Foundations of Artificial Intelligence

M. Helmert
S. Eriksson
Spring Term 2021

University of Basel
Computer Science

Exercise Sheet 4

Due: March 31, 2021

Exercise 4.1 (1+1 marks)

Which of the search algorithms shown in slide 35 of the printout version of chapter 12 would you use for the following problems? For breadth-first and uniform cost also consider whether to use the tree or graph variant. Justify your answer in two to three sentences.

(a) route planning

(b) searching a specific file in a large file system which does not have any symbolic links

Hint: You can assume that the file exists at most once and that there is only one action type `move-to-subdirectory` *with cost 1.*

Exercise 4.2 (2 marks)

Consider a pancakes problem with n pancakes where a state $s = \langle p_1, \ldots, p_n \rangle$ represents the sizes of the pancakes from top to bottom. We additionally define $p_{n+1} = n + 1$ for all states s. Intuitively this means we interpret the table on which the pancakes stand as an additional (immovable) pancake of size $n + 1$. Now consider the following heuristic:

$$h(s) = \sum_{i=1}^{n} d(p_i, p_{i+1})$$

with

$$d(x, y) = \begin{cases} 0 & \text{if } |x - y| = 1 \\ 1 & \text{otherwise} \end{cases}$$

Which of the four properties *safe*, *goal-aware*, *admissible* and *consistent* does h satisfy? Justify your answer for each property.

Exercise 4.3 (1 mark)

Can a heuristic be safe and goal-aware, but not admissible? Justify your answer.

Exercise 4.4 (4+1 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 create a new file `AstarSearch.java`.

(a) Implement A* without node reopening in a new file `AstarSearch.java`. Your class must inherit from `SearchAlgorithmBase`. Make sure that the value of the member variable `expandedNodes` is updated correctly. You can access the heuristic value of a state through a new method in the `StateSpace` interface called `public int h(State s)`.

(b) We altered `PancakeStateSpace` to implement the heuristic described in Exercise 4.2. Test your implementation of `AstarSearch` on the example problem instances provided in the instances directory. 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 pancakes instances/pancakes_prob_01
```

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. In any case, describe in your solution how much RAM was used.

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.

Submission rules:

- Create a single PDF file (ending .pdf) for all non-programming exercises. If you want to submit handwritten parts, include their scans in the single PDF. Put the names of all group members on top of the first page. Use page numbers or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).
- For programming exercises, create only those Java textfiles (ending .java) required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it!
- For the submission, you can either upload the single PDF or prepare a ZIP file (ending .zip, .tar.gz or .tgz; not .rar or anything else) containing the single PDF and the Java textfile(s) and nothing else. Please do not use directories within the ZIP, i.e., zip the files directly.