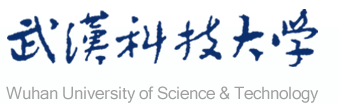
****

《数据结构》实验报告

**专 业：** 计算机科学与技术（卓越计划）

**班 级：** 计算机（卓越）1901

**学 号：** 201913137031

**姓 名：** 杨尔行

**实验4 图**

**一、【实验目的】**

1. 掌握图的定义和图的邻接表存储结构；
2. 掌握图的创建方法；
3. 掌握顶点和边的操作；
4. 掌握图的基本算法并能实际应用；
5. 掌握图的深度优先搜索算法以及实现方法；
6. 使用C/C++语言和图实现“公交线路图”专题。

**二、【实验环境】**

硬件：Intel Core i5 8300H CPU 2.30GHZ

软件：Windows 10 64bit操作系统，编译环境 Visual Studio 2019

**三、【实验内容】**

程序为控制台程序,使用图数据结构和算法,开始运行时,输出菜单,供用户选择。具体实现的功能如下：  
1）创建公交线路图输入站点和公交线路数据,程序根据站点信息和线路信息创建公交线路图。  
(1)站点信息集合(编号、名字)。  
(2)公交线路信息集合(线路编号,线路两端站点编号,路段长度)。  
2）查询公交线路和站点信息  
为验证公交线路图是否创建成功,程序需实现查询公交线路和站点信息功能。

(1)查询公交线路:输入公交线路编号,系统通过公交线路编号查找到该线路途经的所有站点并输出。

(2)查询站点信息:输人站点编号,系统通过站点编号查找到所有经过该站点的公交线路并输出。  
3)查询两站点之间的路线,找到至多换乘 1 次的路线,并输出结果  
用户输人要查询的起点和终点,程序先判断两个站点之间是否有一条路径（即两个站点之间是否连通)。若两个站点之间有路线,则找到所有最多换乘 1 次的路线,然后依次输出,如图 1-39所示。共找到N条路线,如图1-40所示。  
(1)提示共找到几条路线:从|起点站名|到|终点站名|共找到N条路线,如图1-40所示。

(2)循环依次输出每条路线,有路线编号和站点与公交信息。依次输出路线中经过的每一站,并在站点与站点之间输出两站之间所坐的公交车名,如图 1-41所示。

(3)若两个站点之间没有可以找到的路线,则提示用户“两站点之间没有公交路线!”,如图 1-42所示。若两个站点之间有路线,但是不满足最多换乘 1.次的条件,则提示用户“没有满足条件的路线!”,如图 1-43 所示。

**四、【实验程序】**

1)程序源代码

**文件名：main.cpp**

#include "dialog.h"

int main()

{

    Dialog(R"(D:\Projects\AlgorithmLab\e4\test\_data.txt)")

        .begin\_dialog();

}

**文件名：dialog.h**

#ifndef \_DIALOG\_H\_

#define \_DIALOG\_H\_

#include "map.h"

/// 负责UI层的互动

class Dialog

{

Map map;

/// 查询公交线路

void query\_bus();

/// 查询两点间的全部路径

void query\_path();

/// 查询经过车站的公交线路

void query\_stop();

public:

Dialog(const char\* file\_name);

Dialog(const Map& map);

void begin\_dialog();

};

#endif // \_DIALOG\_H\_

**文件名：map\_scanner.h**

#ifndef \_MAP\_SCANNER\_H\_

#define \_MAP\_SCANNER\_H\_

#include <iostream>

#include "map.h"

/// 从文件流读取地图

class MapScanner

{

public:

    Map scan(std::istream& s);

};

#endif // \_MAP\_SCANNER\_H\_

**文件名：map\_scanner.h**

2)程序运行结果截图

AquaLine

GreenAvenue

BlueRoad

RedStreet

YellowLake

BlackHill

OrangeLine

LimeLine

***说明：输入AquaLine，表示用户要显示AquaLine这条公交线路（蓝色）***

***说明：输入1，表示用户要显示某条公交线路***

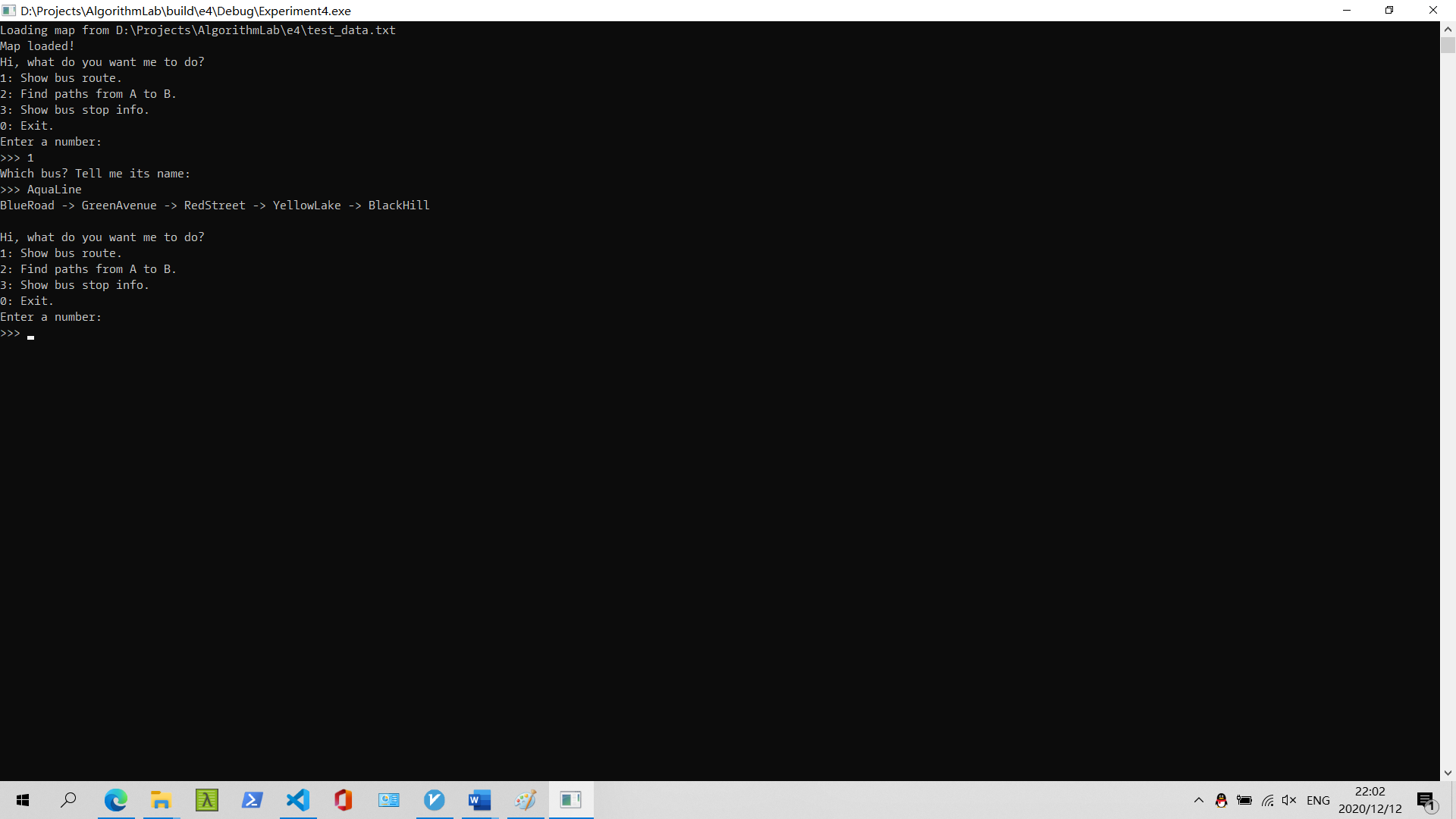
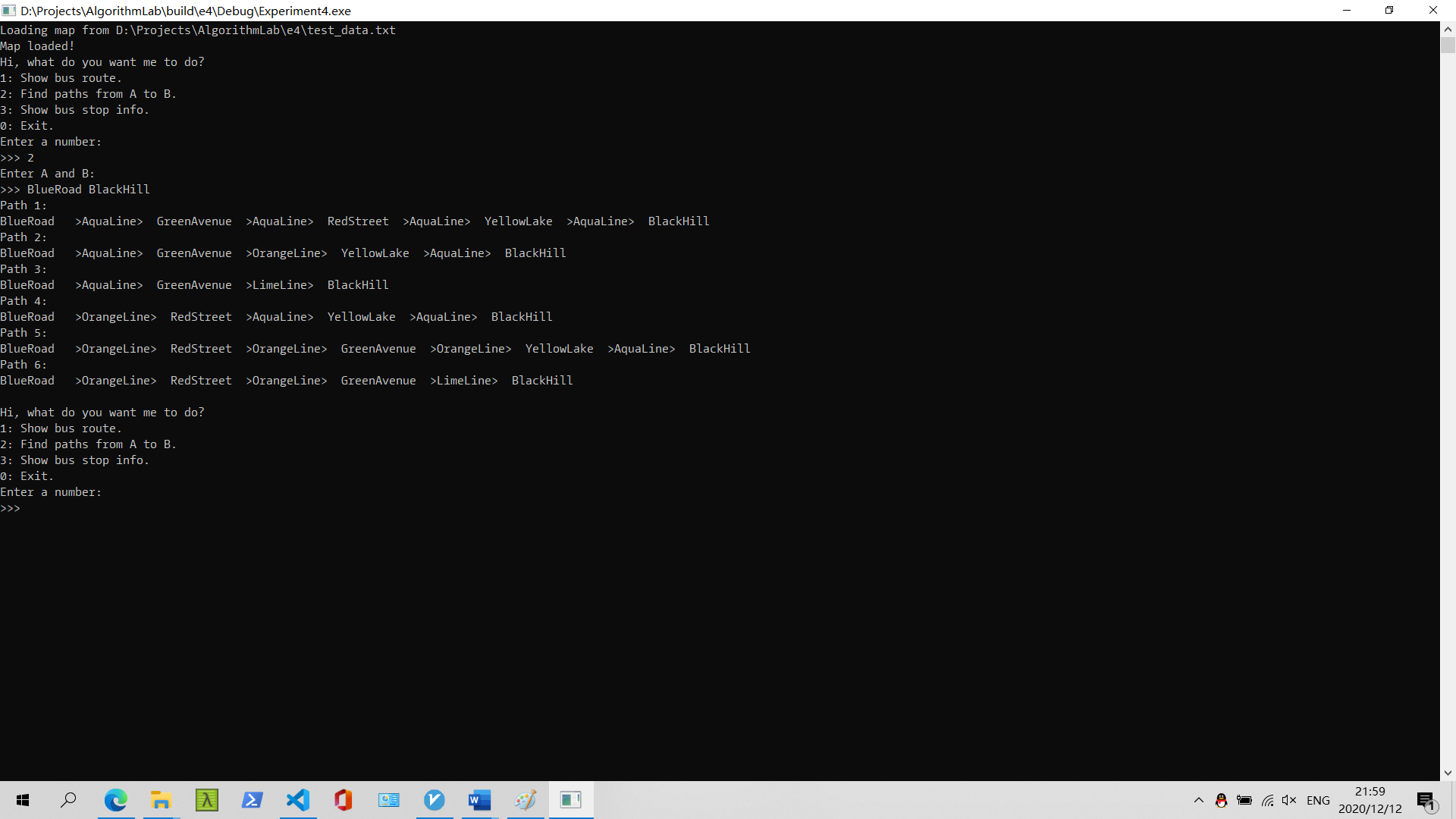


图1：显示某条公交路线



***说明：输入BlueRoad BlackHill，表示用户要显示能从前者到后者的全部路径***

***说明：输入2，表示用户要显示两站点之间的全部路线***

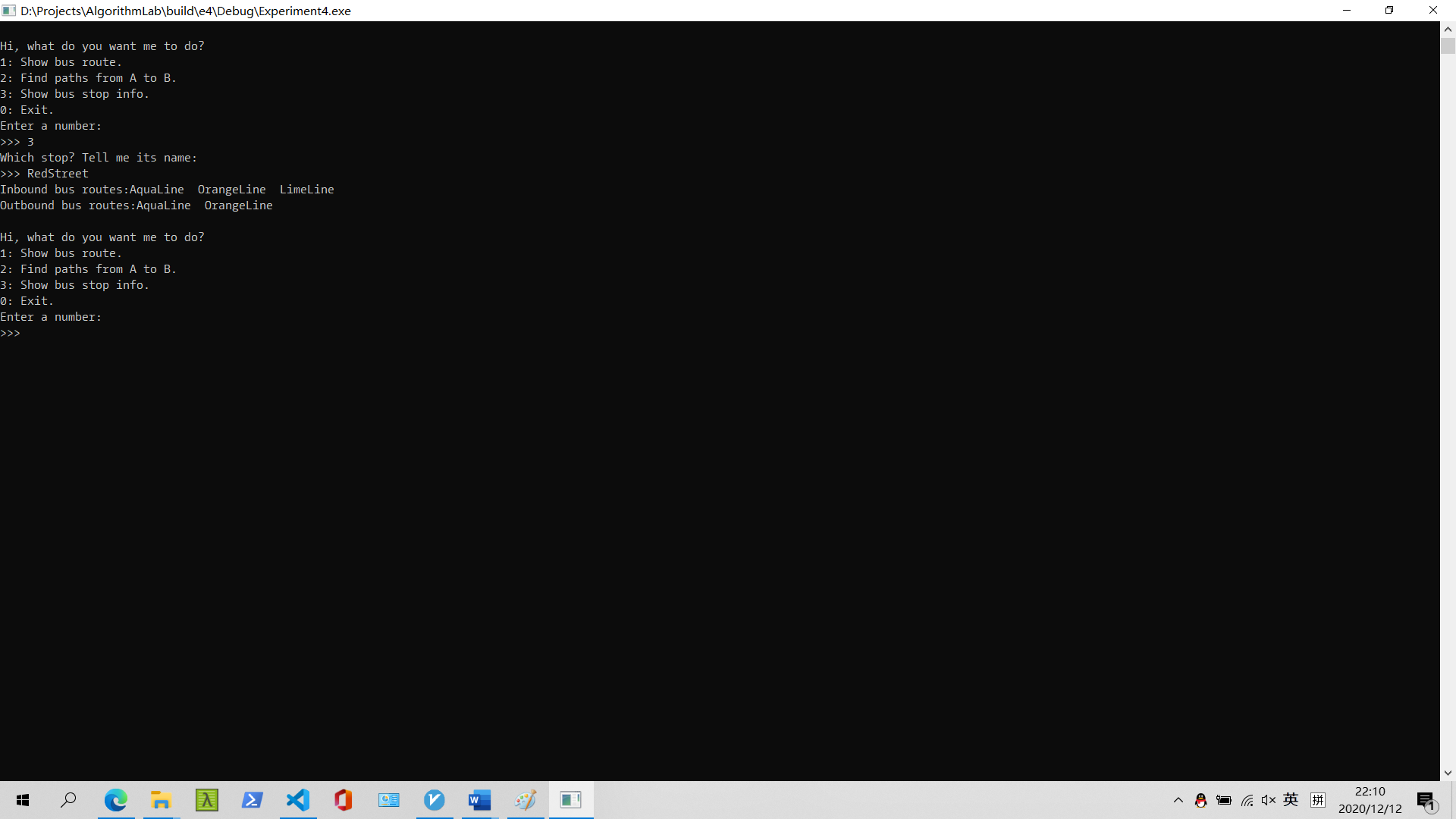


图2：显示两站之间的所有路线

图3：显示经过某站的路线

**五、【实验小结】**

本次实验使用了第2章线性表的知识，实现了链表的一些操作。一元多项式的相加本质上是对两个有序链表进行合并的过程。维护两个指针，分别指向两个加数链表。在循环中，不断取较低幂项加入结果链表，如果等幂，则系数相加再加入结果链表。由于仅需要遍历两个加数链表各一次，算法的时间复杂度为O(n)。

由于算法涉及较多的线性表插入和删除操作，使用链式存储结构显然是最合适的。