### 广州地铁线路查询

李钰 19335112 [liyu256@mail2.sysu.edu.cn](mailto:liyu256@mail2.sysu.edu.cn) 梁睿凯 19335121 liangrk5@mail2.sysu.edu.cn

**摘要**

概述：本实验利用已学习的图论知识，模拟广州地铁线路查询系统，为客户提供最短路径以及最少换乘方案。

主要方法：广度优先搜索，遍历求交集

结论：本程序可实现最短路径以及最少换乘两种乘车方案，但最短路程和实际情况有所出入，因为程序实现时默认两个站点之间的距离相同，最少换乘方案基本一致。

1. **引言**

**解决的问题：**模拟地铁线路图，求两个站点之间的最短路径，以及经过最少换乘站点的线路方案

**解决方法：**

最短路径：广度优先搜索

最少换成站点：遍历可能站点的路线求交集

**实验目的：**通过该实验，深入掌握如何存储点数以及线路复杂的图，熟悉运用广度优先搜索的方法解决最短路径问题等

1. **解决方法**

**输入形式：**通过输入数字（1，2，3）选择功能

1. 输入1后，输入查询线路名称。
2. 输入2后，输入起始点和终点的名称。
3. 输入3后，输入起始点和终点的名称。

**输出形式**：1.查询线路的所有站点名称（按次序）

2.起始点到终点的最短路径以及票价信息。

3.起始点到终点的最少换乘线路以及票价信息。

**使用的数据结构：**station类，vector实现的邻接表，队列

1. **station类**：

成员变量：

order(int)站点序号，name(string)站点名称，linenum(int)所在线路个数，adjacentnum(int)邻接点个数，line(vector<int>)存储所在线路序号的容器，

adjacentsite(vector<int>)存储邻接点序号的容器。

核心成员函数：

station(int order, string name, int linenum, int adjacentnum, vector<int> line, vector<int>adjacentsite) //构造函数

bool isOnLine(int num) //判断站点是否在输入线路中

bool isStation(string name) //判断名称是否匹配

int findNext(int line, vector<station> list, int lastSta) //输入线路序号，返回一个与输入站点不同的邻接点

vector<int> findNext(int line, vector<station> list) //输入线路序号，返回与之相邻的邻接点容器

int findNext(int line, vector<station> list, vector<int> used) //输入线路序号，以及一个已经遍历过的点的容器，返回一个不一样的邻接点

void reachableSta(vector<int>& staList, vector<int> lineSta[]) //将该站点的可达站点加入容器中且与容器中原有不重复

1. **vector实现的邻接表：vector<vector<int>>存储无权无向图**
2. **STL库中的queue**
3. **vector容器存储所有站点对象（序号对应该站点下标）**
4. **string类数组存储所有线路名称（序号对应线路下标）**

**算法描述**：

以下线路算法实现均默认站点与站点之间距离一定，即为无向无权图，故求得最短路径可能会与实际情况有出入，但最少换乘与相关软件答案基本一致。

1. 最短路径实现方法：

采用BFS广度优先搜索方法，利用队列先入后出的特点遍历地铁站点图，同时记录每个站点的父亲结点，当终点被遍历时（即得到最短路径），通过回溯广度优先搜索树得到起始点与站点之间的最短路径序号序列（并不唯一）。

打印线路时，通过序号直接访问存储站点对象的vector得到名称，同时定义currentline与nextline变量，两个变量通过求得两个邻接点的公共线路记录当前线路与下一线路，当二者不相等时打印出“换乘：下一线路”。

1. 最少换乘实现方法：

Step1：将起始点可直接到达的站点存入一个容器A中，遍历查看终点是否在其中，若在，则不需要换乘可直接到达，输出线路。

Step2：将终点可直接达到的站点存入一个容器B中，求A与B的交集，若有交集，即为一次换乘点。

Step3：将起点多换乘一次可到达的点全部存入容器A中，求与B的交集，若有交集，则为两次换乘点。

重复Step3

通过递归回溯线路并打印，算法复杂度较高，但在一定时间可得出正确答案

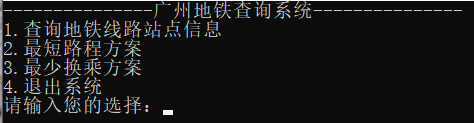
1. 文件读取站点相关信息实现

以读取文件的形式，将所有站点的序号，所在线路序号，邻接点序号存入站点对象中存入vector容器中

1. **程序使用和测试说明**

本程序编译运行前需将命名为 “guangzhou-subway.txt” 的文本文件和程序源代码加在同一文件夹下，再进行编译运行，否则会出现“打开文件出错”的提示。

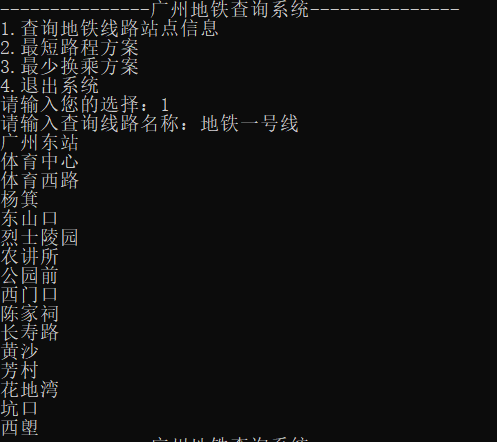
正常编译运行之后，显示主界面如下图



我们为客户提供了4种功能选择，下面分别做详细介绍：

1. 客户输入 1 之后， 可查询指定地铁线路上的所有站点信息。

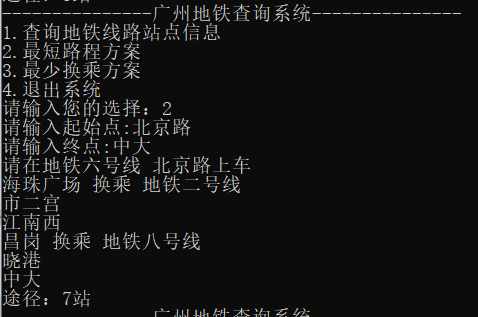
例如：查询 地铁一号线 上所有站点，输入 “地铁一号线” 后其上所有站点打印如下



这里注意，可查询的线路有："地铁一号线"，"地铁二号线"，"地铁三号线"，"地铁三号线北延段" ，"地铁四号线"，"地铁五号线"，"地铁六号线"，"地铁七号线"，"地铁八号线"，"地铁九号线"，"地铁十三号线"，"地铁十四号线"，"地铁二十一号线"，"地铁广佛线"，"地铁APM线"，"地铁十四号线支线" ，客户按需输入即可。

1. 输入2 时，可查询起点到终点的最短路径乘车方案。

例如：起点输入北京路，终点输入中大，程序运行结果为从地铁六号线的北京路上车，并指明在海珠广场站、昌岗站进行换乘，最后统计了所经过的站数

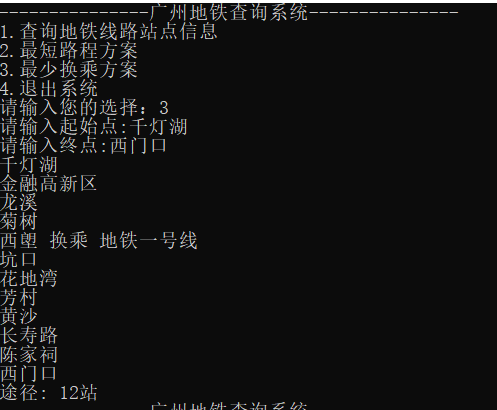


对比百度地图给出的出行方案



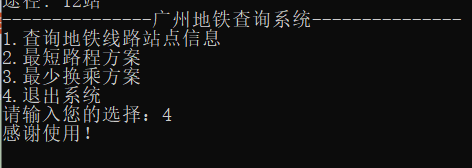
1. 输入 3 ，可查询起点到终点的最少换乘乘车方案。

例如：起点为 广佛线上的千灯湖，终点为西门口，本程序给出和百度地图相同的最少换乘方案，只在西塱换乘地铁一号线即可





1. 输入 4， 结束程序



1. **总结和讨论**

本次实验收获颇丰，对图的算法以及有关问题的解决有了更深刻的理解。然而由于地铁站点数较多，故大部分时间都花在了寻找信息和存储信息上，最短路径实现起来较为简单，而最少换乘算法参考了市面上此类的主流算法，时间复杂度较高，但由于换乘次数有限，故在短时间内依然可以得出正确方案。