华中科技大学

课 程 实 验 报 告

课程名称：C++程序设计

实验名称：面向对象的公交地图导航

院　　系 ：计算机科学与技术

专业班级 ：　计算机 2020xx　

学　　号 ：　U2020xxxxx　

姓　　名 ：　　吴彦祖　　

指导教师 ：　　金良海　　

2021 年 12 月 6 日

一、需求分析

1. 题目要求

假定所有公交车辆从起点到终点都是双向非环路的，且双向线路的所有停靠站点都对应相同。设有 M 路公交路线，第 j 路公交路线有 N_j 个站点。所有公交线路累计共有 S 个站点，第 k 个站点的坐标为（X_k，Y_k），所有坐标均以米为单位标注。邻近站点之间的距离指的是站点坐标之间的欧几里得距离。现有一人处于起点坐标（X_b，Y_b），此人需要步行到最近站点乘车，下车后要步行到达的终点坐标为（X_e，Y_e），而他特别不愿意走路，能坐公交就尽量坐公交。假定公交转乘时的步行距离为 0，试编程求他从起点（X_b，Y_b）到终点（X_e，Y_e）的乘坐线路。建立模型时需要考虑如下几种情形：（1）最少转乘；（2）最短距离。

所有公交线路的站点坐标存放于“stops.txt”文件，其中第 1 行为站点总个数，第 2 行为第 1 个站点的坐标，第 3 行为第 2 个站点的坐标，以此类推。可用图形化的界面显示站点及公交线路。“stops.txt”文件的内容如下。

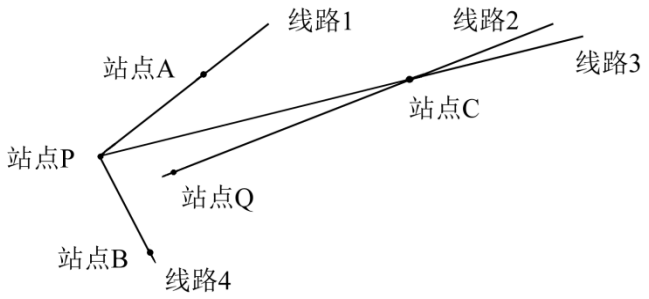
```
39
235  27
358  29
480  34
155  36
222  64
282  62
413  60
457  63
483  60
560  69
131  87
349  61
314  97
420 107
487 125
620 107
666  79
186 107
270 120
350 141
383 148
370 164
```

442 179
496 171
555 167
651 155
775 184
678 272
208 156
296 161
356 190
493 202
490 229
504 262
457 269
249 196
155 190
103 171
112 241

所有公交线路信息存放于“lines.txt”文件，其中第 1 行为公交线路总数，第 2 行为每条公交线路的站点总数，第 3 行为线路 1 经过的站点编号（对应站点坐标参见“stops.txt”），第 4 行为线路 2 经过的站点编号，以此类推。“lines.txt”文件的内容如下。

6
13 11 8 9 7 7
1 6 13 20 22 21 14 8 3 9 15 24 32
4 5 6 12 7 8 9 10 16 26 28
11 18 19 20 21 23 33 34
38 37 36 31 23 24 25 26 27
2 12 13 19 29 37 39
30 31 35 33 25 16 17

采用 Dijkstra 最短路径算法是不合适的，因为它仅考虑了距离而未考虑公交线路行驶约束。如下图所示，对于某站点 P 周围的站点 Q，其欧几里得距离虽近，但可能需要很远的转乘才能到达。例如，从站点 P 出发，要从线路 3 经站点 C 转线路 2 才能到最近的站点 Q，因为站点 P 和站点 Q 之间无直达的公交线路。



通过类型抽象形成站点、公交线路、转乘站点、转乘线路、转乘矩阵、公交系统等类，输入上述文件初始化站点和线路对象，然后通过图形化的界面显示站点和线路地图。用户用鼠标左键在地图上设定起点、鼠标右键确定终点，按照设定的最少转乘或者最短距离选项规划出从起点步行到最近站点上车、到离终点最近站点下车步行到终点的线路，在地图上用不同颜色显示规划线路若干秒，然后消除并恢复原始地图线路的颜色。

假设某个机构或单位的信息存储在“organization.txt”文件，第 1 列存放单位名称，第 2 列单位坐标。“organization.txt”文件中的格式如下，可自己添加更多单位或坐标。

华中科技大学　　990, 370
华乐山庄　　500, 340
光谷中心花园　　631, 367
光谷街北路　　766, 472

用户可输入“华科大”或“华中科大”查询其所在位置，通过最大公共子串算法进行模糊匹配，找到“华中科技大学”及其坐标。在确定始发单位坐标和终点单位坐标后，按照设定的最少转乘或者最短距离选项规划线路，并在地图上用不同颜色显示规划线路若干秒，然后消除并恢复原始地图线路的颜色。

提示：可以构造公交线路转乘矩阵 A（即 A^1）。假定有 5 条公交线路，若线路 i 和线路 j（i≠j）有 r 个转乘站点（即 r 种走法），则矩阵 A^1 的元素 $a^1[i,j]=r$；如 i=j 则规定无须转乘，即有 $a^1[i,i]=0$。则对于上述前 5 条公交线路，从公交线路 i 一次转乘到线路 j，可得如下转乘矩阵 A^1。

0　3　3　1　0
3　0　3　2　1
3　3　0　0　1
1　2　0　0　1
0　1　1　1　0

由上述转乘矩阵 A^1 可知，无法从线路 1 转乘到线路 5，因为 $a^1[1,5]=0$。可以尝试从线路 1 转乘其他线路，再从其他线路转乘线路 5，即从线路 1 经过两次转乘到线路 5。这只需要进行矩阵乘法运算 $A*A=A^2$，从线路 1 经过两次转乘到线路 5，可从 A^2 中得到共有 $a^2[1,5]$种走法。

$$a^2[1,5]=\sum_{k=1}^{5} a^1[1,k]*a^1[k,5]$$

由上述转乘公式，可计算得到 A^2，最后设置 $a^2[i,i]=0$，修正 A^2 使同一线路不用转乘。修正后的 A^2 如下。

0	11	9	6	7
11	0	10	4	5
9	10	0	10	3
6	4	10	0	2
7	5	3	2	0

由上述矩阵可知 $a^2[1,5]=7$，即从线路 1 经过两次转乘到线路 5 共有 7 种走法：（1）从线路 1 到线路 2 共有 3 个转乘点，再从线路 2 经唯一转乘点至线路 5，一共有 3 种转乘方法；（2）从线路 1 到线路 3 共有 3 个转乘点，再从线路 3 经唯一转乘点至线路 5，一共有 3 种转乘方法；（3）从线路 1 经唯一转乘点至线路 4，再从线路 4 经唯一转乘点至线路 5，一共有 1 种转乘方法。

依此类推，可以得到 A*A*A（即 A^3，反映从线路 i 经过 3 次转乘到线路 j 共有几种转乘走法）。对于本题来说，由于总共只有 M=7 条公交线路，故无重复公交的转乘最多只能转乘 M-1 次，所以只需计算到 $A^{M-1}=A^6$ 为止。任意两条线路之间 1 次、2 次……6 次转乘的走法共有 A^+种，$A^+= A^1+ A^2+ A^3+ A^4+ A^5+ A^6$。

上述 A^+运算是一种修正的矩阵“闭包”运算，可以抽象成矩阵类的一个运算。上述 A^+运算只计算了有几种转乘走法，当然，还可以同步 A^n 建立另外一个矩阵，对应元素为某种链表指针类型，用于记载对应转乘线路和转乘站点。

得到上述转乘“闭包”矩阵 A^+以后，只需要搜索离起点最近的站点（可能不止一个站点），找到最近站点所在的线路 i（可能不止一条），并搜索离终点最近的站点（可能不止一个站点），找到最近站点所在的线路 j（可能不止一条），然后在 A^+中找出 a[i, j]所描述的所有转乘线路和转乘站点即可。利用站点坐标可以得到两两站点之间的距离，并利用 A^+分别得到如下两种情形的最优转乘方案：（1）最少转乘；（2）最短距离。

2. 需求分析

根据实验要求，可以将任务分为以下几个模块：

(1)将公交车线路图抽象为图，读取文件到图中；

(2)编写算法实现换乘次数最少或总距离最少的最短路的求解；

(3)以 Qt 框架为基础，编写图形用户界面，将公交车线路和最短路可视化；

(4)设计汉字模糊匹配算法，实现公交站点的模糊匹配；

二、系统设计

1. 概要设计

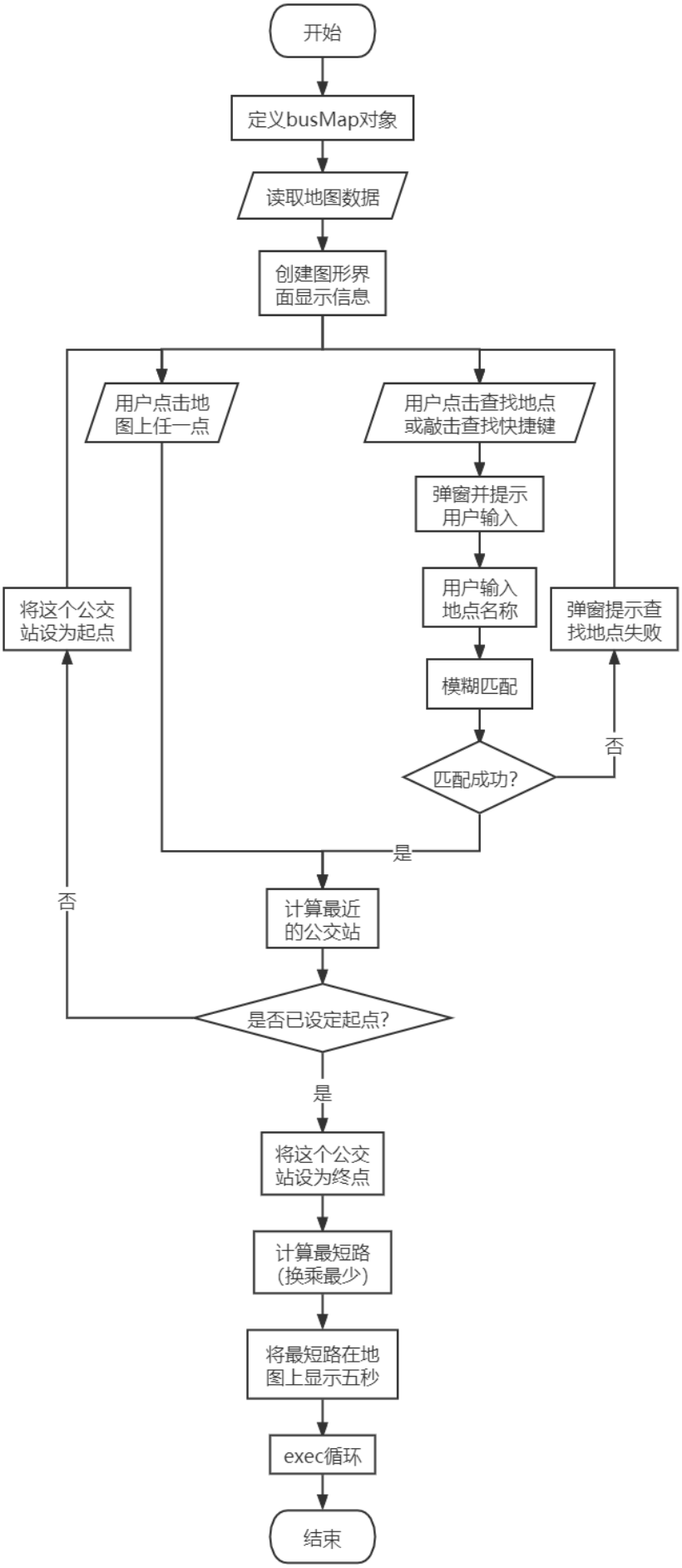
定义 myPoint 类，实现点的数据存储和邻接链表记录边的功能。

定义 busMap 类，存储地图数据，包含公交站点和公交线路等。在 busMap 中定义成员函数实现最短路的求解以及储存。

在 Qt 中创建 mainwindow 类，以 QMainWindow 为基类，作为程序的主界面，显示地图、公交站点和路线等信息。另外创建 dialog 类，以 QDialog 为基类，在其中添加文本框，实现公交站点名称的模糊查询。

在用户单击或输入地点后，程序将匹配最近的公交站点，设为起点或终点，计算最短路并显示 5 秒。

具体实现流程图如下页所示：


开始
定义busMap对象
读取地图数据
创建图形界面显示信息
用户点击地图上任一点
用户点击查找地点或敲击查找快捷键
弹窗并提示用户输入
用户输入地点名称
模糊匹配
匹配成功?
否
弹窗提示查找地点失败
将这个公交站设为起点
否
是
计算最近的公交站
是否已设定起点?
是
将这个公交站设为终点
计算最短路(换乘最少)
将最短路在地图上显示五秒
exec循环
结束

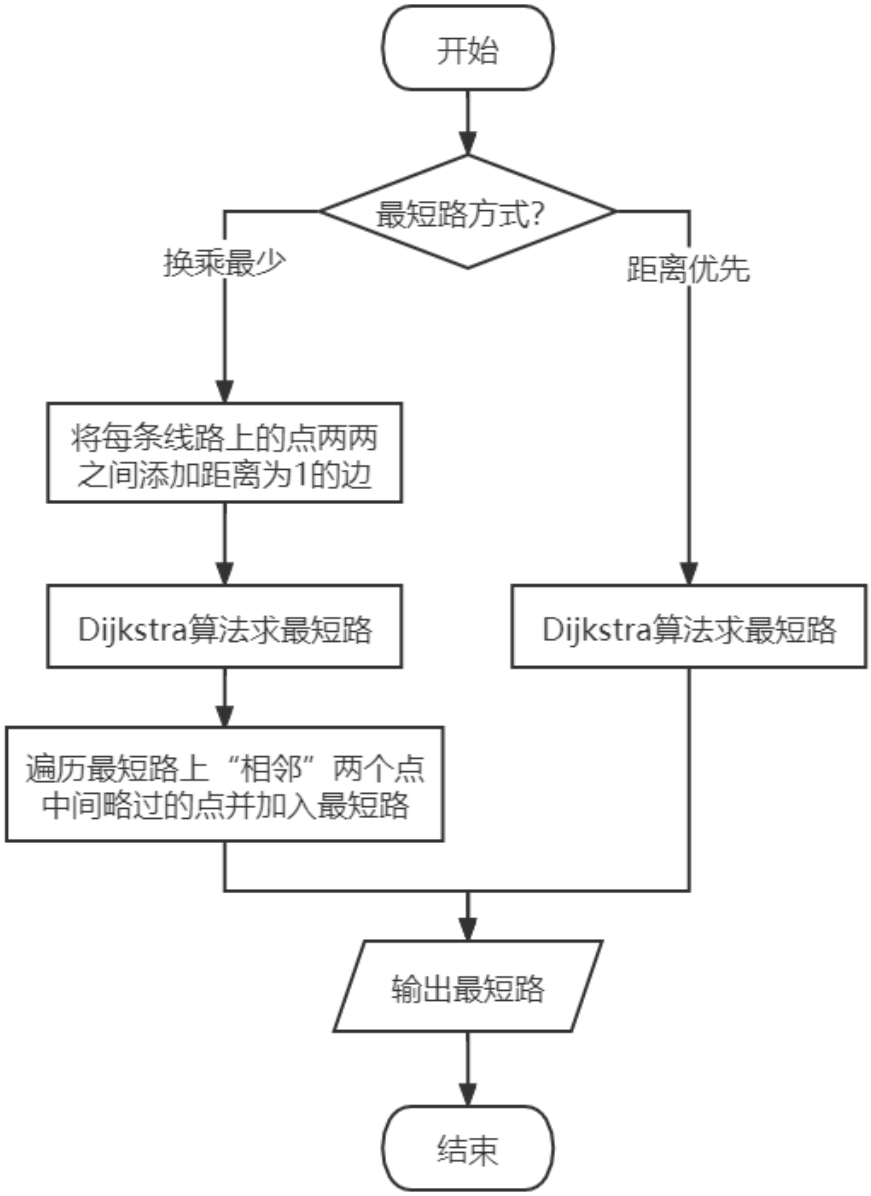
2. 详细设计

(1)最短路的求解思路：

最短路的求解有两种方式，一种是距离优先，另一种是换乘次数最少。距离优先的最短路很容易求解，采用 Dijkstra 算法并且添加数组记录路径即可。换乘次数最少的最短路需要在 Dijkstra 算法的基础上采取一些变化。

首先将同一公交线路上的点两两之间添加一条距离为 1 的边（其他正常边距离都大于 1），然后再用 Dijkstra 算法求解最短路并记录路径即可。这样在求解最短路时，由于同一条线路上的点之间距离都是最小的，所以结果一定是换乘次数最小的。这样求解还会有一个问题，同一公交线路上的点两两之间人为的添加了一条最小边，所以最短路的结果会直接从一个点跳跃到同一线路上另一个不相连的点。因此在存储最短路时，需要遍历一遍每两个“连续”的点之间跳过的点，这样得出的结果才是正确的。

设有 n 个站点，m 条边，则时间复杂度为：O((n+m)logn+n)



(2)myPoint 类的实现：

定义 myEdge 类，存储当前边指向的点和距离，用链表的方式存储下一条边的指针；定义 myPoint 类，存储每个点的坐标和从这个点出发的第一条边。由于最短路有两种方式，换乘最少的求解需要额外添加边，所以为了不混淆两种求解方式的数据，在实现时分别存储了两种不同的边。

另外单独定义了全局函数，用于求解两点间距离。

(3)busMap 类的实现：

定义 busMap 类，定义成员变量存储公交站点的数量和坐标、公交线路的数量和包含的站点以及地点的名称和坐标，另外还需要定义数组存储最短路径。定义成员函数：构造函数（从文件读取数据并且初始化）、两种最短路的求解函数，以及析构函数。在定义时对于每个数组都只定义了指针，在初始化时根据数据规模申请相应大小的空间，在析构时则会释放内存空间。

(4)mainwindow 类的实现：

在 Qt 开发环境下定义 mainwindow 类，以 QMainWindow 为基类，实现主界面。在 Qt Designer 中对 ui 文件进行编写，放置地图框和最短路求解方式的选项，并且在菜单栏添加“文件”、“说明”和“查找”三个选项。“文件”下含有三个数据源文件，点击则会打开文件；“说明”下是对程序操作的说明；“查找”下含有“查找地点”，点击则会弹出 dialog 窗口，“查找地点”的快捷键设为 R。在类中定义槽函数实现距离优先和最少换乘二选一，重载虚函数实现鼠标事件、键盘事件、绘图时间和计时器事件，实现流程图中功能。绘制公交信息时，线路由绿色虚线表示，公交站点由黑色实心点表示；绘制路线时，蓝色代表起点，红色代表终点，路径由蓝色实线表示。最后定义公共函数实现最短路求解函数的调用和数据传输。

主界面如下图所示：

(5)dialog 类的实现：

在 Qt 开发环境下定义 dalog 类，以 QDialog 为基类，实现地点搜索窗口。在 Qt Designer 中对 ui 文件进行编写，放置提示文本、单行文本框和确定按钮。在类中定义槽函数实现输入字符串的传入。

dialog 界面如下图所示：

三、软件开发

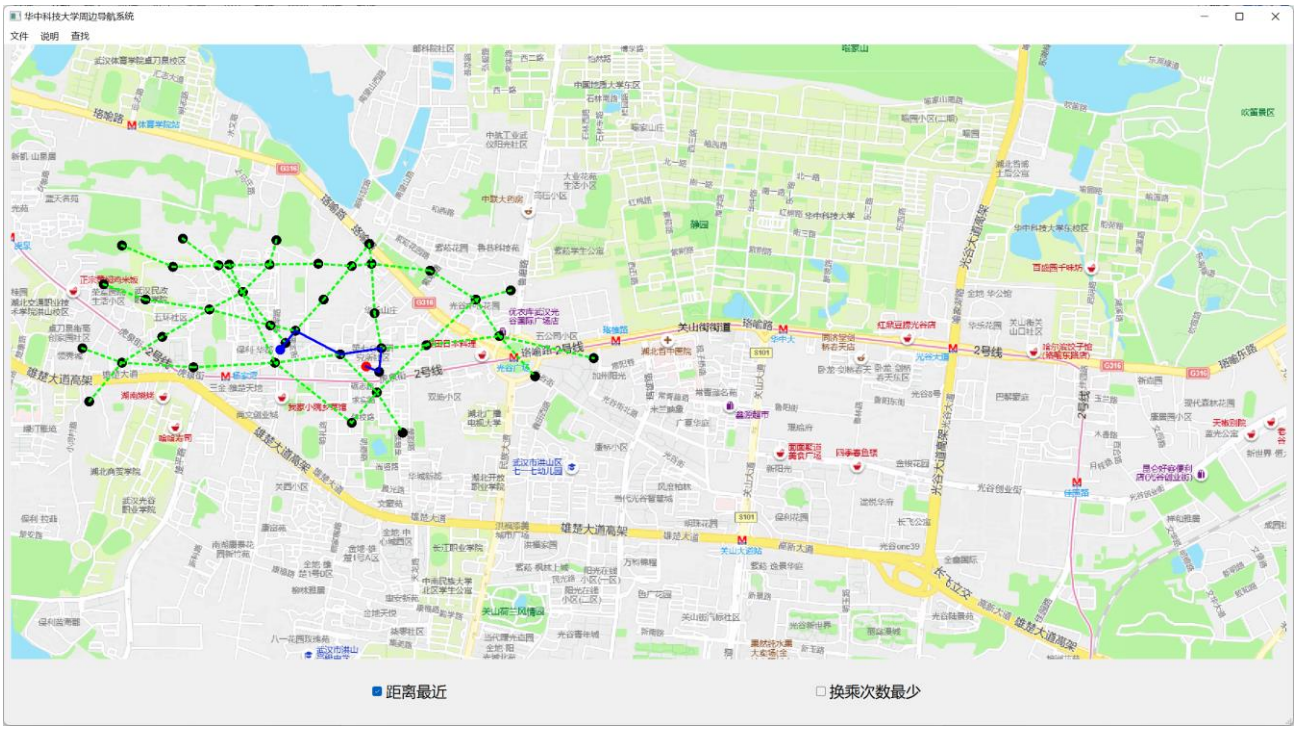
实验采用 Visual Studio 2022 以及 Qt Creator 进行代码编写、编译和调试。在 Visual Studio 2022 中对 myPoint 类和 busMap 类进行编写。在 Qt Creator 中对 mainwindow 类和 dialog 类进行编写。最后由 Qt Creator 生成可执行文件。

四、软件测试

(1)主界面测试：测试程序主界面运行是否正常。黑色是实心点表示公交站点，绿色虚线表示公交线路。

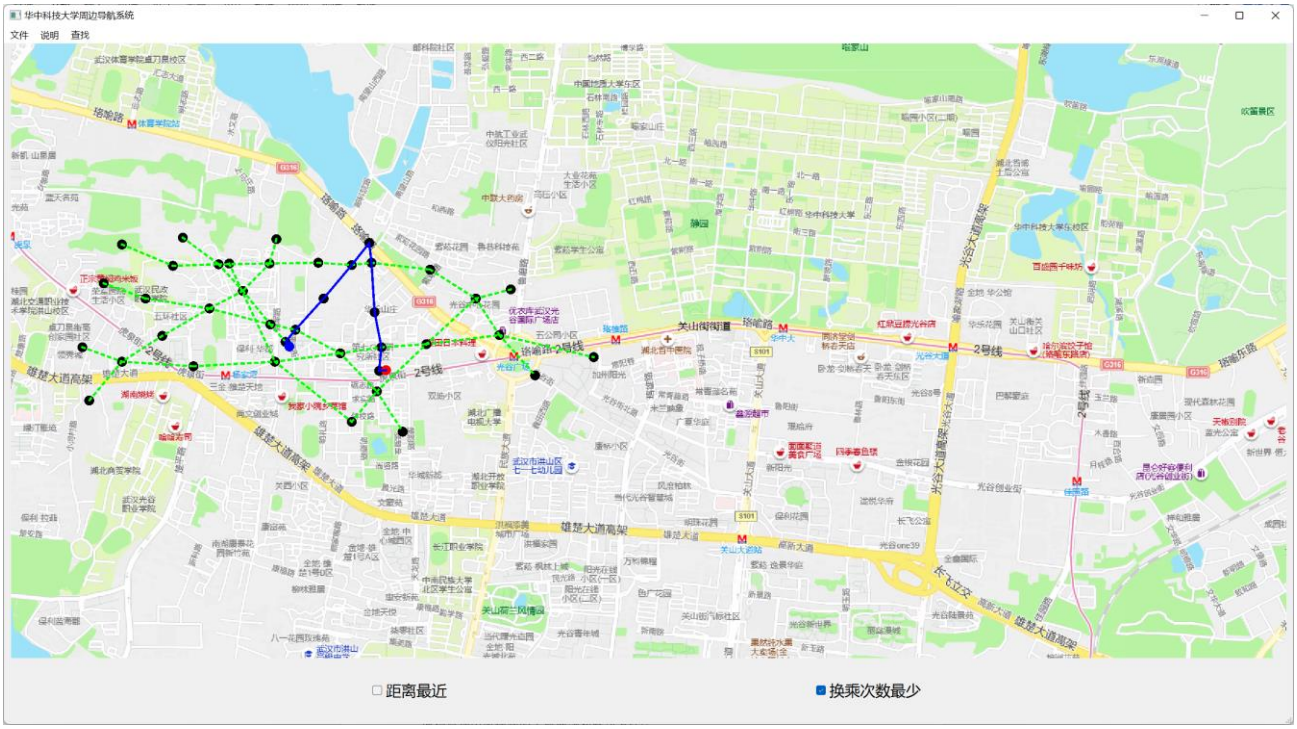
公交站点和路线显示正常，菜单栏功能也正常，经测试程序界面正常。

(2)距离最近的最短路测试：测试距离最短最短路求解和显示是否正确。单击确定起点，再次单击确定终点。蓝色实线表示最短路线，蓝色实心点表示起点，红色实心点表示终点。



图示路线是距离最近的路线，但是需要换乘三次公交车，所以经测试距离最近最短路求解和显示正常。

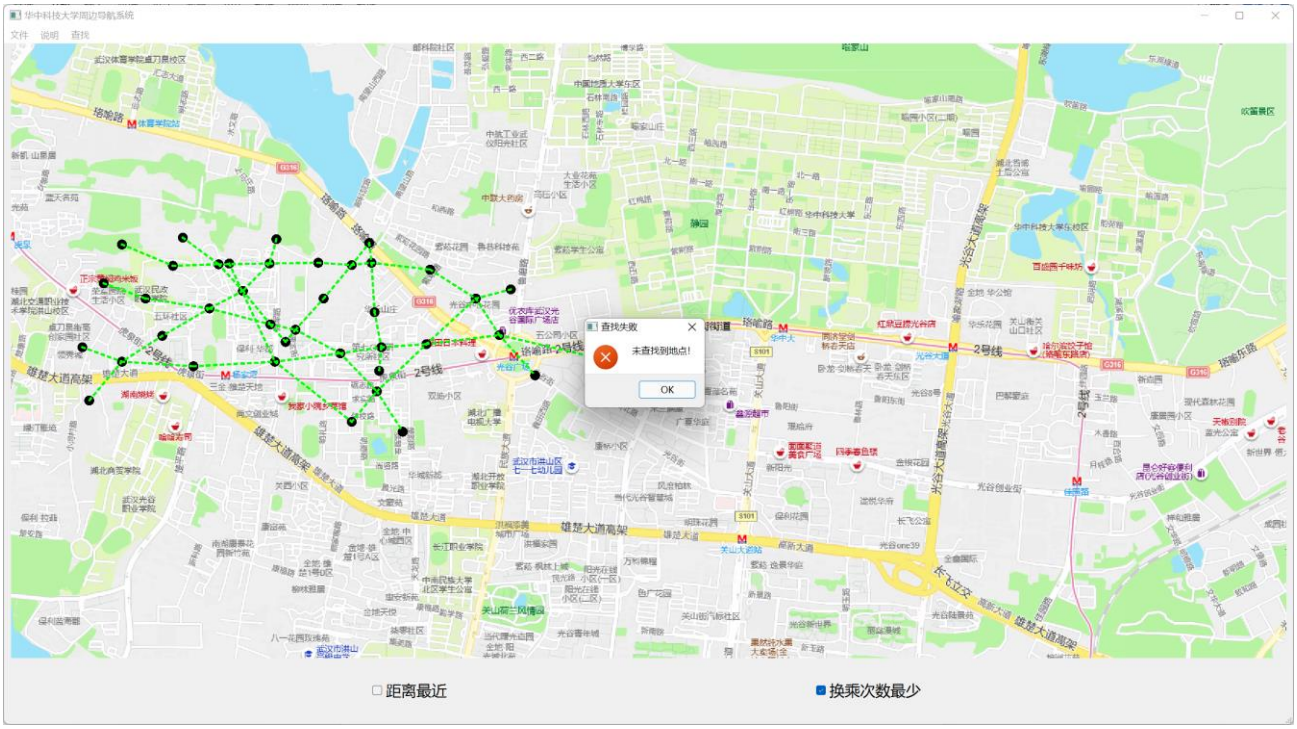
(3)换乘次数最少的最短路测试：测试换乘次数最少最短路求解和显示是否正确。单击确定起点，再次单击确定终点。蓝色实线表示最短路线，蓝色实心点表示起点，红色实心点表示终点。



图示路线距离不是最近，但是只需要换乘一次公交车，换乘次数最少，所以经测试换乘次数最少最短路求解和显示正常。

(4)地点查找测试：单击“查找”下的“查找地点”，或者按 R 键即可弹出查找地点窗口。在文本框中输入地点缩写点击确定即可查询到地点，如果没有对应地点则应该弹出弹窗告知地点查询失败。

如图所示，红色终点即为华中科技大学（华中大为缩写），地点查询成功。



因为地点库内没有天安门，所以弹出查询失败的弹窗。以上测试表明地点查询功能正常。

五、特点与不足

1. 技术特点

本次实验中的最短路算法采取了改进的 Dijkstra 算法，相比于转乘矩阵的闭包运算，更容易理解，而且可以直接观察最短路径的构成和变化，时间复杂度方面也没有额外的增加。

另外，图形用户界面采用了 Qt 框架，相较于 MFC 和 Windows API 的使用简单易懂，而且界面美观、规范性更强，对于新手开发者较为友好。

2. 不足和改进的建议

在决定使用 Qt 后，在公交车站和路线的建模中，应该充分应用 Qt 的内容。比如实际编写中点的坐标使用了 stl 中的 pair，但也可以用 QPoint 来表示，而且 QPoint 更方便，和 Qt 更贴合；另外还有 QString 代替 string 等等。

关于 Qt：虽然 Qt 有很多优点，但是也有美中不足。基于 Qt 编写的桌面程序需要引用 Qt 库才可以运行，体积很大，较为臃肿。另外 Qt 在商业应用中需要付费，学习阶段不适合对 Qt 产生依赖。

六、过程和体会

1. 遇到的主要问题和解决方法

(1)改进的 Dijkstra 算法会略过同一线路上的中间点，实际求解最短路时需要遍历中间点，可以用平衡树保存每条线路的站点，这样遍历时可以快速查询。

(2)Qt 窗口绘图时，需要在绘图事件中完成，而绘图事件在每次页面刷新或调整大小时都要被调用，因此需要考虑好绘图事件中变量的更改时机，最好在外部修改。

(3)Qt 窗口生成后会继续执行下面的内容，如果需要等待用户操作，则需要使用 exec 函数使程序“暂停”。

2. 课程设计的体会

本次实验中复习了最短路算法，学习并使用了 Qt 框架。在今后的学习中，我将继续使用 Qt 编写桌面应用程序，同时学习 Windows API、Linux API 等其他底层 API，更好的理解操作系统和图形用户界面的实现。

七、源码和说明

1. 文件清单及其功能说明

源码文件包括：

头文件：myPoint.h、busMap.h、mainwindow.h、dialog.h

源文件：myPoint.cpp、busMap.cpp、mainwindow.cpp、dialog.cpp

主程序文件：main.cpp

ui 文件：mainwindow.ui、dialog.ui

资源文件：mainwindow.qrc

工程文件：exp_5.pro、exp_5.pro.user

完整程序包括：

生成执行程序：华中科技大学导航系统.exe

Qt 库文件：其他文件

2. 用户使用说明书

使用 Qt Creator 打开 exp_5.pro 即可编译运行。或者打开生成的可执行程序直接进行使用，使用说明如下：单击左键确定起点（蓝色），再次单击左键确定终点（红色），此时会自动生成路线，路线会停留五秒钟。点击“查找”→“查找地点”或者按下快捷键 R 会弹出窗口，此时输入地点名称或缩写再按确定即可查询到地点并标记为起点或终点。