ECS759P Artificial Intelligence
Coursework 1

Due Date: 1st of November 2024 at 10AM

If you have any questions, you are encouraged to ask them and lead discussions via the Student Forum on the module QMplus page. Using the QMplus forum for discussion is preferable as it can help us broadcast essential clarifications to the entire group. You can also ask questions during your assigned "support" lab session, which will happen on Week 6. Additionally, Dr. Iran R Roman is open for inquiries during office hours (Wednesdays, 12pm-1pm, Peter Landin 4.16) and via email to i.roman@qmul.ac.uk.

In your final submission you MUST include the files:

- `XXXXXX_my_report.pdf`
- `XXXXXX_sqrt_agent.py`
- `XXXXXX_search.py`

You must substitute "XXXXXX" for your username ("eer043", for example).

Your submission will be automatically flagged as INCOMPLETE if you do not include ALL these files, properly named, and in the format specified.

General Instructions

The coursework is composed of two parts. Both parts involve writing code, and also answering questions in a written form. You should answer these questions with your choice of typesetting software or by hand on a physical piece of paper. It is preferred if you use a typesetting software (overleaf, for example). If you use pencil and paper, you will need to take pictures of all your work and convert them to a single pdf document that contains ALL your answers to this coursework. Your written answers should be clearly labeled to match the corresponding question number in this coursework description.

The coursework parts are:

1. Square Root Agent: 30%
2. Path-finding algorithms: 70%

1 Square Root Agent

First, you will answer the following questions (in your written report) about the update functions for the square root agent we discussed in lecture. Remember that this agent is given a number x, where $x > 0$, and it must go from an initial guess y_0 to an optimal number y_n such that $J < \epsilon$, where ϵ is a small number (let's say $1e^{-6}$) and J is the goal function.

1.1 Theory questions (submit in your written report)

- Draw (or plot) the cost function $J = |y_n^2 - x|$.
- Derive the update function $y_{n+1} = y_n - \frac{J}{\frac{\partial J}{\partial y_n}}$, where $J = |y_n^2 - x|$. You must show all steps of your derivation in detail.
- Write "pseudo-code" to use this update function and go from y_0 to y_n.
- Draw (or plot) the cost function $J = (y_n^2 - x)^2$.
- Derive the update function $y_{n+1} = y_n - \frac{J}{\frac{\partial J}{\partial y_n}}$, where $J = (y_n^2 - x)^2$. You must show all steps of your derivation in detail.
- Write pseudo-code to use this update function and go from y_0 to y_n.

This subsection goes in your written report and carries 10% of the mark for this coursework.

1.2 Python implementation of the update functions

You are given the file called `sqrt_agent.py`. In this file you will implement three python functions:

- The first function is an implementation of the square root agent using $J = |y_n^2 - x|$ and is called `sqrt_agent_abs_error`. This function takes a positive number x as input and it should output an approximation of its square root, found using Newton's method via the goal function $J = |y_n^2 - x|$.
- The second function is an implementation of the square root agent using $J = (y_n^2 - x)^2$ and is called `sqrt_agent_squared_error`. This function takes a positive number x as input and it should output an approximation of its square root, found using Newton's method via the goal function $J = (y_n^2 - x)^2$.
- The third function is an implementation of basic tests to check your work (you should further test your functions; to determine your grade, we will run a comprehensive battery of tests on your submitted `sqrt_agent.py`).

In these functions, only write code within the blocks delimited as:

```
# * * * * * * * * * * * * * *
# * * * * * * * * * * * * * *
# YOUR CODE STARTING HERE *
# * * * * * * * * * * * * * *
# * * * * * * * * * * * * * *

# YOUR CODE HERE!

# * * * * * * * * * * * * *
# * * * * * * * * * * * * *
# YOUR CODE STOPS HERE  *
# * * * * * * * * * * * * *
# * * * * * * * * * * * * *
```

Do not write code in any other sections and do not modify ANY lines that do not fall within those blocks. Changing code you are not supposed to will result in automatic failure of all tests that determine your grade.

You are provided with a series of steps that you can follow to carry out your implementation of each algorithm. However, you are free to use these steps and follow a different strategy if you prefer. However, the function should use the correct algorithm and return the correct answer.

Given a positive number x, your agents should be able to find $\sqrt{x}$ with a tolerance of $1e^{-3}$.

If you have never worked with '.py' files before, Dr. Roman will do a demonstration during our Week 4 Lecture to show you how to work on this assignment using JHub.

This subsection carries 20% of the mark for this coursework.

2 Path-finding algorithms

Now your task is to build an AI route finder. You are given a data file (tubedata.csv) in CSV format with the London Tube map (which is not necessarily an up-to-date or complete map). The Tube map is defined in terms of a logical relation Tube "step". If you open the data file with any text editor, you will see the content of the file as:

```
Harrow & Wealdstone, Kenton, Bakerloo, 3, 5, 0
Kenton, South Kenton, Bakerloo, 2, 4, 0
...
Bank/Monument, Waterloo, Waterloo & City, 4, 1, 0
```

Each row in the CSV file represents a Tube "step" and is in the following format:
`[StartingStation], [EndingStation], [TubeLine], [AverageTimeTaken], [MainZone], [SecondaryZone]`
where:

- `StartingStation`: a starting station
- `EndingStation`: a directly connected ending station
- `TubeLine`: the tube line connecting the stations above
- `AverageTimeTaken`: the average time, in minutes, taken between the starting and the ending station
- `MainZone`: the main zone of the starting station
- `SecondaryZone`: the secondary zone of the starting station, which is 0 if the station is only in one zone. **Note**: if the secondary zone is 0, use the main zone to define the zone for the ending station; otherwise, use the secondary zone.

Throughout this coursework, you may find it helpful to refer to the London Tube map at:
`https://content.tfl.gov.uk/large-print-tube-map.pdf`

The preferred way to load the data is using the `pandas` Python library method `read_csv` with the parameter `header=None` using the `process_tubedata` method in the `search.py` file attached to this coursework.

Note that in the labs, we used the NetworkX library for graph visualisation. But in this CW you should not use it in the code you submit. You may use it to debug your code, but your final submission should NOT include any use of the NetworkX library.

2.1 Python implementation of search algorithms and their testing

You are given the file called `search.py`. In this file you will implement seven python functions. The function called `process_tubedata` is already implemented and you should not change it at all.

- For the first function you will implement the "breadth first search" algorithm. The function is called `bfs_route_finder`. This function takes as input arguments the starting station, the end station, and the station dictionary (generated by `process_tubedata`). The function outputs the path from the starting station to the end station as a list of station names.
- For the second function you will implement the "depth first search" algorithm. The function is called `dfs_route_finder`. This function takes as mandatory input arguments the starting station, the end station, and the station dictionary (generated by `process_tubedata`). The function outputs the path from the starting station to the end station as a list of station names.

- For the third function you will implement the "uniform cost search" algorithm. The function is called `ucs_route_finder`. This function takes as input arguments the starting station, the end station, and the station dictionary (generated by `process_tubedata`). The function outputs the lowest-cost path from the starting station to the end station as a list of station names.
- For the fourth function you will implement a UCS algorithm that also considers the cost of changing lines at stations. The function is called `ucs_route_finder_with_added_cost_for_changing_lines`. This function takes as input arguments the starting station, the end station, and the station dictionary (generated by `process_tubedata`). The function outputs the lowest-cost path from the starting station to the end station as a list of station names, considering that each time a line is changed at a station the cost increments by 10 minutes.
- For the fifth function you will implement a "heuristics bfs" algorithm. The function is called `heuristic_bfs`. This function takes as input arguments the starting station, the end station, the station dictionary (generated by `process_tubedata`), and the zone information (also generated by `process_tubedata`). The function outputs the lowest-cost path from the starting station to the end station as a list of station names, considering a heuristic of your choice that uses the zone information in the `zone_dict` dictionary.
- For the sixth function you will implement a "counter of all visited nodes". The function is called `count_visited_nodes`.

Do not write code in any other sections and do not modify ANY lines that do not fall within the blocks allocated for you to write your code. Changing code you are not supposed to will result in automatic failure of all tests that determine your grade.

You are provided with a series of steps that you can follow to carry out your implementation of each algorithm. However, you are free to use these steps and follow a different strategy if you prefer. However, the function should use the correct algorithm and return the correct answer.

This subsection carries 50% of the mark for this coursework.

2.2 Compare your search algorithms (submit in your written report)

Make sure you try out a good number of different routes with the algorithms you implemented in python (including the ones already tested in the code). After you are happy with your algorithms and testing, answer the following questions in your written report:

- Is any path finding method consistently better?
- Report the count of the visited nodes by each algorithm for the most interesting routes you tested (minimum 4) and explain your results based on the knowledge of how each algorithm works.
- Report the costs in terms of the time taken for the most interesting routes you tested (minimum 4) and explain your results based on the knowledge of how each algorithm works.
- Explain how you overcame all the issues you had while implementing all path finding algorithms, such as infinite loops, etc. Describe four or five issues you encountered.
- Report the longest route(s) possible without repeated nodes in the state space of this tube map. Describe how you found such route(s).

Note: there's no extra credit for testing more than the 4 trajectories we included in the tests, but understand that increasing the number of tests you carry out will allow you to make sure your functions are robust and will pass the battery of tests we will perform on them.

This question carries 20% of the mark for this coursework.

2.3 Submission

Submit your coursework to QMPlus as a single zip file (filename ending in <username>.zip).

Note that a human will be reading the code and your written report, and cases of plagiarism will be reported.

IMPORTANT NOTE ABOUT YOUR USE OF AI:

While the use of Generative AI (ChatGPT, Github Co-pilot, Claude, etc.) to produce text or code for your coursework is DISCOURAGED, it is NOT prohibited but MUST be acknowledged when used. If you insert AI-generated text or code in your report/code, you must include the name and version of the AI system you used as an in-line comment for each line you generate with AI. Additionally, you must specify the prompt(s) used to produce the content. We recommend using chain-of-thought prompting so that your prompt clearly describes how you critically analysed the problem and the behavior of the AI (also including the analysis of AI-generated mistakes and your subsequent refinements to the prompts).

Failing to acknowledge and thoroughly describe the use of AI-generated code and text will result in an automatic grade of zero. Remember that AI is not perfect and its outcomes can be fallible.

Note that cases of identical (or similar) code and/or text to your peers resulting from the usage of Generative AI (without acknowledgement or critical analysis of this usage) will be considered plagiarism and will be reported.