Exercice 1 :

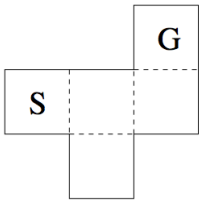
Formalize the following problems as state-space problems: define a suitable state space, a start state, set of actions, the action function, and a goal test.

a. In the movie *Die hard: With a vengeance*, Bruce Willis and Samuel L. Jackson are given a water jug riddle, which they need to solve in order to disarm a bomb: There is a water fountain and two jugs that can hold 3 and 5 liters of water, respectively. How do you measure up 4 gallons of water (to disarm the bomb) using only the two jugs?

b. (From Russel & Norvig, Exercise 3.9): The *missionaries and cannibals problem* is usually stated as follows: Three missionaries and three cannibals are on one side of a river, along with a boat that can hold either one or two people. Find a way to get everyone to the other side of the river without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

Exercice 2 :

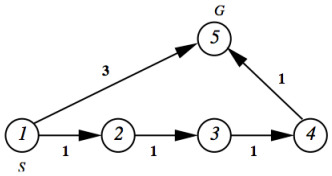
Consider the following robot navigation problem where we want to go from S to G:

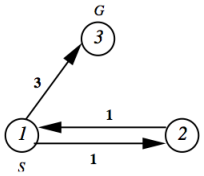


- Draw the search tree for this problem (or as much of the search tree as is needed to answer the following:
- Show how iterative deepening search will solve this problem: show the order in which nodes of the search tree will be selected and expanded.

Exercice 3 :

Consider the following two state spaces:





- Show how Uniform-Cost Search will find for the two problems below an optimal path from start state S to goal state G (draw the relevant part of the search tree, and indicate in which order nodes are expanded).
- For each node n in the two graphs of the exercise above determine the cost $h^*(n)$ of an optimal solution starting from that node.
- Show how A^* finds the optimal solutions for these two problems when $h(n) = h^*(n)$.

Exercice 4 :

Consider the state space from Figure 1.

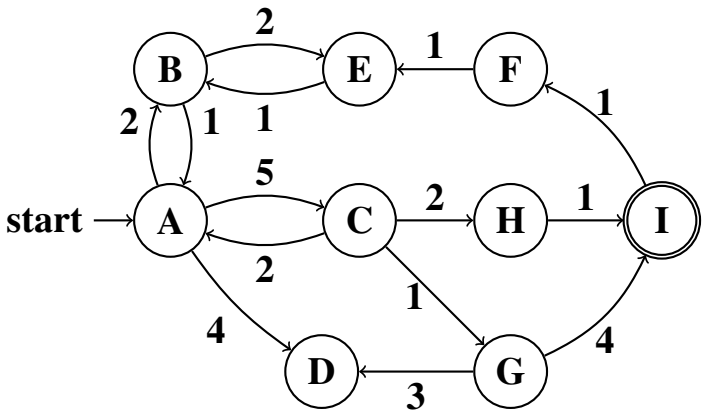


Figure 1: State space.

As a heuristic estimate for a state s, use the minimal number of edges that are needed to reach a goal state from s (or ∞ if s is not solvable, e.g., $h(C) = 2$).

As tie-breaking criteria, if the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first.

(i) Run Uniform-cost-search on this problem. Draw the search graph and **annotate each node with the g value as well as the order of expansion**. Draw duplicate nodes as well, and mark them accordingly by crossing them out. Give the solution found. Is this solution optimal? Justify your answer.

(ii) Run A^* search on this problem. Draw the search graph and **annotate each node with the g and h value as well as the order of expansion**. Draw duplicate nodes as well, and mark them accordingly by crossing them out. Give the solution found by A^* search. Is this solution optimal? Justify your answer.

(iii) Run the hill climbing algorithm, as stated in the lecture, on this problem. For each state, provide all applicable actions and the states reachable using these actions. **Annotate states with their heuristic value. Specify which node is expanded in each iteration of the algorithm.** If the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first. Does the algorithm find a solution? If yes, what is it and is it optimal?

(iv) Could hill-climbing stop in a local minimum without finding a solution? If yes, give an example heuristic $h : \{A, B, ..., I\} \rightarrow \mathbb{N}_0^+ \cup \{\infty\}$ for the state space depicted in Figure 1 that leads hill-climbing into a local minimum, and explain what happens. If no, please explain why.

Exercice 5 :

Consider a variant of the Vacuum Cleaner problem from the lecture where a robot has to clean a 3×3 square room (see Figure 4). There are four possible actions: up, left, right, and down. There is no suck

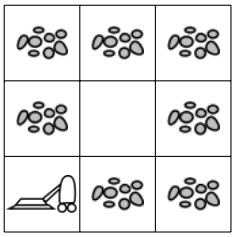


Figure 4: Illustration of the Vacuum Cleaner problem

action since the robot automatically cleans any dirty spot it stands on. Hence, its task is moving around the room and visit all dirty spots. Throughout the exercise, we use Manhattan distance.

Which of the following heuristics are admissible? Why / Why not? Give clear proof arguments. (Formal proofs are not needed.)

(i) $h_1 =$ Number of dirty spots.

(ii) $h_2 =$ Number of clean spots.

(iii) $h_3 =$ Sum of the distances from the robot to all the dirty spots.

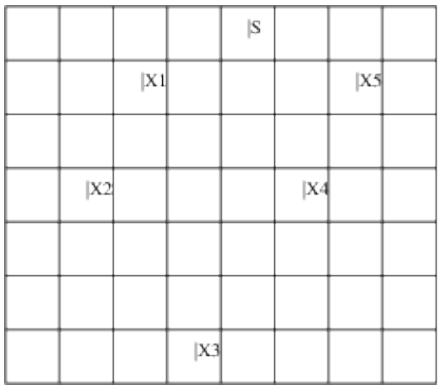
(iv) $h_4 = h_1 + h_2$.

(v) $h_5 =$ Minimum distance from the robot to any dirty spot.

Note: To prove a heuristic admissible, you need to show that it is admissible for every state of the illustrated 3×3 example. To show that a heuristic is not admissible, a counter example is sufficient.

Exercice 6 :

Previous exam question: Consider a robot that can move in the grid below. The cost of moving from one cell to another is equal to the number of steps required for the move, where by a single step the robot can move horizontally or vertically to an adjacent cell (no diagonal steps allowed).



The cells marked X_1, X_2, X_3, X_4, and X_5 are *cells of interest*, each holding one or two of the symbols a, b, and c:

- X_1: (a, c)
- X_2: (b)
- X_3: (b, c)
- X_4: (c)
- X_5: (a)

That is, X_1 holds the symbols a and c. When visiting a cell the robot collects the symbols located at that cell.

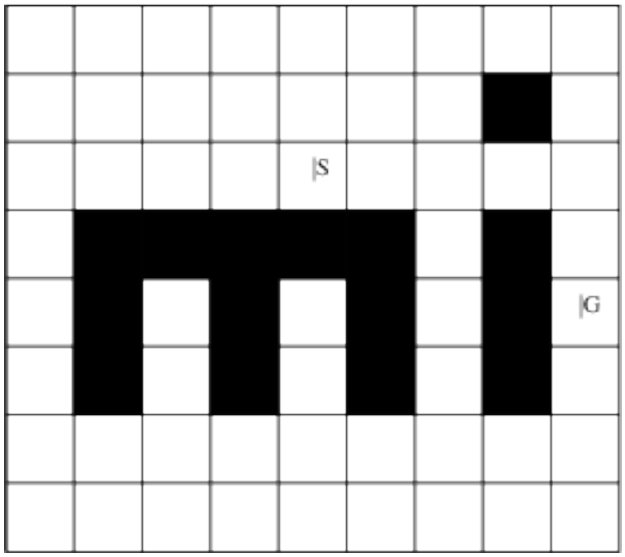
Starting in cell S, the robot's task is to visit cells of interest and collect exactly one copy of each of the three symbols (a, b, c) before returning to cell S. The robot is *not* allowed to move to a cell containing, e.g., an a if the robot has previously visited a cell holding an a.

Determine which of the cells of interest to visit and in which order these cells should be visited by the robot, so that the cost of the path traveled (when starting and ending in cell S) is minimized.

1. Define a suitable state and action representation for this problem.
2. Define a heuristic function that at any given state provides an underestimate of the cost of reaching a goal state (i.e., a state where the robot is located in cell S with the symbols (a, b, c)).
3. Show the search tree for this problem as it would be expanded using A* search.

Exercice 7 :

Previous exam question: Consider the problem of finding a path from position S to G in the grid shown below. You can only move horizontally and vertically and only one step at a time; no step can be made into the shaded areas or outside the grid. The cost of the path between two positions is the number of steps on the path.



1. Number the positions (cells in the grid) in the order in which they are added to the frontier in a uniform-cost search for G starting at S. Assume that the ordering of the operators is *right*, *down*, *left*, and *up*, and that there is multiple path pruning. When multiple lowest-cost paths exists, choose the path first added to the frontier.
2. Define a heuristic function that underestimates the cost of the lowest-cost path.

3. Number the positions (cells in the grid) in the order in which they are added to the frontier according to an A^* search using your heuristic function. Resolve ambiguities using the operator ordering and the procedure above.

NB: When numbering the positions it may be helpful to also annotate the positions with the cost/function values calculated during the search.

Exercice 8 :

Solve Exercise 3.4 in **PM**. Note that the book uses "monotone" instead of "consistent".

Exercice 9 :

You are planning a dinner for three guests. The menu should consist of at least one appetizer, exactly one main dish, and at least one desert (multiple appetizers or deserts are o.k.). You have a list of candidate dishes, and for each guest you know whether they like that dish or not:

Item	Cost	Guest 1	Guest 2	Guest 3
Appetizer 1	5			o.k.
Appetizer 2	5		o.k.	
Appetizer 3	15		o.k.	o.k.
Appetizer 4	30	o.k.		
Main dish 1	90		o.k.	o.k.
Main dish 2	100	o.k.		o.k.
Dessert 1	30		o.k.	
Dessert 2	50	o.k.	o.k.	

Your menu must contain for each guest at least one item that they like. Use A^* to find a minimal cost solution:

- Define the underlying state space problem
- Define a heuristic function that underestimates the true optimal cost function opt.
- Show how A^* will find the minimal cost solution using this heuristic function.
