MATH6005/6182 Coursework

MATH6005: Worth 80%
MATH6182: Worth 80%

Submission date: **13 December 23:59**

Rules

- This is a group work assignment, you must work together on the problem. Each group member will receive the same grade for the coursework.
- You must submit both a single Python file and a document detailing the progress of your team. The Python file will contribute 80 percent of your total mark, the document will count 20 percent.
- You must code the algorithm yourself within your team, you must not use any AI code generating software or obtain help from another team. Any team found to have used AI coding software will be reported to the relevant academic integrity authority.

Introduction

Your task as a team is to write a Python programme that computes the shortest path around a rail network. Transport for London (TFL) require you to write an algorithm that will plan a journey between stations on their network, given the length of time it takes to travel between each station.

Theory

As you have seen in the last lecture we wrote a simple algorithm that given the coordinates of a group of cities, visits each city exactly once, travelling to the nearest neighbour each time. The algorithm that you will code now is a bit more fancy than that - given the travel times and location of the stations the algorithm will compute the shortest path between each of the stations, starting at a designated initial station. Some of you may have heard of this type of algorithm before, named after Dutch mathematician Edsger W. Dijkstra. Before introducing the algorithm, I will introduce a few key terminologies that will be useful in order to code up the algorithm.

Let i be the name of a station in the network and j be the name of another station. We define $l_{i\to j}$ to be the length of time taken to travel on the line from station i to station j. In this coursework the network is not directed (i.e. trains can travel both ways down the line). As an example consider the following mini-network shown in figure 1.

The time taken to travel from Paddington directly to Kings-Cross is 1 minute, which implies that using our notation that $l_{\mathrm{KC}\to P} = 1$. Now consider the problem of finding the shortest path from Waterloo to all other stations in the network. To do this, we will apply Dijkstra's algorithm.

Dijkstra Algorithm

To begin we first describe the data structures we will need. We have a set S which contains a list of the stations visited by the algorithm. We also have a label assigned to each station j which includes the quantities:

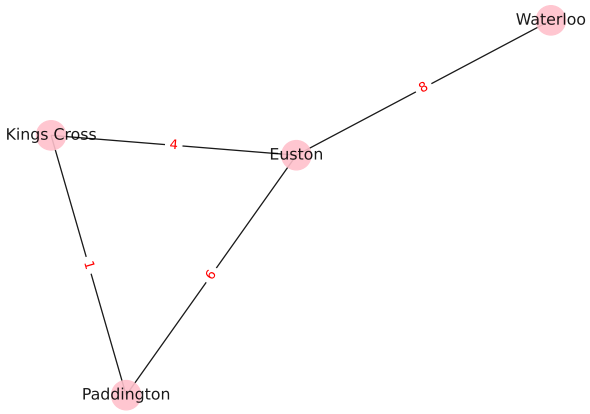

Figure 1: Example rail network

$Y(j)$ which is a label detailing the value of the shortest time path from the starting station to each station j. We also have another label $P(j)$ called the path, which is a label given to each station in the network which contains information about the shortest path in the network to station j. Now, we initialise the data structures.

Step 1:

- $S = \{\text{s}\}$
- $Y(\text{s}) = 0$ - (no time taken to each the starting station)
- $P(\text{s}) =$ - (there is no path taken to get to starting station).

Step 2: We now perform a cut of the network and take a look at all the train lines coming out of the set S (i.e. all the lines that travel from a station inside of S to a station outside of S). For every line we compute

$$(v, w) \in \arg \min_{(i,j)\in\delta^+(S)} \left(Y(i) + l_{ij}\right), \tag{1}$$

where $\delta^+(S)$ denotes a cut of the network corresponding to the lines which travel from a station in S to a station that is outside S. Next, we choose the station w which minimises the above quantity, and then add the station w to the set S and update the label on the station w. We repeat this procedure until all the stations have been visited and every station has been added to the set S.

Worked Example

We will do a worked example here to show how the algorithm should work with the mini network above and find the shortest path from Waterloo to every other station in the network.

Iteration 1:

Initialise the setup.

- $S = \{W\}$
- $Y(W) = 0$
- $P(W) = (W)$.

Iteration 2:

- Perform the cut and and find which lines travel out of the set S. We have that

$$\delta^+(S) = \{(W, E)\}. \tag{2}$$

- Choose the node which minimises $Y(i) + l_{i\to j}$. We have:

$$Y(W) + l_{W\to E} = 0 + 8 = 8. \tag{3}$$

  As this is the only quantity, by definition the line (W, E) is the one we choose. Therefore we travel to Euston station and update the data structures.

- Update: $S = \{w, E\}$.
- Update the labels on Euston as Euston is now in S. $Y(E) = 8$, $P(E) = (W, E)$.

Iteration 3:

- Perform the cut and and find which lines travel out of the set S. We have that

$$\delta^+(S) = \{(E, KC), (E, P)\} \tag{4}$$

- Choose the station which minimises $Y(i) + l_{i\to j}$. We have:

$$Y(E) + l_{E\to KC} = 8 + 4 = 12. \tag{5}$$

$$Y(E) + l_{E\to P} = 8 + 6 = 14. \tag{6}$$

  Clearly travelling to Kings Cross is cheaper, so we add Kings Cross to S and add a label to Kings Cross.

- Update: $S = \{w, E, KC\}$.
- Update the labels on Kings Cross as Kings Cross is now in S. $Y(KC) = 12$, $P(KC) = (W, E, KC)$.

Iteration 4:

- Perform the cut and and find which lines travel out of the set S. We have that

$$\delta^+(S) = \{(KC, P), (E, P)\} \tag{7}$$

- Choose the station which minimises $Y(i) + l_{i\to j}$. We have:

$$Y(KC) + l_{KC\to P} = 12 + 1 = 13. \tag{8}$$

$$Y(E) + l_{E\to P} = 8 + 6 = 14. \tag{9}$$

  Clearly travelling to Paddington is cheaper as it is the only option, but the route we take is via Kings Cross as that gives the minimal time taken. Therefore we add Paddington to S and update the labels.

- Update: $S = \{w, E, KC, P\}$.
- Update the labels on Paddington as Paddington is now in S. $Y(P) = 13$, $P(P) = (W, E, KC, P)$.

Iteration 5

Algorithm terminates as all stations are in S.

To find the shortest path from a designated starting station to any station in the network, we simply read off the label from the algorithm. For example, for the shortest path from Waterloo to Paddington, we read off the Paddington label to obtain: The time taken is 13 minutes and the route is (Waterloo → Euston, Euston → Kings Cross, Kings Cross → Paddington).

YOUR TASK

Your task is to write a Python script that, given the rail network in Figure 2, computes the shortest paths within the network given any starting station. Your script can be written in anyway you like - except it must follow the Dijkstra algorithm presented above. You will be assessed on the following criteria:

- **[30 points]** Correctness and efficiency of the algorithm. Does your algorithm produce the correct result for the network in question? Your algorithm will be tested from a selection of starting stations. Is your algorithm written succinctly and efficiently?
- **[15 points]** Reusability of the algorithm. How easy is your algorithm to apply to another network?
- **[20 points]** Documentation of code. How well is your code documented, can someone who has not written your code understand how to use it?
- **[15 points]** Testing of your algorithm. Before applying your algorithm to the large network, have you tried any tests on a smaller problem as a coding sanity check? TFL do not like applying unknown algorithms to their journey planners.

You must submit a single python file with the code and algorithm contained inside. In your script, you must show how your algorithm works by calculating AT LEAST 4 different journeys within the network. The output of your algorithm must show the length of time the journey takes and the path that one should follow. See the detailed example above. An example output considering the mini-network could be:

$$\begin{aligned} \text{Starting station} &: Waterloo \\ \text{Destination station} &: Paddington \\ \text{Time} &: 13 minutes \\ \text{Route} &: W \to E \to KC \to P. \end{aligned} \tag{10}$$

Along with your script you must submit a document detailing how your group has worked on the task. You must include details of every meeting that you had as a team, who in the group has been delegated each task and any problems that you have encountered along the way. At the end of the document, you must summarise your progress and discuss any limitations of the algorithm that you have written and how it could be improved. You must also make some comments on the following question: Does this route finder accurately model the real world problem trying to be solved?

Your coding will be worth 80 points and your document will be worth 20 points.

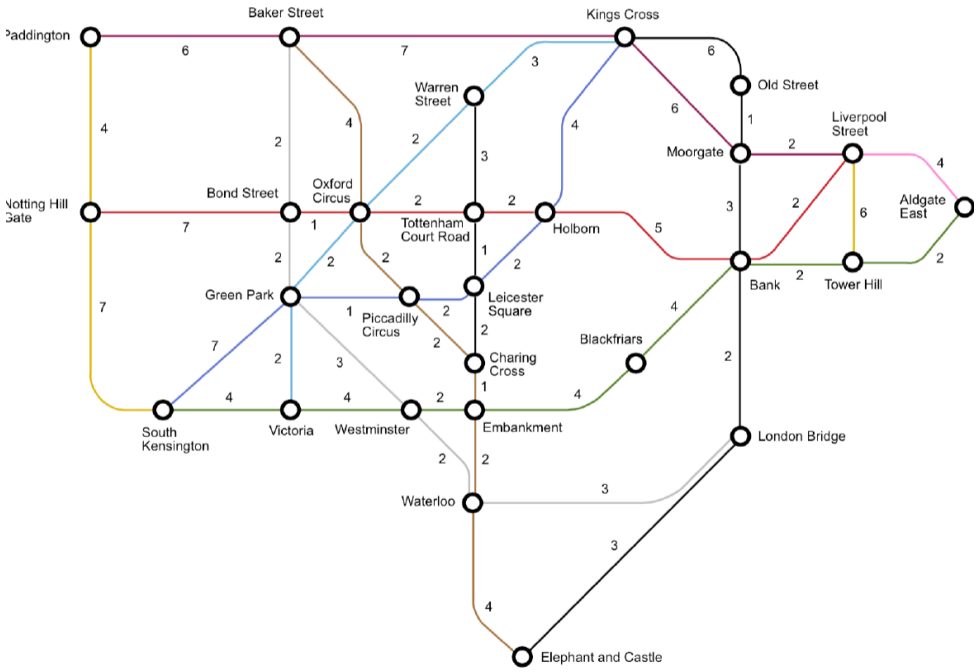

Figure 2: The network you must apply your algorithm to.