Foundations of Artificial Intelligence

M. Helmert, G. Röger
J. Seipp, S. Sievers
Spring Term 2017

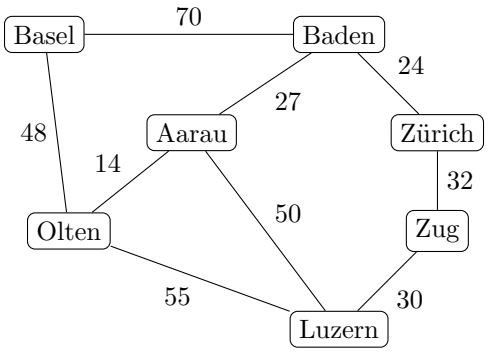
University of Basel
Computer Science

Exercise Sheet 5

Due: April 5, 2017

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 (such as examples you find online). If you encounter technical problems or have difficulties understanding the task, please let us – the tutor or assistants – know *sufficiently ahead of the due date*. Please also read the *hints* below.

The objective of second-to-last week's Exercise 3.2 was to implement uniform cost search for 4-pegs Tower of Hanoi state spaces. This time, you will be asked to work with informed search algorithms. To this end, 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)`). Note that this time, we consider the more common variant of the problem that uses unit costs, i.e., all actions cost 1. You can find the code on the website of the course.

(a) Implement the following heuristic for the 4-pegs Tower of Hanoi state space. Given a state, the heuristic first determines the largest *unfinished* disk d, i.e., the largest disk *not* on its final position because it is either not on the goal peg or because not all larger disks are below it on the goal peg. The heuristic value for any unfinished disk i is $2^{d-i} + x$. The added term x is 0 if the disk is on a non-goal peg and 1 if it is on the goal peg. The overall heuristic value is the sum of the individual heuristic values for all unfinished disks.

Example: Consider the problem with 4 disks. If disk 3 is on the goal peg and all other disks are on any non-goal pegs, then the largest unfinished disk is 2, and hence the heuristic value

is $2^{2-2} = 1$ for disk 2, $2^{2-1} = 2$ for disk 1 and $2^{2-0} = 4$ for disk 0, and hence the overall heuristic value is $1 + 2 + 4 = 7$. If disk 1 is on the goal peg as well, the heuristic value is increased by 1.

The class `FourPegsTowerOfHanoi` already contains the skeleton of the method `public int h(State s)` which you should implement. As an example, we implemented the *blind* heuristic that assigns 0 to goal states and 1 to all other states.

(b) Implement A* without node reopening in a file `AstarSearch.java`. To do so, you may inherit from the provided class in `SearchAlgorithmBase.java`, which also provides code to measure search statistics.

Hint: Note that as for Exercise 3.2, a possible implementation of the open list (yet certainly not the only one) is to use `java.util.PriorityQueue` and one possibility for the closed list is to use a `java.util.HashSet`. Depending on your implementation, it is furthermore possible that you have to implement comparison and/or hashing methods (`equals` and `hashCode`) for all classes that are used to describe a state.

(c) Test your implementation on the example problem instances you can find on the 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 tower-of-hanoi 4toh_inst_5
```

sets the memory limit to 2 GB. If the RAM of your computer is 2GB or less, set the memory limit to the amount of available RAM minus 256 MB instead. You are also free to use higher memory limits. In any case, describe in your solution how much RAM was used.

Report runtime and number of expanded nodes 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 violated.

Furthermore, do the same evaluation with the blind heuristic which is provided in the template of the method `public int h(State s)`. Knowing that A* with the blind heuristic is an uninformed algorithm (similar to uniform cost search or breadth-first search when using unit cost problems), what can you observe when you compare the results of using A* with the blind heuristic and the heuristic you implemented?

Important: The exercise sheets can be submitted in groups of two students. Please provide both student names on the submission. Please create a PDF for exercise 5.1 and a directory containing the Java files for exercise 5.2. Afterwards, please create a zip file containing the PDF and the directory and submit it.