Exercises for MI

Exercise sheet 2

Thomas Dyhre Nielsen

The exercises below can roughly be grouped into two categories. Those focusing on making a formal state-space representation of a problem and those aimed at analyzing or applying a particular search algorithm. As such, all exercises (except those relying on computer access) could be examples of questions for the exam. The questions are, however, of varying difficulty (with the latter group being the more difficult), and that would also be reflected in the 'number of points' each exercise would give at the exam.

I have marked the possible exam questions with an *.

When you have finished with the exercises, you should continue with the exam sheet from 2013 (Question 1) and 2015 (Question 2), which can be found at the course's home page.

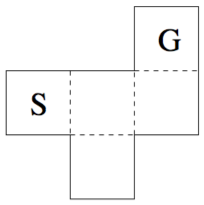
Exercise 1 * Formalize the following problems as state-space problems: define a suitable state space, a start state, set of actions, the action function, and a goal test.

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.

Exercise 2 Solve Exercise 3.2 in **PM**.

Exercise 3 In this exercise we experiment with the Graph Searching applet on `http://www.aispace.org/search/index.shtml`. A robot needs to find a path from the start position S to the goal position G in the following map:



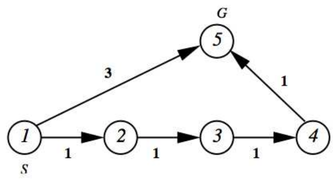
- Create in the Graph Searching applet a graph representing this problem.
- Before you continue, you may want to save your graph into a file, since the applet sometimes experiences crashes that destroy all unsaved input . . .
- Use the 'Fine Step' function under the 'Solve' tab to simulate depth-first search for this problem
- Does depth-first search terminate? If not, is depth-first search unable in principle to find a solution for this problem? How could the implementation in the applet be changed, so that depth-first can run successfully?
- Try depth-first search again with 'Search Options:Pruning:Loop Detection' activated.
- Now try breadth-first search. Is a solution found? What do you observe about the Frontier?

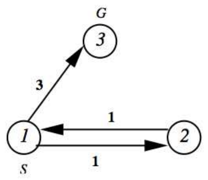
Exercise 4 * Solve Exercise 3.4 in **PM**. Disregard the question about monotone restriction.

Exercise 5 * For the robot navigation problem of the previous exercise:

- 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.

Exercise 6 * Show how Lowest-Cost-First Search will find for the two problems below a minimial cost 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).





Exercise 7 *

- For each node n in the two graphs of the exercise above determine the cost $opt(n)$ of an optimal solution starting from that node.
- Show how A^* finds the optimal solutions for these two problems when $h(n) = opt(n)$.

Exercise 8 * 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.	

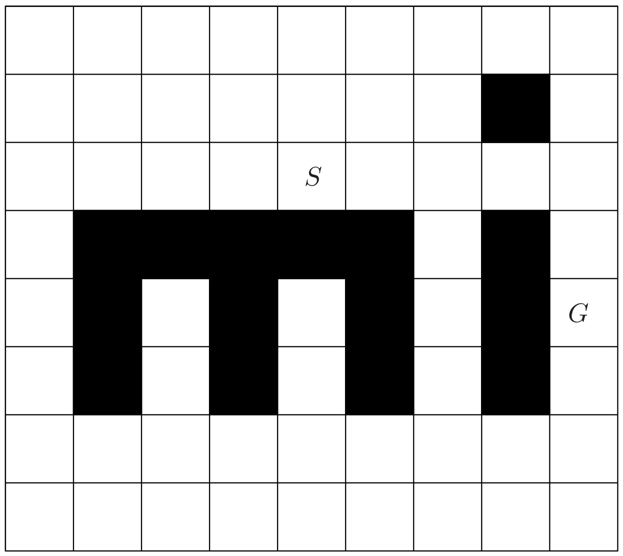
Your menu must contain for each guest at least one item that they like. Use A^* to find a minimal cost solution:

- 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.

Exercise 9

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.

Show how to solve the problem using dynamic programming. Mark each node/ position on the grid with the *cost_to_goal* (see **PM** 3.8.3) and show which path is found.


S
G