Project Report

XvdongWang 518030910021

1. Introduction

In this project, I used QMainWindow as my base class, and there are three dockWidgets corresponding to each mandatory task. I used QSqlite to load the total dataset, and implemented multithreading to load the dataset and plot the trend of traffic inflow and outflow of a station.

2. Implementation Details

(1) load the dataset

I separated this task into two steps. First, load the entire dataset into a database file. Then, import selected data into this program.

To load the dataset, the user needs to click a button called "LoadDataSet". Then a QFileDialog pops out and he/she needs to choose a directory containing the hole *.csv files to load. I used QSqlite to load the dataset into a database file called *metro.db* in the working directory, and I implemented multithreading to do this job, because it may take 2 or 3 minutes to load the entire dataset.

After loading, there will be a *metro.db* file in the working directory, and the program will connect to this database file by a QSqlDatabase* object named *m_db*.

If there already exists the *metro.db* file in the working directory, the program will not build the database file again instead it will just connect to this file. So if the user wants to load another database, he or she needs to delete or rename the current *metro.db* file and then click the "LoadDataSet button".

After loading the dataset, the user can click the "ImportData" button to import the data of his/her interest in the dataset, which will be shown in a QTableView object. The user can choose the start time, end time, specific lineID, stationID, deviceID, status, userID and payType. By default the program will import all of the data in the dataset. I used QqlTableModel object to do this job, and the filter can be implemented with a single sentence: *model->setFilter(filter)*.

I'd like to mention that the data used to plot the traffic trend in task 2 are based on the imported data, that is, the data shown in the QTableView object. So if the user chooses to import the data with stationID 10, and to plot the traffic trend on station 20, warnings will pop out for no suitable data provided.

(2) Plot traffic trend

After loading the dataset and connecting to the database file, traffic trend can be plotted. The user can decide the start time, end time, stationID and time step. I used QSqlQuery object to select the data, which I believe is faster than my hand written code. I also implemented multithreading to do this job, because in my computer to plot the traffic trend from 2019/1/7 0:00 to 2019/1/13 23:59 it will take around 9-10 seconds.

I display the plot with QLineSeries instead of QSplineSeries, because the effect of interpolation can not be shown by the latter way.

I set the minimum number of points to interpolate to be 11. If there are less than 11 points to plot, interpolation will be used. I implemented Lagrange Interpolation, in which another 11 points will be added.

By default, the plot will be displayed in a QGraphView object in the second dockWidget. The user can also choose to display it in a new widget. Also the user can choose always not to implement interpolation.

(3) **Find route**

This task is simpler than the previous two, as a result the third dock widget appears to be quite empty. There are mainly only two widgets and a textBrowser to display the route.

For the input widget, I used a Line Edit and a Spin Box, and connected them together will signals and slots. If the user changes one, the other will change correspondingly. The user can only set the stationID from 0 to 80.

Before finding the route, the user needs to load the load the metro road map first: just click the "Load_roadMap" button and choose the "Metro_roadMap.csv" file in the QFileDialog.

I implemented BFS to search the route, and only one shortest route will be displayed.

3. Results

(1) load the dataset

As I mentioned before, to load the entire database into *metro.db* file, it will take about 2 or 3 minutes, while the time to import data is much faster, which can be ignored. There is only one exception in which it will take around 10 seconds to import the data: the user chooses to filter a specific userID. This is due to the length of userID: more than 30.

(2) Plot traffic trend

To plot the traffic trend from 2019/1/7 0:00 to 2019/1/13 23:59 it will take around 10 seconds. And the figure below is an example of the result:

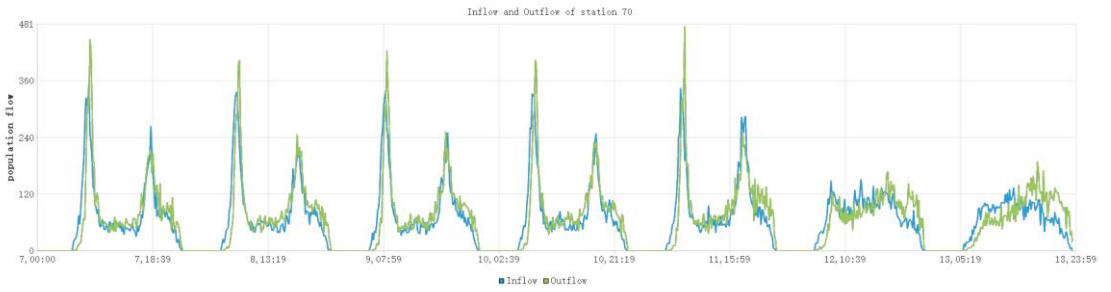

Figure 1: traffic flow in station 70

(3) Find route

The shortest route will be displayed in the text browser.

4. Discussions

For the performance of my program, first I think that loading the dataset takes quite a long time, which is also the reason why I implemented multithreading. The benefits

of loading the dataset into a database file is that the user only needs to do this once. The second time he/she run this program, he/she only needs to connect to the database file, whose time cost can be ignored.

The plotting time is also not very fast. I used QSqlQuery object to select the data, which I believe is faster than my hand written code. I don't know very clearly about the data structure of sqlite database, so I also don't know how data is selected. I've asked some of my classmates about their performance, and I found there is not too much difference.

As for the interesting results I revealed from the data, the most obvious one is as *Figure1* shows. The population flow in most station is much larger during morning and evening rush hours in weekdays than in weekends, and the traffic trend is quite similar during weekdays and during weekends.

Also, as shown below in *Figure2* and *3*, there are stations where most people go in during morning rush hours and go out during evening rush hours (*Figure2* of station 8), while some other stations display the opposite trend (*Figure3* of station 20).

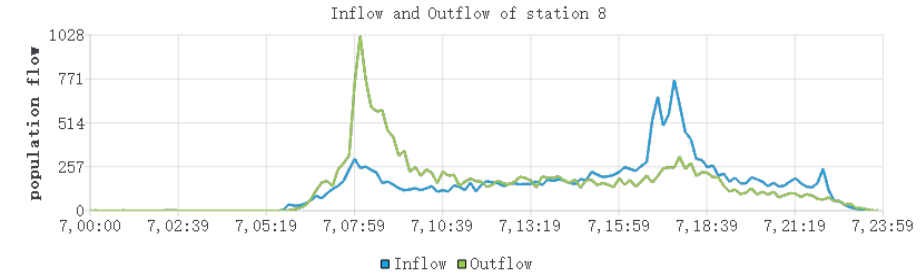

Figure2: traffic flow on station 8

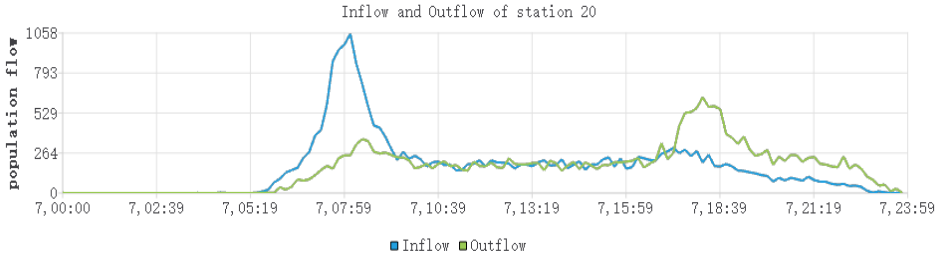

Figure3: traffic flow on station 20

Another interesting result is that the population flow during morning rush hours is usually larger than the flow during evening rush hours. In *Figure2*, at most there are about 1000 people going out of station 8 during morning rush hours within 10 minutes (the time step), while there are only about 770 people going in the station during evening rush hours. From this we can infer that traffic is more concentrated during the morning rush hours, and more dispersed in evening rush hours, which is consistent with common sense:

In the morning, a person has to go to work, and he/she choose the subway, which is often fixed. While after work, in the evening, he/she may choose to return home, also he/she probably wants to have a relax, such as seeing a movie or going out for dinner, as a result the transportation way is diverse.

5. Acknowledgement

First I'd like to thank Dr. Jin and Dr. Ling for their excellent lectures. The algorithm class is usually a bit tedious, while Dr. Jin still gave very vivid lectures. And Dr. Ling always asked us about our understanding. Also I found the reference learning materials quite useful, for example, the CS162 Operating System class in Berkeley.

Then I'd like to thank TA for patiently checking our homework, and when there was something wrong, TA would tell us what went wrong. I'm not very good at coding, and I really appreciate TA looking at my codes.

In short, this is a very meaningful class, and thanks for everyone's help.