数据结构课程设计报告

计算机科学与技术 19307130266 林山力

一、选题：交通查询系统

【问题描述】

为满足广大复旦教师和学生的出行需要，复旦大学计算机学院邀请你为其开发一款交通查询 APP。出于不同的目的，教师和学生对交通工具有着不同的要求。 例如，因公出差的教师希望在旅途中耗费的时间尽可能短，出门旅游的学生则希 望旅费尽可能省，而行动不便的人则要求中转次数最少。所以，你开发的 APP 需要具备在各个城市间进行线路规划的能力，能够实时为教师和学生提供两种或三种最优决策的交通咨询。

程序规定有如下设置：

(1)为评估 APP 是否已经达到预定设计目标，请给出 APP 的运行实例。如对于路线咨询功能，需要展示：由用户输入起始站、终点站、最优决策原则和交通工具，程序输出最快需要多长时间才能到达或者最少需要多少旅费才能到达，并详细说明依次于何时乘坐哪一趟列车或哪一次班机到何地，整个运行流程需在报告中附上截图；

(2) 要求程序有一定鲁棒性，能应对一些错误的输入；

(3) 不允许使用数据库系统来存储各类信息，只能使用普通文件；

(4) 只需要实现核心数据结构和算法，能从控制台进行输入即可，不需要提供图形化界面。

二、需求分析

叙述每个模块的功能要求

1.实现对城市信息进行编辑（如添加或删除）的功能。

2.假定城市之间有两种交通工具：火车和飞机。实现对列车线路以及飞机航班 班次信息进行编辑（增设或删除）的功能。班次信息最少应包括：起始站、终点站、出发时间、到达时间、票价。

3.打印时刻表。输入城市名称，打印由该城市出发的所有列车路线以及飞机航班班次信息。

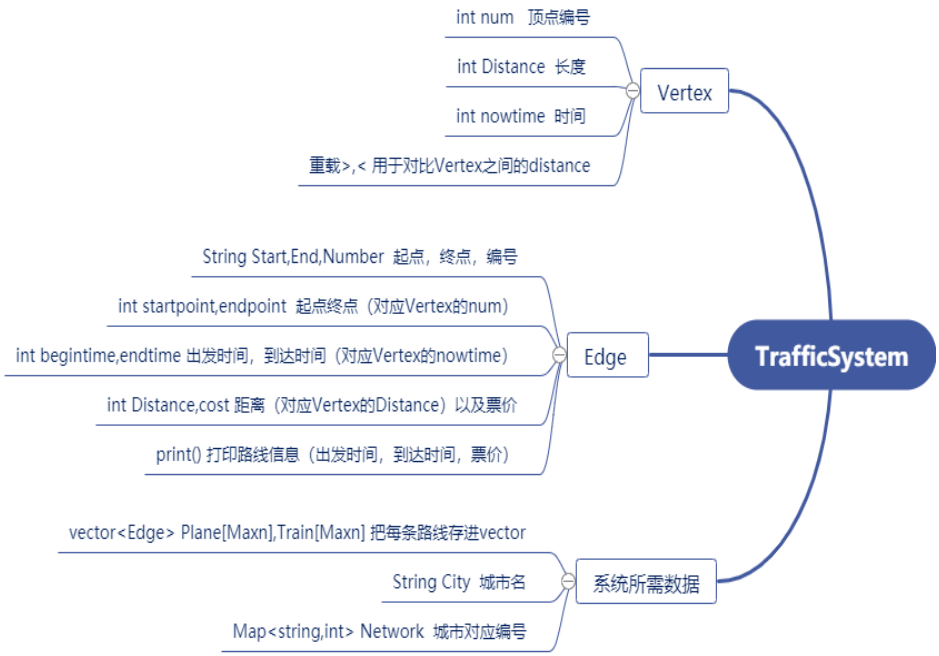
4.在确定起始站和终点站以及交通工具（全程不变）的情况下，系统应能提供两种最优决策：最快到达或最省钱到达。（旅途中耗费的总时间应该包括中转等候的时间）

三、概要设计

1.将交通系统设计为一个类，需要的东西全放入类定义。

2.用文件存放所有需要的数据，即火车时刻表和航班时刻表，以此搭建交通系统图，整个交通系统图在这些数据的基础上跑起来。

3.再使用两层循环不断调用类定义中的功能函数，实现 app 的启动和结束。



交通系统的类定义需要的数据结构 Vertex 代表图中的顶点。Edge 代表图中的边。加上系统中所需要的数据就构成抽象的交通系统图，再将各个数据结构的抽象类定义具化后得到以上内容，定义旁都有对应的注解。

PS: 这个图里的"系统所需数据"少写了一个 Total，代表城市个数。

以下为基于以上关键数据结构的功能函数。

```
struct Vertex {
    int num, Distance, nowtime;
    bool operator < (const Vertex& b)const { return this->Distance < b.Distance; }
    bool operator > (const Vertex& b)const { return this->Distance > b.Distance; }
};
```

```
struct Edge {                                                                    //起
    string Start, End, Number;
    int startpoint, endpoint, begintime, endtime, Distance, cost;
    int change(string s) {                                                       //用
        int res = ((s[0] - '0') * 10 + (s[1] - '0')) * 60;
        res += ((s[3] - '0') * 10 + (s[4] - '0'));
        return res;
    }
    Edge() :Start(""), End(""), Number(""), startpoint(0),
        endpoint(0), begintime(0), endtime(0), Distance(0), cost(0) {} //初始化
    Edge(string num, string a, string b, string c, string d, int s, int t, int p) : //边
        Number(num), Start(a), End(b),
        begintime(change(c)), endtime(change(d)), startpoint(s), endpoint(t), cost(p) {}
    void PrintTime(int t) { printf("%02d:%02d", t / 60, t % 60); }                //打
    void print() {                                                               //打
        cout << Number << " " << Start << " " << End << " ";
        PrintTime(begintime); cout << " ";
        PrintTime(endtime); cout << " " << cost << endl;
    }
};
```

```
int Total = 0;
map<string, int> Network;
string City[Maxn];
vector<Edge>Plane[Maxn], Train[Maxn];
```

功能函数对应所有基本要求。

四、详细设计

类定义代码都在 Traffic.h

类定义功能函数 #begin#

```
TrafficSystem::TrafficSystem() {
    Total = 0;
    Network.clear();
    for (int i = 0; i < Maxn; i++)  Plane[i].clear();
    for (int i = 0; i < Maxn; i++)  Train[i].clear();
}
```

构造函数，初始化 Network 以及 Plane[Maxn],Train[Maxn];

```
int TrafficSystem::PassTime(int x, int y, bool f) {
    if (f) return 0;
    x = x % 1440;
    y = y % 1440;
    return (y - x + 1440) % 1440;
}
```

计算需要耗费的时间。

```
int TrafficSystem::Scheme(Edge k, int type, Vertex intemp, int startpoint, int Distance) {//1 for 时间最短,2 for 路费最少
    if (type == 1)
        return PassTime(k.begintime, k.endtime, 0) + PassTime(Distance, k.begintime, startpoint == intemp.num);
    else if (type == 2)
        return k.cost;
    else
        return 0;
}
```

用来选择方案，用对应方案返回路线权值（时间最短/花费最少）

```
void TrafficSystem::PrintPath(int startpoint, int v, Edge path[]) {
    if (v == startpoint)    return;
    PrintPath(startpoint, path[v].startpoint, path);
    path[v].print();
}
```

递归调用，打印整条路线上的所有信息。

```
void TrafficSystem::Conversion(int Time) {
    int h = Time / 60, m = Time % 60;
    printf("%d小时%d分", h, m);
}
```

换算时间，然后输出。

```
void TrafficSystem::ReadInFile() {
    string Number, startpoint, endpoint, sttime, edtime, kind;
    int p, N;
    ifstream fin("shuju.txt");
    fin >> kind >> N;
    for (int i = 0; i < N; i++) {
        fin >> Number >> startpoint >> endpoint >> sttime >> edtime >> p;
        if (Network[startpoint] == 0) {
            Network[startpoint] = ++Total;
            City[Total] = startpoint;
        }
        if (Network[endpoint] == 0) {
            Network[endpoint] = ++Total;
            City[Total] = endpoint;
        }
        int S = Network[startpoint], E = Network[endpoint];
        Edge New(Number, startpoint, endpoint, sttime, edtime, S, E, p);
        Train[S].push_back(New);
    }
    fin >> kind >> N;
    for (int i = 0; i < N; i++) {
        fin >> Number >> startpoint >> endpoint >> sttime >> edtime >> p;
        if (Network[startpoint] == 0) {
            Network[startpoint] = ++Total;
            City[Total] = startpoint;
        }
        if (Network[endpoint] == 0) {
            Network[endpoint] = ++Total;
            City[Total] = endpoint;
        }
        int S = Network[startpoint], E = Network[endpoint];
        Edge New(Number, startpoint, endpoint, sttime, edtime, S, E, p);
        Plane[S].push_back(New);
    }
};
```

从文本文件中读取数据，用以搭建基本系统。文件内的数据是从网上找的。

```
void TrafficSystem::DeleteEdge(vector<Edge> edge[]) {
    vector<Edge>::iterator It;
    string Number, startpoint, endpoint;
    cin >> Number >> startpoint >> endpoint;

    It = edge[Network[startpoint]].begin();
    for (It = edge[Network[startpoint]].begin(); It != edge[Network[startpoint]].end(); It++) {
        if (It->Number == Number) {
            edge[Network[startpoint]].erase(It);
            return;
        }
    }
}
```

删除路线，用城市名去 Network 里找到这个城市对应的 int，然后回到 edge[]也就是存放路线信息的 vector 里删除对应路线。

```
void TrafficSystem::Solution(string a, string b, int type, string traffic) {
    if (traffic == "火车")
        ShortestPath(Network[a], Network[b], Train, type);
    else
        ShortestPath(Network[a], Network[b], Plane, type);
}
```

根据输入获取对应交通工具的最短路径（这里 type 就是对应时间或者钱最少）

```
void TrafficSystem::ShortestPath(int startpoint, int endpoint, vector<Edge> edge[], int type) {
    bool Visit[Maxn];
    int Distance[Maxn], i, v;
    Edge path[Maxn];
    Vertex intemp{ startpoint,0,0 }, newtemp;
    priority_queue<Vertex, vector<Vertex>, greater<Vertex> > prior_queue;
    for (i = 1; i <= Total; i++) {
        Visit[i] = false;
        Distance[i] = inf;
    }
    Distance[startpoint] = 0;
    prior_queue.push(intemp);
    Visit[startpoint] = true;
    while (!prior_queue.empty()) {
        intemp = prior_queue.top();
        prior_queue.pop();
        Visit[intemp.num] = false;
        for (v = 0; v < edge[intemp.num].size(); v++) {
            Edge k = edge[intemp.num][v];
            if (Distance[k.endpoint] > Distance[intemp.num] + Scheme(k, type, intemp, startpoint, intemp.nowtime)) {
                Distance[k.endpoint] = Distance[intemp.num] + Scheme(k, type, intemp, startpoint, intemp.nowtime);
                path[k.endpoint] = k;
                if (Visit[k.endpoint] == false) {
                    Visit[k.endpoint] = true;
                    newtemp.nowtime = k.endtime;
                    newtemp.num = k.endpoint;
                    newtemp.Distance = Distance[k.endpoint];
                    prior_queue.push(newtemp);
                }
            }
        }
    }
    if (Distance[endpoint] == 0) {
        cout << "未查找到这条路线，可以选择编辑从而新增这条路线！" << endl;
        return;
    }
    if (type == 1) {
        cout << "旅行时间最短:"; Conversion(Distance[endpoint]); cout << endl;
        cout << "选择如下方案:" << endl;
        PrintPath(startpoint, endpoint, path);
    }
    if (type == 2) {
        cout << "旅行费用最少:" << Distance[endpoint] << "元" << endl;
        cout << "选择如下方案:" << endl;
        PrintPath(startpoint, endpoint, path);
    }
    cout << endl;
}
```

用 Dijkstra 算法求最短路径，借助优先级队列来的最小堆获取最小值。结果存在 Distance 数组中，查不到的退出，查到了根据 type 打印最短路径。

```
void TrafficSystem::AddEdge(vector<Edge> edge[]) {
    string Number, startpoint, endpoint, sttime, edtime;
    int p;
    cin >> Number >> startpoint >> endpoint >> sttime >> edtime >> p;
    if (!Network[startpoint]) {
        Network[startpoint] = ++Total;
        City[Total] = startpoint;
    }
    if (!Network[endpoint]) {
        Network[endpoint] = ++Total;
        City[Total] = endpoint;
    }
    int S = Network[startpoint], E = Network[endpoint];
    Edge New(Number, startpoint, endpoint, sttime, edtime, S, E, p);
    edge[S].push_back(New);
}
```

增加一条路线。

```
void TrafficSystem::DeleteCity(string a) {
    Train[Network[a]].clear();
    Plane[Network[a]].clear();
    Network[a] = 0;
}
```

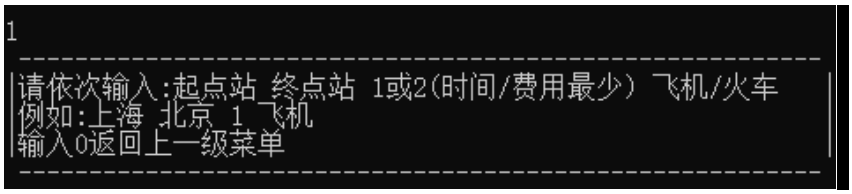
删除城市以及相关的路线信息。

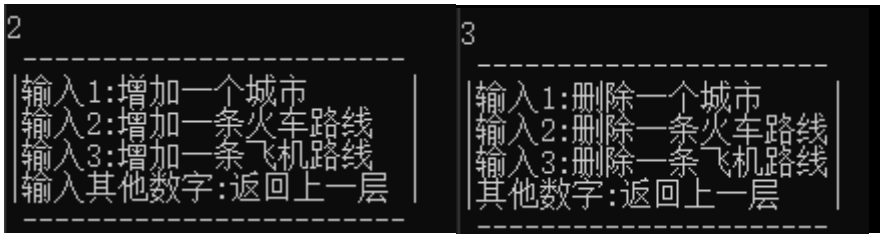
```
void TrafficSystem::print(vector<Edge> edge[]) {
    for (int i = 1; i <= Total; i++)
        for (int j = 0; j < edge[i].size(); j++)
            edge[i][j].print();
    cout << endl;
}
```

打印火车时间表/航班信息表。
类定义功能函数 #end#

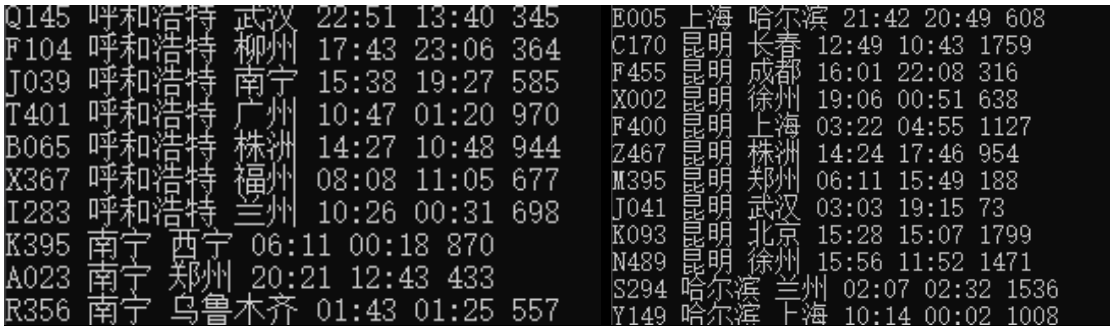
主函数会在附在.cpp 文件里
程序运行界面#begin#

```
------欢迎进入交通查询系统-------
|       输入1:查询最优路线       |
|       输入2:新增路线           |
|       输入3:删除路线           |
|       输入4:输出火车时间表     |
|       输入5:输出飞机时间表     |
|       输入其他数字:退出系统    |
 --------------------------------
```





输入 4 和 5 之后出现时刻表，然后返回主菜单。



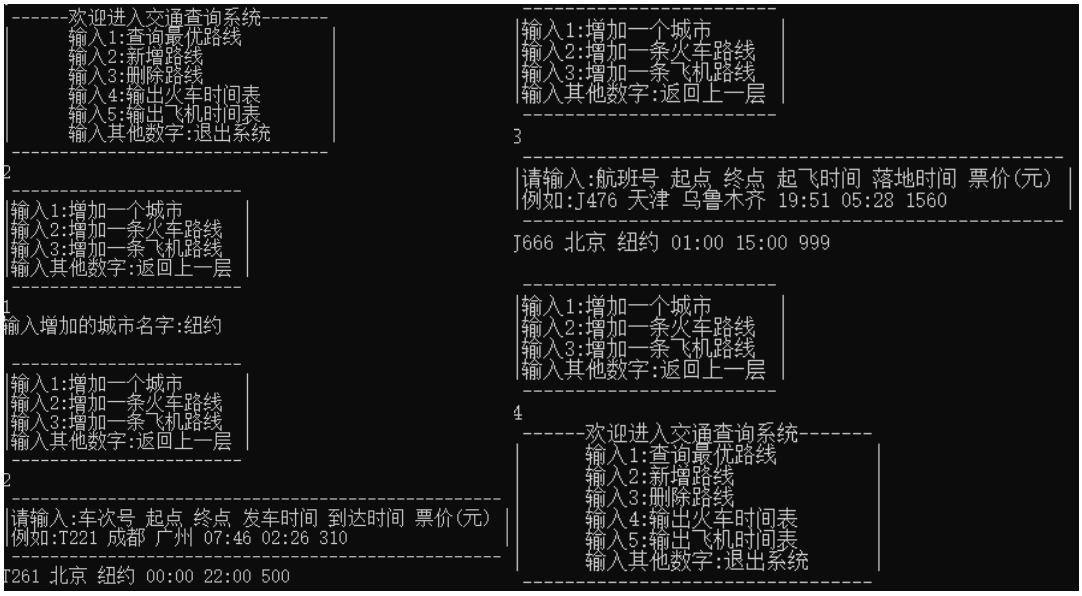
程序运行界面#end#

五、调试分析

包括测试数据、测试输出的结果（输出内容要求附上截图）、时间复杂度分析、对每个模块设计和调试时所存在问题的思考（有哪些问题？问题如何解决？）、 及算法是否可以改进等。

1.关于最优路线的测试数据以及结果:

2.关于新增路线的测试数据以及结果:

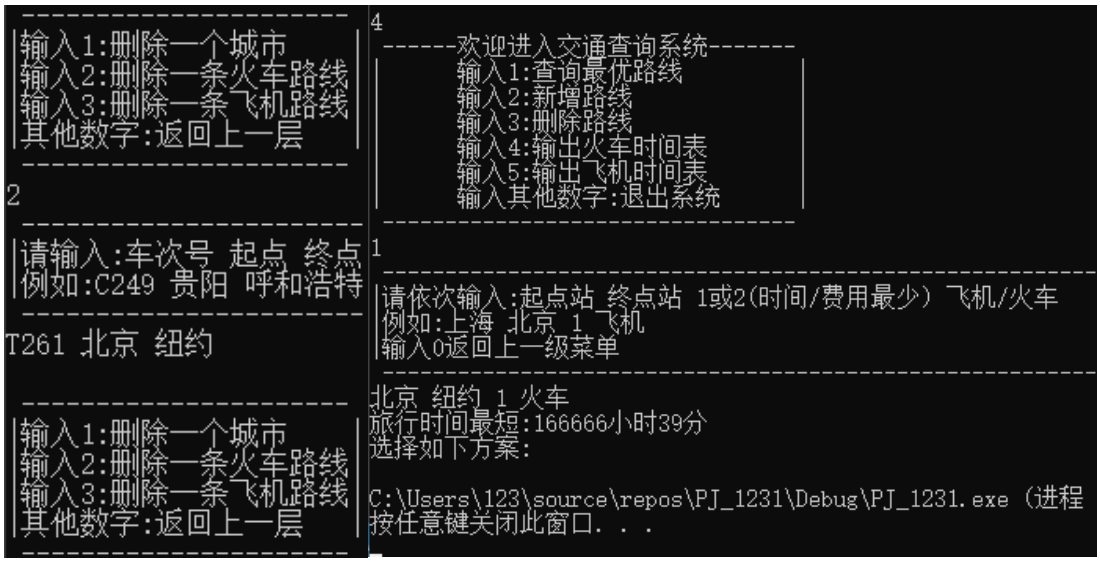


新增成功

现测试新增的路线是否存在并能否使用

路线加入成功，并且可以正常使用

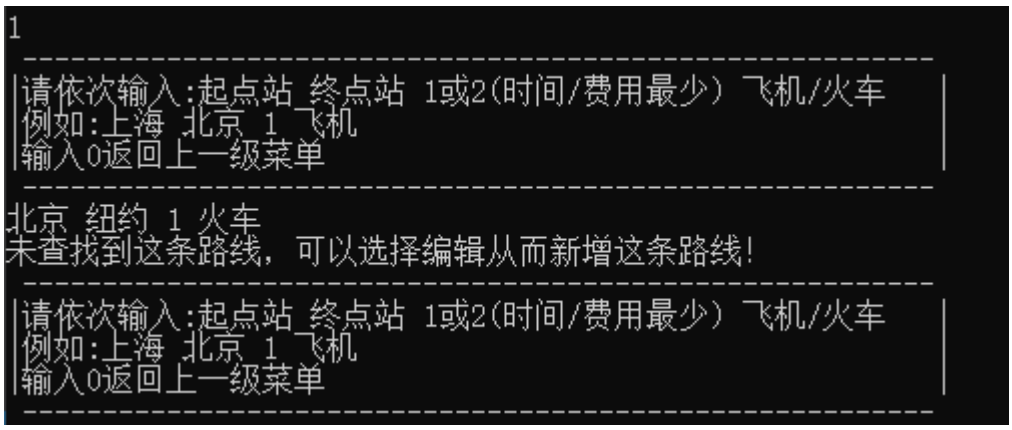
3.现在删除去纽约的火车路线，检测删除路线是否正常



发现可以正常删除，但是再次查找该路线时，出现了异常，并异常结束了。

经过寻路发现出现问题的是 shortestpath（）函数，我们之前增加了这条路线，然后有删除了，对于函数中的 Distance[endpoint] 没有置零而是在置了 inf 一个很大的数，所以在这里修改异常退出条件就是 Distance[endpoint] >= inf/2 。然后重新运行程序，按之前的操作加入纽约这个城市，加入火车路线，再删除，再查找，发现正常运行了！

```
if (Distance[endpoint] == 0 || Distance[endpoint] >= inf/2) {
    cout << "未查找到这条路线，可以选择编辑从而新增这条路线！" << endl;
    return;
}
```



4.关于输出火车时间表/航班信息表

```
4
P467 大连 乌鲁木齐 07:29 22:04 479
Q493 大连 天津 13:00 14:31 705
A335 大连 贵阳 06:45 13:04 182
H496 大连 上海 11:24 05:02 76
M162 大连 株洲 04:33 20:56 453
F143 大连 兰州 17:48 13:48 992
H168 乌鲁木齐 昆明 00:42 03:35 411
H093 乌鲁木齐 沈阳 21:32 02:22 223
I064 乌鲁木齐 徐州 19:20 13:33 975
N285 乌鲁木齐 上海 12:14 20:55 625
H152 乌鲁木齐 徐州 13:40 08:06 411
S381 乌鲁木齐 成都 13:21 18:55 856
H329 乌鲁木齐 哈尔滨 08:12 03:57 550
Z084 乌鲁木齐 郑州 18:18 08:10 288
E341 乌鲁木齐 郑州 15:57 21:46 691
Q278 乌鲁木齐 柳州 07:25 06:14 72
K037 乌鲁木齐 南宁 10:58 14:02 808
J257 乌鲁木齐 武汉 21:58 18:18 662
E462 武汉 天津 13:05 04:01 828
L477 武汉 深圳 08:54 23:14 373
E058 武汉 呼和浩特 20:40 10:37 325
O171 武汉 上海 10:44 17:42 620
Y471 武汉 成都 16:35 23:05 713
P198 武汉 南宁 02:58 05:17 411
S146 武汉 柳州 17:08 14:09 639
S214 武汉 昆明 20:29 06:55 506
O466 天津 长春 23:11 12:07 338
L215 天津 上海 17:30 02:39 213
P162 天津 乌鲁木齐 13:12 12:41 755
A070 天津 武汉 13:57 02:42 162
X289 天津 福州 03:54 12:30 344
D491 福州 贵阳 08:27 18:36 392
O072 福州 北京 14:17 19:33 98
F012 福州 西安 01:58 14:33 140
J333 福州 徐州 15:37 05:18 178
Z003 福州 沈阳 09:44 14:32 686
W391 福州 武汉 14:34 22:44 903
W487 福州 呼和浩特 22:11 13:38 214
P369 贵阳 南昌 23:36 04:41 999
Z368 贵阳 乌鲁木齐 07:55 09:14 550
G159 贵阳 武汉 23:33 21:44 155
Q223 贵阳 北京 12:31 19:01 176
A110 贵阳 长春 12:32 07:35 170
C249 贵阳 呼和浩特 12:39 11:04 422
Q330 贵阳 乌鲁木齐 01:18 10:39 353
```

```
5
U109 大连 南昌 02:58 00:17 16
A225 大连 徐州 14:16 22:55 763
D407 大连 深圳 01:19 14:30 477
S063 大连 乌鲁木齐 19:42 00:40 1912
A328 大连 南宁 23:58 14:40 1099
Y401 大连 株洲 05:08 22:33 793
S223 大连 贵阳 17:47 23:28 470
Y129 大连 呼和浩特 19:23 19:25 1401
L091 大连 上海 10:56 16:01 1013
H391 乌鲁木齐 北京 22:30 20:17 920
U069 乌鲁木齐 北京 09:18 11:45 258
T138 乌鲁木齐 天津 17:01 09:58 206
Q217 武汉 乌鲁木齐 05:30 07:35 760
J028 武汉 西宁 05:23 08:50 1605
K250 武汉 成都 06:03 12:55 698
N481 武汉 贵阳 20:13 20:36 1162
U311 武汉 上海 18:27 11:43 1605
E478 武汉 徐州 00:39 11:54 1021
A162 天津 哈尔滨 19:54 23:00 98
D187 天津 北京 09:21 12:58 1352
C486 天津 呼和浩特 13:34 09:00 1254
S476 天津 乌鲁木齐 19:51 05:28 1560
F462 天津 郑州 11:57 16:02 509
N094 天津 柳州 11:07 10:45 806
U293 天津 大连 15:21 13:36 1607
V403 天津 西宁 19:59 04:15 140
C151 天津 兰州 04:16 17:56 1179
O193 天津 徐州 22:45 00:07 1408
J412 天津 西安 22:13 21:12 1403
P121 福州 柳州 21:16 21:18 179
R061 福州 广州 15:14 01:50 354
E084 福州 乌鲁木齐 23:41 18:04 1782
D436 福州 上海 05:54 08:40 836
L222 福州 昆明 08:40 04:38 589
U422 福州 西安 12:08 14:01 743
B469 福州 天津 03:53 09:24 1711
B159 福州 郑州 16:11 18:05 1602
Y038 福州 成都 21:56 23:15 918
R474 福州 沈阳 13:58 05:09 1859
W161 福州 株洲 15:38 19:47 114
H291 贵阳 西安 09:49 23:38 980
C396 贵阳 南宁 09:22 02:10 1642
R000 贵阳 徐州 20:38 20:28 1356
V307 贵阳 南昌 04:14 16:31 1458
L245 贵阳 乌鲁木齐 15:47 03:49 543
P390 贵阳 上海 22:23 20:46 884
```

6.关于退出系统:

```
------欢迎进入交通查询系统-------
|       输入1:查询最优路线       |
|       输入2:新增路线           |
|       输入3:删除路线           |
|       输入4:输出火车时间表     |
|       输入5:输出飞机时间表     |
|       输入其他数字:退出系统    |
 --------------------------------
0
感谢您的使用，再见!

C:\Users\123\source\repos\PJ_1231\De
按任意键关闭此窗口. . .
```

7. 关于主要算法的时空复杂度分析

时间复杂度：使用邻接链表处理数据，使用 Dijkstra 算法，复杂度 O(n 平方)。n 是城市数量。

空间复杂度：用邻接链表 vector 存储了每一条边，复杂度 O（m）。m 是边数。

六、课程设计总结

课程设计让我从头到尾再次学习了一遍数据结构，无论是学过的类定义，vector 的用法，文件输入，优先级队列，还是没学过的 map 容器，以及一直云里雾里的 Dijkstra 算法。其实也尝试过 floyed，还有 prim 算法，但是 bug 更多，就打消了，怕耽误更多时间。但是对于这些算法的思想的理解是更加深入了。

遇到不会的模块，先翻看了清华的教材以及老师课上的 ppt 解决了很多关于图的问题。解决不了的，书上没有的，就在网上搜索相关资料，STL 库的一些用法。

关于程序调试，最大的感悟就是，在写程序的时候就尽力将思绪理清楚理透彻，花时间写出逻辑严密的代码，比草草写完后花更多的时间调 bug 更值得。

关于数据结构课程的认识，从理论向实践迈进。我觉得对于理论的了解更加深入了，并且我可以使用我学习到的理论知识去解决实际问题，这是让我有最大成就感的地方。

做这个课程设计是从十二月中旬靠后开始的，每天去完成一部分，前七天大概构建了框架和类定义，接下来每天写一两个功能函数，期间学习了网上他人关于从文件中读入数据的格式以及文件内容，这个是让我大大加速的地方，文件内容也会附录。最后用了三天左右的时间优化代码以及完成菜单界面的构建。在这里我也感谢我的舍友帮我纠出程序中的一些优化问题和 bug，以及边界不完善的漏洞。

七、参考资料

参考书籍:

1.《数据结构（用面向对象方法与 C++语言描述)》第二版 殷人昆

2.《Database System Concepts》Sixth Edition

参考网站:

1. https://www.runoob.com/w3cnote/cpp-vector-container-analysis.html
2. http://c.biancheng.net/view/215.html
3. https://blog.csdn.net/xulingxin/article/details/81335030
4. https://www.runoob.com/cplusplus/cpp-classes-objects.html
5. https://blog.csdn.net/wkq0825/article/details/82255984
6. https://www.w3cschool.cn/cpp/cpp-fu8l2ppt.html
7. https://www.cnblogs.com/fnlingnzb-learner/p/5833051.html
8. https://blog.csdn.net/sevenjoin/article/details/81943864
9. https://blog.csdn.net/Rice__/article/details/106598336
10. https://blog.csdn.net/qq_39630587/article/details/83240036
11. http://c.biancheng.net/view/7591.html
12. https://www.runoob.com/cplusplus/cpp-files-streams.html
13. https://blog.csdn.net/zhangzhikang_zzk/article/details/50405035?utm_medium=distribute.pc_relevant.none-task-blog-OPENSEARCH-9.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-OPENSEARCH-9.control
14. https://blog.csdn.net/geter_CS/article/details/102580332
15. https://blog.csdn.net/nisxiya/article/details/45725857