Introduction to Artificial Intelligence

Exercise Sheet 2

Handed out: 20.11.2020 Prof. Dr. Susanne Biundo-Stephan
Due: 04.12.2020 Conny Olz

Exercise 1 (4 points)

Implement a program to solve the Labyrinth Problem (as given on the last exercise sheet) by using Breath First Search (BFS).

Read the problem definition from the *stdin* (not from a file). In Linux, you can test your program by using the following line of Bash code:

```
java -jar labyrinth.jar < lab.txt
```

The definition describes a $n \times m$ labyrinth that includes the following characters:

- Space: a field the agent might go to (i.e. passable by the agent)
- Period: goal position (passable by the agent)
- @ symbol: starting position (passable by the agent)
- # symbol: not passable by the agent

The following encoding describes the labyrinth given on the last exercise sheet:

```
#####
# . #
#   #
# # #
#@  #
#####
```

Your program must return the *number of steps* needed to reach the goal. This must be the only output of your entire program. In the version that is handed in, do not output any debug information, etc. Just output a single line that contains a single number and end the line with a line break.

Introduction to Artificial Intelligence

Exercise Sheet 2

Handed out: 20.11.2020 Prof. Dr. Susanne Biundo-Stephan
Due: 04.12.2020 Conny Olz

Exercise 2 (6 points)

Implement a program that solves Sokoban[1] instances by using Depth First Search (DFS) and Breath First Search (BFS). If you want to, you can reuse code from the last exercise.

Read the problem definition from the *stdin* (not from a file). In Linux, you can test your program by using the following line of Bash code:

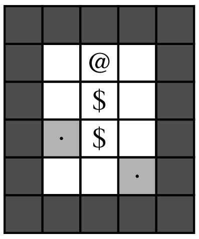
```
java -jar sokoban.jar < soko.txt
```

If you have solved the instance, output a sequence of the characters L, R, U, D describing the moves the agent needs to do to solve it. I.e., when the agent performs the moves one after the other in the initial situation, all boxes are placed on a goal position. Do *not* separate the characters by spaces or commas (a solution to the given problem might e.g. be RDDLRUULLDRURDD).

A Sokoban instance is encoded in the following way [2]

- Space: a field the agent might go to (i.e. passable by the agent)
- $ symbol: current position of a box (passable by the agent)
- Period: goal position for a box (passable by the agent)
- @ symbol: starting position (passable by the agent)
- # symbol: not passable by the agent

The following figure shows a Sokoban instance and its encoding in the given format:



```
#####
# @ #
# $ #
#.$ #
#  .#
#####
```

[1] http://de.wikipedia.org/wiki/Sokoban
[2] For a more detailed explanation, see http://sokobano.de/wiki/index.php?title=Level_format