中山大学数据科学与计算机学院本科生实验报告

（2019-2020 学年第一学期）

课程名称： 数据结构与算法分析　任课教师：乔海燕

年级&班级	18 级计科 8 班	专业(方向)	计算机类
学号	18340208	姓名	张洪宾

一、 实验名称:

广州地铁线路查询

二、 实验目的:

（1） 将图论知识进行运用，加深对知识的理解

（2） 提高我们面向用户的意识，设计出用户友好的程序

三、 实验内容:

设计并实现一个地铁线路查询程序，输出最佳乘车方案等信息。

此次实验我希望实现广州地铁 app 中的两个功能：线路查询和站点导航，并写出一个比较简单的 UI 交互界面。

线路查询即选择我在 UI 界面中放置的广州地铁的所有线路，选择某一个线路，输出该线路的起点至终点的所有站点。站点导航即输入起点和终点，程序将生成一条最佳的线路并输出。

四、 解决方法:

（1） 获取广州地铁数据

通过网上的 python 代码爬取广州地铁的数据，并将数据做了一定的处理，使数据变成如下格式，并存在 subway_information.txt 中：

第一行是广州地铁线路总数 n

然后输出 n 条线路的信息，每条线路的信息如下：

信息的第一行是线路的名称和该线路站点的数量 m，然后一下 m 行是该线路所有站点的名称及它到下一个站所需要的时间。

（2） 将数据读入程序并处理，构建出地铁的线路图

首先我按层次定义了三个类或结构体，一个是表示地铁站，一个表示线路，一个表示整个城市的地铁。我们不妨定义一个站为一个点，定义相邻站 x,y 之间的距离（用地铁的运行时间（单位：分钟）表示）为 T(x,y)。而由于地铁不是单线的，所以我们可以将整个地铁线网抽象成为一个简单无向图。特别注意的是，虽然我们从一端开始构建一条线，但是到最后其实我们得到的图是双向的。

注意到，在广州地铁线网中有大量的换乘站，并且 3 号线在体育西路站处有两个分支，14 号线在新和站处有两个分支。为了处理这两种情况，我想了应该办法，就是我们将换乘站当作多个站点，出现在它相连接的站点中，分别记为A_0、A_1、……、A_n。其中对$\forall i,j<n$，都有$T(A_i,A_j)=1$，即我们认为换乘时间为一分钟。以四号线和八号线相交处的万胜围站为例子,从大学城北到客村站需要在万胜围站转车,在我的模型中有万胜围 1 站和万胜围 2 站，其中万胜围 1 站所在线路为 4 号线， 万胜围 2 站所在线路为 8 号线。从万胜围 1 站转车到万胜围 2 站，便实现了从 4 号线转车到 8 号线。而对于分支，类似的思想，我将它定义为两条线路。

三个层次如图 1 所示：

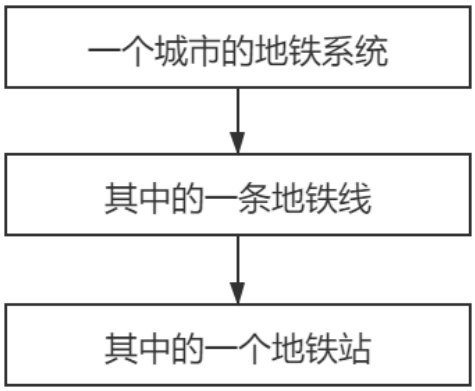

图 1

先分析最底层的 station 层次：

首先 station 应该含有站点的名称，站点所在的线路信息，然后还有离下一个站的距离。为了便于最后生成邻接矩阵，我还给每个站点按输入的顺序进行编号。

于是最终的 station 结构如下：

```
    //地铁站结构
struct station {
    string station_name;//站名
    int station_id;//站编号
    bool transfer;//是否为中转站
    int dist_to_next;//到下一站的距离
    string line_name;//在哪条线路
    station() = default;
    station(string name, int id_, int dist,string linename);
};
```

再分析中间的地铁线层：

首先作为一条地铁线应该有它的线路名称，然后还需要有它的具体线路信息，即它所有的站点信息。

所以 line 结构如下：

```
//地铁线结构
struct line {
    string line_name;//线路名称，如 1 号线
    vector<station> all_station;
    line() = default;
    line(string name, vector<station>stations);
};
```

最后是最顶层的地铁线网层次。作为一个城市的地铁线网，它应该包含所有的地铁线的信息。为了便于建图，我在该层次中加入了所有中转站的信息。具体类的结构如下：

```
//对整个地铁进行抽象
class subway {
private:
    vector<line> all_lines; //所有的地铁线
    vector<station> transfer_station; //所有的中转站
public:
    //用所有线路建立地铁网，并找到所有中转站
    subway(vector<line>lines);
    vector<station> get_transfer();
```

```
    //建立矩阵来表示地铁线网
    Graph mkgraph();
    //建立 map 用于找到地铁站的编号
    map<string, vector<int>>find_station_id();
    map<int, station>find_station();
    //计算一条从出发点到终点的距离
    int calcuate_dist(vector<int>res);
    //列出所有站点
    vector<station> list_all_station();
};
```

至此，所有的数据结构已经定义完毕，需要进行下一步操作来完成我们需要的功能。

（3） 寻找最佳路径

通过 subway 中的 mkgraph 函数，建立一个矩阵，用于储存整个图。而该图是以两个站点之间的权为元素，即它是一个无向有权图。于是我们可以用 Dijkstra 算法来解决这个问题。

我们先定义了结点类：

```
class Vertex {
public:
    int order;//结点的编号
    int dist;//结点到出发点的距离
    int from;//该结点的来源
    Vertex(int o, int d) {
        order = o;
        dist = d;
        from = -1;
    }
};
```

Dijkstra 算法：

```
设 d[0]=0，其他 d[i]=INF;//INF 是一个很大的值，用来替代正无穷
循环 n 次 {
    在所有未标号结点中，选出 d 值最小的结点 x;
    给结点 x 标记;
    将 x 的父母结点记为 d 值;
    对于从 x 出发的所有边(x,y)，更新 d[y] = min{d[y], d[x]+w(x,y)
}
```

然后我们从终点出发，将父母结点不断放到一个 stack 中，直到找到起点，然后再将 stack 不断 pop，将其中元素倒序放到一个 vector 中。

所以我们可以得到一条用路径编号表示出来的最优路径，它是以时间最小为依据的排序

的。而我们只需要再用 subway 类中生成的 map 将站点根据编号找到，就可以生成我们最终的乘车路线了。

（4） 设计 UI 交互界面

在这里我使用了 Windows 系统的 API 函数，通过不断地屏幕刷新和打印不同颜色的字体，已经使用 getc()函数读取方向键、Enter 键还有 Esc 键，最终使得我们可以用方向键、Enter 键还有 Esc 键对它进行选择。

五、 程序使用和测试说明：

我设计的用户界面比较简单。点开界面，如图 2 所示：程序会提示用方向键与回车键还有 Esc 键选择，第一个功能是地铁线网查询，第二个功能是出行导航。可以通过方向键控制选择，当前选择用彩色标识。而用回车键进行选择，用 Esc 键选择退出。

图 2

而当选择第一个功能后，如图 3 所示：地铁线网查询后会出现广州地铁的所有线路，同样也是用方向键与回车键还有 Esc 键进行选择.

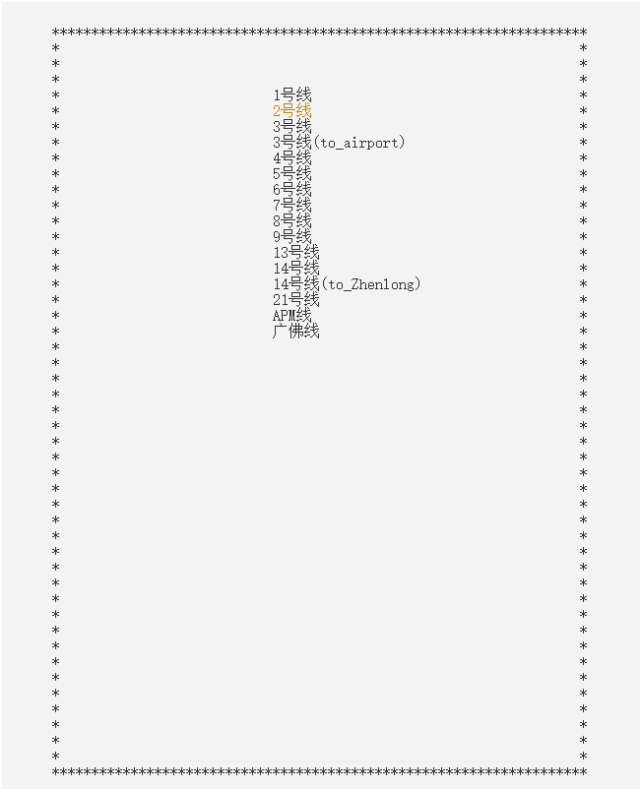

图 3

此时点击回车就可以看到我们查询的 2 号线的线路情况，如图 4 所示：

图 4

点击 Esc 回到起始界面，选择导航功能，可以得到输入界面，输入起点和终点后界面如图 5 所示：

图 5

点击 Enter 键我们可以看到查询的结果，如图 6 所示

图 6

此时按 Esc 可以返回初始界面，而再按 Esc 可以退出程序。

六、 总结与讨论：

此次实验主要分为两个 part，一个是用 Dijkstra 算法计算出最优路径，一个是写 UI 交互界面。Dijkstra 算法在之前的练习中有反复学习，所以问题的关键是怎么设计出一个用户友好的用户界面。

通过此次研究 windows.h，我稍微了解了一些关于 API 函数的知识，了解了操作系统的博大精深。而我本来是想要用 Qt 写一个图形界面的，迫于时间限制只能先写一个比较简单的 UI，我会在寒假花时间再用 Qt 做一个用户界面弥补本次的遗憾。

七、 参考文献：

[1]乔海燕、蒋爱军、高集荣和刘晓铭.数据结构与算法实验实践教程[M].北京：清华大学出版社，2012.