中山大學 SUN YAT-SEN UNIVERSITY

实 验 报 告

实验题目： 广州地铁线路查询

一. 实验目的

百度地图、高德地图等软件在导航时，如果选择出行方式为地铁，通常将提供从起点站到终点站的最短路径或者是多条规划路径的具体线路名称。本次实验旨在实现一个以广州地铁为基础数据的地铁线路查询程序，输入广州地铁中的起点站和终点站之后，输出从起点到终点的最优地铁乘坐路径，并根据实际统计数据给出一共经历的站点总数。

二. 实验环境

本实验可基于Visual Studio Code等平台开发，参考主流的编码规范，如Google C++Style Guide（中文版）

2.1 编程语言和开发工具

编程语言： ANSI C/C++

开发工具： Visual Studio Code、编译器G++

2.2 编码规范

遵循良好的程序设计风格来设计和编写程序。基本编码规范：

1. 标识符的命名要到达顾名思义的程度；

2. 关键代码提供清晰、准确的注释；

3. 程序版面要求：

   a) 不同功能块用空行分隔；

   b) 一般一个语句一行；

   c) 语句缩进整齐、层次分明。

三、实验要求

1. 原则上两人一组完成，每人写一份报告。报告应该详实规范清晰，内容应该数据结构和算法设计细节、程序测试、程序使用和总结。

2. 你的程序应该解决实际问题，并给出正确答案。

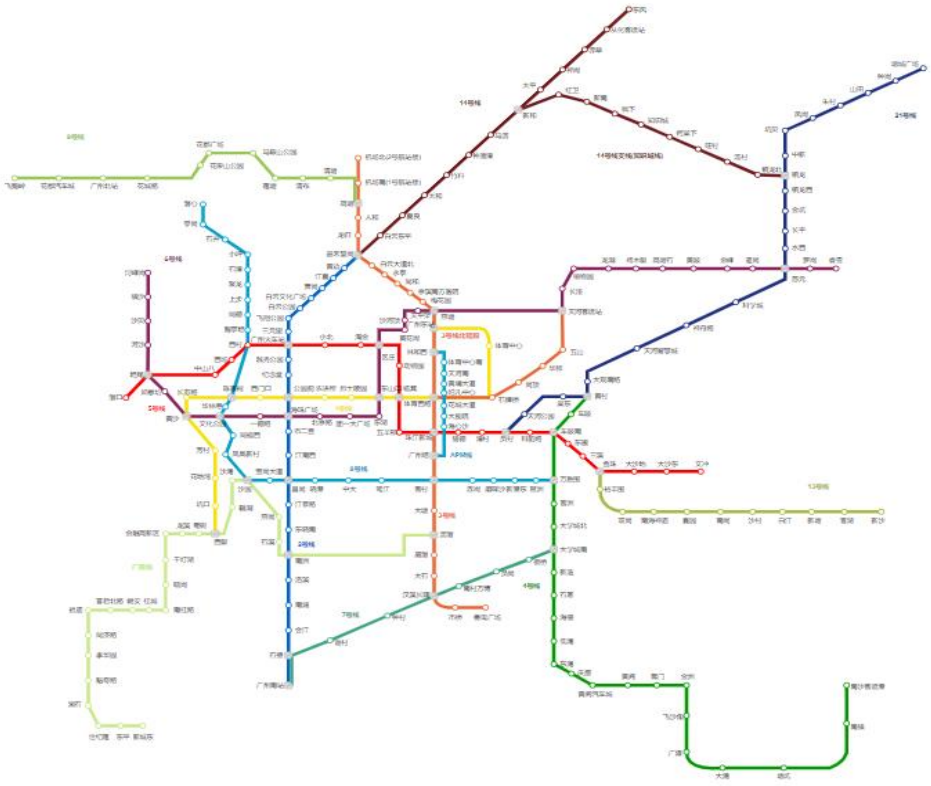
3. 程序输出信息可参照广州地铁线路查询、坐车网（zuoche.com）和百度地图线路规划等给出有关信息，可以自行设计，最低要求给出一种最佳乘车方案。

4. 我们将择时分组进行讲解演示，并根据完成情况打分。

四、实验内容

分析与设计

参考广州地铁线路查询和百度地图规划，得到如下广州市地铁线路图，本次实验数据以该图为标准。



广州市地铁线路图

实验原理

广度优先搜索（也称宽度优先搜索，缩写 BFS，以下采用广度来描述）是连通图的一种遍历算法这一算法也是很多重要的图的算法的原型。Dijkstra 单源最短路径算法和 Prim 最小生成树算法都采用了和宽度优先搜索类似的思想。其别名又叫 BFS，属于一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。基本过程，BFS 是从根节点开始，沿着树(图)的宽度遍历树(图)的节点。如果所有节点均被访问，则算法中止。一般用队列数据结构来辅助实现 BFS 算法。

广度优先搜索算法的搜索步骤一般是：

（1）从队列头取出一个结点，检查它按照扩展规则是否能够扩展，如果能则产生一个新结点。

（2）检查新生成的结点，看它是否已在队列中存在，如果新结点已经在队列中出现过，就放弃这个结点，然后回到第（1）步。否则，如果新结点未曾在队列中出现过，则将它加入到队列尾。

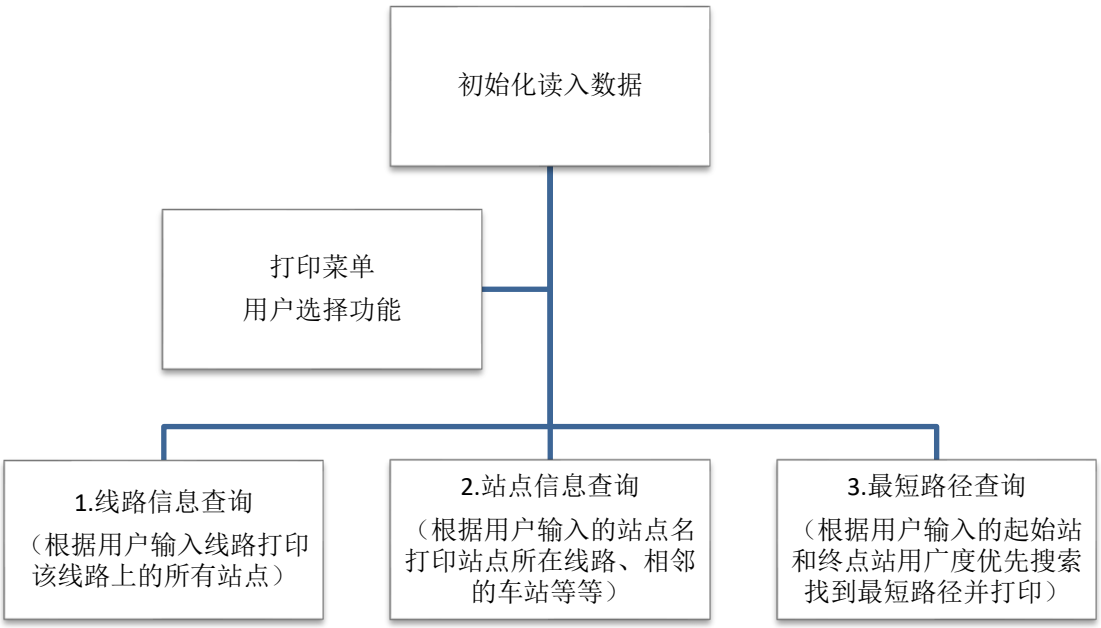
（3）检查新结点是否目标结点。如果新结点是目标结点，则搜索成功，程序结束；若新结点不是目标结点，则回到第（1）步，再从队列头取出结点进行扩展。

最终可能产生两种结果：找到目标结点，或扩展完所有结点而没有找到目标结点。

若找到目标结点，查询所经过的路径即最短路径。

本系统共实现 3 个功能，除了实验要求所必实现的最短路径查询功能外，还包括线路信息查询和站点信息查询

程序流程图



类的设计

1、Station 类

```
class Station{
public:
    string name;//站点名称
    string prev;//从哪个站来
    Station(){  name = "";}
    void set(const string &n) { name = n; }
    void setPrev(const string &p) { prev = p; }
};
```

Station 用于记录路径信息，即记录广度优先搜索树的每个结点的前驱

name 为站点名，prev 记录从哪一站来，搜索到目标站点时根据 prev 回溯，将所经过站点的信息存入最短路径的容器中

2、SubwaySystem 类

```
class SubwaySystem
{
//邻接表记录地铁线路图
    map< string, vector<string> > adj_table;
//用于记录 Station 信息：来自哪里；key=站点名称
    map<string, Station> stations;
//用于记录线路信息：线路上的所有站点；key=线路名称
   map<string, vector<string>> routes;
//用于记录站点的线路信息：在哪些线路上；key=站点名称
    map< string, set<string> > lines;
//用于记录最短线路，包括起始站和终点站
    list<string> theShortestPath;
private:
//根据线路符合打印线路名称
    void print_line(const string &);
//根据可换乘的线路数排序，在遍历时先遍历可换乘线路数较少的站点
    void sort_by_lines(vector<string> &);
//返回两个站点的相同的线路，用于记录换乘
    string get_line(const string &, const string &);

//处理文件信息
    void Handle_File(const char* name);
//广度优先搜得到以得到最短路径
    void BFS_Dispose(map<string,bool> &vis_table, const string& from,co
nst string& dest);
//打印最短路径
    void Print_ShortestPath();
 //查询最短路径
    void Find_Route(string& start,string& dest);

public:
    SubwaySystem(string filename);  //从文件读入地铁信息
    void Line_Inquiry();            //线路信息查询
    void Site_Inquiry();            //站点信息查询
    void Route_Inquiry();           //查询最短路径
};
```

功能实现

1、线路信息查询

```
void SubwaySystem::Line_Inquiry(){
    cout << endl;
    string line;
    cout << "Line:" << endl;
    cin >> line;
    if (routes.find(line) != routes.end())
    {
        cout << "Stations of ";
        print_line(line);
        cout << " :" << endl;
        for (int i = 0; i < routes[line].size(); ++i){
            cout << routes[line][i];
            if(i!=routes[line].size()-1)
                cout << " - ";
        }
        cout << endl;
    }
}
```

线路信息存储在 SubwaySystem 类成员 map< string, vector<string> > routes 中，

实现时根据用户输入的线路号打印输出该线路上的所有站点信息。

2、站点信息查询

```
void SubwaySystem::Site_Inquiry(){
    cout << endl;
    string site;

    cout << "Site:" << endl;
    getline(cin, site);

    if (lines.find(site) != lines.end()){
        cout << "Line Information:" << endl;//打印所属线路
        for(const auto&a:lines[site]){
            print_line(a);
            cout << endl;
        }
```

```
        cout << "Adjacent station(s):" << endl;//打印相邻站点名称
        for(const auto&a:adj_table[site])
            cout << a << endl;
    }
}
```

站点所属线路信息存储在 SubwaySystem 类成员 map< string, set<string> > lines

中，实现时根据用户输入的站点名称打印输出该站点的所属线路信息。

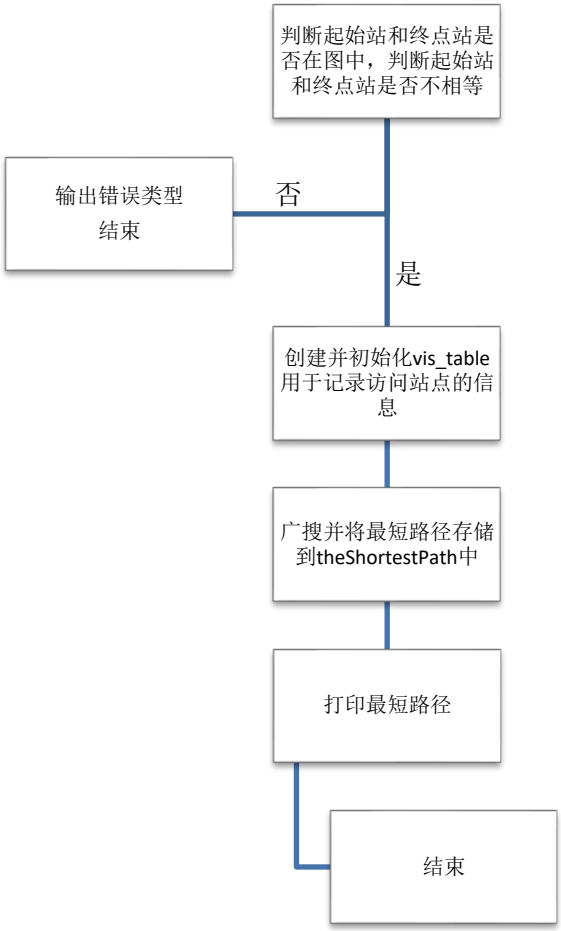
每个站的相邻站点信息存储在 SubwaySystem 类成员

map< string, vector<string> > adj_table 中，实现时根据用户输入的站点名称打印

输出该站点的所属线路信息。

3、查询最短路径

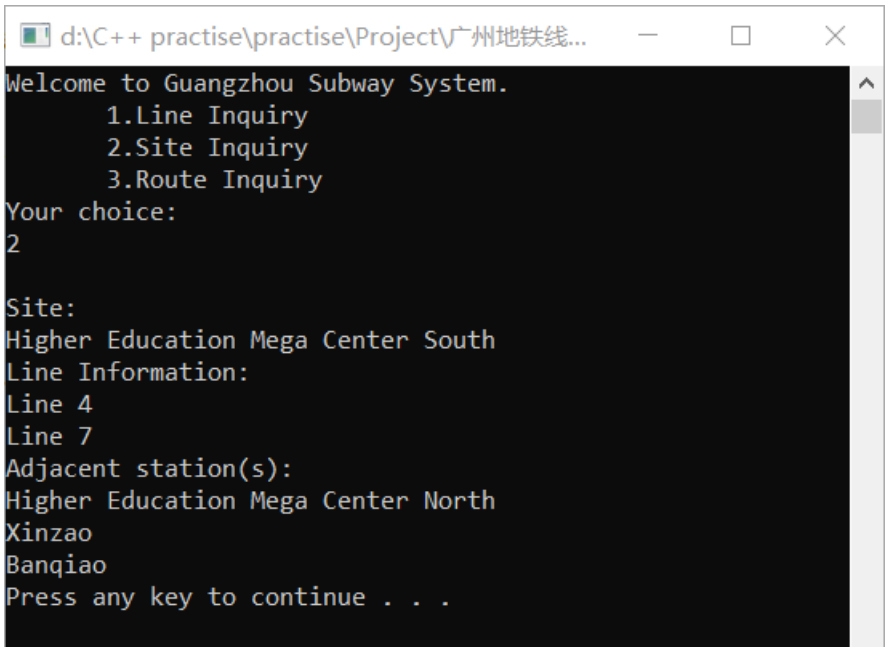
```
void SubwaySystem::Find_Route(string& start,string& destination)
{
    if(stations.find(start)!=stations.end()&&stations.find(destination)
!=stations.end()&&start!=destination){
        map<string, bool> vis_table;
        for(auto a:stations)
            vis_table[a.first] = false;

        BFS_Dispose(vis_table,start,destination);
        cout << endl;
        cout << "The shortest path:" << endl;
        Print_ShortestPath();
    }
    else if(stations.find(start)==stations.end())
        cout << "Missing departure station." << endl;
    else if(stations.find(destination)==stations.end())
        cout << "Missing destination." << endl;
    else
        cout << "The departure station is the same as the destination."
<<endl;
}
```


判断起始站和终点站是否在图中，判断起始站和终点站是否不相等
否
输出错误类型
结束
是
创建并初始化vis_table用于记录访问站点的信息
广搜并将最短路径存储到theShortestPath中
打印最短路径
结束

实验结果：


d:\C++ practise\practise\Project\广州地铁线路查询\bin\main.exe
Welcome to Guangzhou Subway System.
1.Line Inquiry
2.Site Inquiry
3.Route Inquiry
Your choice:
1
Line:
4
Stations of Line 4 :
Huangcun - Chebei - Chebeinan - Wanshengwei - Guanzhou - Higher Education Mega Center North - Higher Education Mega Center South - Xinzao - Guanqiao - Shiqi - Haibang - Dichong - Dongchong - Qingsheng - Huangge Auto Town - Jiaomen - Jinzhou - Feishajiao - Guanglong - Dachong - Tangkeng - Nanheng - Nansha Passenger Port
Press any key to continue . . .

线路信息查询



站点信息查询

五、实验心得

在这一次实验过程中，我对于图的构建以及map、vector等容器的构成有了更多的了解，对于遍历、搜索等算法的原理有更深刻的认识；此外，也让我更加熟悉了数据结构的设计流程和测试流程。

在这次实验中遇到更多的是一些问题和挑战。最大的障碍是如何合理的处理数据以及其

线路站点信息，如何选择合适的算法计算出最短路径并输出等等，对于这些问题，在网络上查找相关资料、参考别人的做法必不可少，还需要比较不同做法在实现速度上、在代码可观性等方面的优劣，这种解决问题的方法在之后的学习过程中也会持续使用到。参考别人的方法，是相对比较高效的学习方法之一。对于许多之前不熟悉但通过学习别人的应用方法觉得巧妙的方式方法，也可以记录跟学习，扩充自己的知识。

除此之外，在本次实验中也遇到了许多大大小小的问题，比如VScode的输出框输出中文会出现乱码，经过网上的资料查找和搜索发现需要在全局变量中进行设置，考虑到效率等问题，选择放弃中文输出而采用英文的导语输出。同时本次的实验实现还有许多不完善的地方和许多可以进一步进行优化的内容，例如换乘方案的时间延迟没有考虑在内，还有应该如何进一步优化算法、如何减少程序的运行时间等，可以用自然语言等描述的算法逐步细致化、精确化、形式化，还有特别注意特殊情况、边界条件的考虑，尽可能地增强程序的容错性。这些问题都值得进行进一步的思考、处理。

最后是具体的测试过程中，或许应该考虑增加一些输出让使用者了解整个程序的实现过程。

五、参考资料

https://blog.csdn.net/qq_19782019/article/details/82659964

https://blog.csdn.net/weixin_43388615/article/details/105539003

https://blog.csdn.net/zyf__2015/article/details/106950186