Foundations of Artificial Intelligence

M. Helmert — University of Basel
S. Eriksson — Computer Science
Spring Term 2021

Exercise Sheet 3

Due: March 24, 2021

Exercise 3.1 (1+1+1 marks)

Show that nontrivial state spaces with the following properties exist by providing a corresponding state space and explaining why it satisfies the required property. Your state space must have at least 5 nodes reachable from the initial state and all solutions must have at least length 2.
Note: You can omit action names and instead directly use action costs as labels in the graph.

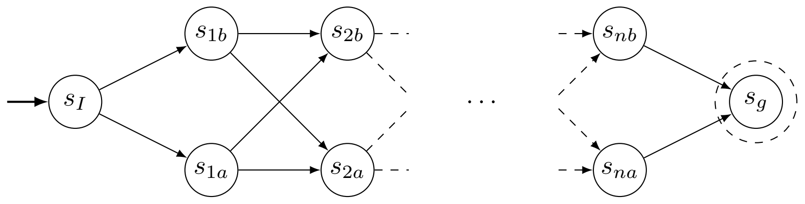
(a) Tree search and graph search expand the same amount of nodes (when using the same expansion strategy).

(b) Breadth-first search expands $|S| - 1$ nodes.

(c) Breadth-first search finds a suboptimal solution.

Exercise 3.2 (1+1 marks)

Consider a breadth-first search in the following state space:



(a) How many search nodes are at least inserted into the open list until the agent finds a plan if duplicate detection is not used? Give an answer as a function of n and justify your answer.

(b) How does that answer differ if duplicate detection is used?

Exercise 3.3 (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 (such as examples you find online) except for what is provided by us. If you encounter technical problems or have difficulties understanding the task, please let us – the tutor or assistant – know *sufficiently ahead of the due date.*

The archive required for this exercise will be uploaded to the course website and to the ADAM instruction files only on Thursday morning, since it contains the solution to Exercise 2.4.

(a) Implement *uniform cost search.* Only create a single new file called `UniformCostSearch.java`. The new class `UniformCostSearch` must derive from `SearchAlgorithmBase`. Make sure that the value of the member variable `expandedNodes` is updated correctly. A possible implementation of the open list (yet certainly not the only one) is to use a `java.util.PriorityQueue` and one possibility for the closed list is to use a `java.util.HashSet`.

Important: Do not implement breadth-first search, even though all actions in the Pancake problem have equal cost. Your algorithm should produce optimal solutions for any black-box state space, which means it cannot know whether all actions have the same cost.

(b) 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 UniformCostSearch 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 violated.

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.