LINGI2261: Artificial Intelligence
Assignment 2: Solving Problems with Informed Search

François Aubry, Gael Aglin, Yves Deville
October 2018

⚠ Guidelines

- This assignment is due on **Wednesday 24th October 2018, 6:00 pm**.

- *No delay* will be tolerated.

- *Document* your source code (at least the difficult or more technical parts of your programs). Python docstrings for important classes, methods and functions are also welcome.

- Indicate clearly in your report if you have *bugs* or problems in your program. The online submission system will discover them anyway.

- Copying code or answers from other groups (or from the internet) is strictly forbidden. Each source of inspiration must be clearly indicated.

- Source code shall be submitted on the online *INGInious* system. Only programs submitted via this procedure will be graded. No report or program sent by email will be accepted.

- Respect carefully the *specifications* given for your program (arguments, input/output format, etc.) as the program testing system is *fully automated*.

ⓘ Deliverables

- The following files are to be submitted on *INGInious* inside the *Assignment 2* task(s):

  - `report_A2_group_XX.pdf`: Answers to all the questions in a single report, named. Remember, the more concise the answers, the better.

  - The file `pacmen.py` containing your Python 3 implementation of the Pacmen problem solver. Your program should take the paths to the instance files as only argument. The search strategy that should be enabled by default in your programs is A^* **with your best heuristic**. Your program should print the solution to the standard output in the format described further. The file must be encoded in **utf-8**.

✎ Anti plagiat charter

As announced in the class, you'll have to electronically sign an anti plagiat charter. This should be done ***individually*** in the **INGInious** task entitled *Assignment 2: Anti plagiat charter*. Both students of a team must sign the charter.

1 Search Algorithms and their relations (3 pts)

1.1 $A^\star$ versus uniform-cost search

Consider the maze problem given on Figure 1. The goal is to find a path from 🚶 to € moving up, down, left or right. The black positions represent walls. This question must be answered by hand and doesn't require any programming.

> ✒ **Questions**
>
> 1. Give a consistent heuristic for this problem. Prove that it is consistent. Also prove that it is admissible.
>
> 2. Show on the left maze the states (board positions) that are visited during an execution of a uniform-cost graph search. We assume that when different states in the fringe have the smallest value, the algorithm chooses the state with the smallest coordinate (i, j) $((0, 0)$ being the bottom left position, i being the horizontal index and j the vertical one) using a lexicographical order.
>
> 3. Show on the right maze the board positions visited by $A^\star$ graph search with a manhattan distance heuristic (ignoring walls). A state is visited when it is selected in the fringe and expanded. When several states have the smallest path cost, this uniform-cost search visits them in the same lexicographical order as the one used for uniform-cost graph search.
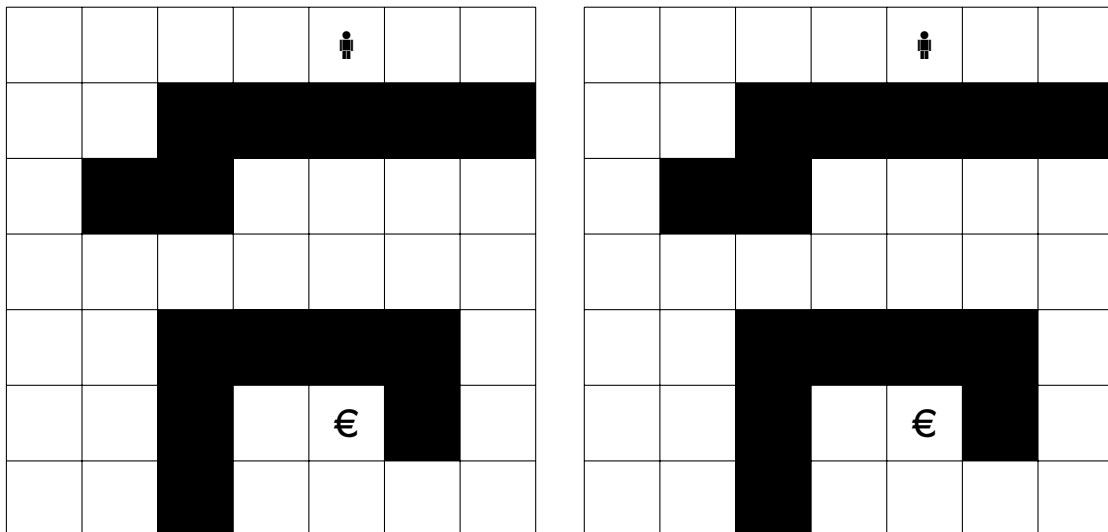

Figure 1

2 Pacmen problem (17 pts)

The problem you will solve for this assignment is the Pacmen problem. Again, the search procedures from aima-python3 will help you implement the problem! The Pacmen problem we consider is a simplified version of the online Pacman game version available at `http: //www.zebest-3000.com/jeux/jeu-256.html`. In this simplified version, there is no ghost. The game consists in a movable pacman, which must be moved to eat some foods in a maze. The pacman can move either horizontally or vertically but it cannot pass through the maze walls. The game is won as soon as all the foods in the maze are eaten.

Furthermore, like the name can let think, our version of the game introduces some new feature. We can have some mazes with many pacmen. In each step, each pacman moves into some adjacent positions. They can be at the same position in the maze. The objective is to eat all the foods in the maze using the pacmen with optimal number of moves. For this assignment, we will consider the number of pacman will be at most 3.

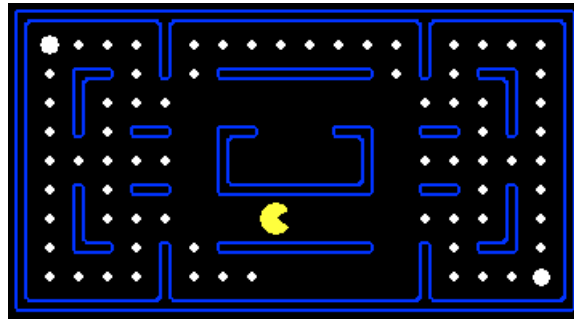Figure 2 shows an instance of the simplified version of our Pacmen.

Figure 2: The Pacmen game with single agent.

Input and output format

Your program must take one argument : the initial state (e.g. `i01`). We use ASCII symbols in order to represent a state. Figure 3 shows an example of initial state, for the test instance *i01*.

```
###############
#             @ #
# @ x x    x    #
#   x       x   #
#   x x x x $ #
#   x       x   #
#   x           #
#               #
###############
```

Figure 3: The initial instance file `i01`.

The solid wall structures are represented by x's. Pacmen are indicated by $ whereas @ symbols represent foods. The # symbols just delimit the environment borders. The output of the program should be a minimal sequence of every intermediate grid, represented in the same way, starting with the initial state and finishing with the goal state. Figure 4 gives an example of a valid solution for instance *i01*. This solution, obtained by executing python3

pacmen.py instances/i01, is also optimal that is, involves a minimal number of actions (or moves).
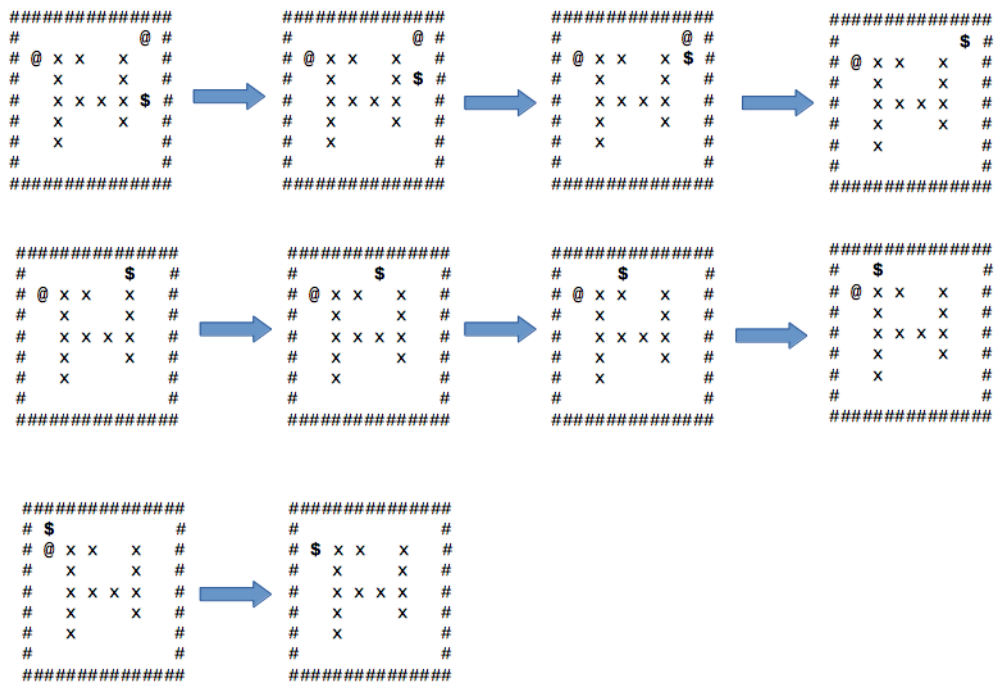

Figure 4: A possible optimal solution output for instance *i01*.

Be careful with the solution output format ! Your solver must respect the exact same format as in Figure 4. For this, do not modify the output printing code in the example file pacmen.py and do not print anything added to it.

Questions

1. Model the Pacmen problem as a search problem; describe:

   - States

   - Initial state

   - Actions / Transition model

   - Goal test

   - Path cost function

2. What is the maximum branching factor for this problem considering there are k pacmen in the maze ?

3. Give an admissible heuristic for a case of one pacman and n foods. Prove that it is admissible. What it is complexity ?

4. Give an admissible heuristic for a case of k pacmen and n foods. Prove that it is admissible. What it is complexity ?

5. **Implement** the problem with k pacmen and n foods. Extend the *Problem* class and implement the necessary methods and other class(es) if necessary.

6. **Experiment**, compare and analyze informed (*astar_graph_search*) and uninformed (*breadth_first_graph_search*) graph search of aima-python3 on the 10 instances of Pacmen provided. Report in a table the time, the number of explored nodes and the number of steps to reach the solution.

   Are the number of explored nodes always smaller with *astar_graph_search*? What about the computation time? Why?

   When no solution can be found by a strategy in a reasonable time (say **1 min**), indicate the reason (time-out and/or swap of the memory).

7. **Submit** your program on INGInious, using the A^* algorithm with your best heuristic(s). Your file must be named *pacmen.py*. You program must print to the standard output a solution to the Pacmen instance given in argument, satisfying the described output format.