Introduction

**The main idea for the data analysis and visualization of Hangzhou Metro Traffic is to use the computer to analyze the current dataset. So in this project, I design a graphic user interface, which allows the user to load the current dataset, select a station and plot the trend of traffic inflow and outflow over time, input the entering station ID and exiting station ID and then plan one or more possible routes for him/her.**

Implementation Details

**1、Load the dataset**

**Similar to importing adjacency matrix, which will be discussed later.**

**2、Draw the traffic flow chart according to the read in. CSV file**

**Each column in the read. CSV file has been stored in the corresponding global array s [75500], S1 [75500], S2 [75500], S3 [75500], S4 [75500], S5 [75500], S6 [75500], and the element type is qstring.**

**The function is implemented as follows:**

void MainWindow::**on\_pushButton\_2\_clicked**()

{

int i, j;

int timeStep=str3.toInt();**//Convert the input of the “timestep” textEdit to int**

int num=0; **//Record how many passengers enter or leave the station in the timestep period**

sort();**//Sort 7 global arrays by time**

QLineSeries \*series = new QLineSeries();**//New a qlineseries instance**

series->append(f(str1),0); **//There are 0 people entering or leaving the station at the specified starting time point**

**//Every timestep from the start time point to the end time point, count the number of inbound or outbound people in the period**

if (flag==1){

for (i=f(str1); i<f(str2); i+=timeStep){

for (j=0; j<75000; j++)

**//Judge the arrival time of the j-th element in which time period; whether it is from the station queried by the user;**

if (f(s[j])>i && f(s[j])<(i+timeStep) && s4[j]=="1" && s2[j]==str4)

num++;

series->append(i,num);

num=0;}}

else{

for (i=f(str1); i<f(str2); i+=timeStep){

for (j=0; j<75000; j++)

if (f(s[j])>i && f(s[j])<(i+timeStep) && s4[j]=="0" && s2[j]==str4)

num++;

series->append(i,num);

num=0;}}

QChart \*chart = new QChart();

chart->legend()->hide();**//Hide legend**

chart->addSeries(series); **//Associate series with qchart and render the data**

series->setUseOpenGL(true); **//Open OpenGL**

chart->createDefaultAxes();**//Create the default coordinate system (Cartesian coordinates)**

chart->setTitle(QStringLiteral("Traffic Flow"));**//Set chart titl**e

QChartView \*view = new QChartView(chart);

view->setRenderHint(QPainter::Antialiasing); **//Turn on anti aliasing for better display**

view->resize(400,300);

view->show();

chart->show();**//Show chart**}

**//Convert time to minutes（0～1439）**

int **f**(const QString s){

int hour, min;

hour=(s[11].unicode()-'0')\*10+s[12].unicode()-'0';

min=(s[14].unicode()-'0')\*10+s[15].unicode()-'0';

return hour\*60+min;}

**3、Route planning**

**The function is implemented as follows:**

void MainWindow::**on\_pushButton\_3\_clicked**()

{

**//Import adjacency matrix**

QString fileName = QFileDialog::getOpenFileName(this, tr("Excel file"), qApp->applicationDirPath (),tr("Files (\*.csv)"));

QFile file(fileName);

if (!file.*open*(QIODevice::ReadOnly))

return;

else{

QStringList list;

list.clear();

QTextStream in(&file);

int i =0;

while(!in.atEnd())

{

QString fileLine = in.readLine();

list = fileLine.split(",", QString::SkipEmptyParts);

**//Except the header, all elements in the adjacency matrix are stored in a global two-dimensional array**

if(i) {

for(int a = 1; a < 82; a++)

{

M[i][a]=list.at(a);

}

}

i++;

}

file.*close*();

test();

**//Output route to textBrowser**

ui->textBrowser->insertPlainText("顶点");

QString s = QString::number(headNode.key, 10);

ui->textBrowser->insertPlainText(s);

ui->textBrowser->insertPlainText("到顶点");

s = QString::number(endNode.key, 10);

ui->textBrowser->insertPlainText(s);

ui->textBrowser->insertPlainText("路径数目为：");

s = QString::number(pathNum, 10);

ui->textBrowser->insertPlainText(s+"\n");

for (int i=0;i<pathNum;i++)

{

s = QString::number(i+1, 10);

ui->textBrowser->insertPlainText("第"+s+"条: ");

for(int j=0;j<NUM;j++)

{

if (resultPath[i][j]==0)

{

break;

}

s = QString::number(resultPath[i][j], 10);

ui->textBrowser->insertPlainText(s+" ");

}

ui->textBrowser->insertPlainText("\n");

}

return;

}

void **FindAllPath**(int Matrix[NUM][NUM],Node startNodeKey,Node endNodeKey){

result[nPos]=startNodeKey.key; **//Put the start node in the result set**

Mark[startNodeKey.key-1]=true; **//Mark start node as accessed**

nPos++; **//Result set index plus 1**

while(nPos!=0)

{

int tempVal=result[nPos-1]; **//Get previous element**

if (tempVal==endNodeKey.key) **//If the current element is a termination node**

{

for (int j=0;j<nPos;j++)

resultPath[pathNum][j]=result[j]; **//Copy the result set to the path matrix**

nPos--; **//Go back to the previous node**

result[nPos]=0; **//Result set index is set to null**

pathNum++; **//Number of paths plus 1**

Mark[endNodeKey.key-1]=false;

break;

}

while(startNodeKey.flag<NUM-1) **//Using flag to indicate the index of each element**

{

if (Matrix[tempVal-1][startNodeKey.flag]==1)

{

if (Mark[startNodeKey.flag]==false) {

Node tempNode;

tempNode.key=startNodeKey.flag+1;

FindAllPath(Matrix,tempNode,endNodeKey);**//DFS**，

}

startNodeKey.flag++;**//Add one corresponding to index value**

}

if (startNodeKey.flag==NUM-1) **//If it is the last node**

{

nPos--; **//Back up again**

startNodeKey.flag=0; **//The index of the node is set null**

result[nPos]=0; **//The corresponding index in the result is set null**

Mark[startNodeKey.key-1]=false; **//Mark as not accessed after access. Because the following element has been accessed, it is convenient for next visit**

break;

}

void **test**()

{

int Matrix[NUM][NUM];

int i, j;

**//Mapping adjacency matrix M to matrix Matrix**

for (i=0; i<NUM-1; i++)

for (j=0; j<NUM-1; j++)

Matrix[i][j]=M[i+1][j+1].toInt();

headNode.key=str6.toInt();**//start node**

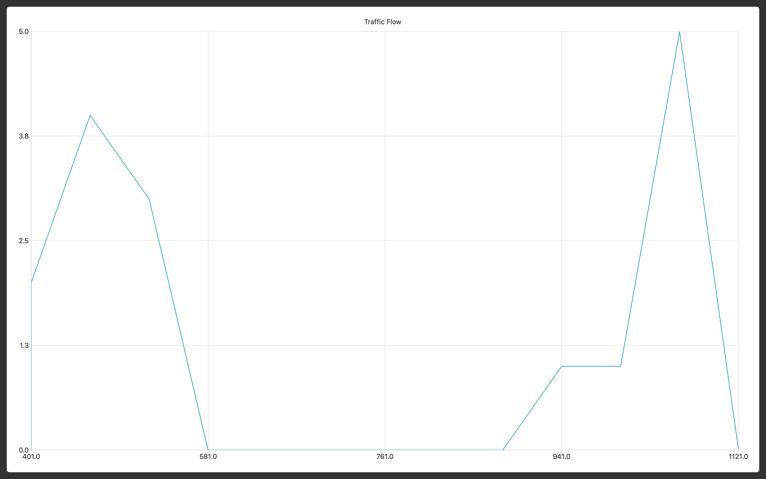
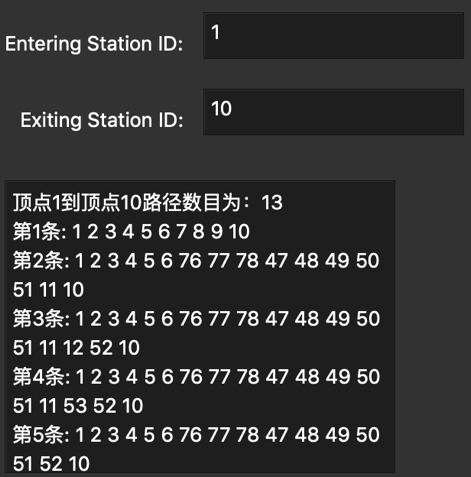
headNode.flag=1;

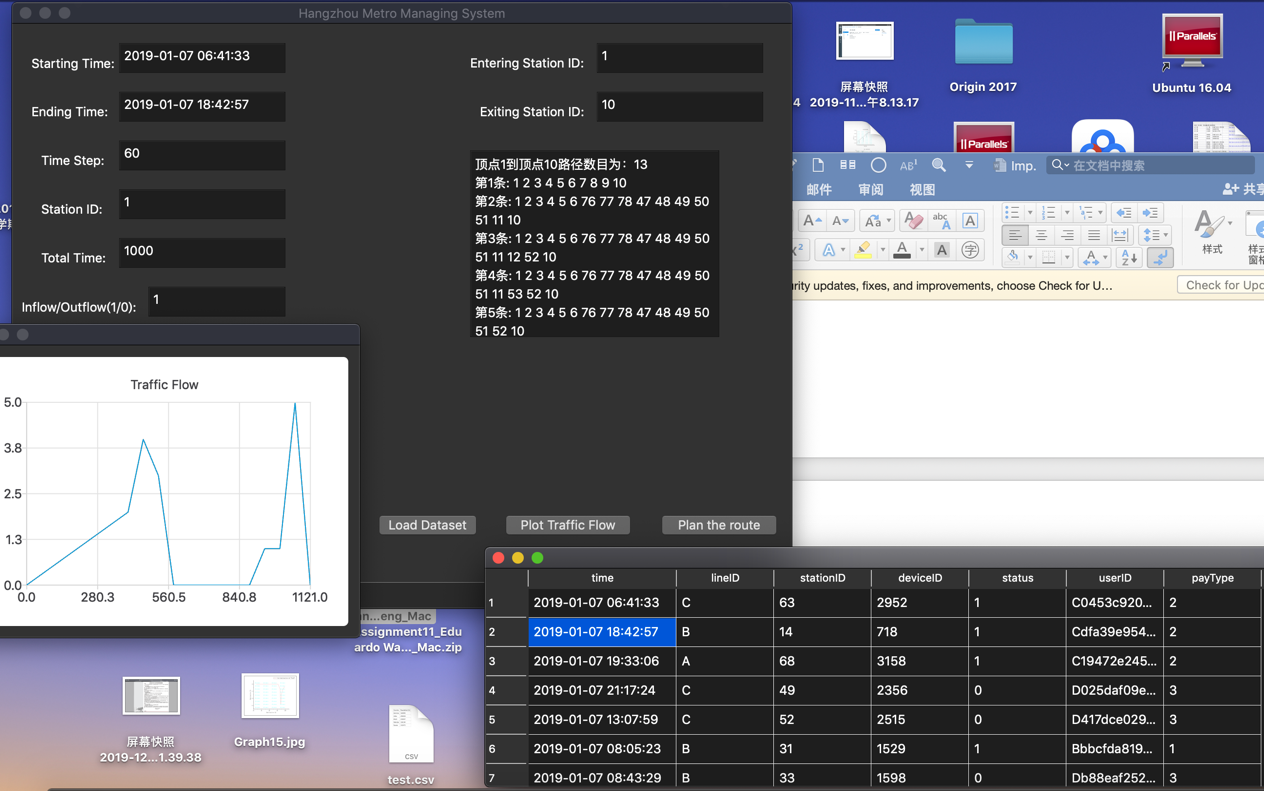
endNode.key=str7.toInt();**//end node**

FindAllPath(Matrix,headNode,endNode);

}

Results





Discussions

1. **Why does the compiler report errors for such statements：a=f(s[j])?**
2. **The performance of the application: Time complexity of the algorithm for loading the dataset is O(n), spatial complexity is O(n); Time complexity of the algorithm for drawing the traffic flow chart according to the read in. CSV file is O(n^2)(because time complexity of the algorithm for sorting is O(n^2)), spatial complexity is O(n^2); Time complexity of the algorithm for route planning is O(n^2) (because time complexity of DFS algorithm is O(n^2)), spatial complexity is O(n^2)**
3. **Static QWidget, its subclass and global QWidget cannot be used in class. Because you must construct a QApplication instance before using the QWidget of UI. However, static and global objects are generated before entering the main function, so they are constructed earlier than the QApplication object in the main function.**
4. **When there is a compilation error, you can sometimes use the rebuild method to solve it. And after adding new .qrc, you should rebuild it.**