**ECS759P Artificial Intelligence: Coursework 1**

**Due date: 16 Nov 2022 at 10.00 am**

*If you have any questions, you are encouraged to ask any question and discuss via Student Forum on the module QMPlus page. You can also come to the online drop-in session*[1] *(Wednesday 2 November 2022 at 2-3pm) or email Dr. Huy Phan at* ***h.phan@qmul.ac.uk*** *but using forum for discussion is preferable.*

# 1 Introduction

The coursework is composed of two parts, involving coding exercises:

- Agenda-based search: 60% + 2% extra
- Genetic algorithm: 40%

The total mark will be capped to 100%.

# 2 Agenda-based Search

This part of the coursework is intended to help you understand how the agenda-based search works and to give you practice in its implementation.

You task is to build an AI route finder using agenda-based search mechanism. You are given a data file (`tubedata.csv`) in CSV format with the London Tube map (which is not necessarily to be up-to-date and complete). 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 named stations
- `MainZone`: the main zone of the starting station
- `SecondaryZone`: the secondary zone of the starting station, which is `"0"` if the station is in only one zone. **Note:** to define the zone for the ending station use the main zone if the secondary zone is `"0"` 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

**Hint:** In the labs, we used `NetworkX` to construct a graph for search algorithms because it is useful for visualizing the graph. However, it is not necessary for this coursework. Here, the preferred way to load the data is using the Pandas Python library method `read_csv` with the parameter `header=None` and build the graph as follows:

[1] Join the drop-in session via this link:
https://teams.microsoft.com/l/meetup-join/19%3aIMiUgQ8Axq3CIKJ3RCUJO3IbuiKREoGDEcOivdzxbxc1%40thread.tacv2/1666697154623?context=%7b%22Tid%22%3a%22569df091-b013-40e3-86ee-bd9cb9e25814%22%2c%22Oid%22%3a%22defdf3f7-df69-4c8e-9032-baebea448caf%22%7d

```
import pandas as pd
df = pd.read_csv("tubedata.csv", header=None)
df.head()

from collections import defaultdict

station_dict = defaultdict(list)
zone_dict = defaultdict(set)

# get data row by row
for index, row in df.iterrows():
    start_station = row[0]
    end_station = row[1]
    act_cost = int(row[3])
    zone1 = row[4]
    zone2 = row[5]

    # station dict.  of child station tuples (child_name, cost from parent to the child)
    # {"Mile End":  [("Stepney Green", 2), ("Wembley", 1)]}
    station_list = station_dict[start_station]
    station_list.append((end_station, act_cost))

    # we add the main zone
    zone_dict[start_station].add(zone1)
    # we add the secondary zone
    if zone2 != "0":
        zone_dict[start_station].add(zone2)
        # if the secondary zone is not 0 it's the main zone for the ending station
        zone_dict[end_station].add(zone2)
    else:
        # otherwise the main zone for the ending station is the same as the starting station
        zone_dict[end_station].add(zone1)
```

## 2.1 Implement DFS, BFS, and USC

Implement DFS, BFS, and USC to find routes from a starting station to a destination station. For a path found, your program should print/display meaningful information: path in stations, cost in average time, number of nodes expanded. The route from Euston to Victoria is a good one for testing. Briefly describe how to launch the program and mention its main files in your report.

Firstly you need to think about how to represent a state and how to construct a new state from each station reachable from the current state but not already visited in the current path so far (this may include current station, line and zone, path and cost so far). Describe the state representation in your report.

**This question carries 20% of the marks for this coursework.**

## 2.2 Compare DFS, BFS, and USC

Make a detailed comparison of the three search methods For this, make sure you try out a good number of different routes, including the ones below, and base your comparison on the cost (even if not considered by DFS and BFS) and the number of nodes expanded for the different methods. Answer the following questions in your report:

- Is any method consistently better?
- Report the returned path costs in terms of the count of the visited nodes for one route below (or any route of your choice) and explain your results based on the knowledge of costs each algorithm considers.

- Report the returned path costs in terms of the average time taken for one route below (or any route of your choice) and explain your results based on the knowledge of costs each algorithm considers.
- Report the returned path costs in terms of the visited nodes and the average time taken for one route below (or any route of your choice) for two different orders to process the nodes (direct and reverse the order of the explored nodes at each level) and explain your results for the three algorithms.
- Explain how did you overcome the loop issue in your code.

Examples of routes you may include in your comparison:

- Canada Water to Stratford
- New Cross Gate to Stepney Green
- Ealing Broadway to South Kensington
- Baker Street to Wembley Park

**This question carries 15% of the marks for this coursework.**

## 2.3 Extending the cost function

Improve and implement the current UCS cost function to include the time to change lines at one station (e.g., 2 minutes). Using one of the routes mentioned above (or any route of your choice), give an example in your report how this new cost has changed the paths returned by each algorithm. Explain which of the algorithms were not affected and why.

**This question carries 10% of the marks for this coursework.**

## 2.4 Heuristic search

Given that you know the zone(s) a station is in, consider how you might use this information to focus the search in the right direction and implement your heuristic **Best-First Search (BFS)** (*Note:* not A* search). Explain in the report the motivation for your heuristic, its formula and illustrate its performance in terms of the average time taken as compared to the solution returned by UCS using one of the routes mentioned above or any route of your choice (even if it does not work in practise).

**This question carries 15% of the marks for this coursework.**

## 2.5 Extra

Beyond the heuristic design in Section 2.4, you are encouraged to participate in an extra peer-review activity. To do that, you are required to complete the following steps:

- **Step 1.** Submit the description of the heuristic you designed in Section 2.4.
  The deadline for this step is **16 Nov 2022 at 10.00 am**.
- **Step 2.** After the deadline of **Step 1**, comment on pros and cons of the heuristics designed by two other students that are assigned to you.
  The deadline for this step is **23 Nov 2022 at 10.00 am**.

Both the steps can be completed using the link "Heuristic peer-review" under the "COURSEWORK 1" on QM+.

**This question carries 2% extra marks for this coursework.**

# 3 Genetic Algorithm

This part of the coursework is intended to help you understand how a genetic algorithm works and to give you practice in its implementation.

**Let's play a guessing game!** I have given you a secret text phrase and your goal is to find it using a genetic algorithm. The phrase was exactly **10 characters long** and could only contain upper-case English letters, numbers between 0 and 9, and the underscore symbol _ (underscore). For instance, `1_PHRASE22` and `MYPA51WASE` are legitimate candidates.

I can give you hints (unlimited times) by telling how close to the secret phrase your guess is. So here is the deal: For each guess phrase you show me, I am going to give a score between 0.0 and 1.0 indicating how close to the secret phrase it is (where 1.0 means that the guess is exactly correct).

To be more precise, you have access to a function called `closeness` as follows:
**input:** a list of guesses, e.g. `["___TEST___", "IS_I7_L0V3"]`, along with your own student **username**[2] (e.g `"eey111"`).
**output:** an ordered dictionary containing the guess phrases as keys and the associated scores as values (e.g. `{"___TEST___": 0.045, "IS_I7_L0V3": 0.247}`)

Below is the example Python3 code demonstrating how you can access this function and obtain the hints for the above example (*Note:* you need to replace `"eey111"` with your username)

```
from hints import closeness
scores = closeness(["___TEST___", "IS_I7_L0V3"], "eey111")
>>> scores = {"___TEST___": 0.045, "IS_I7_L0V3": 0.247}
```

## 3.1 Implement a Genetic Algorithm

Implement a Genetic Algorithm to find your secret phrase (recall: you should use your own student username whenever invoking `closeness`). Report the secrete phrase that you got.

**This question carries 20% of the marks for this coursework.**

## 3.2 Elements of the algorithm

Briefly explain how specifically you chose to do the state encoding, selection, crossover, and mutation.

**This question carries 10% of the marks for this coursework.**

## 3.3 The number of reproductions

Report the number of reproductions you had to do to converge to the secret phrase? Ideally, run your code multiple times and report on the average and variance instead of a single run (since Genetic Algorithms involve randomisation).

**This question carries 5% of the marks for this coursework.**

## 3.4 Hyper-parameters

Explore the effect of at least one hyper-parameter on the performance of your algorithm. This could for instance be the choice of the mutation rate, or the selection/pairing scheme, the population number, etc. Ideally, provide a table or a figure with a very concise discussion.

**This question carries 5% of the marks for this coursework.**

[2] The list of students in the module is available here https://qmulprod.sharepoint.com/:x:/s/ECS759P-ArtificialIntelligence20222023SEMA/EUd2Vxlqr9hNgFlJjpUbfPoBfsW-O9m-Ge1fJ5VDKpYruw?e=9ouNxo. If you find that you are not in the list, let us know immediately.

# 4 Submission

Submit your coursework on QMPlus as a single zip file (filename ending in `<yourStudentNumber>.zip`) containing:

- Source code (Python)
- Report (a single PDF format for both parts)

Note also that a human will be reading the code and cases of plagiarism will be reported. Strategy, algorithms, originality, code quality, and report quality will be also assessed.