ECS759P Artificial Intelligence: Coursework 1

Due date: 19 Nov 2021 at 10.00 am

If you have any questions, you are encouraged to ask any question and discuss via Student Forum on the module's QMPlus page. You can also email Dr. Huy Phan at `h.phan@qmul.ac.uk` *but using forum for discussion is preferable (what it is used for!).*

1 Introduction

The coursework is composed of two parts. The first one involves coding exercises. The last one is a writing exercise about adversarial search.

- Agenda-based search: 70%
- Adversarial search: 30%

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 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 that defines 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 named stations
- `AverageTimeTaken`: the average time, in minutes, taken between the named stations
- `MainZone`: the main zone of the stations
- `SecondaryZone`: the secondary zone of the stations, which is `"0"` if the station is in only one zone

Throughout this coursework, you may find it helpful to refer to the London Tube map at
`http://www.afn.org/~alplatt/tube.html`

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 (e.g. stations, cost, number of node expanded). The route from Euston to Victoria is a good one to test on. Describe in the report how your program works.

You firstly need to think about how to represent a state and how to construct a new state from each station reachable from the one in the current state but not already visited in the current path so far (hint: if you are unsure, think about the common features of DFS, BFS, and USC).

This question carries 25% of the marks for this coursework.

2.2 Compare DFS, BFS, and USC

Now make a detailed comparison of the three search methods, and write a short analysis of what you find. Make sure you try out a good number of different routes, including the ones below, and, once you've done so, base your comparison on the relative costs, and the relative number of nodes expanded, of the different methods. Is any one method consistently the best? What are the advantages and disadvantages of the methods, in comparison one to another? How does the way the Tube Map is structured affect the outcomes? Write a short report (no more than 2 pages of 10 point font for this part) of what you find and explain what you have discovered.

Routes to 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

Consider how you might improve the cost function over simply adding the time required between stations. Implement and use your new cost function in USC. Explain your new cost function and what you found in the report.

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 search. You should explain in detail in the report your heuristic - if your implementation has gone astray, but your idea was good, you will get at least some of the credit.

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

3 Adversarial search

You are now engaged in a thrilling two-player adversarial game which is represented by the following tree. Suppose you are the MAX player (who goes first).

3.1 Play optimally (MINIMAX algorithm)

Knowing that you start (top of the tree is you), complete the blank nodes in the provided tree using the MINIMAX approach.

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

3.2 Play optimally, quicker thinking! (α-β pruning)

Suppose there is a mean arbiter that doesn't like too much hesitation at the beginning. Perform α-β pruning on the tree you filled, searching from left to right. Which branches are pruned and why?

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

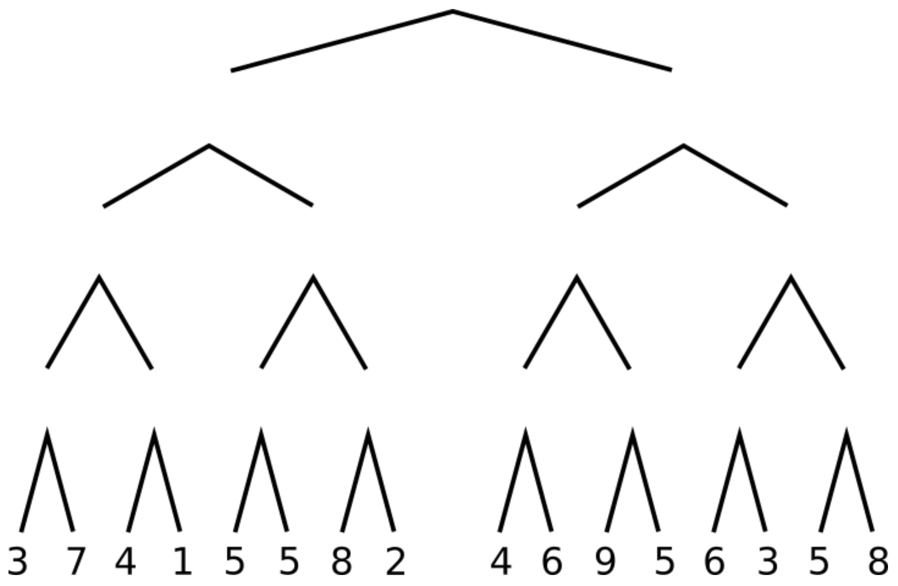

Figure 1: Tree of the game

3.3 Entry free to play

Suppose the entry fee to play the game is $x \geq 0$ units. In particular, this is the money that the MAX player pays the MIN player to be allowed to start the game. The rest of the game is as before. So for instance, if the terminal state of the game is the left-most leaf in Figure 1, then the MAX player wins from the MIN player 3 units (so the MIN player has to pay MAX player 3 units). Note that this is independent of the entry fee. So effectively, in this scenario, the MAX player has won $3 - x$ units overall, exactly how much the MIN player has lost.

1. What are the ranges of values for x that will be still worth playing the game for the MAX player (to enter the game at all)?

2. For a fixed value of x, do you prefer to be the first player (MAXimiser) or the second player (MINimiser)? Very briefly explain your answer.

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

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.