目录

1 需求分析.....2

1.1 问题分析.....2

1.2 目前程序完成功能.....2

1.3 需要处理的数据.....2

1.4 程序开发运行选用的环境.....3

1.5 用户界面的设计.....3

2 数据结构设计.....3

2.1 Vertex 类（存储地点信息）.....3

2.2 路径维护.....4

2.2.1 使用泛型集合来存储维护路径.....4

2.3 方法.....4

3 详细设计.....4

3.1 信息导入;.....4

3.1.1 功能.....5

3.1.2 框图.....6

3.2 距离矩阵的建立：julijuzhen().....6

3.2.1 功能.....6

1 需求分析

1.1 问题分析

旅行商问题：

一个售货员必须访问 n 个城市，恰好访问每个城市一次，并最终回到出发城市。售货员从城市 i 到城市 j 的旅行费用是一个整数，旅行所需的全部费用是他旅行经过的的各边费用之和，而售货员希望使整个旅行费用最低。（等价于求图的最短哈密尔顿回路问题）

令 G=(V, E)是一个带权重的有向图，顶点集 V=(v0, v1, ..., vn-1)。从图中任一顶点 vi 出发，经图中所有其他顶点一次且只有一次，最后回到同一顶点 vi 的最短路径。

算法分析：

在设计中，需要考虑时间复杂度及空间复杂度，考虑用户的路径可能存在转弯、圆圈形状。若直接以两点直线距离作为路径则不合规范且估算不符合现实情况，所以在设计的过程中需要考虑到路径内的拐点，在每个路口单独设计位置信息点，这样才能模拟出实际用户行走线路。同时，直线可直接用两端点为信息坐标存储。

页面设计方面，需要考虑用户操作复杂度，用户可通过下拉框选择减少操作复杂度，并且可以使用文本框输入出发及停留时间，附加功能设计时，使用 push button 使用户仅需要点击按钮即可实现功能，无需输入过多信息。

1.2 目前程序完成功能

- 展示校园平面图，供用户选择性输入自己的若干出行地点(不少于 5 个)、停留时间和出发时刻；
- 按用户选择，能提供合理出行线路和相关信息。信息包括每个地点的到达时刻，每个地点停留的时间，并预估总时间，并正确输出；
- 在校园平面图上展示出行线路和各种时间信息。
- [网购订票]：提供场馆（例如羽毛球馆）网址、供用户提前预订；
- 附加功能：景点信息显示
- 附加功能：地图变换（放大缩小）
- 附加功能：校园美景图片展示
- 附加功能：路径颜色变换

1.3 需要处理的数据

地点信息：Didian
位置点坐标：Guanjiandian
边：Edge
邻接矩阵：juli[Guanjiandiannumber][Guanjiandiannumber]
最短路径：zuiduan_lujing[Guanjiandiannumber]
景点信息：xinxi_weizhi

1.4 程序开发运行选用的环境

工程选用环境：QT Creator 4.0.3（Community）

1.5 用户界面的设计

2 数据结构设计

2.1 Vertex 类（存储地点信息）

使用邻接矩阵存储图信息，计算出的最短路径信息存储于矩阵中。

```
class Vertex
{
    int  ditu_x;
    int  ditu_y;
    struct Didian{
        double x,y;
    }didian[DidianNumber];              //地点信息
    struct Guanjiandian{
        double x,y;
```

```
    int duiyingjianzhu=-1;
}guanjiandian[GuanjiandianNumber];//位置信息
struct Edge{
    int start,end;
}edge[EdgeNumber];                  //边的信息
struct Xinxi_weizhi{
    double x1,y1,x2,y2;
}xinxi_weizhi[XinxiNumber];         //景点信息
int juli[GuanjiandianNumber][GuanjiandianNumber];   //邻接矩阵
int zuiduan_lujing[GuanjiandianNumber];             //存最短路径

}
```

2.2 路径维护

2.2.1 使用泛型集合来存储维护路径

```
QVector<int> path;                          //用于路径的输出
QVector<int>::iterator    it=path.begin();
```

2.3 方法

```
void didianzuobiao();               //导入地点信息
void guanjiandianzuobiao();         //导入位置点信息
void biandejianli();                //导入边信息
void julijuzhen();                  //导入邻接矩阵信息
void jisuanZuiduanLujing(int s);    //计算最短路径
void xinxijianli();                 //导入景点信息
void TSP();                         //TSP 算法
bool isVisited(bool visited[]);
void printPath();
void getPath();
void hamilton();                    //生成 Hamilton 回路
```

3 详细设计

3.1 信息导入；

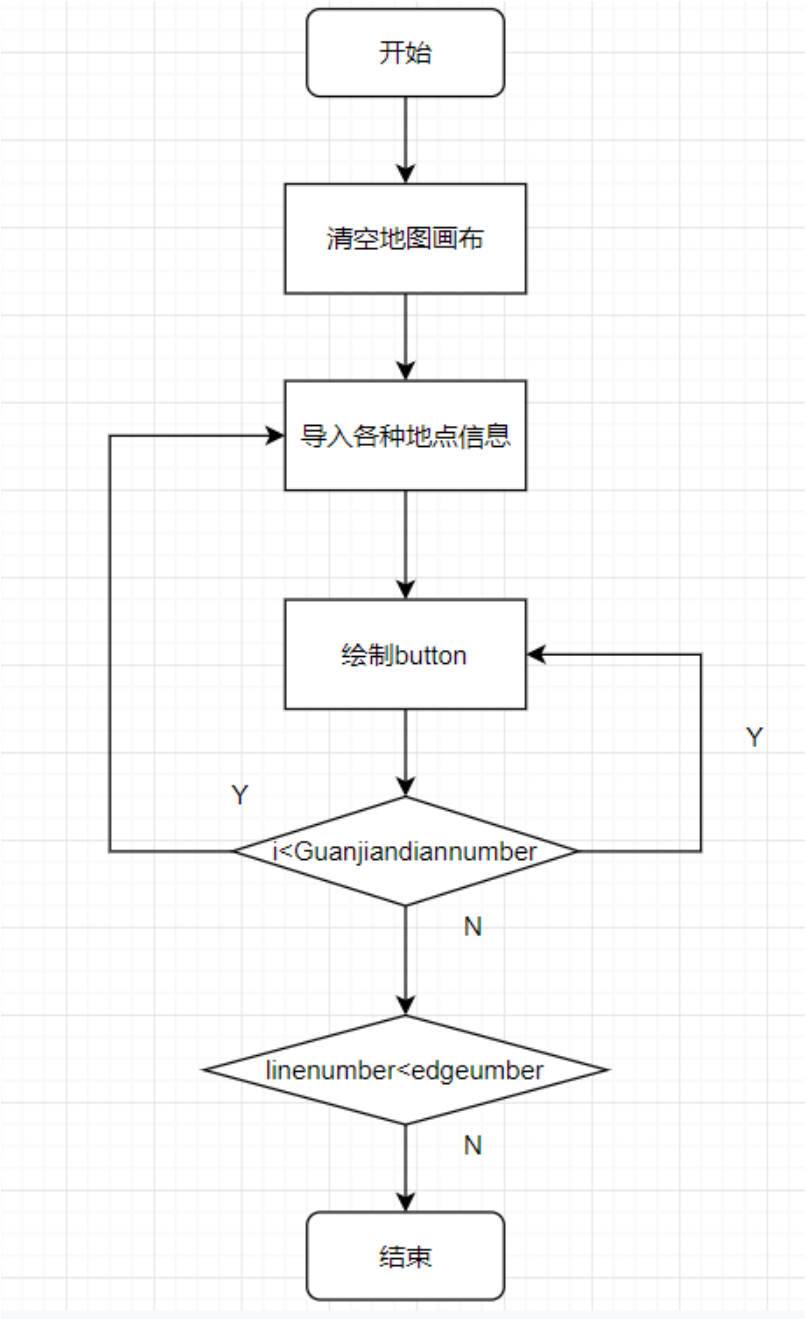
```
void didianzuobiao();               //导入地点信息
void guanjiandianzuobiao();         //导入位置点信息
void biandejianli();                //导入边信息
void julijuzhen();                  //导入邻接矩阵信息
```

3.1.1 功能

将所有地点位置信息导入，作为 button 显示在地图上。

```
QPixmap temp(":/ditu/ditu.png");        //导入地图
temp.save("temp.png");
yuantu=temp;
yuantu=yuantu.scaled(ui->label->width()*1.331,ui->label->height()*1.331,Qt::Ign
    oreAspectRatio, Qt::SmoothTransformation);
ui->label->setGeometry(0,0,ditu_width,ditu_height);     //画标签
ui->label->setPixmap(yuantu);
```

3.1.2 框图



3.2 距离矩阵的建立：julijuzhen()

```
int dis[GuanjiandianNumber][GuanjiandianNumber];      //求最短路径并存储
```

3.2.1 功能

选择要去的地点个数，依次选择起点及中途点

点击路径查询，使用 Dijkstra 算法，求出最短距离矩阵，再在每个每个距离路径已知

的情况下，组成路径根据起点及中途点查询最短路径

输入停留时间，系统自动计算时间输出

迪杰斯特拉算法：