Assignment 2: Heuristic Search

Artificial Intelligence
WS 2023
Due: 2023-11-13, 12:00 noon

Katharina Hoedt
Verena Praher
Florian Schmid

1 Environment Activation / Framework re-installation

You will use the **same** `conda` environment that you set up for assignment 1. However, we need to download the **new version** of the Python framework from MOODLE, and **re-install** the package it contains in this virtual `conda` environment.

- Activate the virtual environment you previously created:
  `$ conda activate py311_ai_assignments`
- For most shells, your command prompt should now have changed, to indicate that the virtual conda environment named `py311_ai_assignments` is now active.
- Download the file `ai_assignment2.zip` from the MOODLE course page.
- Unzip the `ai_assignment2.zip` into an empty directory.
- We will refer to this directory as your **base** directory.
- In your shell, navigate to the base directory.
- Issue the following command:
  `$ pip install -e .`
  (it will install the new `ai_assignments` package and its dependencies into your active `conda` environment)
- You are now ready to tackle the practical part of assignment 2!

2 Theoretical Questions (10 pts)

There is a **separate quiz** on MOODLE for theoretical questions, covering properties of heuristics for search algorithms. For the theoretical questions you have an **unlimited number** of attempts, but **no feedback** whether or not your answers are correct.

Hint: Note again, that successors of a node are added to the fringe from left to right (if relevant).

3 GBFS, ASTAR (Quiz + Code = 4 + 10 pts)

For the **practical** parts of the assignments, you will be asked to implement the following search algorithms that you already heard about in the lecture, as well as one heuristic:

- GBFS, Greedy Best First Search (`ai_assignments/search/gbfs.py`)
- ASTAR, A^* Search (`ai_assignments/search/astar.py`)
- Chebyshev distance as a heuristic (`ai_assignments/search/gbfs.py`)

Please **only modify and upload the two indicated source files**!

All practical assignments have a corresponding MOODLE quiz. Those quizzes contain questions, which ask you to provide the solution to a specific problem instance.

Here is what to remember in general:

- Each problem instance is encoded as a **JSON file**.
- This file is attached to a question in the MOODLE quiz.
- You will need to **download** the problem instance and run your code to **solve** it.
- You will then **copy the answer** into the answer field of the question in the quiz.
- In your shell, navigate to the base directory.
- Make sure your `conda` environment is active:
  (and you followed all instructions in section 1)
  `$ conda activate py311_ai_assignments`
- After you are done solving all the problem instances, upload `gbfs.py` and `astar.py` in a **zip** file to MOODLE!

Here is what to do for GBFS:

For GBFS, there are two variants. Those variants correspond to **two different heuristics** that give us lower bounds on the distance to the goal state. We have provided you with one heuristic implementation, and you will do the second one yourself:

- The first heuristic – which is implemented – uses the **Manhattan distance** from the current state to the goal state, and registers a solver named `gbfs_mh`.

- The second heuristic – which **you implement** – uses the **Chebyshev distance** from the current state to the goal state, and registers a solver as `gbfs_ch`.

You only need to implement GBFS **once**! It will automatically register as two solvers with names as above, for the two different heuristics. This is important for solving the problem instances. You can follow these steps:

- Implement Greedy Best First Search in `ai_assignments/search/gbfs.py` (first **TODO**)
- Download the `GBFS_MH` and `GBFS_CH` problem instances from MOODLE
- In a shell, try:
  `$ python solve.py <the-problem-instance-from-the-quiz> gbfs_mh`
  to test your implementation of GBFS.
- Then, implement the Chebyshev distance (second **TODO** in `gbfs.py`), which we defined in class as $d_{chebyshev}(\boldsymbol{a}, \boldsymbol{b}) = \max_i (\mid a_i - b_i \mid)$.
- In a shell, you can then check this implementation by running the second problem instance,
  `$ python solve.py <the-problem-instance-from-the-quiz> gbfs_ch`
- Finally, copy and paste the solution hashes to their respective `GBFS_MH` or `GBFS_CH` questions in the quiz.

Here is what to do for ASTAR (A^*):

For ASTAR, there are also two variants. Those variants correspond to **two different heuristics** that give us lower bounds on the distance to the goal state. In this scenario, we have provided you with two heuristic implementations:

- The first heuristic uses the **Euclidean distance** from the current state to the goal state, and registers a solver named `astar_ec`.
- The second heuristic uses the **Manhattan distance** from the current state to the goal state, and registers a solver as `astar_mh`.

You only need to implement ASTAR **once**! It will register as two solvers with names as above, for the two different heuristics.

- Implement A^* in `ai_assignments/search/astar.py` (see **TODO**)
- Download the `ASTAR_EC` and `ASTAR_MH` problem instances from MOODLE
- In a shell, try:
  `$ python solve.py <the-problem-instance-from-the-quiz> astar_ec`
  or
  `$ python solve.py <the-problem-instance-from-the-quiz> astar_mh`
  for the respective problem instances.
- Copy and paste the solution hashes to their respective `ASTAR_EC` or `ASTAR_MH` questions in the quiz.