Artificial Intelligence

A.Y. 2016/2017

Daniele Nardi and Federico Nardi

Homework 1

1 Deadline

The homework must be delivered before *November, 6th 2016 23:59 UTC*. **The system will not accept neither new submissions nor modifications of already submitted material, when deadline is expired**.

In order to submit your solution, go to the course website

`https://elearning2.uniroma1.it/course/view.php?id=4478`

and search for **Homework-Search**.

2 Preliminaries

- **AIMA Project**
  - Download the **AIMA project** from the Berkeley University at `http://aima.cs.berkeley.edu/code.html`. We refer to Java in the homework description but, in case you choose a different programming language you should provide an equivalent solution of the problem
  - Follow the instructions in the README.md to compile the code and to run the AIMA applications and demos

- **Home Code**
  - Download the **Home code** provided on the course website (Homework-Search section).

3 Homework

Part 1

1.0 In order to familiarize with the AIMA framework go to:
`aima-gui/src/main/java/aima/gui/applications/`
and test the various example programs. In particular, solve the problems by using all available uninformed, heuristic and local search algorithms to test their capabilities.

1.1 Consider the *NQueens* problem implemented in the `NQueensDemo.java` demo. Plot a graph in which you show the capabilities of each search algorithm in solving the problem with an increasing size of the state space. In particular, plot the number of nodes generated as well as the time needed by the algorithm to find the solution for solving the 8-, 10- and 12-Queens problem.

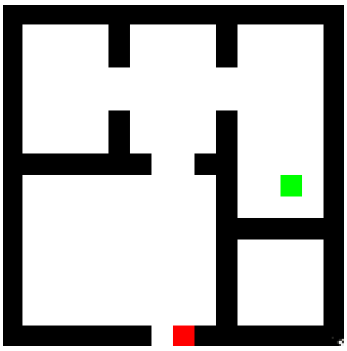
1.2 Consider the *Eight Puzzle* problem implemented in the `EightPuzzleApp.java` application. Choose the **Medium** configuration of the board and solve the problem with A^* by using the **MisplacedTileHeuristic** and the **ManhattanHeuristic**. Then, expand the state space by creating a 4×4 board and a start configuration and repeat the previous procedure. Evaluate, for each run, the number of nodes generated and the time needed to find the solution.

The outcome of this section of the homework should be a commented report of the results (Section 1.1) and plots (Section 1.2) obtained using all available search algorithms on the chosen applications and demos.

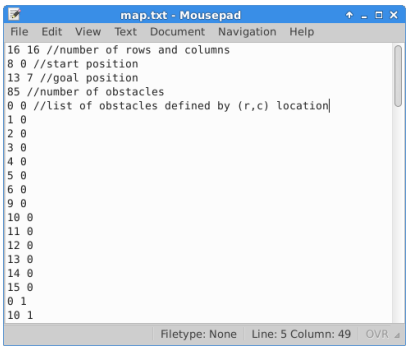
Part 2

Consider the problem of a domestic robot that has to go from a *start* position to a *goal* position avoiding obstacles. The domestic environment is defined on an *Occupancy Grid* of $R \times C$ cells, as shown in Fig. 1(a), where the black

cells represent walls and furniture, i.e. they cannot be traversed, while white cells are the admissible positions. The cell (r, c) is located in row r and column c, with $1 \leq r \leq R$ and $1 \leq c \leq C$.



(a) An example of occupancy grid



(b) The corresponding textual description, in your file avoid comments

Figure 1: Domestic robot problem.

In order to navigate the environment the robot can move in four directions (`UP`,`DOWN`,`RIGHT`,`LEFT`), as long as the resulting position is not outside of the grid or on a black cell. A *solution* to the problem is a sequence of cells that describes the path from *start* to *goal*. Use the `AIMA` framework and the `Home` code to formulate the problem:

2.1 Define a 16×16 occupancy grid by drawing a map of your home (or room).

2.2 Convert the grid in a text file by using the format specified in Fig. 1(b).

2.3 Use the code in the `Home` package to load the text file containing the map and store it in a matrix.

2.4 Use the `AIMA` framework to formulate the problem by completing the **Environment** class and extending the **ActionsFunction**, **ResultFunction** and **GoalTest** classes.

2.5 Solve the problem using *Depth First Search* and A^* with an admissible heuristic. As a result, you will get the list of actions that bring the robot to the *goal*, print the solution path by adding blue cells to the grid and use the provided function to visualize the results. Then, increase

the resolution of the grid as explained in the code documentation and repeat the previous procedure.

The outcome of this section of the homework should be the source code that implements the formulation and the solution of the problem along with a brief report that explains the conducted experiments and the corresponding results.

The solution of the homework must be zipped in an archive named *studentName_homework1*.zip and updated on the website.