DEEC
DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Instituto Superior Técnico

Artificial Intelligence and Decision Systems (IASD)
Lab classes, 2014/2015

Assignment #1

Version 1.0 - September 11, 2014

The goal of this assignment is to develop a Python program to solve a labyrinth puzzle, using Search methods. The program should include: i) at least one uninformed search method; and ii) at least one informed search method with an heuristic specially designed to this problem.

The labyrinth is full of doors that will often block the way, and switches that can be used to open and close the doors. Every door and switch is marked by a letter (or a pair of letters) and/or a color. Pressing a switch toggles all doors that have the same marking.

The objective is to find the way through a labyrinth from an initial cell until the finish cell, both specially marked in the labyrinth definition.

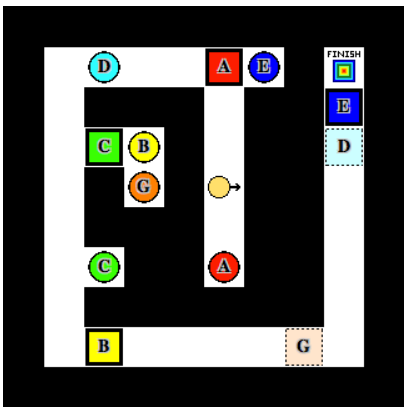

Figure 1: Example of a labyrinth. The small yellow circle with an arrow is the agent. The bigger circles are switches. The squares are doors, where a full square is a closed door, whereas a dashed square is an open door (for instance, door "B" is closed but door "G" is open). The black cells are walls and the white cells are corridors.

Rules Summary

- The agent can freely move in corridors, *i*.e., it can move from a white cell to an adjacent white cell or a switch cell. To move the agent, there are four commands: "**Up**", "**Down**", "**Left**", and "**Right**".

- The agent cannot move to a black cell.

- If the agent is in a switch cell it can issue the command "**Push**" in order to toggle all doors with the same marking of the switch. "Toggle the doors" means that all closed doors become open and vice versa.

- When in a switch cell, the agent is not obliged to use (push) the switch.

- If the agent is near a door cell, it can only move to that cell if the door is open.

- For each door there is only one switch, but for each switch there may be more than one door.

- The problem is considered solved when the agent reaches the finish cell.

Input File

The initial state of the labyrinth to be solved should be defined in an input file. For the purpose of this project, it is assumed that the problem does not have more than 100 switches and no more than 200 doors (no more than 100 open doors and 100 closed doors). The structure of the input file should be as follows:

1. the initial configuration of the labyrinth is defined through a $L \times C$ matrix of numbers;

2. the first line of the input file has two integers representing the values of L and C;

3. the next L lines in the input file, each one with C integers, define the mentioned matrix;

4. black cells are represented by a "0" (zero);

5. white cells are represented by a "1" (one);

6. the initial position of the agent is represented by a "2" (two);

7. the goal cell (finish position) is represented by a "3" (three);

8. switch cells are represented by a number between 100 and 199;

9. closed door cells are represented by a number between 200 and 299;

10. open door cells are represented by a number between 300 and 399;

11. the marking of a switch/door is univocally defined by the last two digits of the switch/door number (for instance, if there is a switch represented by the number 1xy, the corresponding door(s) should be represented either by 2xy (in which case is(are) closed) or 3xy (open)).

12. if needed you may assume that the limits of the labyrinth are always represented in the input file by zero numbers.

For instance, the labyrinth presented in Figure 1 is described by the following input file:

```
10  10
 0  0    0    0  0    0    0    0    0  0
 0  1  103    1  1  200  104    0    3  0
 0  1    0    0  0    1    0    0  204  0
 0  1  202  101  0    1    0    0  303  0
 0  1    0  106  0    2    0    0    1  0
 0  1    0    0  0    1    0    0    1  0
 0  1  102    0  0  100    0    0    1  0
 0  1    0    0  0    0    0    0    1  0
 0  1  201    1  1    1    1  306    1  0
 0  0    0    0  0    0    0    0    0  0
```

Output File

The output of the solver should also be given in a file, with the number of actions (movements or push), and the sequence of the agent's actions. The actions should be represented by the following letters $\{U, D, L, R, P\}$.

For instance, a solution for the example labyrinth can be described by:

```
50
DDPUUUUURPLLLLLDDDDDRPLUUURRPLLDDDDDRRRRRRRRUUUUUUU
```

Assignment Goals

1. Develop a Python program for solving this type of labyrinth problems using Search methods. The solver must include at least one uninformed search method, and one informed search method with an heuristic function.

   In your program the domain-independent part *should be* explicitly isolated from the domain-dependent one.

   Suggestion: Use the General Search to develop a generic search algorithm (domain-independent part) and then particularize it to the labyrinth problem (domain-dependent part).

   Note 1: All code should be adequately commented. In particular, for each function you should include comments describing the function's purpose, inputs and outputs.

   Note 2: Do not forget to identify the version of Python used.

2. The answers to the following points should be presented in a short report, delivered together with the source code.

   (a) Describe how do you represent the problem state, the operator(s), the initial state, the goal state, and the path cost;

   (b) Identify and justify each uninformed and informed search algorithms implemented;

   (c) Describe how did you separate the domain-independent part of the solver from the domain-dependent one;

   (d) Describe and justify the heuristic function(s) developed;

   (e) Compare quantitatively and comment the performance of the different search techniques implemented.

The deliverable of this Lab Assignment consists of two components:

- the Python source code
- the short report (pdf format) with no more than 2 pages.

Deadline*: 24h00 of 17-Out-2014, by email to lmmc@isr.ist.utl.pt*
(Projects submitted after the deadline will be penalized.)