University of Royal Holloway CS5960-2020
Prof. Sara Bernardini, Dr. Santiago Franco, Dr. Ali Akbari, Dr. Ferdian Jovan

lab 3.

Solutions due Tuesday, **Dec 1, 23:59** uploaded in the AI CMS. See submission instructions below.

This exercise sheet is accompanied with some source files, which we provide through a separate `zip` archive. You can download this archive from the AI CMS under the Materials category.

Exercise 6: Heuristic Functions. ()

The planning task for this exercise is based on the 2D grid world from an earlier exercise sheet. The task is about finding the minimal number of agent movements to collect all coins on the grid. The agent's actions are given by moving to horizontally and vertically adjacent cells (i.e., diagonal movements are not allowed). The agent cannot move onto walls. A coin is automatically collected when the agent moves onto the respective cell. The goal is to collect all coins.
The provided `heuristic_search.py` scripts already contains an implementation of the planning model, as well as an implementation of the A^* search algorithm. Figure 1 shows an illustration of a state in the GUI. The agent's current position is indicated by `A`. The locations of coins are indicated by numbers. Coins which have been collected already are listed below the grid. Running

```
python3 heuristic_search.py -h
```

will print all relevant usage information.
Your task in this exercise pertains to the four heuristic functions described next. For each of them:

(a) Fill in the implementation in `heuristic_search.py`. The relevant places in the code are indicated by `TODO` comments. The file contains detailed comments for the classes and functions that you will have to use to do so. **Include comments, explaining what you did, and (if not obvious) why it is correct!**

(b) Justify, on paper, whether the heuristic is actually admissible or not. If admissible, provide a formal argumentation why this is so. If not admissible, provide a counter example showing why it is not.

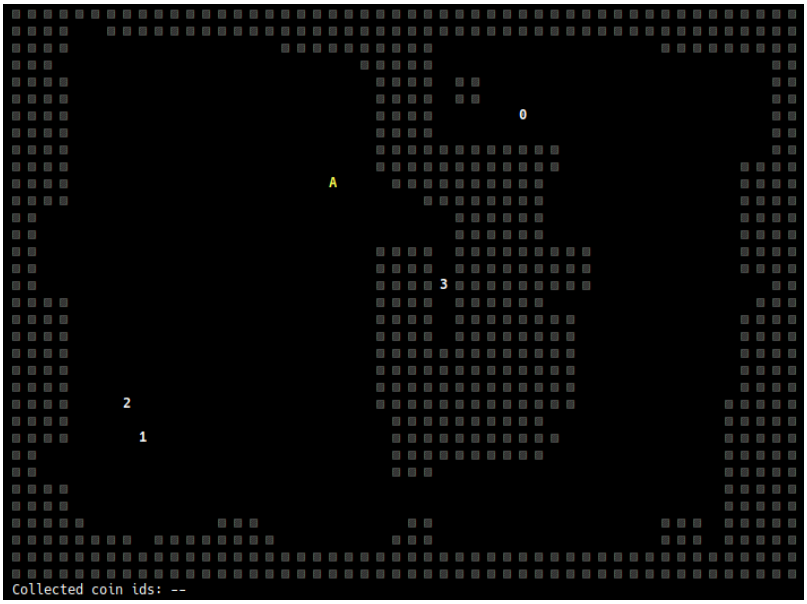

Figure 1: Graphical illustration of the initial state in the grid world planning task of Exercise 6. Script parameters:
`python3 heuristic_search.py den009d.map 20,10 32,6 8,25 7,23 27,16`

Your code must be compatible with Python 3!
Some of the heuristics make use of the Manhattan distance. The Manhattan distance is defined as $d(\langle x, y\rangle, \langle x', y'\rangle) = |x - x'| + |y - y'|$. Let $\mathcal{C}$ be the set of all coins. For every coin $c \in \mathcal{C}$, let $\langle c_x, c_y\rangle$ denote its position on the grid. A state s is triple $\langle s_x, s_y, C_s\rangle$ where $\langle s_x, s_y\rangle$ gives the agent's current position, and $C_s \subseteq \mathcal{C}$ is the set of all collected coins. The heuristic functions are formally defined as follows:

(i) The maximal Manhattan distance between the agent's current location to any coin that has not been collected. Formally, $h_1(s) = \max_{c \in \mathcal{C} \setminus C_s} d(\langle s_x, s_y\rangle, \langle c_x, c_y\rangle)$. To use h_1 in the script, append `--heuristic ManhattanMax` to the command line parameters.

(ii) The sum of the Manhattan distances to all coins that have not been collected. Formally, $h_2(s) = \sum_{c \in \mathcal{C} \setminus C_s} d(\langle s_x, s_y\rangle, \langle c_x, c_y\rangle)$. To use h_2 in the script, append `--heuristic ManhattanSum` to the command line parameters.

(iii) Let $\mathcal{C} \setminus C_s = \{c^1, \ldots, c^n\}$, and assume that $c^1, \ldots, c^n$ are ordered so that

(1) for all $1 < i \leq n$, it holds that $d(\langle s_x, s_y\rangle, \langle c^1_x, c^1_y\rangle) \leq d(\langle s_x, s_y\rangle, \langle c^i_x, c^i_y\rangle)$,

(2) for all $1 \leq i < n$ and $i+1 < j \leq n$ it holds that $d(\langle c_x^i, c_y^i\rangle, \langle c_x^{i+1}, c_y^{i+1}\rangle) \leq d(\langle c_x^i, c_y^i\rangle, \langle c_x^j, c_y^j\rangle)$.

The heuristic is given by $h_3(s) = d(\langle s_x, s_y\rangle, \langle c_x^1, c_y^1\rangle) + \sum_{i=1}^{n-1} d(\langle c_x^i, c_y^i\rangle, \langle c_x^{i+1}, c_y^{i+1}\rangle)$. To use h_3 in the script, append `--heuristic ManhattanOrderedSum` to the command line parameters.

(iv) Come up with and implement an arbitrary other heuristic with the following properties:

(1) Your heuristic is admissible. Justify why it is admissible (on paper)!

(2) Your heuristic (and implementation) works for arbitrary maps, arbitrary initial positions, arbitrarily many coins, and arbitrary coin locations.

(3) For the map, initial agent location, and coins as shown in Figure 1, A^* with your heuristic expands less than 5000 states (less than about 50% of the states expanded by breadth-first search).

To use your heuristic in the script, append `--heuristic CallMeFancy` to the command line parameters.

Submission Instructions

Solutions need to be packaged into a `.zip` file and uploaded toe Moodle. The file should be named as:

`CS5960-FirstName-LastName-StudentID-lab3`

The `.zip` file needs to contain the following files

- a PDF file with the solutions to the theoretical questions of this sheet.
- the python file with your implementation of the solution.