**西 安 邮 电 大 学**

**（计算机学院）**

数据结构课程设计报告

# 题 目：全国交通咨询模拟系统

**专业名称： 计算机科学与技术**

**班 级： 计科1801**

**学生姓名： 王琛**

**学号（8位）： 04181017**

**指导教师： 徐舒婷**

**设计起止时间：** 2019年12月30日—2020年1月3日

## 一. 设计目的

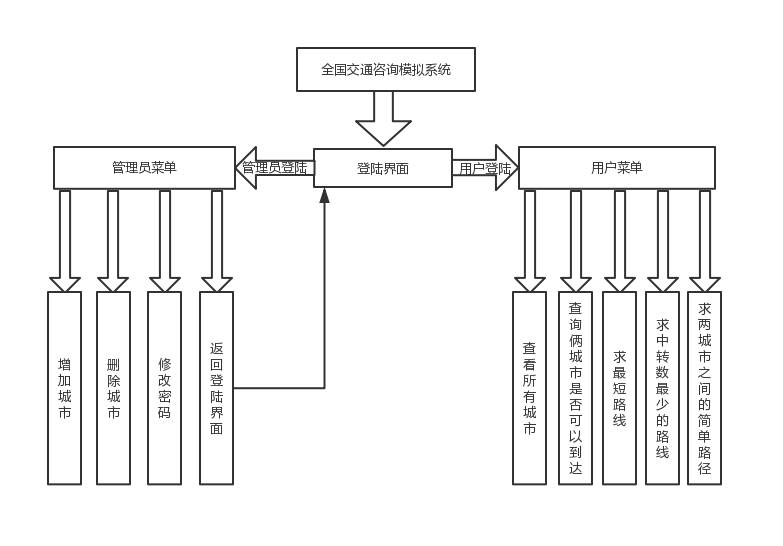
处于不同目的的旅客对交通工具具有不同的要求。例如，因公出差的旅客希望旅途时间尽可能短，出门旅游的游客则期望旅费尽可能省，而老年旅客则需要中转次数最少。编制一个全国城市间的交通咨询程序，为旅客提供两种或三种最优决策的交通咨询。

## 二. 设计内容

提供用户以及管理员功能，用户可以对交通图进行查询，而管理员可以对交通图进行增删查改，同时管理员可以登陆、修改密码等待操作，有一定的交互界面。

## 三．概要设计

### 功能模块图；



### 2．各个模块详细的功能描述。

(1).void login()//登陆

登陆界面，登陆到管理员和用户使用界面的函数，登陆到管理员使用界面时需要输入用户名和密码，与admin.txt文件中的内容进行比对，若比对成功，则进入，比对失败，则报错，重新输入。用户界面不需要进行比对。

1. void admin\_nav()//管理员使用界面

管理员使用的菜单，对管理员需要的功能进行选择。

1. void user\_nav()//用户使用界面

用户使用的菜单，对用户需要的功能进行选择。

1. void addmap()//增加城市的函数

增加城市即节点，以及节点信息描述，并且直接可以手动增加该增加节点与其他文件中已存节点之间的边和权值。

1. void deletemap()//删除城市的函数

删除城市即节点，以及节点描述，和与节点相连的边与权值，可直接都删除掉。

1. void load()//写入文件

将城市名字和描述写入文件，以及城市到各城市的距离也就是权值。

1. void save()//文件保存

每次进行对文件操作完都得进行保存操作。

1. int main()//主函数

程序的主函数，负责调用各函数功能。

1. void all\_map()//城市列表

显示文件中所有城市的名字，以及查询某一个城市的节点信息即城市描述

1. void ask()//查询函数

查询两节点是否可以到达。

1. void Dijkstra()//迪杰斯特拉

迪杰斯特拉求最短路径，求权值最小的路径并输出。

1. void bfs()//广度优先遍历

广度优先遍历算法求两节点之间中转次数最少的路径，并输出该路径。

1. void \_dfs()//深度优先遍历

深度优先遍历求所有简单路径，此处由于路径过多，所以给加一个限制条件，输出其路径。

1. void dfs()

对void \_dfs()中的参数进行初始化。

## 四．详细设计

### 1．功能函数的调用关系图

addmap

load

deletemap

save

全国交通咨询模拟系统

main

save

login

admin\_nav

main

all\_map

user\_nav

ask

Dijkstra

bfs

dfs

### 2．各功能函数的数据流程图

(1)登陆流程图

login

选择登陆方式

返回登陆选择

admin\_nav

user\_nav

(2)添加城市和相关信息

addmap

添加城市名称

添加城市描述

添加与其他城市的距离

添加成功

(3)删除城市和相关信息

deletemap

输入需要删除的城市编号

删除成功

(4)Dijkstra求最短路径

Dijkstra

求出最短路

存入栈中

正序输出

(5)深度优先遍历求所有简单路径

\_dfs

将路径存入数组

加入限制条件

输出符合条件的所有路径

### 重点设计及编码

(1)迪杰斯特拉算法

void Dijkstra(int v0, int s) //迪杰斯特拉求最短路径，并输出路线

{

int min, i, j, u, v;

int p[N], l[N];

for (i = 0; i < N; i++)

{

p[i] = -1;

l[i] = 0;

visit[i] = 0;

}

for (i = 0; i < n; i++)

{

dis[i] = map[v0][i];

if (dis[i] < inf) //v0能直接到达，即上一站点为v0

p[i] = v0;

}

visit[v0] = 1;

for (i = 0; i < n; i++)

{

min = inf;

for (j = 0; j < n; j++) //每次找出距离v0最近点

{

if (visit[j] == 0 && dis[j] < min)

{

min = dis[j];

u = j;

}

}

visit[u] = 1; //标记该点

for (v = 0; v < n; v++)

{

if (visit[v] == 0 && dis[v] > dis[u] + map[u][v]) //通过最近点更新其他边

{

p[v] = u; //存储更新的边，即为路线

dis[v] = dis[u] + map[u][v];

}

}

}

v = s;

i = 1;

while (p[v] != v0) //将路线存入栈中，正序输出

{

l[i++] = p[v];

v = p[v];

}

printf("\n");

u = i - 1;

printf(" \*|路线为：\n");

printf(" %s--->", name[v0]);

for (i = u; i >= 1; i--)

printf("%s--->", name[l[i]]);

printf("%s\n", name[s]);

printf(" 最短路径长度为：%d 公里\n", dis[s]);

}

(2)广度优先遍历算法

void bfs(int v0, int s) //广度优先遍历 最短中转次数

{

int min, i, j, u, v;

int tot = 0;

int p[N], l[N], q[N], front = 0, tail = 0;

int cur;

for (i = 0; i < N; i++)

{

dis[i] = -1;

p[i] = -1;

l[i] = 0;

visit[i] = 0;

}

q[tail++] = v0;

visit[v0] = 1;

while (front != tail)

{

cur = q[front++];

if (cur == s)

break;

for (i = 0; i < n; i++)

{

if (!visit[i] && map[cur][i] && map[cur][i] != inf)

{

dis[i] = dis[cur] + 1;

q[tail++] = i;

visit[i] = 1;

p[i] = cur;

}

}

}

for (i = s; p[i] != -1; i = p[i])

l[tot++] = i;

l[tot++] = v0;

printf(" \*|路线为：\n ");

for (i = tot - 1; i >= 0; i--)

{

if (i != tot - 1)

printf("--->");

printf("%s", name[l[i]]);

}

printf("\n 最短中转数为：%d 个\n", dis[s]);

}

(3)深度优先遍历算法

void \_dfs(int cur, int end, int visit[], int path[], int len, int limit)//深度优先遍历求所有简单路径

{

int i;

if (limit >= 0)

{

visit[cur] = 1;

path[len++] = cur;

if (cur == end)

{

for (i = 0; i < len; i++)

{

if (i)

printf("--->");

printf("%s", name[path[i]]);

}

printf("\n ");

}

else

{

for (i = 0; i < n; i++)

{

if (!visit[i] && map[cur][i] && map[cur][i] != inf)

{

\_dfs(i, end, visit, path, len, limit - 1);

}

}

}

visit[cur] = 0;

}

}

(4)增加节点函数

void addmap()

{

int i;

int num, dis;

printf(" \*| 请输入城市名称：");

scanf("%s", name[n]);

printf(" \*| 请输入城市描述：");

scanf("%s", ms[n]);

all\_map();

printf(" 请依次输入与以上各大城市的距离，格式为：（曾是编号，距离），以-1 -1结束\n");

for (i = 0; i < n; i++) // 全都设置为不可达

map[i][n] = map[n][i] = inf;

map[n][n] = 0;

scanf("%d %d", &num, &dis);

while (num != -1 && dis != -1)

{ // 依次输入景点和距离，设置距离

scanf("%d %d", &num, &dis);

if (num >= 0 && num < n)

map[n][num] = map[num][n] = dis;

}

n++;

printf("\n 添加成功!\n");

}

(5)删除节点函数

void deletemap()

{

int id, i;

all\_map();

printf(" \*| 请输入要删除的城市编号：");

scanf("%d", &id);

for (i = id; i < n; i++)

strcpy(name[i], name[i + 1]);

for (i = 0; i < n; i++)

{ // 将距离信息对应删除

map[id][i] = map[id + 1][i];

map[i][id] = map[i][id + 1];

}

n--;

printf("\n 删除成功！\n");

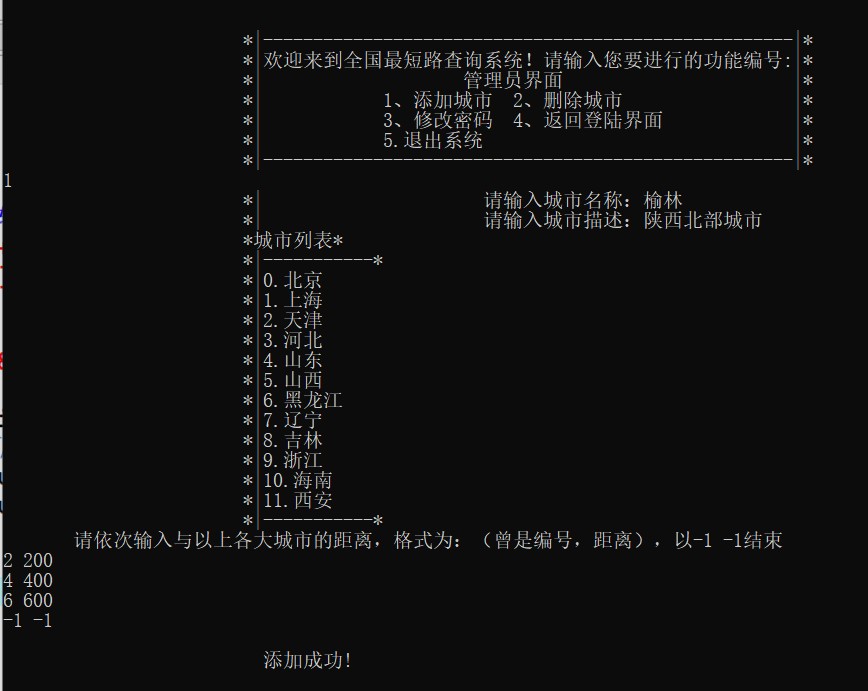
}

## 五．测试数据及运行结果

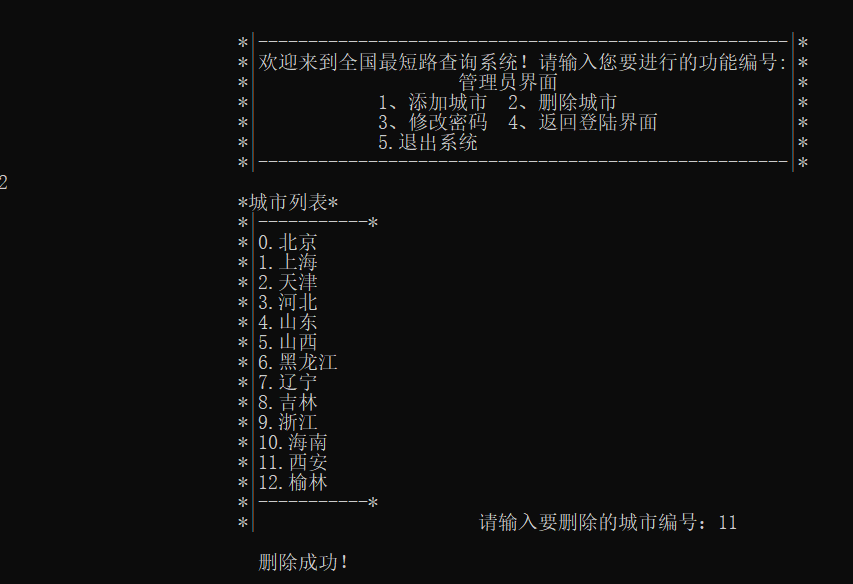
### 1．正常测试数据和运行结果

(1)增加城市：榆林 增加该节点信息：陕西北部城市

删除城市：西安（且删除其节点信息和边）









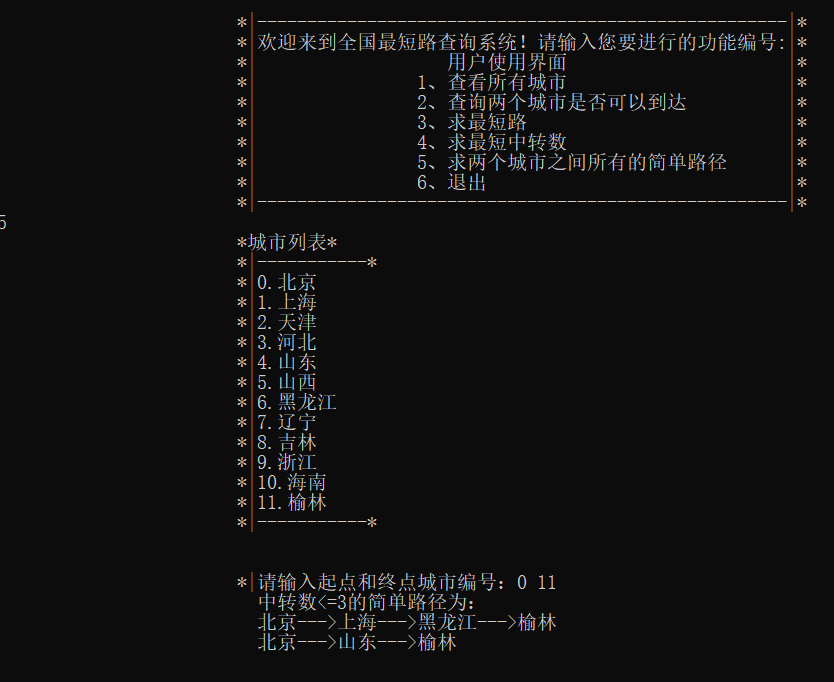
1. 求最短路：北京--榆林



1. 求最小中转数：北京--榆林



1. 求所有简单路径(中转数<=3的)：北京--榆林

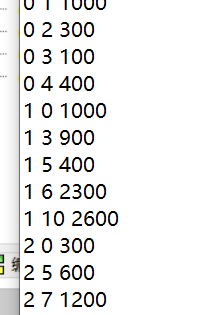


### 2．异常测试数据及运行结果

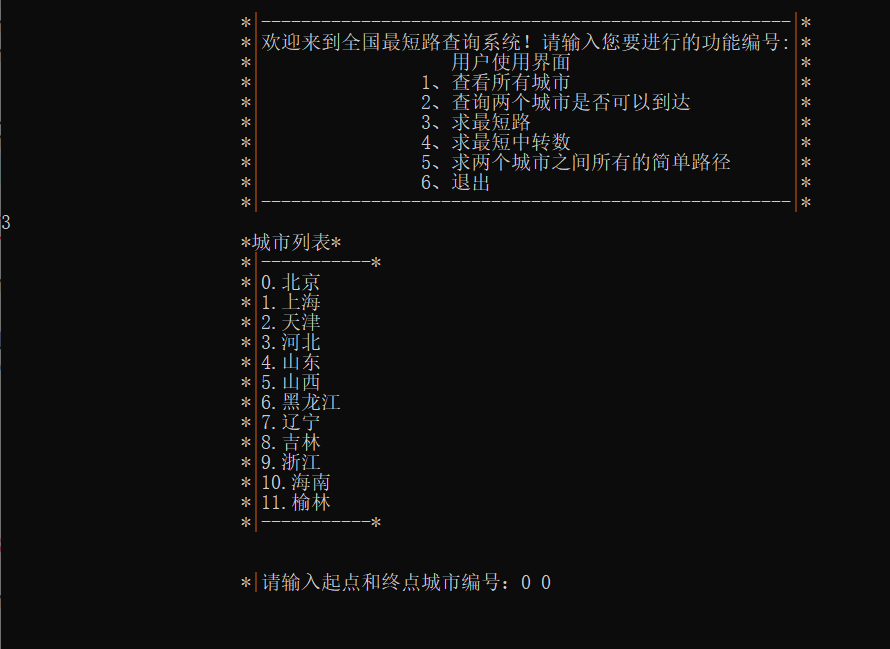
(1)计算两个城市是否可以到达：上海--河北



显示不可到达，但是我存的时候，明显可以看出这两个城市是可以到达的。



(2)输入起点和终点如果是同一个的话，好多功能都不能实现。

****

## 六．调试情况，设计技巧及体会

### 1．改进方案

(1)此次课设，时间紧迫，所以整体设计框架写的不太好，函数都是想到啥添啥，没有规律顺序可言，代码整体比较难，改进方案是对函数顺序摆放整齐，可观性增强。

(2)全局变量应用较多，导致bug频出，调试的时候不好调试，出错不好修改。改进方案是减少全局变量的使用，尽可能不用全局变量。

(3)基本功能齐全，还有一个最小生成树没有写完，写的比较粗糙。改进方案是增强其可视化程度。

### 2．体会

在此次课程设计中，我收获颇丰。

初期进行总体设计时，没时间思考太多框架，。最后才发现框架设计是最为重要的，因为后面的工作只是在框架内完成各个模块，然后在完成各个模块的调用。

除了最为重要的框架设计，模块设计也仅次于框架设计。进行模块设计时，除了要设计好该模块的功能，还应考虑到该模块能否与其他模块组成更大的模块组，或者能否与其他模块完成信息交互。

总结下来，在此次课程设计中，完成了预期的课程设计目的，提高了自己设计程序、完成程序、检测程序的能力。

## 七．参考文献

《数据结构与算法》