# ECS759P Artificial Intelligence: Coursework 1

**Due date: 17 November 2023 at 10.00 am**

If you have any questions, you are encouraged to ask them and lead discussions via the Student Forum on the module QMPlus page. You can also come to the support lab session (30 October, 9-11am at Eng: 3.52 and 3.56 and 11am-13pm at Eng Building: B10 - IoC). Additionally, Dr. Julia Ive's office is open for inquiries during office hours (Fridays 12pm-1pm, CS 309) and via email to `j.ive@qmul.ac.uk`. That said, using the forum for discussion is preferable as it can help us broadcast essential clarifications to the entire group.

## 1 Introduction

The coursework is composed of two parts. Both parts involve coding exercises.
The task distribution is:

- Agenda-based search: 60%
- Adversarial search: 40%

## 2 Agenda-based Search

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

Your task is to build an AI route finder using an agenda-based search mechanism. 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` as follows (we strongly advise you to copy this code from the `undirected_map.py` file attached to this coursework):

```
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 dictionary 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))

  # the following two lines add the other direction of the tube "step"
  station_list = station_dict[end_station]
  station_list.append((start_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 for the starting station
    zone_dict[end_station].add(zone1)
```

Note that in the labs, we used the `NetworkX` library for graph visualisation. Its usage is not required for this coursework.

## 2.1 Implement DFS, BFS and UCS

Implement DFS, BFS and UCS to find routes from a starting station to a destination station. For a path found, your program should print/display the following 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.

First, 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 mark for this coursework.**

## 2.2 Compare DFS, BFS and UCS

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 costs in terms of the average time (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 inverse order of explored nodes at each level) and explain your results for the three algorithms.
- Explain how you overcame 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 mark 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 of 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 mark 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 and 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 practice).

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

### 2.5 Extra: Peer Review

You are encouraged to participate in an extra peer-review activity. Discuss with a classmate of your choice their heuristic design (you do not need to mention the name of the classmate in your report). Briefly (2-3 sentences) describe the heuristic of your classmate. Provide two-three advantages and two-three disadvantages of this heuristic. There should be an equal number of advantages and disadvantages.

You can also comment on one of the heuristics described below instead of discussing with a classmate:

1. Zone-based heuristic which assigns a constant time of 10 minutes for travel within one zone and a constant time of 20 minutes for travel across zones.
2. Average expected time of travelling within each zone and from one zone to another. The heuristic is based on the average minimum time as estimated for samples of 5 station pairs within each zone and for each zone combination.

**Extra: This question carries 2% extra marks for this coursework.**
The total coursework mark will be capped at 100%.

## 3 Genetic Algorithm

This part of the coursework is intended to help you understand how genetic algorithms work and to practise their implementation.

You are given the code in Python below (we strongly advise you to copy this code from the `password_fitness.py` file attached to this coursework) to generate your password from your EECS username (e.g., "eey349").

```
def get_password(student_username, l=10):
    # possible characters include upper-case English letters,
    # numbers between 0 and 9 (inclusive),
    # and the underscore symbol
    options = string.digits + string.ascii_uppercase  + "_"

    h = hashlib.sha256(("ECS759P-AI"+student_username).encode("utf-8"))
    d = h.digest()
    s = ""
    for n in d:
      s += options[n%len(options)]

    return s[0:l]

# TO DO: replace *** with your EECS username and uncomment the code
# student_password = get_password('***')
print(student_password)
```

For example, by calling this method for the username "eey349" we get the password "2Q4ZQYUOTJ". You need to call this method for your own username by replacing the "***" in the code above (**do not forget to uncomment the relevant code, otherwise you will get an error message**) and provide your password in your report.

This password is exactly 10 characters long and it can contain only upper-case English letters, numbers between 0 and 9 (inclusive), and the underscore symbol (_).

The fitness function for any candidate password could be computed as provided below (we strongly advise you to copy this code from the `password_fitness.py` file attached to this coursework):

```
# Distance function
def distance_function(string_one, string_two):
    score = 0
    for i, j in zip(string_one, string_two):
        # Square of the absolute difference between two Unicode codes
        score += math.sqrt(abs(ord(i) - ord(j)))
    return score


# Upper bound of the distance value
MAX_VALUE = distance_function('0000000000', '__________')

# Compute normalised fitness for a list of candidate passwords
def get_normalised_fitness(list_of_phrases, student_password):
    ordered_dict = dict()
    phrase_to_find = student_password
    for phrase in list_of_phrases:
        # Return 1 when a candidate matches the true password
        # (string distance between them is zero)
        ordered_dict[phrase] = 1 - distance_function(phrase, phrase_to_find) / MAX_VALUE
    return ordered_dict

# Example of how to get fitness values for a list of candidates
get_normalised_fitness(['2Q4HHHHOTJ', '2HHZQYUOTJ'], student_password)
```

## 3.1 Implement a Genetic Algorithm

Implement a Genetic Algorithm to find your password. Report the true password (obtained as indicated above) and the password found (they should ideally match).

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

## 3.2 Algorithm components

Describe the chosen state representation and the methods you chose to perform selection, crossover and mutation.

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

### 3.3 Number of reproductions

Report the number of reproductions you had to perform to converge to the true password. Run your code 5-10 times and report the average and standard deviation of the number of reproductions (since Genetic Algorithms involve randomisation).

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

### 3.4 Hyperparameters

Explore the effect of at least one hyperparameter on the performance of your algorithm. This could be for instance trying at least two different mutation or crossover rates, two selection schemes, two population sizes, etc. Provide a table that compares the average and the standard deviation of reproductions until convergence for each hyperparameter choice. Comment on which hyperparameter value worked better and why.

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

## 4 Submission

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

- *Commented* source code in Python $\geq 3.7$ (extension .py, Jupiter Notebook file with the extension .ipynb). Comments are important as they indicate your understanding of the code.
- Report (a single PDF format for both parts, no more than 3 pages of font-size 10 point).

Note 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.

**IMPORTANT RE USE OF AI:**

While the use of Generative AI (ChatGPT, Github Co-pilot, etc) to produce text or code for your coursework is NOT prohibited, it MUST be acknowledged. A detailed explanation of how you used those tools while preparing your coursework does not count towards the page limit. If you insert AI-generated text or code in your report/code, you must include a share link or name the AI system. Additionally, you must specify the prompt(s) used to produce the content and how you critically analysed it (including the analysis of AI-generated mistakes and your subsequent refinements to the prompts).

Failing to acknowledge and describe the use of AI-generated code and text will mean that mistakes and issues in this code and text will be considered as your own. If you fail to acknowledge the use of Generative AI and thoroughly evaluate its results, you can not contest the grade given on the grounds that the use of Generative AI is not prohibited, and its outcomes can sometimes 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.

When adding citations, note that fraudulent reporting of source material is considered academic misconduct (see the latest academic misconduct policy): for example, if you provided an inaccurate summary of an existing paper written by generative AI, that would be considered fraudulent reporting of source material, which in turn qualifies as academic misconduct.