

## 《数据结构》课程设计总结

学 号 165277

姓 名 郑玄词

专业班级 计算机2班

2018年9月

装

订

线

## 目录

第一部分	算法实现设计说明.....	1
1.1	题目.....	1
1.2	软件功能.....	1
1.3	设计思想.....	1
1.4	逻辑结构与物理结构.....	2
1.4.1	数据结构.....	2
1.4.2	插入结点.....	3
1.4.3	删除结点.....	4
1.4.4	显示操作结果.....	6
1.5	开发平台.....	7
1.6	系统的运行结果分析说明.....	7
1.6.1	调试及开发过程.....	7
1.6.2	成果分析.....	8
1.7	操作说明.....	10
第二部分	综合应用设计说明.....	12
2.1	题目.....	12
2.2	软件功能.....	12
2.3	设计思想.....	12
2.3.1	系统需求.....	12
2.3.2	系统分析.....	13
2.3.3	系统架构模式.....	13
2.3.4	任务分解，逐步求精.....	14
2.3.5	数据结构分析.....	14
2.4	逻辑结构与物理结构.....	14
2.4.1	显示地铁线路图.....	14
2.4.2	定义站点.....	16
2.4.3	后端实现类.....	17
2.5	开发平台.....	18
2.6	系统的运行结果分析说明.....	19
2.6.1	调试及开发过程.....	19
2.6.2	成果分析.....	19
2.7	操作说明.....	26

装

订

线

2.7.1	换乘查询.....	26
2.7.2	动态添加线路.....	27
第三部分	设计总结.....	29
3.1	工作总结.....	29
3.2	软件改进.....	29
3.3	心得与收获.....	29
第四部分	参考文献.....	31

装

订

线

## 第一部分 算法实现设计说明

### 1.1 题目

题号 7：二叉排序树的建立和删除，输入一组关键值，建立相应的二叉排序树，完成结点的查找和删除操作。

要求：

- (1) 可以实现删除根结点、叶子结点以及其它任意结点的功能；
- (2) 可随时显示操作结果。

### 1.2 软件功能

系统包含的主要功能：

1. 建立二叉排序树，输入一个元素后二叉树仍然保证是二叉排序树，即任意结点的左子结点值小于右子结点的值；
2. 删除二叉排序树中的任意指定结点，包括根结点和叶子结点，删除后的二叉树仍然保证是二叉排序树；
3. 随时显示任意操作结果。

### 1.3 设计思想

#### 1. 算法分析<sup>[1]</sup>

算法实现包含三部分：建立二叉排序树、删除二叉排序树中的指定元素、显示操作结果。

二叉排序树的关键是任意结点的左子结点值小于右子结点的值，则没输入一个元素的值都首先与原有二叉树中的元素值进行比较，如果小于则在左子树中继续比较，如果大于则在右子树中继续比较，在子树中重复上述操作，直至找到确切位置则将此结点插入<sup>[1]</sup>。

插入算法如下：

- 1) 若 root 是空树，则将结点 s 作为根结点插入；否则
- 2) 若  $s \rightarrow data < root \rightarrow data$ ，则把结点 s 插入到 root 的左子树中；否则
3. 把结点 s 插入到 root 的右子树中<sup>[1]</sup>。

删除算法如下：

- 1) 若结点 p 是叶子，则直接删除结点 p；

2) 若结点 p 只有左子树, 则只需重接 p 的左子树; 若结点 p 只有右子树, 则只需重接 p 的右子树;

3) 若结点 p 的左右子树均不空, 则

(1) 查找结点 p 的右子树上的最左下结点 s 以及结点 s 的双亲结点 par;

(2) 将结点 s 数据域替换到被删结点 p 的数据域;

(3) 若结点 p 的右孩子无左子树, 则将 s 的右子树接到 par 的右子树上; 否则, 将 s 的右子树接到结点 par 的左子树上;

(4) 删除结点 s;

利用 Qt 的视图控件完成显示功能。

## 1.4 逻辑结构与物理结构

### 1.4.1 数据结构

```
typedef struct TNode
{
    int data;
    TNode*left, *right;
    bool found_cond;
}TNode, *Bisetree;
```

```
/******
```

类 名 称: TNode

功 能: 二叉搜索树

成员及描述: data: 节点数值

left, right: 左右子节点

found\_cond: 被查找状态

说 明:

```
*****/
```

```
class BSTree
```

```
{
```

```
public:
```

```
BSTree();
```

```
status Insert_Bisearchtree(elemtype e, int(*compare)(elemtype, elemtype));
```

```
status Search_Bisearchtree(elemtype e, int(*compare)(elemtype, elemtype));
```

```
status Delete_Bisearchtree(elemtype e, int(*compare)(elemtype, elemtype));
```

```
status Clear_Bisearchtree(Bisetree T);
```

```
Bisetree return_tree();
```

```
private:
    Bisetree tree;
};
```

## 1.4.2 插入结点

```

/*****
函数名称: BSTree::Insert_Bisearchtree
功    能: 插入元素
输入参数: 插入元素内容, 比较函数
返 回 值:
说    明:
*****/
status BSTree::Insert_Bisearchtree(elemtype e, int(*compare)(elemtype, elemtype))
{
    Clear_srch(tree);
    Bisetree t = tree, q;
    if (t == NULL)
    {
        tree = new TNode;
        if (tree == NULL)
        {
            qDebug() << "Space Overflow";
            exit(ERROR);
        }
        tree->data = e;
        tree->left = NULL;
        tree->right = NULL;
        tree->found_cond = NOTFOUND;
        return OK;
    }
    for (; t != NULL;)
    {
        if (compare(t->data, e) == MORE)
        {
            q = t;
            t = t->left;
        }
        else
        {

```

装

订

线

```

        q = t;
        t = t->right;
    }
}

t = new TNode;
if (t == NULL)
{
    qDebug() << "Space Overflow";
    exit(ERROR);
}

t->data = e;
t->left = NULL;
t->right = NULL;
t->found_cond = NOTFOUND;
if (compare(q->data, e) == MORE)
    q->left = t;
else
    q->right = t;
return OK;
}

```

### 1.4.3 删除结点

```

/*****
函数名称: BSTree::Delete_Bisearchtree
功    能: 删除元素
输入参数: 元素名称, 比较函数
返 回 值: 删除结果
说    明:
*****/
status BSTree::Delete_Bisearchtree(elemtype e, int(*compare)(elemtype, elemtype))
{
    Clear_srch(tree);
    Bisetree t = tree, q = tree;
    for (; t != NULL;)
    {
        if (compare(t->data, e) == MORE)
        {
            q = t;
            t = t->left;
        }
    }
}

```

```

else if (compare(t->data, e) == LESS)
{
    q = t;
    t = t->right;
}
else
{
    if (t->left == NULL && t->right == NULL)
    {
        if (q == t)
        {
            tree = NULL;
            return OK;
        }
        delete t;
        if (compare(q->data, e) == MORE)
            q->left = NULL;
        else
            q->right = NULL;
        return OK;
    }
    else if (t->left == NULL)
    {
        Bisetree q = t;
        t = t->right;
        q->data = t->data;
        q->left = t->left;
        q->right = t->right;
        delete t;
        t = NULL;
        return OK;
    }
    else if (t->right == NULL)
    {
        Bisetree q = t;
        t = t->left;
        q->data = t->data;
        q->left = t->left;
        q->right = t->right;
        delete t;
        return OK;
    }
}

```



```

else
{
    Bisetree p = t, s = p->left;
    for (; s->right != NULL;)
    {
        p = s;
        s = s->right;
    }
    t->data = s->data;
    if (p != t)
        p->right = s->left;
    else
        p->left = s->left;
    delete s;
    return OK;
}
}
}
if (t == NULL)
    return ERROR;
return OK;
}

```

## 1.4.4 显示操作结果

/\*\*\*\*\*\*

函数名称: BST\_Demo::paintEvent

功 能: 画二叉树

输入参数:

返 回 值:

说 明: 窗口: x(200-780), y(30-600)

\*\*\*\*\*/

```

void BST_Demo::paint_tree(QPaintEvent *event, Bisetree t, Bisetree parent, int xp, int yp, int deep)
{
    Q_UNUSED(event);
    double r = 17;
    if (t == NULL)
        return;
    if (t == mod->return_tree())
    {
        paint_node(event, r, xp, yp, t->data, t->found_cond);
    }
}

```

```

    paint_tree(event, t->left, t, xp, yp, 1);
    paint_tree(event, t->right, t, xp, yp, 1);
    return;
}
double x, y = yp + 125;
if (t == parent->left)
    x = xp - (140 / deep);
else
    x = xp + (140 / deep);
paint_branch(event, x, y, xp, yp, r);
paint_node(event, r, x, y, t->data, t->found_cond);
paint_tree(event, t->left, t, x, y, deep + 1);
paint_tree(event, t->right, t, x, y, deep + 1);
}

```

装  
订  
线

## 1.5 开发平台

开发平台：

计算机型号：联想 ThinkPadT460

计算机内存：4.00GB

CPU：Intel Core i5 2.6GHz

操作系统：Windows 10 家庭版

开发语言：C++ (C++11 标准以上)

开发框架：Visual Studio + Qt designer

集成开发环境：Qt Version 5.11.1

编译器：MinGW 64bit

运行环境：

可在上述集成环境下运行；

通过 windeployqt.exe 及 Enigma Virtual Box 进行整合压缩为发布为了一个 LinkListVisualizer.exe 文件，可在普通 windows 机型下运行。

## 1.6 系统的运行结果分析说明

### 1.6.1 调试及开发过程

#### 1. 调试

算法开发采用的是新技术框架 Qt，同时也是跨平台的，在 Visual Studio + Qt designer 中开发调试，Qt 中包含了大量的库类，类似于 java 开发简便，Qt 有较好的调试器<sup>[3]</sup>。

## 2. 开发

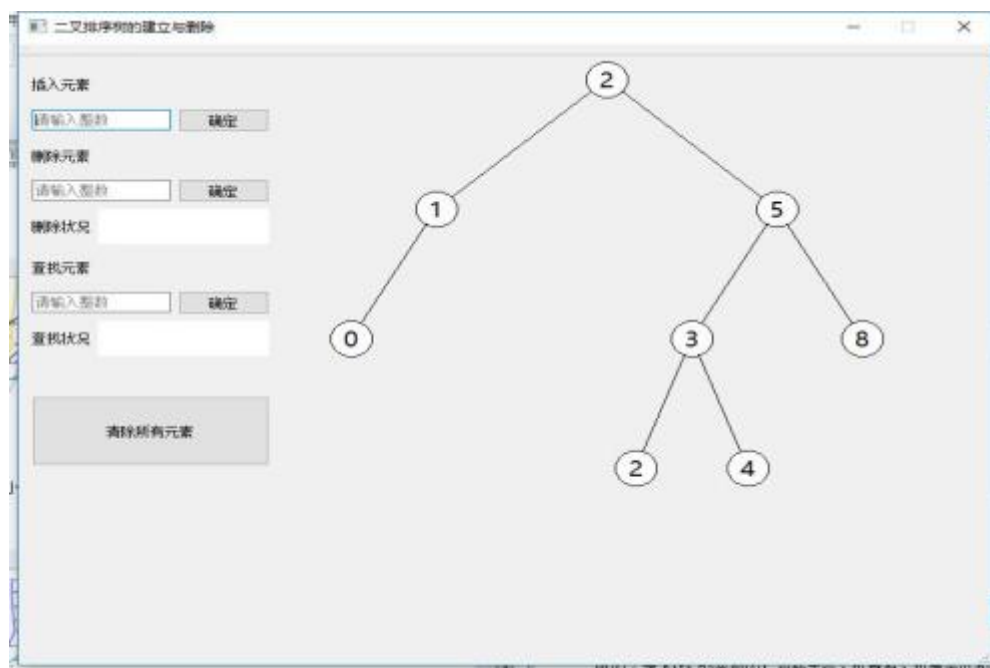
算法有三个小題，分别是在二叉排序树中进行插入、删除操作和显示操作结果，采用分模块逐步开发方式。先确定主窗口，控制整体界面，然后完成二叉排序树设计和开发，二叉排序树完成以后集中完成界面显示。

### 1.6.2 成果分析

#### 1. 正确性

通过采用不同类型数据，进行测试，程序运行正确，与预期没有任何差错。

##### 1) 建立二叉排序树

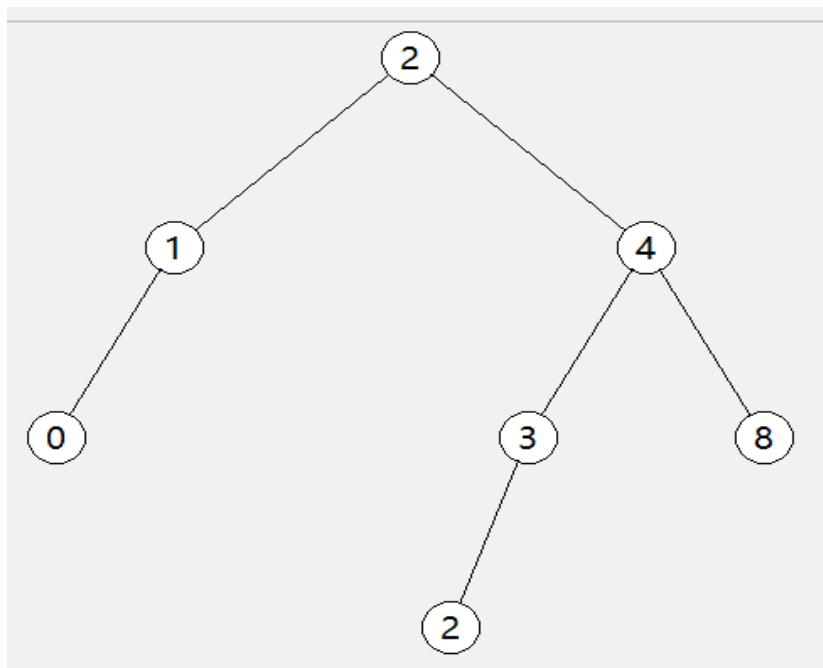


##### 2) 删除结点

删除元素

51

删除状况

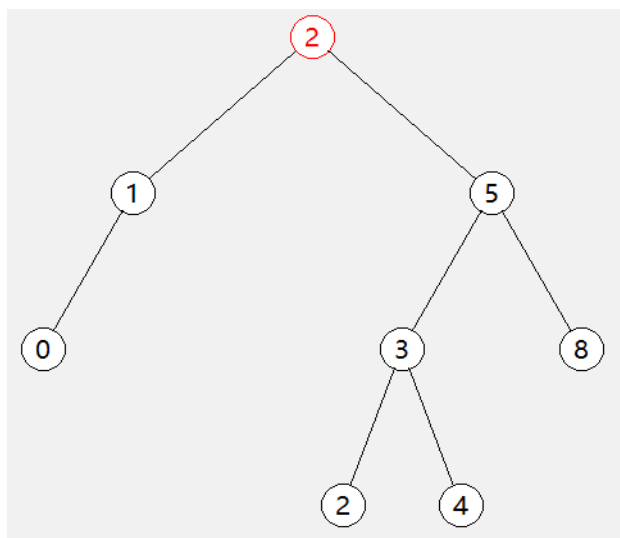


### 3) 查找指定值的元素

查找元素

查找状况

高亮显示找到的元素



查找元素

查找状况

2节点搜索成功

## 2. 稳定性

- 1) 系统响应较快，所有操作都是及时响应和得出结果；
- 2) 每项操作都能及时更新，例如：添加新的线路和站点后，所有用户接口界面中的数据都会进行立马更新，视图也会立马显示新加线路和站点；
- 3) 经过反复测试调试，程序目前无异常情况。

## 3. 容错能力

系统设计中加入了异常情况判断，进行了容错处理，所以系统具有很好的容错能力，不至于导致系统崩溃。一些样例如下：

- 1) 非法输入均有提示：

查找元素

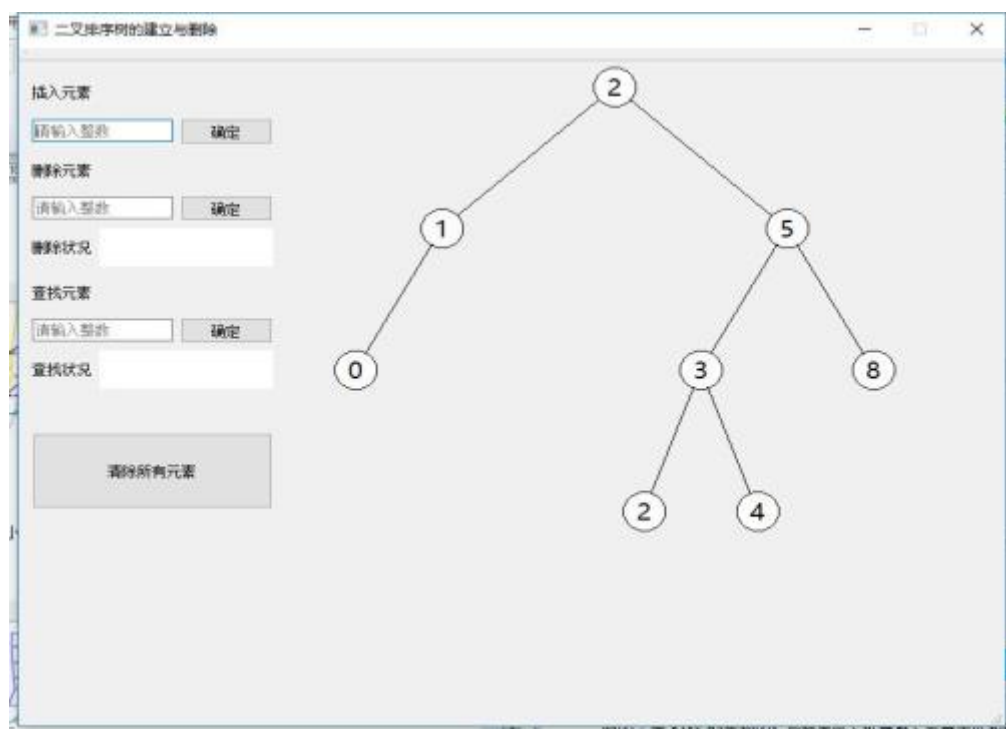
查找状况

0节点搜索失败

## 1.7 操作说明

本题目分为了三个部分，分别是二叉排序树的插入、删除、查找的基本操作演示。通过前面介绍的运行环境下进行程序的运行。

操作比较简单，在一个界面即可操作所有操作，优点是操作简单，不需要切换界面，而且可以直接操作结果显示在同一个界面窗口。



装  
订  
线

## 第二部分 综合应用设计说明

### 2.1 题目

题号 2★★★★：上海的地铁交通网络已经基本成型，建成的地铁线十多条，站点上百个，现需建立一个换乘指南打印系统，通过输入起点和终点站，打印出地铁换乘指南，指南内容包括起点站、换乘站、终点站。

- (1) 图形化显示地铁网络结构，能动态添加地铁线路和地铁站点。
- (2) 根据输入起点和终点站，显示地铁换乘指南。
- (3) 通过图形界面显示乘除路径。

### 2.2 软件功能

软件功能主要包含：通过输入起点和终点站，显示出地铁换乘指南，指南内容包括起点站、换乘站、终点站。

#### 1、线路查询

- 1) 显示上海地铁网络线路图，浏览地铁网络线路图，包括通过键盘、鼠标拖放、查看操作；
- 2) 查找地铁线路信息，包括线路段、包含站点等；
- 3) 查看地铁站的详细信息，包括站点地理坐标、所属线路等；
- 4) 突出显示查到的路线。

#### 2、换乘指南查询：

- 1) 提供地铁换乘查询，可通过视图输入起点和终点显示乘坐路线和换乘路线；
- 2) 显示起点到终点的最短路径的换乘策略指南。

#### 3、添加线路：

- 1) 动态添加线路，可根据需要新增线路；
- 2) 添加有分支的线路和环线。

### 2.3 设计思想

#### 2.3.1 系统需求

- 1) 众所周知上海是国际大都市，商业金融都很发达，占地面积达 6340 万平方公里，主要交

通工具当属地铁，上海的地铁交通很发达，十几条地铁线，交织成网络，四通八达遍布全上海市，包含上百个站点，本系统需要根据输入的起始站点和终点，寻找最短路径，显示最佳换乘指南，显示内容包括起点站、换乘站、终点站。

2) 图形化显示地铁网络结构，能动态添加地铁线路和站点。

## 2.3.2 系统分析

系统分为前后端两部分处理，前端负责功能的界面图形显示，后端通过最短路径算法，处理数据结构计算起点和终点之间的最短路径，如图 2-1 系统结构图。

系统的总体构思即为图最短路径查询，站点即为图中结点。核心算法通过迪杰斯特拉算法求短距离。数据结构采用静态存储线性链表结构。

网络线路的连接情况需用图结构、算法实现有队列，名字到存储位置有哈希映射<sup>[1]</sup>。

采用面向对象的方法通过 C++ 语言程序结合 QT 框架实现。

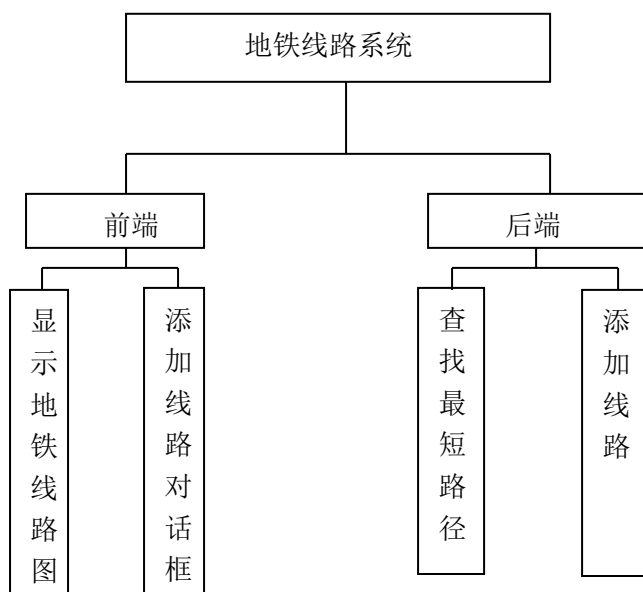


图 2-1 系统结构图

## 2.3.3 系统架构模式

系统采用 MVC 模式架构，即为模型-视图-控制器结构模式。在本次系统设计中，前后端实现分离，前端主要考虑与用户的交互，例如设计视图，地铁网络线路图展现等，后端主要是数据和算法的处理，将复杂的功能和数据交由后端处理，前后端实现函数接口，完成整个软件系统的



架构设计<sup>[2]</sup>。

## 2.3.4 任务分解，逐步求精

系统前端与后端分块设计实现，任务有繁化简，根据系统需求，先进性系统总体分析，根据分析结构进行设计，从前端开始，对每个功能要求的前端界面和交互进行细化，实现前端代码，对每个功能的前端实现后，完善后端接口函数，完成所有前端后，后端也基本同步完成，从而完成整个程序的实现。

## 2.3.5 数据结构分析

系统设计站点、线路及其关系的采用现行链表存储表示，所有线路和站点名字是唯一的，表示线路的网络关系时，每个站点可即为一个节点，站点的连接即为一条边，地铁网络结构显然用图结构存储，对每个站点和线路来讲，不会有删除操作，会有添加需求，重点是查询需求，采用顺序表存放最为合适，将站点和线路名和存储位置的下标值进行哈希关联，可方便访问。<sup>[1]</sup>

## 2.4 逻辑结构与物理结构

### 2.4.1 显示地铁线路图

```
void SubwayDemo::dispfile2display()
{
    QPen pen;
    QVector<QString> namelist;
    QFont font("Microsoft YaHei", 8, 50, false);
    for (int i = 0; i < this->linelist.size(); i++)
    {
        pen.setColor(linelist[i].color);
        scene->addText(this->linelist[i].name->setPos(this->linelist[i].vlist[0].lon,
this->linelist[i].vlist[0].lat);
        for (int j = 0; j < this->linelist[i].vlist.size(); j++)
        {
            scene->addEllipse(this->linelist[i].vlist[j].lon/20-bias_x,
this->linelist[i].vlist[j].lat/20-bias_y, 1, 1, pen);
            int t = 0;
            for (; t < namelist.size(); t++)
```

```

        {
            if (this->linelist[i].vlist[j].name == namelist[t])
                break;
        }
        if (t == namelist.size())
        {
            namelist.insert(namelist.begin(), this->linelist[i].vlist[j].name);

            scene->addText(this->linelist[i].vlist[j].name, font)->setPos(this->linelist[i].vlist[j].
lon / 20 - bias_x, this->linelist[i].vlist[j].lat / 20 - bias_y);
        }
    }
    for (int j = 0; j < this->linelist[i].s2slist.size(); j++)
        scene->addLine(this->linelist[i].vlist[this->linelist[i].s2slist[j].id1].lon /
20 - bias_x,
                        this->linelist[i].vlist[this->linelist[i].s2slist[j].id1].lat / 20 - bias_y,
                        this->linelist[i].vlist[this->linelist[i].s2slist[j].id2].lon / 20 - bias_x,
                        this->linelist[i].vlist[this->linelist[i].s2slist[j].id2].lat / 20 - bias_y,
pen);
    }
}

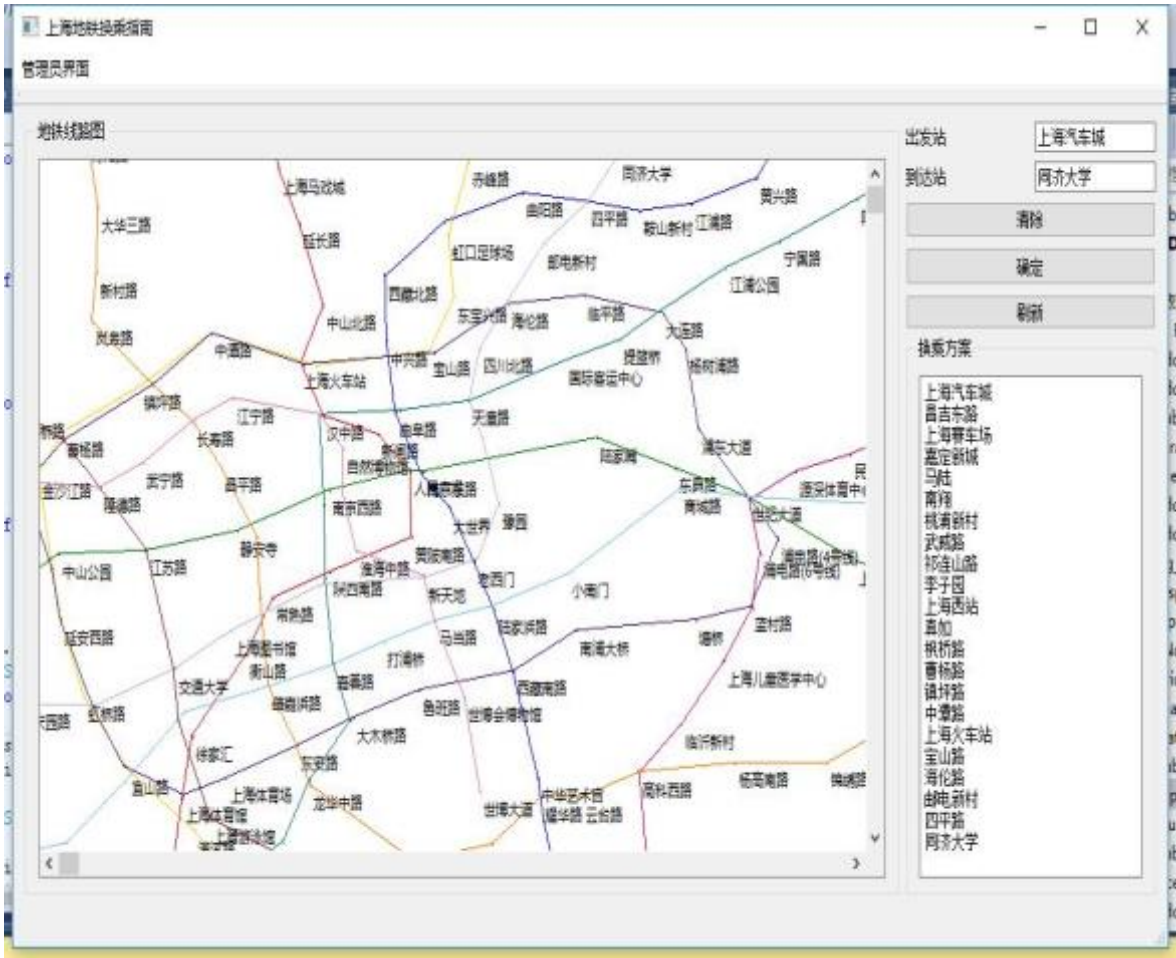
```

装

订

线

装  
订  
线



2.4.2 定义站点

```
struct sta
{
    QString name;
    double lon, lat;
};
struct sta2sta
{
    int id1, id2;
};
struct line
{
    QString name;
    QColor color;
    QVector<sta>vlist;
```

```
QVector<sta2sta>s2slist;
};
```

## 2.4.3 后端实现类

```

/*****
结构体名称: arc_info
功    能: 邻接表边节点信息
成员及描述: adjvex: 对应弧头在顶点表中的位置 (0开头)
power: 边节点权值
说    明:
*****/
struct arc_info
{
    double power;
    int adjvex;
};
/*****
结构体名称: ver_info
功    能: 邻接表边节点信息
成员及描述: color: 节点访问状况 (白, 灰)
            dist: 到起点的距离
            lon: 纬度
            lat: 经度
说    明:
*****/
struct ver_info
{
    int color;
    double dist;
    double XY[2];
    QVector<QString>rd_id;
    QString name;
};
struct ANode
{
    arc_info data;
    ANode*next;
};
/*****
结构体名称: Vnode, *V_p

```

功 能：顶点节点

成员及描述：infor：顶点相关信息

first\_arc：下属第一个边节点指针

pre：依据需求的前驱节点指针

说 明：

\*\*\*\*\*/

```
struct VNode
{
    ver_info* info;
    ANode* firstarc = NULL;
    VNode*pre;
};

class bg_graph
{
public:
    bg_graph();
    ~bg_graph();
    int vexnum;
    int arcnum;
    VNode*vertices;
    status Relax_Dij(VNode&dest, VNode&path);
    status Dijkstra(int start, int end);
};
```

装

订

线

## 2.5 开发平台

开发平台：

计算机型号：联想 ThinkPad T460

计算机内存：4.00GB

CPU：Intel Core i5 2.6GHz

操作系统：Windows 10 家庭版

开发语言：C++（C++11 标准以上）

开发框架：Visual Studio + Qt designer

集成开发环境：Qt Version 5.11.1

编译器：MinGW 64bit

运行环境：

可在上述集成环境下运行；

通过 windeployqt.exe 及 Enigma Virtual Box 进行整合压缩为发布为了一个 LinkListVisualizer.exe 文件，可在普通 windows 机型下运行。

## 2.6 系统的运行结果分析说明

### 2.6.1 调试及开发过程

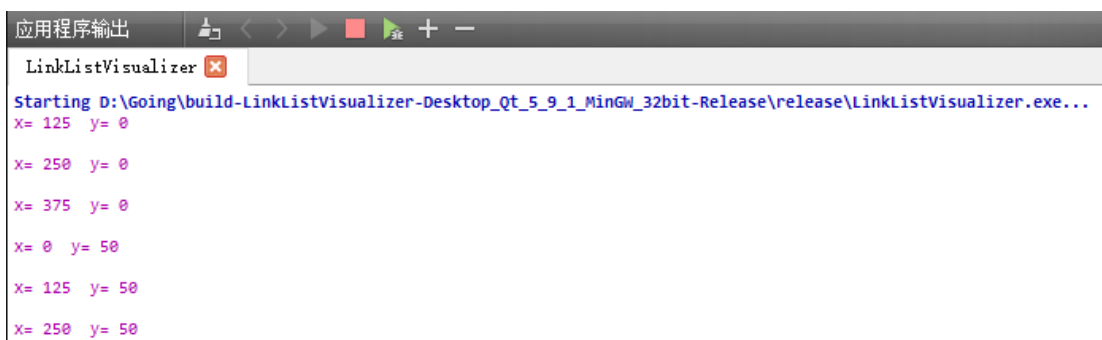
#### 1. 调试

系统开发采用的是新技术框架 Qt，同时也是跨平台的，在 Qt Creator 中开发调试，Qt 中包含了功能丰富的库类，类似于 java 开发简便，Qt 有较好的调试器<sup>[3]</sup>。

例如：

```
//添加节点的GraphicsItem
void LinkList::addLNodeGraphicsItem(LNode *p1, QPoint coord)
{
    int x=coord.x(), y=coord.y();
    qDebug()<<"x="<<x<<" y="<<y<<"\n";
    p1->valueRect =scene->addRect(x,y+SPACING,VALUE_RECT_W,RECT_H,QPen(),LinkList::markBrush);
    p1->pointerRect =scene->addRect(x+VALUE_RECT_W,y+SPACING,POINTER_RECT_W,RECT_H);
    p1->valueText =scene->addText(p1->data,LinkList::dataFont);
    p1->valueText->setPos(x,y+SPACING+5);
}
```

输出：



```
应用程序输出
LinkListVisualizer
Starting D:\Going\build-LinkListVisualizer-Desktop_Qt_5_9_1_MinGW_32bit-Release\release\LinkListVisualizer.exe...
x= 125 y= 0
x= 250 y= 0
x= 375 y= 0
x= 0 y= 50
x= 125 y= 50
x= 250 y= 50
```

#### 2. 开发

通过对 Qt 的不断学习，了解也由浅入深，开发不断熟练。开发时，遵循将后端系统和前端用户交互分开，先思考后端系统的核心数据结构的表示和存储，采用迪杰斯特拉算法对结点数据进行处理，寻找最短路径，实现题目所要求的相应功能，设计核心算法，设计前端和后端的程序接口，搭建整个综合应用程序的主体框架，分模块逐步实现，不断迭代求精，完成整个程序的开发。

### 2.6.2 成果分析

## 1. 正确性

通过采用不同类型数据，进行测试，程序运行正确，与预期没有任何差错。

1) 输入起始站点和终点，查询最短路径，并突出显示：

例如选择“上海汽车城”到“同济大学”，线路如下：

出发站

到达站

清除

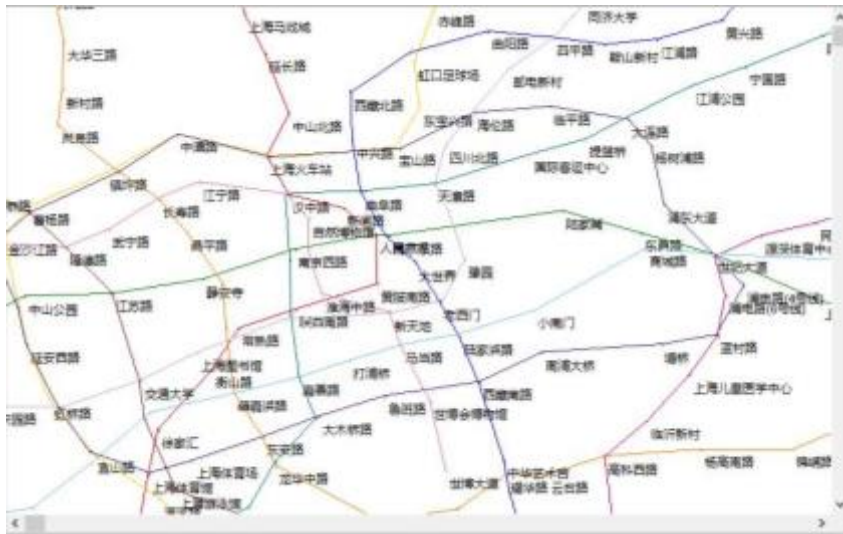
确定

刷新

换乘方案

上海汽车城  
昌吉东路  
上海赛车场  
嘉定新城  
马陆  
南翔  
桃浦新村  
武威路  
祁连山路  
李子园  
上海西站  
真如  
枫桥路  
曹杨路  
镇坪路  
中潭路  
上海火车站  
宝山路  
海伦路  
邮电新村  
四平路  
同济大学

2) 视图显示线路：



3) 添加新线路，添加新线路要先添加站点：

我的线路1

添加站点

1

我的站点1

121.220375

31.292236

确定添加

取消

修改站点信息

1

请输入站点名称

0.000000

0.000000

确定修改

取消

删除站点

1

确定删除

取消

下一步

清空

退出

站点添加成功



The screenshot shows the 'Add Station' dialog box. The 'Route Name' is '我的线路1'. The 'Add Station' section has 'Station Number' set to 2, 'Station Name' as '请输入站点名称', and 'Station Longitude' and 'Station Latitude' both set to 0.000000. The 'Modify Station Information' section has 'Station Number' set to 1, 'Station Name' as '请输入站点名称', and 'Station Longitude' and 'Station Latitude' both set to 0.000000. The 'Delete Station' section has 'Station Number' set to 1. The list on the right contains one item: '1: 我的站点1 (121.22, 31.2922)'. Buttons at the bottom include '确定添加', '取消', '下一步', '清空', and '退出'.

The screenshot shows the 'Add Station' dialog box after adding three stations. The 'Add Station' section has 'Station Number' set to 4, 'Station Name' as '请输入站点名称', and 'Station Longitude' and 'Station Latitude' both set to 0.000000. The 'Modify Station Information' section has 'Station Number' set to 3, 'Station Name' as '我的站点3', 'Station Longitude' set to 121.240785, and 'Station Latitude' set to 31.291742. The 'Delete Station' section has 'Station Number' set to 1. The list on the right contains three items: '1: 我的站点1 (121.22, 31.2922)', '2: 我的站点2 (121.224, 31.2837)', and '3: 我的站点3 (121.241, 31.2917)'. Buttons at the bottom include '确定添加', '取消', '确定修改', '取消', '下一步', '清空', and '退出'.

1: 我的站点1 (121.22, 31.2922)  
2: 我的站点2 (121.224, 31.2837)  
3: 我的站点3 (121.241, 31.2917)

**添加通路**

通路端点1:

通路端点2:

☒ 双向 ☐ 单向

**确认添加** **取消**

**修改通路**

通路编号:

通路端点1:

通路端点2:

☒ 双向 ☐ 单向

**确认修改** **取消**

**删除通路**

通路编号:

**确认删除** **取消**

**通路情况**

**站点情况**

- 1: 我的站点1 (121. 22, 31. 2922)
- 2: 我的站点2 (121. 224, 31. 2837)
- 3: 我的站点3 (121. 241, 31. 2917)
- 4: 我的站点4 (121. 245, 31. 2829)
- 5: 我的站点5 (121. 262, 31. 2816)

**返回** **清空** **确定**

**添加站点**

线路名称: 我的线路1

添加站点

站点序号:

站点名称:

站点经度:

站点纬度:

**确定添加** **取消**

**修改站点信息**

站点序号:

站点名称:

站点经度:

站点纬度:

**确定修改** **取消**

**删除站点**

站点序号:

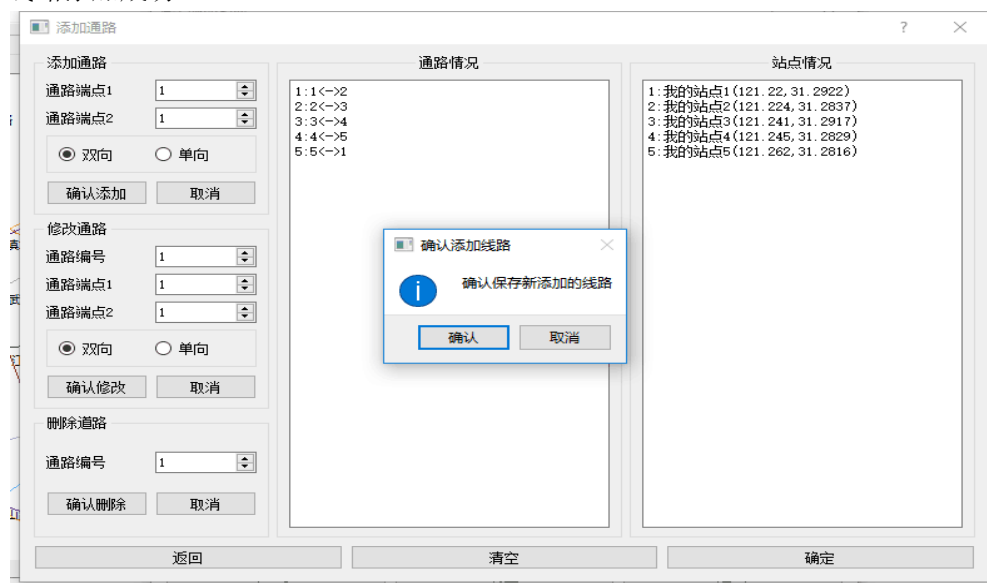
**确定删除** **取消**

**下一歩** **清空** **退出**

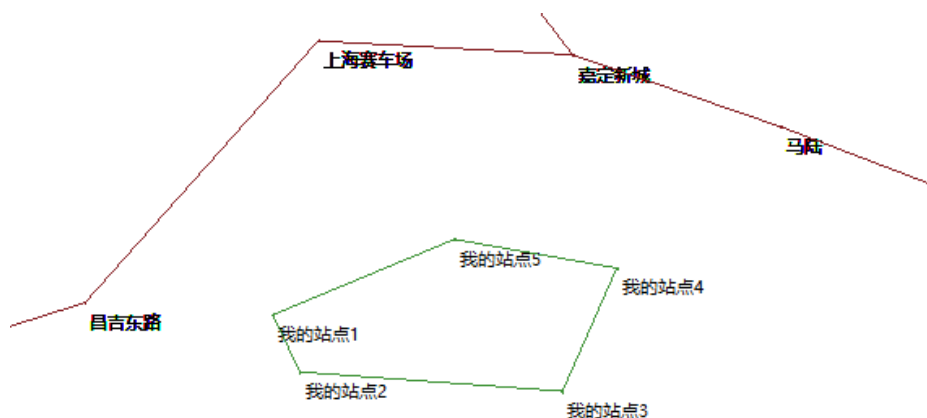
1: 我的站点1 (121. 22, 31. 2922)  
2: 我的站点2 (121. 224, 31. 2837)  
3: 我的站点3 (121. 241, 31. 2917)  
4: 我的站点4 (121. 245, 31. 2829)  
5: 我的站点5 (121. 262, 31. 2816)



线路添加成功:



#### 4) 添加有分支的线路和环线



#### 2. 稳定性

- 1) 系统响应较快, 所有操作都是及时响应和得出结果;
  - 2) 每项操作都能及时更新, 例如: 添加新的线路和站点后, 所有用户接口界面中的数据都会进行立马更新, 视图也会立马显示新加线路和站点;
  - 3) 经过反复测试调试, 程序目前无异常情况。
- #### 3. 容错能力

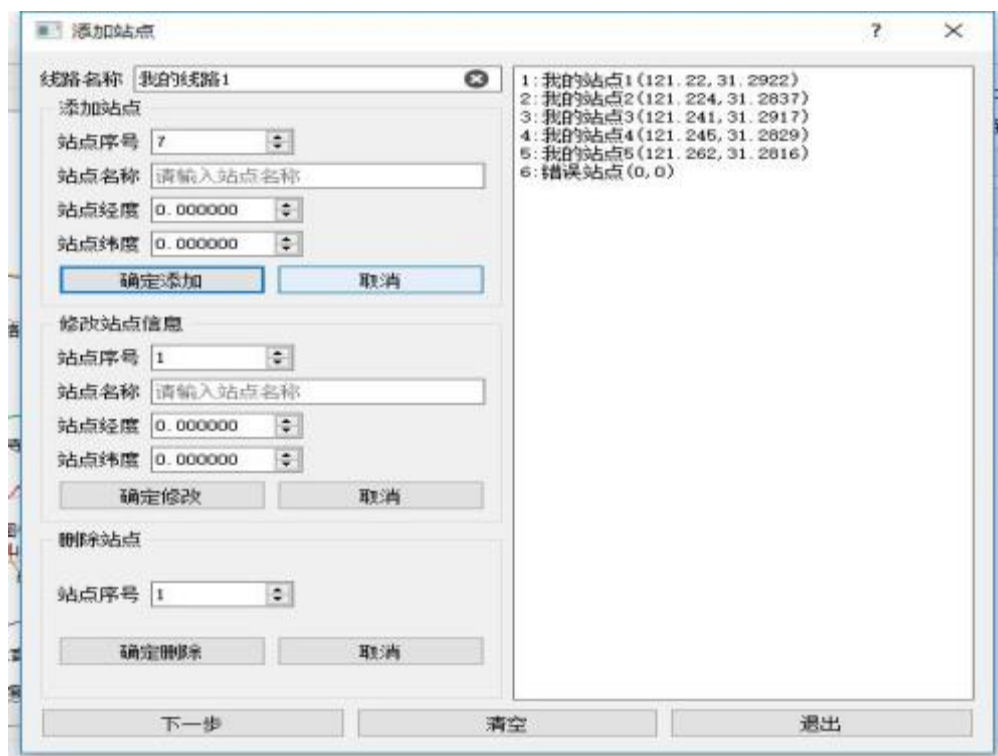
系统设计中加入了异常情况判断，进行了容错处理，所以系统具有很好的容错能力，不至于导致系统崩溃。一些样例如下：

1) 非法输入均有提示：

This screenshot shows the system's response to an invalid departure station. The '出发站' (Departure Station) field contains the text '错误站点' (Error Station). The '到达站' (Arrival Station) field contains '同济大学' (Tongji University). Below the input fields are three buttons: '清除' (Clear), '确定' (Confirm), and '刷新' (Refresh). Under the '换乘方案' (Transfer Scheme) section, a message box displays '未找到出发站' (Departure station not found).

This screenshot shows the system's response to an invalid arrival station. The '出发站' (Departure Station) field contains '张江高科' (Zhangjiang High-tech). The '到达站' (Arrival Station) field contains the text '错误站点' (Error Station). Below the input fields are three buttons: '清除' (Clear), '确定' (Confirm), and '刷新' (Refresh). Under the '换乘方案' (Transfer Scheme) section, a message box displays '未找到到达站' (Arrival station not found).

装  
订  
线



#### 4. 利用控件保证输入数据的正确性

输入数据的合理性通过输入控件保证，例如输入经纬度时，输入其它非法字符将不会处理，输入范围被确定，其它范围是无法输入的。

地理坐标(P):

东经E

北纬N

有效范围:

min [ 121.0 , 30.9 ]

max [ 122.0 , 31.5 ]

## 2.7 操作说明

按照题目的要求，系统主要包含两大功能，换乘查询和动态添加线路。

### 2.7.1 换乘查询

1. 输入起始和目标站点，直接键入即可，如下：

选择“起点站”、“终点站”、“换乘策略”为“所需时间最短”后，点击“确定”，即可显示相应站点，并突出显示路线

出发站

到达站

换乘方案

- 隆德路
- 江苏路
- 静安寺
- 常熟路
- 陕西南路
- 新天地
- 马当路

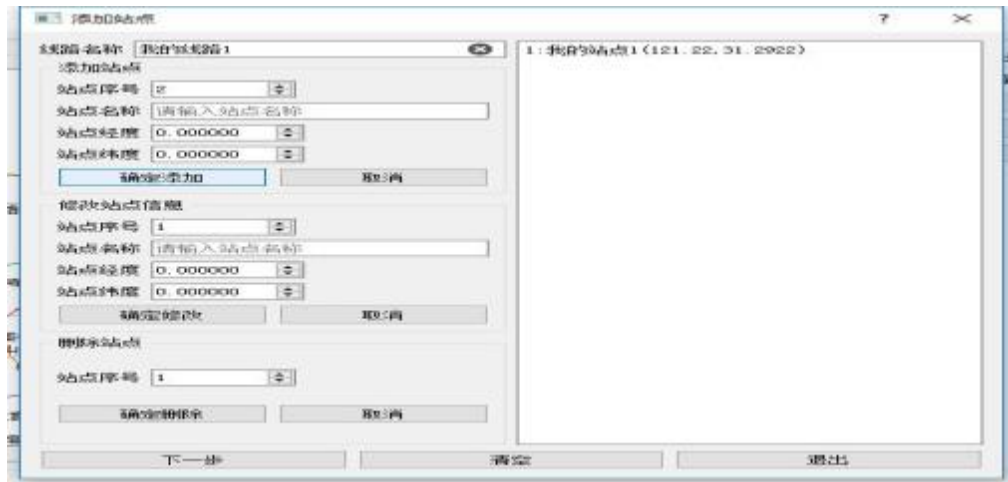
装  
订  
线

## 2.7.2 动态添加线路

动态添加主要包含添加线路，添加站点，添加连接，文本简易添加，在工具栏中如下：

添加线路

先添加站点



添加连接



## 第三部分 设计总结

### 3.1 工作总结

通过本系统学习和运用了 Qt5 的编程架构和相关知识；独立完成课程设计中的算法实现题目的所有设计、编程实现、测试等工作；独立完成课程设计中的综合应用题目的所有设计、编程实现、测试等工作；完成课程设计报告一份。

### 3.2 软件改进

没有对于换乘的次数进行限制，算法为了缩短路径会进行多次换乘，尤其是在市区内网格内状的地铁网络。如下图：



上图中，红线为本软件给出的路径，蓝色线和棕色线为百度地图生成的路线，相比较而言，距离有缩短，但增加了 3 次换乘，会导致乘客体验下降。

改进：可以在后台最短路径的权值计算中，加入换乘次数。

### 3.3 心得与收获

通过开发本系统，在学习和实现的过程中体会和领悟。第一，做事情一定要坚持，克服困难，主动思考，有困难要勇于动脑钻研，想办法将任务进行分解，降低问题难度，一步步解决，终将解决整个问题，学会找最重要的事情先做，这样做事情才有效率，不拖拉。对程序的用户界面和



一些细节很是用心，但是这样就好浪费很多时间，效率不高，应该先做好最重要的事情，例如最不可缺少的功能实现，主体实现然后细节修改，而不是本末倒置。第二，学会自学，提高自学能力，提高独立解决问题的能力，学知识是一个层面，能够灵活运用是跟深入的能力，学会不等于会用，会用并且能够灵活运用需要一个不懈的过程，需要肯下功夫，才是真正学到自己的脑子。

完成数据结构课程设计后，感觉自身的自学能力有了很大提升。自学能力也锻炼的少。但是这次的课程设计的所有工作，都是由自己去选择相应的技术学习实现，尤其是计算机专业的大学生所需要的自学能力得到了必须的锻炼，对以后所有的技术和知识的自学，这将是一个有意义的开端，以后继续努力，不断丰富自己的能力。

虽然课程设计只有看似算法设计和综合应用实现两个题目，但是从布置课程设计题目到完成的整个过程中，不仅仅是学习到了以前未曾接触过的标准用户交互程序，从简陋的控制台程序提升到界面友好、美观、人性化的应用程序，打开了编程的一扇大门，同时也从心态、自学能力、解决未知问题能力等内在的综合实力有了全新的提升。具体来说，学习了 Qt 的编程框架，包括元对象机制、基本控件的掌握、绘图框架、模型编程、文件操作等相关知识；学习到了 Github 开源项目托管平台的使用，将其运用到了实际的项目；学习到了 openstreetmap 等开源地图数据的获取和提炼等知识，自身技术能力又上了一个台阶。在软实力上，打开了用户界面编程的大门，激发自身对编程的热情和后面学习的动力；通过自学实现了课程设计，自身的学习能力得到了有效锻炼；开发的过程中，无论是设计还是编程，都会遇到各种各样的问题，通过独立解决这些疑难杂症，在独立解决问题的能力方面也得到了很大提升；最重要的是，树立不惧所有未知知识、解决疑难问题、接受各种挑战的信心。

## 第四部分 参考文献

- 【1】 殷人昆. 数据结构（用面向对象方法与 C++ 语言描述）[M]. 北京：清华大学出版社, 2007
- 【2】 仇国巍. Qt 图形界面编程入门 [M]. 北京：清华大学出版社, 2014-1
- 【3】 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
- 【4】 Thomas H.Cormen, Charles E.Leiserson , Ronald L.Rivest, Clifford Stein 算法导论[M]. 机械工业出版社, 2017-6

装

订

线