同济大学计算机系

数据库课程设计报告

姓名：张天然

学号：1751237

专业：信息安全

任课老师：关佶红

日期：2020 年 6 月 25 日

一、运行及环境搭建说明

开发环境：

操作系统：Windows 10家庭版

集成开发环境：Qt Version 5.9.1

编译器：MinGW 32

开发语言：C++ 11

二、应用功能

线路图查看：

●查看上海地铁网络线路图，包括通过键盘、鼠标拖放、放大缩小等简易查看操作；

●查看地铁线路信息，包括线路段、包含站点等；

●查看地铁站的详细信息，包括站点地理坐标、所属线路等

换乘指南查询：

●提供地铁换乘查询，可通过视图方便的查看乘坐路线和换乘路线；

●提供最小出行时间的换乘策略指南

●提供最小换乘次数的换乘策略指南

动态添加线路：

●动态添加线路，可根据需要新增线路；

●动态添加站点，可根据需要新增站点；

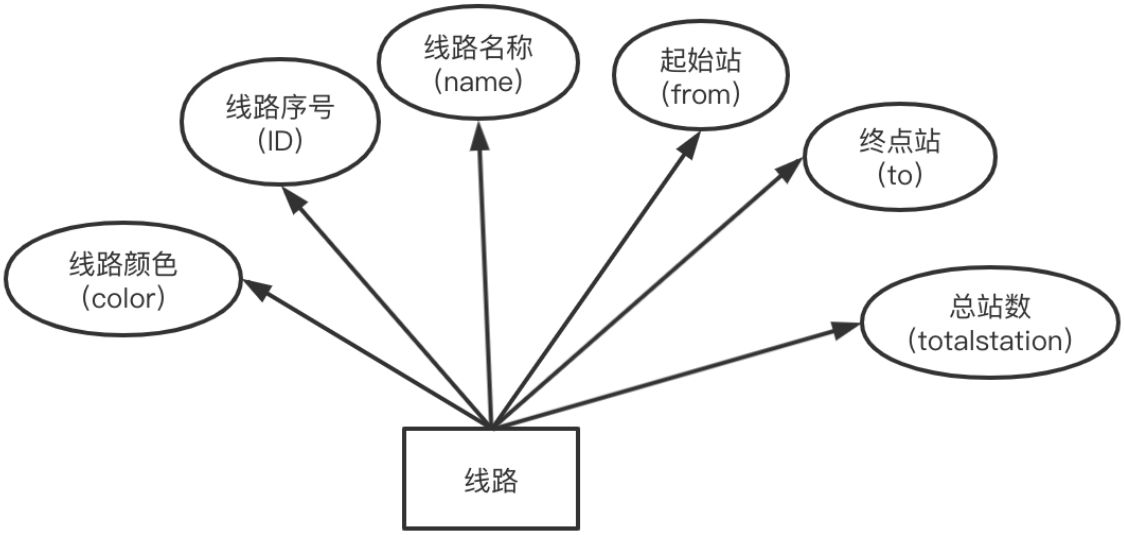
●动态添加连接，可根据需要新增站点连接；

●文本方式简易添加，方便快捷；

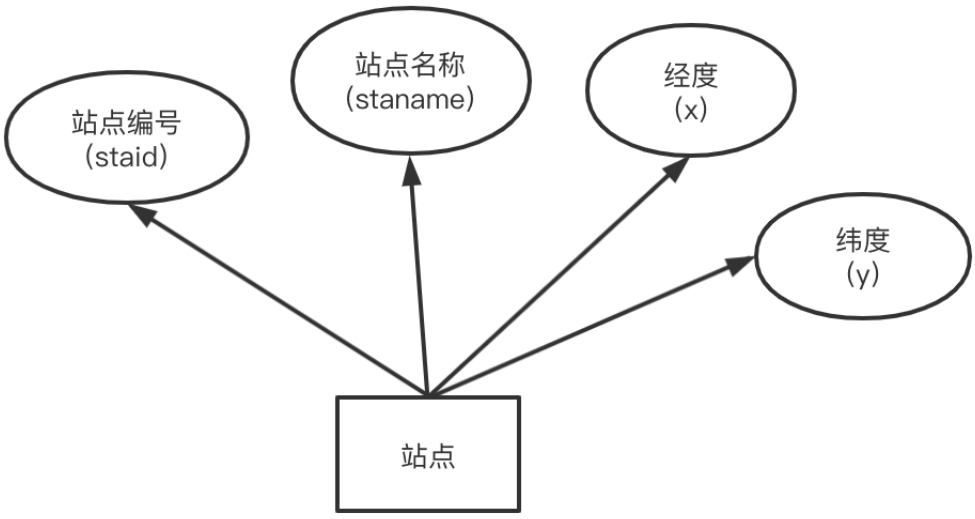
三、概念设计

3.1 E-R图设计

3.1.1 线路



3.1.2 站点



四、逻辑设计

4.1 关系模型设计

4.1.1 线路

1. 实体：线路
2. 属性：线路序号，线路颜色，线路名称，起始站，终点站，总站数
3. 主码：线路序号
4. 关系模型：线路（线路序号，线路颜色，线路名称，起始站，终点站，总站数）

4.1.2 站点

1. 实体：站点
2. 属性：站点编号，站点名称，经度，纬度
3. 主码：站点编号
4. 关系模型：站点（站点编号，站点名称，经度，纬度）

4.2 数据库表结构设计

4.2.1 线路数据库表结构设计

字段名	数据类型	是否主键	可否为空	描述
LineID	int	是	NOT NULL	线路序号
LineName	varchar(20)	否	NOT NULL	线路名称
LineColor	varchar(6)	否	NOT NULL	线路颜色
From	varchar(20)	否	NOT NULL	起始站
To	varchar(20)	否	NOT NULL	终点站
TotalSta	Int	否	NOT NULL	总站数

4.2.2 站点数据库表结构设计

字段名	数据类型	是否主键	可否为空	描述
StaID	varchar(10)	是	NOT NULL	站点编号
StaName	varchar(20)	否	NOT NULL	站点名称
X	float	否	NOT NULL	经度
Y	Float	否	NOT NULL	纬度

4.3 代码实现

4.3.1 线路

```
//线路类
class Line
{
protected:
    int id;                     //线路ID
    QString name;               //线路名称
    QColor color;               //线路颜色
    QVector <QString> fromTo;   //线路起始站点
    QSet<int> stationsSet;      //线路站点集合
    QSet<Edge> edges;           //线路站点连接关系

public:
    //构造函数
    Line(){};
    Line(QString lineName, QColor lineColor):name(lineName), color(lineColor)
    {};

    //声明友元
    friend class SubwayGraph;
    friend class QTextStream;
};
```

4.3.2 站点

```
//地铁站点类定义
class Station
{
protected:
    int id;                     //站点ID
    QString name;               //站点名称
    double longitude, latitude; //站点经纬度
    QSet<int> linesInfo;        //站点所属线路
```

```
    //所有站点的边界位置
    static double minLongitude, minLatitude, maxLongitude, maxLatitude;

public:
    //构造函数
    Station();
    Station(QString nameStr, double longi, double lati, QList<int> linesList);

protected:
    //求取站点间实地直线距离
    int distance(Station other);

    //声明友元
    friend class SubwayGraph;
    friend class QTextStream;
};
```

4.3.3 线路管理

```
class ManageLines : public QDialog
{
    Q_OBJECT

private slots:
    //线路编辑内容改变
    void on_lineEditLineName_textChanged(const QString &arg1);
    //点击选择颜色按钮
    void on_pushButtonChooseColor_clicked();
    //站点编辑内容改变
    void on_lineEditStationName_textChanged(const QString &arg1);
    //经度编辑内容改变
    void on_doubleSpinBoxLatitude_valueChanged(double arg1);
    //纬度编辑内容改变
    void on_doubleSpinBoxLongitude_valueChanged(double arg1);
    //列表部件选择项改变
    void on_listWidget_itemClicked(QListWidgetItem *item);

public:
    //构造函数
    explicit ManageLines(QWidget *parent = 0);
    //析构函数
    ~ManageLines();
    //设置所有部件可见
    void setAllVisible();
    //设置添加线路部件可见
    void setAddLineVisible();
    //设置添加站点部件可见
```

```
    void setAddStationVisible();
    //设置添加连接部件可见
    void setAddConnectionVisible();
    //设置文本添加部件可见
    void setAddByTextVisible();
    //更新线路列表信息
    void updateLinesListWidget();
    //更新选择部件
    void updateComboBox();

protected:
    Ui::ManageLines *ui;                //UI
    QVector<QWidget*> tabWigetsVector;  //保存tab部件指针
    QVector<QIcon> tabIconVector;       //保存tab部件Icon
    QVector<QString> tabTextVector;     //保存tab部件标题

    QString lineName;                   //保存输入线路名
    QColor lineColor;                   //保存输入线路颜色
    QString stationName;                //保存输入站点名
    double longitude;                   //保存输入站点经度
    double latitude;                    //保存输入站点纬度
    QList<QString> linesNameList;       //保存选择线路名表
    QList<QString> linesSelected;       //保存选择的线路名
    QList<QString> stationsNameList;    //保存选择站点名表

    //声明友元
    friend class MainWindow;
};
```

五、物理设计

5.1 用户查询特定线路

【事务内容】

用户在主页面时，可以选择查看地铁1-16号线路的全观，需要访问LineID，FirstStation，TerminalStation，StationID，StationName，Longitude，Latitude。连接的属性为LineID，即线路编号，则可以在访问线路编号时，同时通过查询站点的LineID属性，确定在同一线路上的站点，从而连贯的访问线路上所有站点。如果还想查询起点、终点的信息，则可以通过LineID在起点、终点两张表中查询到相关信息。

【操作频率】

1000次/天

【性能要求】

1s之内完成

5.2 用户查询站点信息

【事务内容】

查询站点非常简单，只需要查询站点对应的Station表即可。可以访问Station的属性Station ID，StationName，Longitude，Latitude，LineID，ExitAmount。如果还需要查询站点所在路线，那么根据级联的Line ID属性查询对应的Line表即可。

【操作频率】

1000次/天

【性能要求】

1s内完成

5.3 用户查询班次信息

【事务内容】

查询班次的操作和查询站点的操作不太相同，虽然都可以通过查询班次编号来定位到Run表中的相应条目，但是查询班次编号的操作可行性比较差，大多数时候是通过线路编号和出发时间来确定的。因此根据级联的线路编号属性Line ID和LeavingTime可以确定需要查询的班次信息。

【操作频率】

1000次/天

【性能要求】

1/s内完成

5.4 用户查找目标线路

【事务内容】

此操作是系统的核心操作。该操作需要访问一条或多条线路。

要完成此操作，在接收到用户输入的两个站点之后，立即在Station表中查询两个站点A和B。若两者在同一线路上，则显示该线路从A到B的段落。若其中一个站点查询到多个对应站点（站点接入了多条线路），则逐一进行比较，确认是否在同一线路上。

若两站点不在同一线路上，此时需要运用迪杰斯特拉算法进行计算，此处不做详述。

【操作频率】

100次/天

【性能要求】

5s内完成

5.5 添加、修改线路信息

【事务内容】

若线路有变更，可以修改相关信息，需要访问Line表。如果需要修改线路的站点信息，则还需要访问Station表。如果删除了Station表或Line表中的某些信息，那么级联对应的信息也要删除。

【操作频率】

50次/天

【性能要求】

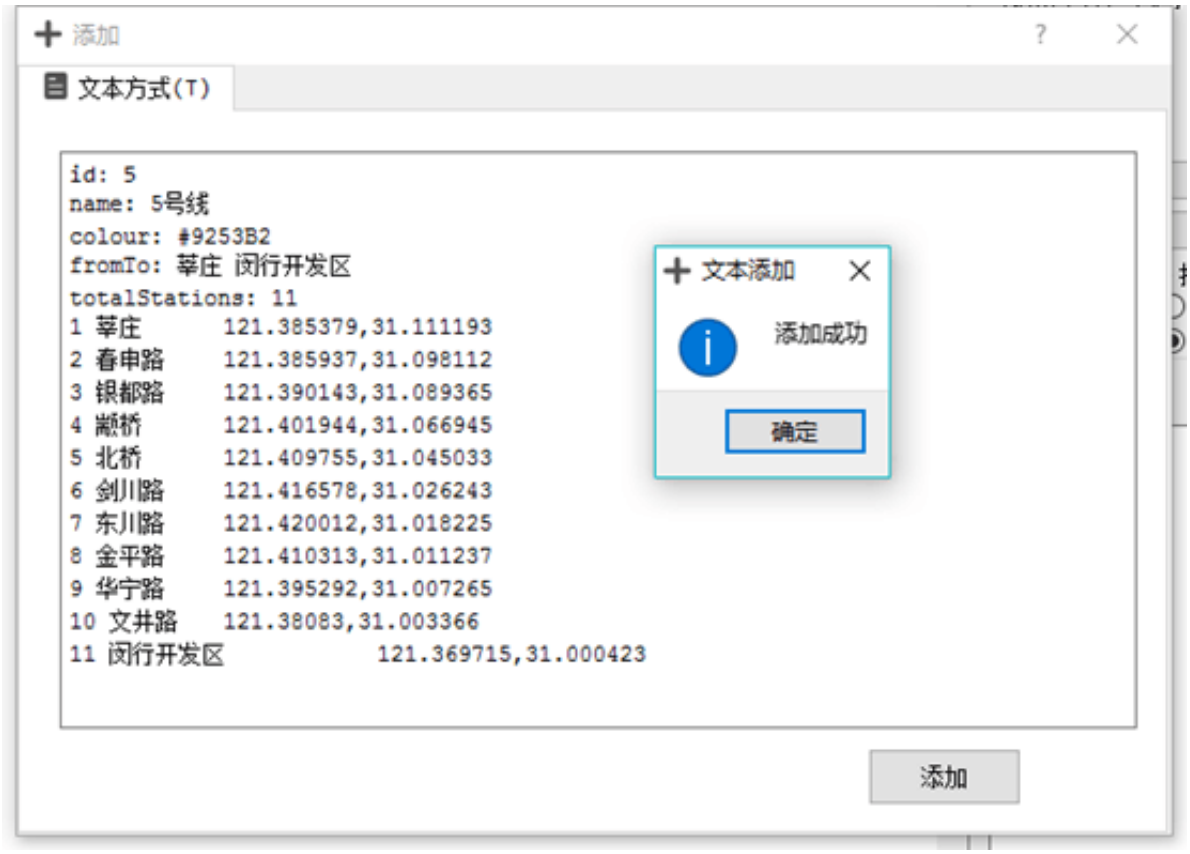
5s内完成

六、详细设计

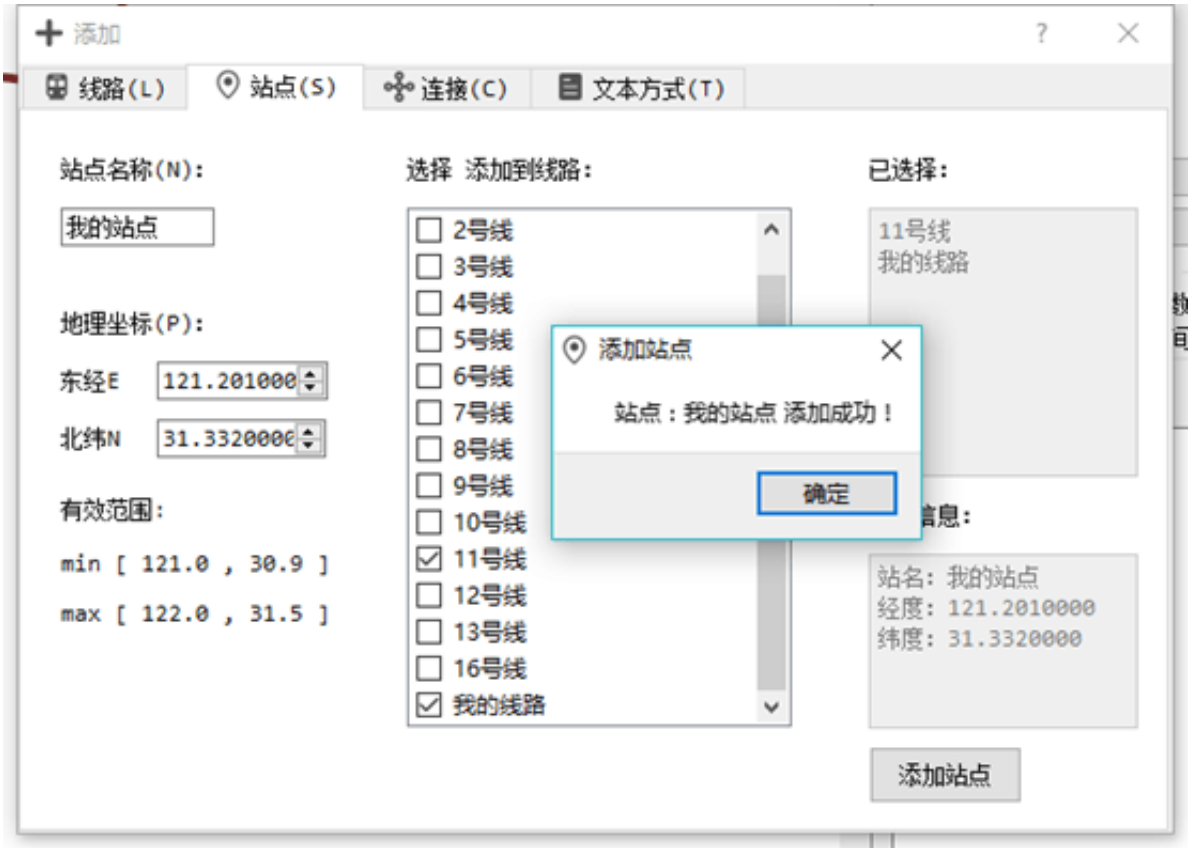
6.1 UI设计

6.1.1 换乘查询

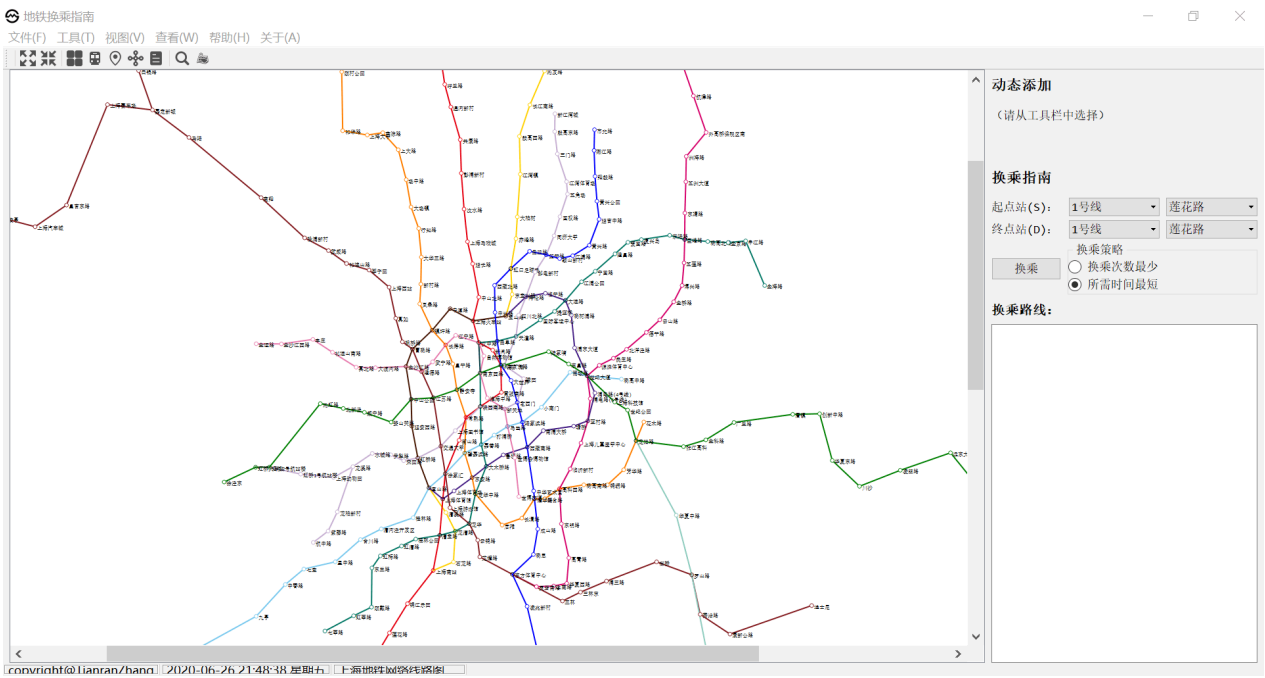
6.1.2 添加线路



6.1.3 添加站点



6.1.4 整体UI



6.3 后端设计

```
//后端管理类
class SubwayGraph
{
protected:
    QVector<Station> stations;          //存储所有站点
    QVector<Line> lines;                //存储所有线路
    QHash<QString, int> stationsHash;   //站点名到存储位置的hash
    QHash<QString, int> linesHash;      //线路名到存储位置的hash
    QSet<Edge> edges;                   //所有边的集合
    QVector<QVector<Node>> graph;       //地铁线路网络图

public:
    //构造函数
    SubwayGraph();

    //获取线路名
    QString getLineName(int l);
    //获取线路颜色
    QColor getLineColor(int l);
    //获取线路hash值
    int getLineHash(QString lineName);
    //获取线路集合hash值
    QList<int> getLinesHash(QList<QString> linesList);
    //获取线路名集合
    QList<QString> getLinesNameList();
    //获取线路的所有包含站点
    QList<QString> getLineStationsList(int l);
```

```
    //获取站点名
    QString getStationName(int s);
    //获取站点地理坐标
    QPointF getStationCoord(int s);
    //获取站点最小坐标
    QPointF getMinCoord();
    //获取站点最大坐标
    QPointF getMaxCoord();
    //获取站点所属线路信息
    QList<int> getStationLinesInfo(int s);
    //获取两个站点的公共所属线路
    QList<int> getCommonLines(int s1, int s2);
    //获取站点hash值
    int getStationHash(QString stationName);
    //获取站点集合hash值
    QList<QString> getStationsNameList();

    //添加新线路
    void addLine(QString lineName, QColor color);
    //添加新站点
    void addStation(Station s);
    //添加站点连接关系
    void addConnection(int s1, int s2, int l);

    //获取网络结构，用于前端显示
    void getGraph(QList<int>&stationsList, QList<Edge>&edgesList);
    //获取最少时间的线路
    bool queryTransferMinTime(int s1, int s2,
                              QList<int>&stationsList,
                              QList<Edge>&edgesList);
    //获取最少换乘的线路
    bool queryTransferMinTransfer(int s1, int s2,
                                  QList<int>&stationsList,
                                  QList<Edge>&edgesList);
    //从文件读取数据
    bool readFileData(QString fileName);

private:
    //清空数据
    void clearData();
    //插入一条边
    bool insertEdge(int s1, int s2);
    //更新边界经纬度
    void updateMinMaxLongiLati();
    //生成图结构
    void makeGraph();
};
```

6.4 算法设计

最短路径的计算由Dijikstra算法完成，代码如下：

```
bool SubwayGraph::queryTransferMinTime(int s1, int s2,
QList<int>&stationsList, QList<Edge>&edgesList)
{
#define INF 999999999
    stationsList.clear();
    edgesList.clear();

    if(s1==s2)
    {
        stationsList.push_back(s2);
        stationsList.push_back(s1);
        return true;
    }
    makeGraph();

    std::vector<int> path(stations.size(), -1);
    std::vector<double> dist(stations.size(), INF);
    dist[s1]=0;
    std::priority_queue<Node, std::vector<Node>, std::greater<Node>> priQ;
    priQ.push(Node(s1, 0));
    while(!priQ.empty())
    {
        Node top=priQ.top();
        priQ.pop();
        if(top.stationID==s2)
        {
            break ;
        }

        for (int i=0; i<graph[top.stationID].size(); ++i)
        {
            Node &adjNode=graph[top.stationID][i];
            if(top.distance+adjNode.distance<dist[adjNode.stationID])
            {
                path[adjNode.stationID]=top.stationID;
                dist[adjNode.stationID]=top.distance+adjNode.distance;
                priQ.push(Node(adjNode.stationID, dist[adjNode.stationID]));
            }
        }
    }

    if(path[s2]==-1)
    {
```

```
        return false;
    }
    int p=s2;
    while(path[p]!=-1)
    {
        stationsList.push_front(p);
        edgesList.push_front(Edge(path[p],p));
        p=path[p];
    }
    stationsList.push_front(s1);

//    qDebug()<<"s1="<<s1<<" s2="<<s2<<" size= "<<stationsList.size()<<" "
<<edgesList.size()<<"\n";
    return true;
}

//获取最少换乘的线路
bool SubwayGraph::queryTransferMinTransfer(int s1, int s2,
QList<int>&stationsList, QList<Edge>&edgesList)
{
    stationsList.clear();
    edgesList.clear();

    if(s1==s2)
    {
        stationsList.push_back(s2);
        stationsList.push_back(s1);
        return true;
    }

    std::vector<bool> linesVisted(lines.size(),false);
    std::vector<int> path(stations.size(),-1);
    path[s1]=-2;
    std::queue<int> que;
    que.push(s1);

    while(!que.empty())
    {
        int top=que.front();
        que.pop();
        for (auto &l: stations[top].linesInfo)
        {
            if(!linesVisted[l])
            {
                linesVisted[l]=true;
                for (auto &s: lines[l].stationsSet)
                {
                    if(path[s]==-1)
                    {
```

```
                        path[s]=top;
                        que.push(s);
                    }
                }
            }
        }
    }

    if(path[s2]==-1)
    {
        return false;
    }
    int p=s2;
    while(path[p]!=-2)
    {
        stationsList.push_front(p);
        edgesList.push_front(Edge(path[p],p));
        p=path[p];
    }
    stationsList.push_front(s1);
    return true;
}
```

七、参考文献

【1】 霍亚飞.Qt Creator快速入门第二版[M].北京：北京航空航天大学出版社,2014-1
【2】 霍亚飞，程梁. QT5 编程入门[M]. 北京：北京航空航天大学出版社,2015-1
【3】 严蔚敏，吴伟民.数据结构（C语言版）[M].北京：清华大学出版社,2007