Report on Group Assignment of Introduction to Python MATH6005

Submitter: Group 28

Deran Kong, Wenkai Peng, Sachin Suresh, Leyi Yang

1. Introduction:

In response to the task of designing an efficient route finder for a rail network based on Dijkstra's algorithm, our team, comprised of four dedicated members, embarked on a collaborative effort to develop a robust and scalable solution. This document details our team's experience, outlining the responsibilities given to each member, the format of our meetings, and the difficulties we had while working on the project. From the initial exploration of Dijkstra's algorithm to the finalization of a working script, our collective goal was to create a route finder that not only meets the provided specifications but also demonstrates adaptability to real-world rail network scenarios. The document explores the work that our team has done, highlights the shortcomings of our current implementation, suggests some possible enhancements, and ends with some thoughts on how well our route finder captures the intricacies of real-world transportation networks.

2. Process record:

First meeting (December 15th):

In this meeting, we discussed the following points:

- Principle of Dijkstra Algorithm and implement in code.
- How to save the transport network data in Python.
- Task allocation.

After this meeting, we planned to pair up and write two codes and then compare. Our task allocation is followed:

Table 1 Task allocation

Wenkai Peng Sachin Suresh	Coding	Write document
Deran Kong Leyi Yang	Coding	Make transport network

Second meeting (December 22th):

- Compare our codes and determine the final code.
- Test the functions in the smaller network.
- Apply the functions to the target network.
- Discuss how to write the document.
- In postscript, we also tested the codes from every station to every station.

3. Problems in progress:

How to generate the network in Python:

The first problem we met is saving the network data in Python. At first, we planned to use matrix to save the network, but after reviewed relevant tutorials, we decided to save the network in a dictionary. In this way,

we did not need to consider stations without connection, and it was better to simulate the network.

How to optimize the code to reduce computation:

At start we used many if statements and variables to realize the algorithm, it took us long time to optimize our code to implement it in a compact way. And we also used heap to reduce computational complexity.

4. Summary:

Our team successfully applied Dijkstra's algorithm to the train network, producing an efficient script with a logical structure. In the progress, we had clear task allocation and we wrote the code in pairs to learn from each other, and after comparison we could operate our final code. To ensure its robustness, the script was thoroughly tested across a wide range of network sizes. To improve clarity and simplify future reuse, code documentation, including comments and explanations, was added.

5. Limitations of algorithm:

There are still some limitations that can be improved:

1. There may be more than one optimal path between two stations (There may be two shortest paths with the same time). But in our code, we just generate one optimal path between two stations. It may be better to generate other possible solutions to give users more options.

2. The script presently functions in a static mode, not taking into account real-time data, dynamic schedule changes, or disruptions. To provide more accurate and up-to-date route recommendations, improve the script to dynamically include real-time issues such as delays, maintenance, and schedule changes.
3. The script's performance may suffer when used with large-scale train networks. Improve the algorithm's scalability by employing techniques such as graph trimming or parallelization. Consider adopting data structures that can handle huge graphs more effectively.

6. Application in reality:

The route finder accurately reflects the core principles of train network traversal. It gives accurate answers for a variety of scenarios and is well-suited for smaller networks. However, for large-scale and real-world scenarios, we recognize the need for additional considerations. We think that this model does not consider the waiting time at stations. To apply the model in real world, it may be better to add the average waiting time at every station into consideration.