabetical order.

i. BFS with limited-graph-based implementation and early goal test.

ii. BFS with limited-graph-based implementation and late goal test.

iii. DFS with limited-graph-based implementation (note that DFS does not use an early goal test).

(b) Assume a heuristic h, such that the heuristic value, $h(x)$, for each grid position, x, is shown below. Additionally, all step costs have a value of 1.

$S, h(S) = 3$	$A, h(A) = 1$	$B, h(B) = 1$	$C, h(C) = 1$
$D, h(D) = 1$	$E, h(E) = 2$	$F, h(F) = 2$	$H, h(H) = 2$
$I, h(I) = 1$	$J, h(J) = 1$	$G, h(G) = 0$	$K, h(K) = 3$

i. Trace the A* Search with limited-graph-based implementation using h by specifying the order of the nodes in terms of when the goal test is performed (i.e., first node tested, then second, then third, etc.). Assume that when pushing, or popping, from the frontier, ties (i.e., nodes with the same evaluation function value) are broken based on alphabetical order.

ii. Using A* search with limited-graph-based implementation using h, determine the path taken by the Knight from its starting position S to the goal position G.

Note that you should express your answers in the form S–B–A–F–G (i.e., no spaces, all uppercase letters, delimited by the dash (–) character), which, for example, corresponds to the order S, B, A, F, and G.

3. Let C^* be the cost of an optimal path. Assuming that every action costs at least some small positive constant ϵ. Explain why the time and space complexity of the UCS algorithm is given as: $O(b^{1+\lfloor C^*/\epsilon \rfloor})$.

4. Suppose you have two admissible heuristics, $h1$ and $h2$. You are also given an inadmissible heuristic $h3$. You decide to create the following new heuristic functions:

   - $h_4(n) = min(max(2 * h_1(n), h_3(n)), h_2(n))$
   - $h_5(n) = max(h_1(n), min(h_2(n), h_3(n)))$
   - $h_6(n) = min(h_3(n), max(h_1(n), 1.1 * h_2(n)))$
   - $h_7(n) = max((h_1(n) + h_2(n))/2, h_3(n))$

   For each of the new heuristics, specify if it is admissible or not. Justify your answer.

5. Given a set of admissible heuristics: h_1, h_2, h_3, $\ldots$, h_n, define a new heuristic g where g is a function of the set of the heuristics that will result in A* expanding a minimal number of nodes while still guaranteeing admissibility.

6. UWordle is a word game where players attempt (with unlimited guesses) to guess a 5-letter English target word. After every guess, feedback will be in the sequence of 5 colored tiles corresponding to the characters in the guessed word, indicating:

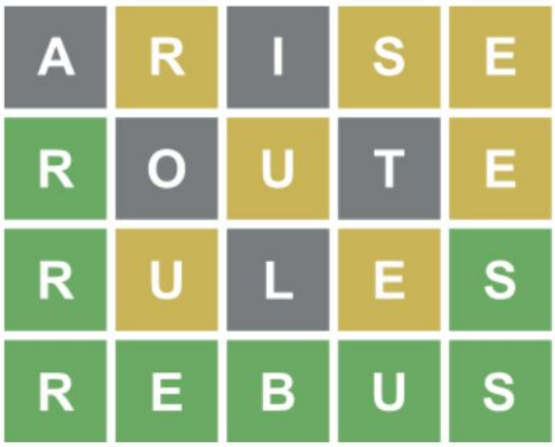
   - Green - Letter is in the word and in the correct position.
   - Yellow - Letter is in the word but in the wrong position.
   - Grey - Letter is not in the word at all.

   Note that in the original game (i.e., Wordle), there are only 6 guesses; our version, UWordle has unlimited guesses.

   Each word guessed, as well as the target word, are valid English words of 5-letters. (Also note that for the purposes of the UWordle problem, only consider capitalised English letters of the alphabet.)

   The following example illustrates a game of UWordle completed in 4 guesses. The goal is to guess the word "REBUS". The first guess of "ARISE" yielded the following information:

   - The characters A, I are not in the target word.
   - The characters R, S, E are in the target word but not in the correct positions.



   This question is concerned with solving the UWordle puzzle as a search problem.

Assume we have a valid dictionary, D, of all valid 5-letter English words, and that the target word can be found in D. Given a fixed but unknown target word, we represent the search problem as follows.

- **State Space**: Each state is represented by the set of candidate words C_i, where C_i is the set of possible target words (which will include the actual target word) for that state.
- **Initial State**: Entire dictionary D.
- **Goal State**: The size of candidate words $|C| = 1$.
- **Action**: Pick a word in D.
- **Transition Model**: From the observation (i.e., sequence of colored tiles received as feedback) following each action, we exclude all words in C_i that violate the new observed constraints to create C_{i+1}.
- **Cost**: The cost of every guess is 1.

(a) Determine the environment characteristics of the above problem.

i. Fully or Partially Observable?
ii. Single or Multi-Agent?
iii. Deterministic or Stochastic?
iv. Static or Dynamic?
v. Discrete or Continuous?

(b) Heuristic h_1 is such that for any state s_i, $h_1(s_i) = k$, where k is a constant value. State and prove that for your chosen value of k, h_1 is admissible.

(c) Propose another distinctly different but useful[1] heuristic h_2 that dominates h_1. Explicitly state h_2, explain how h_2 can be computed and proof dominance.

Discuss how h_2 is useful to a local search algorithm.

If your proposed heuristic is admissible, provide a proof. If your proposed heuristic is not admissible, proof and justify why another admissible heuristic cannot be found.

[1] Similar to Tutorial 3, Q1, your heuristic must be computable, explicitly defined and non-trivial (i.e., your heuristic may not be $h(s) = 0$ for all states s, the (abstract) optimal heuristic, $h*$, or a linear combination/simple function thereof).