

LINFO1361: Artificial Intelligence

Assignment 2: Solving Problems with Informed Search

Amaury Fierens, Auguste Burlats, Nicolas Golenvaux, Yves Deville
March 2023



Guidelines

- This assignment is due on **Wednesday 15th March 2023, 18h00**.
- *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.
- 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 *INGlrious* system. Only programs submitted via this procedure will be graded. No 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*.
- The answer to questions must be given by filling in the latex template provided. The final file must be submitted on *gradescope*. No report sent by email will be accepted.
- Nothing must be modified in the template except your answer that you insert in the *answer* environments as well as your names and your group number. The names are provided through the command *students* while the command *group* is used for the group number. The dimensions of *answer* fields *must not* be modified either. For the tables, put your answer between the "&" symbols. Any other changes to the file will *invalidate* your submission.
- To submit on gradescope, go to <https://gradescope.com> and click on the "log in" button. Then choose the "school credentials" option and search for *UCLouvain Username*. Log in with your global username and password. Find the course LINFO1361 and the Assignment 2, then submit your report. Only one member should submit the report and add the second as group member.
- Check this link if you have any trouble with group submission <https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members>



Deliverables

- The following files are to be submitted:
 - report_A2_group_XX.pdf: Answers to all the questions in a single report following the template given on moodle. Remember, the more concise the answers, the better. This deliverable should be submitted on gradescope as mentioned before.
 - The file `softflow.py` containing your Python 3 implementation of the Pacmen problem solver. Your program should take the path 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^* 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. (1 pt)
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. (1 pt)
3. Show on the right maze the board positions visited by A^* 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. (1 pt)

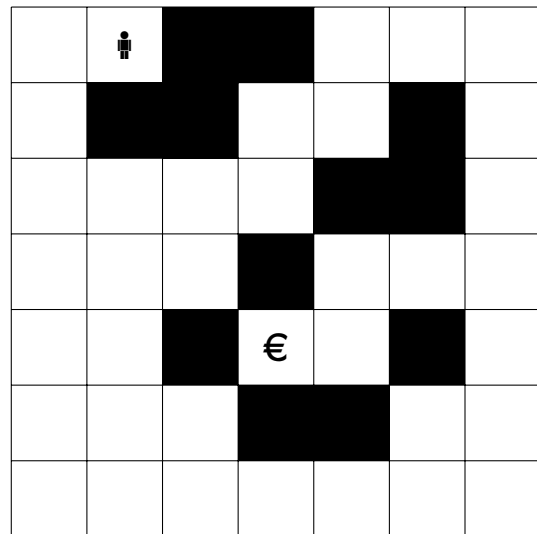
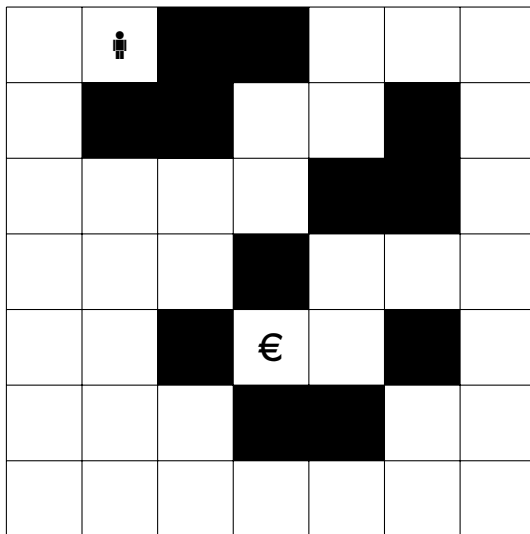


Figure 1

2 SoftFlow problem (17 pts)

The problem you will solve for this assignment is the SoftFlow problem. Again, the search procedures from `aima-python3` will help you solve it!

In this problem, we model a technician who's asked to connect optical fiber cables to several offices of the INGI department, passing them through false ceilings. Each cable should be used to connect a pair of points that correspond to the entry point and the desktop. The cables being expensive, the technician is asked by the Accounting Service to minimize the total quantity of cable used. An additional constraint is that false ceilings of the INGI department are quite narrow, which prevents him from overlapping his cables.

As you may have noticed, this problem is inspired from the game Flow Free¹, but the goal is different as we do not force you to fill the whole grid.

Input and output format

Your program must take one argument : the path to the file containing the initial state (e.g. `i01`). We use ASCII symbols in order to represent a state. Figure 2 shows an example of initial state, for the test instance `i01`.

```
#####  
#   #a   #  
# 0 #####  
#           #  
#   # # # #  
#   #   b # #  
#   #   1 #  
#####
```

Figure 2: The initial instance file `i01`.

The different symbols in this problem as follows:

- The `a` and the `b` are the entry point of the cables. As there could be up to 10 cables, the letters can go up to `j`. Those letters changes with the number of cables (1 cable = `[a]`, 2 cables = `[a, b]`, 3 cables = `[a, b, c]`, ...).
- The `0` and the `1` are the exit point of the cables (i. e. the desktop that must be connected). As there could be up to 10 cables, the symbols can go up to 9. Those symbols increase with the number of cables (1 cable = `[0]`, 2 cables = `[0, 1]`, 3 cables = `[0, 1, 2]`, ...).
- `#` marks a part of the false ceilings that is not accesible (too narrow or a wall).
- A space represents an empty position, that can be used by at most one cable.

Hint: As `"a"` correspond to `"0"`, `"b"` correspond to `"1"`, etc in the grid, the spacing between entrance and exit symbol is constant. Therefore, an easy way to convert characters to integers/integers to characters is to use the built-in functions from python `ord()` and `chr()`. The ASCII Chart can also be helpful in this purpose².

A solution to this problem is composed of the successive states of the game, induced by installation of the cables and ending with them all placed with a minimal length of cable used.

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,

¹https://en.wikipedia.org/wiki/Flow_Free

²<https://www.commfront.com/pages/ascii-chart>.

separated by an empty line. Each tile occupied by a part of cable except the extremity of the cable (i. e. a character between "a" and "j") must be represented by the same number as the exit point of the given cable. When a cable connect to a desktop, the symbol of the extremity of this cable should be immediately transformed to the number of the cable, to indicate that the cable is connected.

Figure 3 gives an example of a valid solution for instance *i01*. This solution, obtained by executing `python3 softflow.py instances/i01`, is also optimal because it involves a minimal number of actions (or moves).

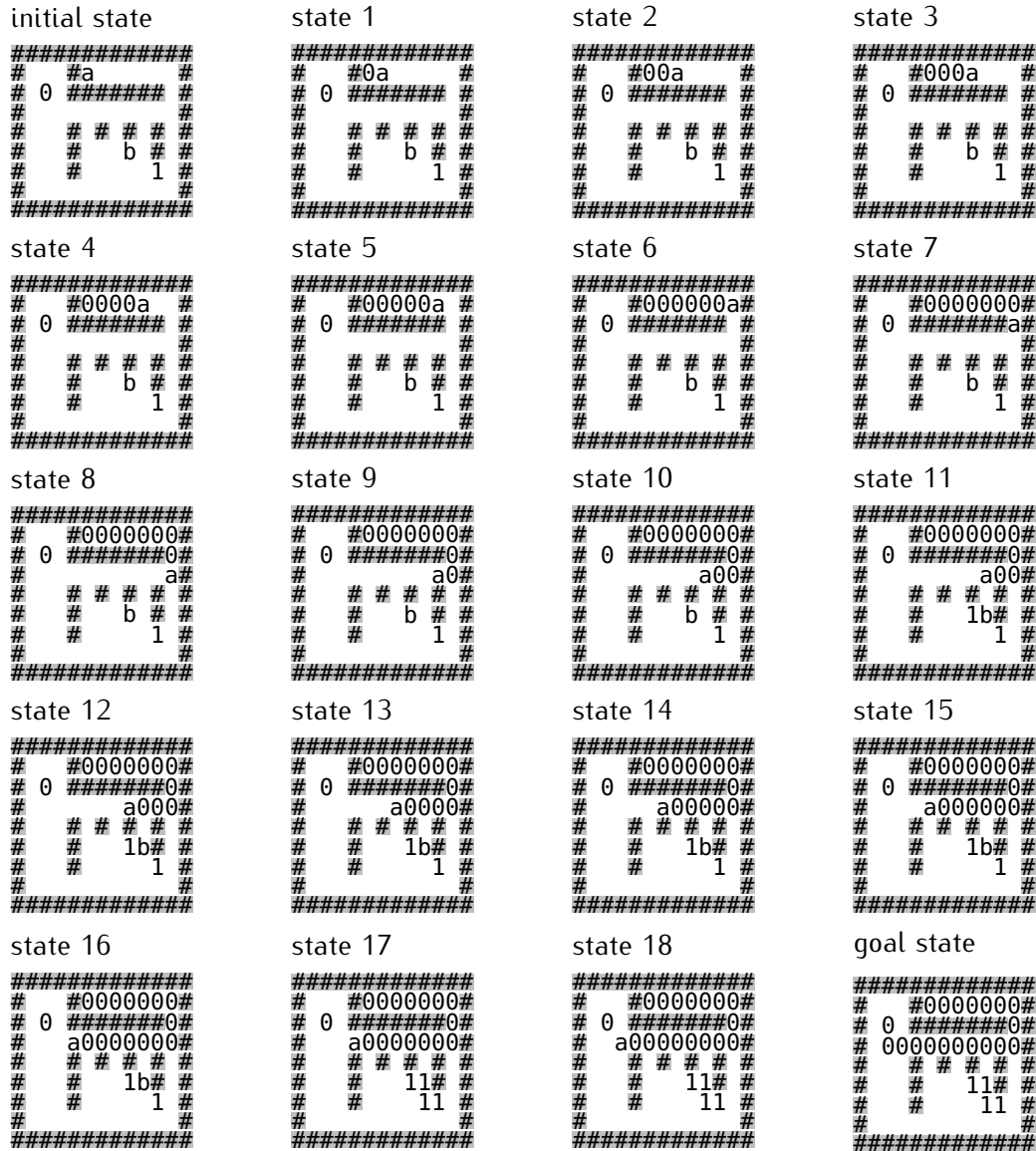


Figure 3: A possible optimal solution output for instance *i01*.

Be careful with the solution output format ! Your solver must respect the exact same format as in Figure 3, except that we put several states per line in this figure for display purpose. For this, do not modify the output printing code in the example file `softflow.py` and do not print anything in addition.



Questions

1. Model the SoftFlow problem as a search problem; describe: (2 pts)
 - States
 - Initial state
 - Actions / Transition model
 - Goal test
 - Path cost function
2. Give an upper bound on the number of different states for a SoftFlow problem with a map of size $n \times m$, with k cables to place. Justify your answer precisely. (1 pt)
3. Give an admissible heuristic for a SoftFlow instance with k cables. Prove that it is admissible. What is its complexity ? (2 pts)
4. **Implement** your solver. Extend the *Problem* class and implement the necessary methods and other class(es) if necessary. (1 pt)
5. **Experiment**, compare and analyze informed (*astar_graph_search*) and uninformed (*breadth_first_graph_search*) graph search of aima-python3 on the 10 instances of SoftFlow provided. (4 pts for the whole question)

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 3 min), indicate the reason (time-out and/or exceeded the memory).
6. **Submit** your program on INGIInious, using the A^* algorithm with your best heuristic(s). Your file must be named *softflow.py*. Your program must print to the standard output a solution to the SoftFlow instance given in argument, satisfying the described output format. (5 pts)