Foundations of Artificial Intelligence

M. Helmert
J. Seipp
Spring Term 2019

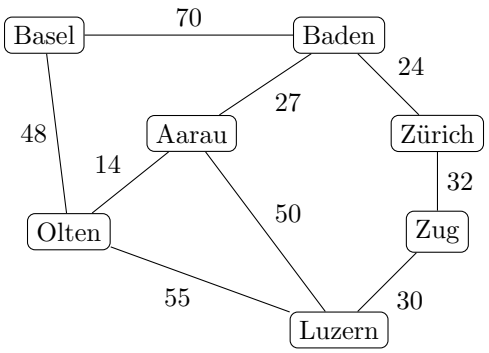
University of Basel
Computer Science

Exercise Sheet 5

Due: April 3, 2019

Exercise 5.1 (2+2 marks)

Consider the following map:



Let the air-line distance between Zug and the other cities be given by the following table:

city	distance
Aarau	44
Baden	38
Basel	83
Luzern	21
Olten	51
Zug	0
Zürich	23

Consider the heuristic that maps each state to its air-line distance to Zug.

(a) Provide the search tree of A* (without reopening) when queried for the shortest path from Basel to Zug. Indicate the order in which nodes are expanded and annotate each node with its f-, g-, and h-values.

(b) Provide the search tree of greedy best-first search (without reopening) when queried for a path from Basel to Zug. Indicate the order in which nodes are expanded. Compare the result to the result of (a).

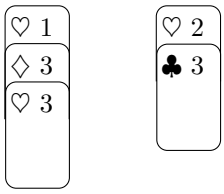
Exercise 5.2 (3+3+2 marks)

The task in this exercise is to write a software program. We expect you to implement your code on your own, without using existing code you find online. If you encounter technical problems or have difficulties understanding the task, please let us know. For this exercise, you only have to change the files `FreecellStateSpace.java` and `AstarSearch.java`.

(a) We have extended the interface `StateSpace` with a method that returns a heuristic value for the given state (the method is called `public int h(State s)`). The `FreecellStateSpace` class already contains the skeleton of this method. Implement a heuristic that computes the sum of

- the number of cards that have not been moved onto a foundation pile yet, and
- the number of cards that have to be moved to another (non-foundation) pile at least once because there is a card of the same suit and lower rank below it in the same pile.

To understand how the heuristic is computed, consider the following example state s for a Freecell instance with cards up to rank 3:



All cards that are not depicted have already been moved to the corresponding foundation pile. The first part of the proposed heuristic counts the number of cards that are not yet on a foundation pile, which evaluates to 5. The second part evaluates to 1, as there is only one card (♡ 3) that has to be moved onto another non-foundation pile because there is a card of the same suit and lower rank below it (♡ 1). The heuristic value of s is therefore $h(s) = 6$.

(b) Implement A* without node reopening in `AstarSearch.java`. Make sure that the value of the member variable `expandedStates` of the parent class `SearchAlgorithmBase` is updated correctly.

(c) Test your implementation on the example problem instances provided in the tarball on the course website. Set a time limit of 10 minutes and a memory limit of 2 GB for each run. On Linux, you can set a time limit of 10 minutes with the command `ulimit -t 600`. Running your implementation on the first example instance with

java -Xmx2048M AstarSearch freecell freecell_inst_1

sets the memory limit to 2 GB. You are free to use higher memory limits. In any case, mention the limit in your solution.

Report runtime, number of node expansions, solution length and solution cost for all instances that can be solved within the given time and memory limits. For all other instances, report if the time or the memory limit was hit.

Important: Solutions should be submitted in groups of two students. However, only one student should upload the solution. Please provide both student names on each file and each page you submit. We can only accept a single PDF or a ZIP file containing *.java or *.pddl files and a single PDF.