Foundations of Artificial Intelligence

M. Helmert
J. Seipp
Spring Term 2019

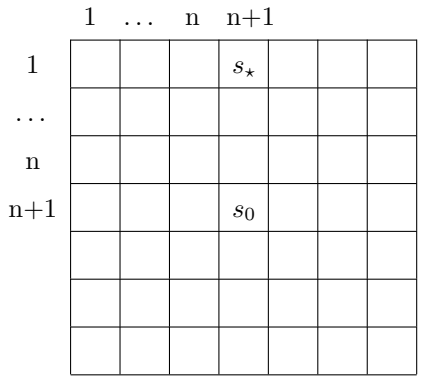
University of Basel
Computer Science

Exercise Sheet 3

Due: March 20, 2019

Exercise 3.1 (1+1+1 marks)

Consider a search problem on a square grid of size $2n+1$ (for $n \in \mathbb{N}$). An agent is initially located in state s_0 in the grid cell with coordinates $(n+1, n+1)$ and has the goal to reach state $s_\star$ located in the grid cell with coordinates $(n+1, 1)$. The agent has the possibilities to move north, east, south, and west in the grid if there is a grid cell in the corresponding direction (otherwise, the corresponding action is not applicable). We assume that the agent uses breadth-first search to compute a solution.



(a) How many search nodes are at least inserted into the open list until the agent finds a solution using tree search? Give an answer as a function of n and justify your answer.

(b) How does that answer differ when using graph search?

(c) Compare the number of search nodes that is inserted into the open list in the last search layer that is created completely for grids with $n = 10$ and $n = 20$ for tree search and graph search and discuss the results.

Exercise 3.2 (6+3 marks)

The task in this exercise is to write a software program. We expect you to implement your code without using existing code you find online. If you encounter technical problems or have difficulties understanding the task, please let us know.

The objective of last week's Exercise 2.4 was to implement a blocks world variant with a fixed number of towers of a given maximal capacity. We have made some additional changes, resulting in a closely related problem: a variant of the game *Freecell* (`https://en.wikipedia.org/wiki/FreeCell`) where

- towers are called *piles*,
- blocks are replaced by *cards* with differing *suits*, *ranks* and *colors*,
- moving the top card of a pile onto another is only allowed if the destination pile is empty or if the top card of that pile has a different color (in contrast to the usual game rules, we ignore ranks when moving cards),

- an additional *discard action* allows the removal of a card from the game (which is our way to model that a card is put on a foundation pile) if it has the lowest rank among all cards that share the same suit, and
- as opposed to the usual game rules, moving a card between piles incurs a cost that is equal to the rank of the card plus one (we model ranks as numbers from $\mathbb{N}_0$).

You can find the code for the Freecell state space on the website of the course. (Since the Freecell state space is built on the solution for last week's exercise, we will make the code available on Thursday, March 7.)

(a) Implement *uniform cost search* to solve Freecell problems. You should only need to create a single new file called `UniformCostSearch.java`. The new `UniformCostSearch` class must derive from `SearchAlgorithmBase`. 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`.

(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 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 violated.

Hint: Please test your solution. Your code must compile and we must be able to run it.

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.