# 02285 - Assignment 1

Emil Rask Maagaard - s142955

February 9, 2015

# 1 Introduction

# 2 Exercise 1 (Search strategies)

**Benchmark**

| Level | Client | Time | Memory used | Solution length | Nodes explored |
|---|---|---|---|---|---|
| SAD1 | BFS | 0.07 s | 11.52 MB | 19 | 78 |
| SAD1 | DFS | 0.05 s | 8.96 MB | 27 | 44 |
| SAD2 | BFS | DNF | DNF | DNF | DNF |
| SAD2 | DFS | 4.57 s | 509.2 MB | 5781 | 6799 |
| custom | BFS | DNF | DNF | DNF | DNF |
| custom | DFS | 0.06 s | 11.52 MB | 45 | 60 |

It can seem like there are many more solutions, because of the now 4 A's, and not just one. However, only one of the A's can be pushed into the goal due to the "Sobako ??????" rule. - So even with 8GB of RAM allocated it is not possible.

Quite a few nodes have been explored.

Level: The custom designed level is a derivation of SAD2, where some of the possible spaces are filled to limit the possible movements. The possible A's are also moved a little bit.

# 3 Exercise 2 (Optimisations)

**Sub title**

| Level | Client | Time | Memory used | Solution length | Nodes explored |
|---|---|---|---|---|---|
| SAD1 | BFS | 0.01 s | 5.12 MB | 19 | 78 |
| SAD1 | DFS | 0.01 s | 5.12 MB | 27 | 44 |
| SAD2 | BFS | DNF | DNF | DNF | DNF |
| SAD2 | DFS | 1.23 s | 23.10 MB | 5781 | 6799 |
| custom | BFS | DNF | DNF | DNF | DNF |
| custom | DFS | 0.01 s | 5.12 MB | 45 | |

1) ????  2) The number of columns are used when instantiating the node Benchmark:

b) “The locations of boxes in a level are not static. Explain which data structure would allow you to save memory in most levels, while still offering good performance when it comes to lookup. In terms of running time, what would the impact of such a modification be on isGoalState() and getExpandedNodes()”

???

c)

d)

Something about the box structure not being fitting for the case, since there are very few boxes compared to the possible large 2-dimensional arrays they are stored in.

# 4 Exercise 3 (Heuristics)

### Sub title

a) “Write a best-first search client by implementing the StrategyBestFirst class. The Heuristic argument in the constructor must be used to order nodes. As it implements the Comparator<Node> interface it integrates well with the Java Collections library. Make sure you use an appropriate data structure for the frontier, and state which one you have used.”

The frontier is a List. When a node (the one with the lowest f(n)) is poped from the frontier, the frontier is cleared in order to make a new frontier based on the chosen node. The remaining nodes in the frontier, before clearing it, is stored as an old frontier - in order to go back to that, if the new one proves to be a dead end.

Actually I use two frontiers.

| Level | Evaluation | Time | Memory used | Solution length | Nodes explored |
|---|---|---|---|---|---|
| SAD1 | A* | 0.01 s | 5.12 MB | 19 | 35 |
| SAD1 | WA* | 0.01 s | 5.12 MB | 19 | 19 |
| SAD1 | Greed | 0.01 s | 5.12 MB | 19 | 19 |
| SAD2 | A* | 0.01 s | 5.12 MB | 19 | 19 |
| SAD2 | WA* | 0.01 s | 5.12 MB | 19 | 19 |
| SAD2 | Greedy | 0.01 s | 5.12 MB | 19 | 19 |
| custom | A* | 0.01 s | 5.12 MB | 19 | 19 |
| custom | WA* | 0.01 s | 5.12 MB | 19 | 19 |
| custom | Greedy | 0.01 s | 5.12 MB | 19 | 19 |
| FireFly | A* | DNF | DNF | DNF | DNF |
| FireFly | WA* | DNF | DNF | DNF | DNF |
| FireFly | Greedy | DNF | DNF | DNF | DNF |
| Crunch | A* | DNF | DNF | DNF | DNF |
| Crunch | WA* | DNF | DNF | DNF | DNF |
| Crunch | Greedy | DNF | DNF | DNF | DNF |

b) Heuristic function The heuristic functions functions calculates the distance from the agent to the box and from the box to the goal. The goal and box coordinates are found in the pre-processing.

Issue when multiple of same box or goal - because it is stored in a map with character as key, where keys are unique.

Benchmark:

c) Admissibility:

The heuristic function is admissible, as it calculates the Pythagorean distance between agent, box and goal without regarding any possible obstacles.

d) Competition levels:

# 5 Conclusion