

沈阳航空航天大学

# 课 程 设 计 报 告

课程设计名称：数据结构课程设计

课程设计题目：校园导游程序

学 院	计算机学院
专 业	计算机科学与技术
班 级	计算机 1901
学 号	193408020202
姓 名	马丹
指导教师	李照奎

2021 年 01 月

## 目 录

<b>1</b>	<b>题目介绍.....</b>	<b>1</b>
1.1	问题描述.....	1
1.1.1	问题背景.....	1
1.1.2	主要任务.....	1
1.2	问题分析.....	2
<b>2</b>	<b>系统总体设计.....</b>	<b>3</b>
2.1	系统总体功能.....	3
2.2	系统总体流程.....	4
<b>3</b>	<b>数据结构设计.....</b>	<b>5</b>
3.1	存储结构.....	5
3.2	结构体类型.....	5
<b>4</b>	<b>功能模块设计.....</b>	<b>6</b>
4.1	初始化信息模块.....	6
4.2	景点信息查询模块.....	6
4.3	最短路径查询模块.....	7
4.4	所有路径查询模块.....	8
4.5	删除或增加更新模块.....	9
<b>5</b>	<b>系统测试与运行结果.....</b>	<b>11</b>
5.1	调试及调试分析.....	11
5.2	测试用例.....	11
<b>6</b>	<b>总 结.....</b>	<b>15</b>
	参考文献.....	16
	附 录 （程序清单） .....	17

# 1 题目介绍

## 1.1 问题描述

根据题目内容，需要用无向网表示所在学校的景点平面图，即校内每个景点为无向网中的各个结点，景点的编号、名称、简介等信息存储在结点中，图中的边表示景点间的道路，存放景点之间的路径长度等信息。需要回答有关景点介绍、游览路径等问题。具体要求为：

- (1) 能够查询各个景点的相关信息。
- (2) 查询图中任意两个景点间的最短路径。
- (3) 查询图中任意两个景点间的所有路径。
- (4) 增加、删除、更新有关景点和道路的信息。

### 1.1.1 问题背景

轻快是时代需求，当然也更符合游客的心理。出去旅行时，人们往往更希望能将所有的景色都游览一遍，但是景区非常庞大，路径特别多的时候，人们在对景区不熟悉的情况下，凭感觉乱逛，往往参观几个景点以后就精疲力竭了。一个能为来访客人提供帮助的校园导游程序，能快速让他们找到要去地点的最短路径，让他们能轻松地了解两个景点之间的所有路径，以便于来访校园者参观校园。

### 1.1.2 主要任务

首先，用无向网表示所在学校的校园景点图，需要建立一个图的存储结构来为后续存储结点以及边的信息即景点信息和景点路径的信息作准备。这个图的存储结构包括结点的存储结构和边的存储结构。查询各个景点的相关信息，即对图进行一次遍历，从而得到可以查询的景点，达到查询景点的信息的目的，然后将其输出到显示屏上；查询图中任意两个景点间的最短路径，需要对图进行遍历查找两个结点之间的所有路径，同时将得到的边的信息即路径的长度信息和路径的信息分别存储到数组中，当查找所有路径结束后，对记录路径长度的数组进行遍历查找其中的最小值即为要查询的最短路径。

## 1.2 问题分析

查询各个景点的相关信息即对图中的景点信息进行查询，属于使用图这一存储结构的基本操作。查询任意两个景点间的最短路径，这一问题涉及到最短路径的求解算法，需要根据程序的实现选择并学习合适的算法进行对程序的实现。查询图中任意两个景点之间的所有路径，这一问题需要用到图的遍历算法，这样才能把所有的路径都找出来。增加、删除、更新有关景点的信息。这一功能也是对图的存储结构的基本操作的考察。

要解决上述问题，首先需要对校园的景点图信息进行收集，在百度地图中可以搜索到相应的信息，其次先进行对图的存储结构的选择，图的操作都是以两种存储结构为基础的，邻接矩阵存储结构和邻接表存储结构。四种图（有向图，有向网，无向图，无向网）的创建，其他操作都是在图的创建以后开始进行，选择了图的邻接矩阵存储形式，图的邻接矩阵存储结构即图的数组表示法，借助于邻接矩阵容易判定任意两个顶点之间是否有边或弧相连，这样在对路径的查找上比较便利，路径和距离也都由数组形式进行存储。找最短路径算法的选取，选择了弗洛伊德算法来实现，最后遍历图找两点之间所有路径的算法的选取，选择了深度优先遍历算法来实现。

## 2 系统总体设计

### 2.1 系统总体功能

沈航导游咨询系统由六大部分模块组成，其中包括初始化景点图信息模块，以及景点信息查询功能，两个景点之间最短路径查询功能，两景点之间所有路径查询功能，景点之间所有路径查询功能，景点或路径信息的增加或更新或者删除功能共四个功能模块，退出系统模块。初始化景点图信息模块由两个函数组成，分别为初始化景点信息函数、初始化景点图函数。景点信息查询功能由两个函数实现，分别为展示校园景点平面图函数、查询景点信息函数。最短路径查询功能由弗洛伊德算法实现，由两个函数组成，分别为查询并记录最短路径函数和输出最短路径函数。查询两景点之间所有路径功能由一个函数实现，使用深度优先遍历算法结合栈的用法实现查询并输出所有路径。景点或者路径信息的删除或者增加（也可实现对已有景点或者路径信息的更新）由一个函数实现，最后可根据程序菜单提示的内容可选择进行退出程序系统来退出程序，如图 2.1 所示。

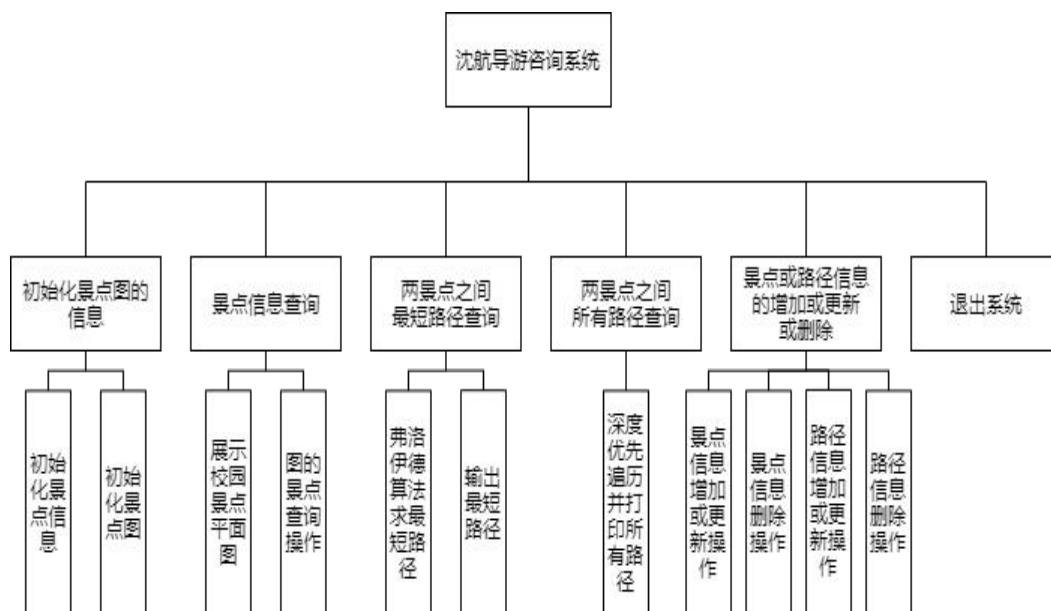


图 2.1 系统总体功能模块

## 2.2 系统总体流程

系统首先初始化景点图的信息，显示主菜单，程序使用者需要按要求输入选项，进行对功能的使用，这一过程由主函数调用其它功能对应的函数进行实现即主模块对其它功能模块进行调用，被调用的功能实现以后，返回到主函数再次显示主菜单进行功能的选择，此处为一个循环过程，使用程序者可以再次选择其他功能使用或者选择结束程序的使用。当选择退出程序，循环结束，程序运行结束。如图 2.2 所示。

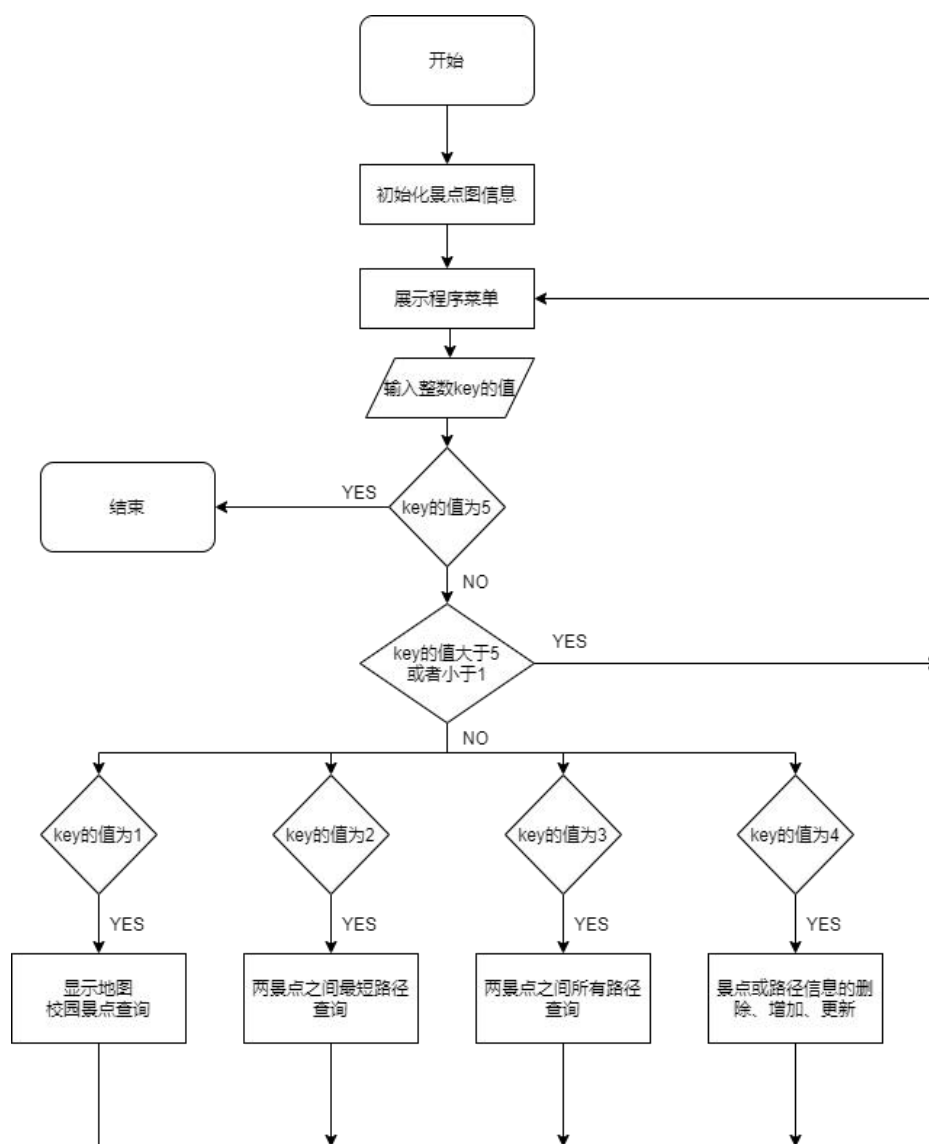


图 2.2 系统总体流程

## 3 数据结构设计

### 3.1 存储结构

总体采用图的顺序存储方式即邻接矩阵的形式对景点图的信息进行存储。采用两个整型二维数组，分别对路径和路径之间的距离进行存储，同时也使用了栈的存储结构，方便后续对最短路径以及所有路径进行求解。

### 3.2 结构体类型

#### 3.2.1 定义景点信息结构体类型。

```
typedef struct VexNode//景点信息结构体
{
    int vertex;//景点编号
    char name[20];//景点名称
    char vertex_message[400];//景点介绍
}VexNode;
```

定义了一个整型成员、二个字符型数组成员，分别为景点编号、景点名称、景点介绍，用来描述景点的具体信息。

#### 3.2.2 定义景点图结构体类型

```
typedef struct maps//景点图的结构体
{
    VexNode v[MAX];
    int vexnum;//点数
    int arcnum;//边数
    int eds[MAX][MAX];//邻接矩阵
}AdjList;
```

定义了一个景点信息类型数组成员、两个整型成员、一个整型二维数组成员，分别为景点、景点数量、路径数量、邻接矩阵，用来描述景点图的具体信息。

## 4 功能模块设计

### 4.1 初始化信息模块

初始化信息模块由两个函数组成，其功能为预置校园景点图的信息，对校园景点图进行初始化，为后续操作提供基础的景点图信息。其运行过程由图 4.1 所示。



图 4.1 初始化信息模块流程

### 4.2 景点信息查询模块

景点信息查询模块由二个函数组成，其功能为展示景点平面图并查询已有的景点信息。该模块运行时，首先输出校园平面景点图，然后输出可查询的景点及其编号，使用者之后根据提示需输入要查询的景点编号，即可查询到想了解的景点信息。模块流程如图 4.2 所示。



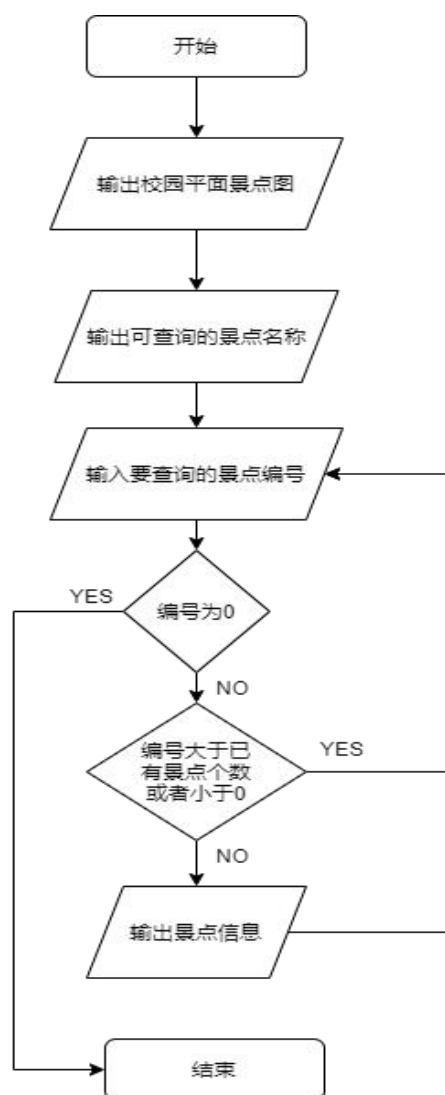


图 4.2 景点信息查询模块流程

### 4.3 最短路径查询模块

最短路径查询模块由二个函数组成，其功能为求两景点之间的最短路径。首先需要输入起点的景点编号，然后输入终点的景点编号，此模块采用弗洛伊德算法求最短路径，最后输出两景点之间的最短路径。如图 4.3 所示。

弗洛伊德（Floyd）算法又称为插点法，是一种利用动态规划的思想寻找给定的加权图中多源点之间最短路径的算法。它的核心思路为：从图的带权邻接矩阵  $A=[a(i, j)]$   $n \times n$  开始，递归地进行  $n$  次更新，即由矩阵  $D(0)=A$ ，按一个公式，

构造出矩阵  $D(1)$ ；又用同样地公式由  $D(1)$  构造出  $D(2)$ ；……；最后又用同样的公式由  $D(n-1)$  构造出矩阵  $D(n)$ 。矩阵  $D(n)$  的  $i$  行  $j$  列元素便是  $i$  号顶点到  $j$  号顶点的最短路径长度，称  $D(n)$  为图的距离矩阵，同时还可引入一个后继节点矩阵  $path$  来记录两点间的最短路径。

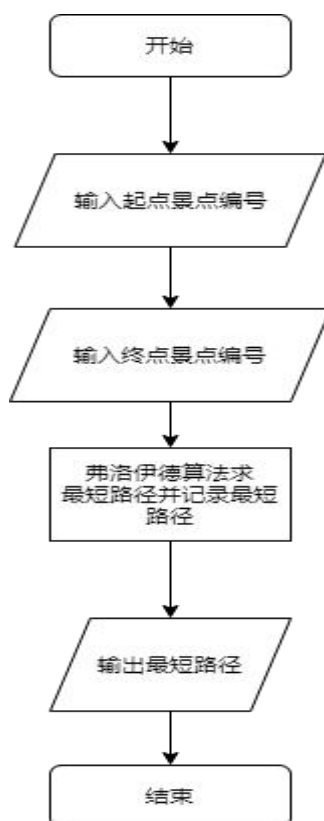


图 4.3 最短路径查询模块流程

## 4.4 所有路径查询模块

所有路径查询模块由一个函数组成，其功能为求两景点之间所有路径。此模块需要输入起点和终点的景点编号，然后通过深度优先遍历算法能够求得所有满足起点终点是使用者所输入的景点的路径并输出。如图 4.4 所示。

深度优先遍历算法简介：深度优先搜索属于图算法的一种，英文缩写为 DFS 即 Depth First Search. 其过程简要来说是对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次。深度优先遍历图的方法是，从图中某顶点  $v$  出发：

- (1) 访问顶点  $v$ ；

(2) 依次从  $v$  的未被访问的邻接点出发，对图进行深度优先遍历；直至图中和  $v$  有路径相通的顶点都被访问；

(3) 若此时图中尚有顶点未被访问，则从一个未被访问的顶点出发，重新进行深度优先遍历，直到图中所有顶点均被访问过为止。当人们刚刚掌握深度优先搜索的时候常常用它来走迷宫。

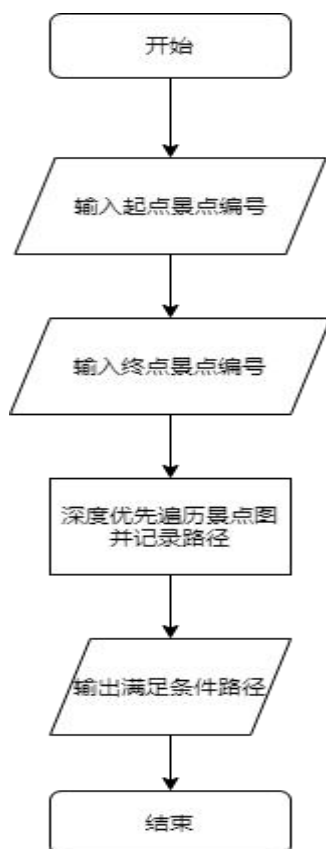


图 4.4 所有路径查询模块流程

## 4.5 删除或增加更新模块

删除或增加更新模块由下面一个函数组成，其功能为删除或增加或更新景点信息和路径信息。使用者根据模块菜单，选择输入对应功能的序号，功能 1 为增加或者更新景点信息，使用者需要输入要增加信息的景点的编号（如果景点为新增的景点编号需要在原来基础上递增）、景点名称、景点介绍。功能 2 为增加或者更新一条路径的信息，以景点编号形式输入，并赋予这条路径长度。功能 3 为删除景点信息，使用者只需要输入景点编号即可，程序中将该景点信息全部置 0

表示该景点已经被删除。功能 4 为删除路径信息，输入景点编号表示对应路径即可，程序中将该路径在邻接矩阵的值赋值为无穷大，表示两景点之间无路径。该模块流程如图 4.5 所示。

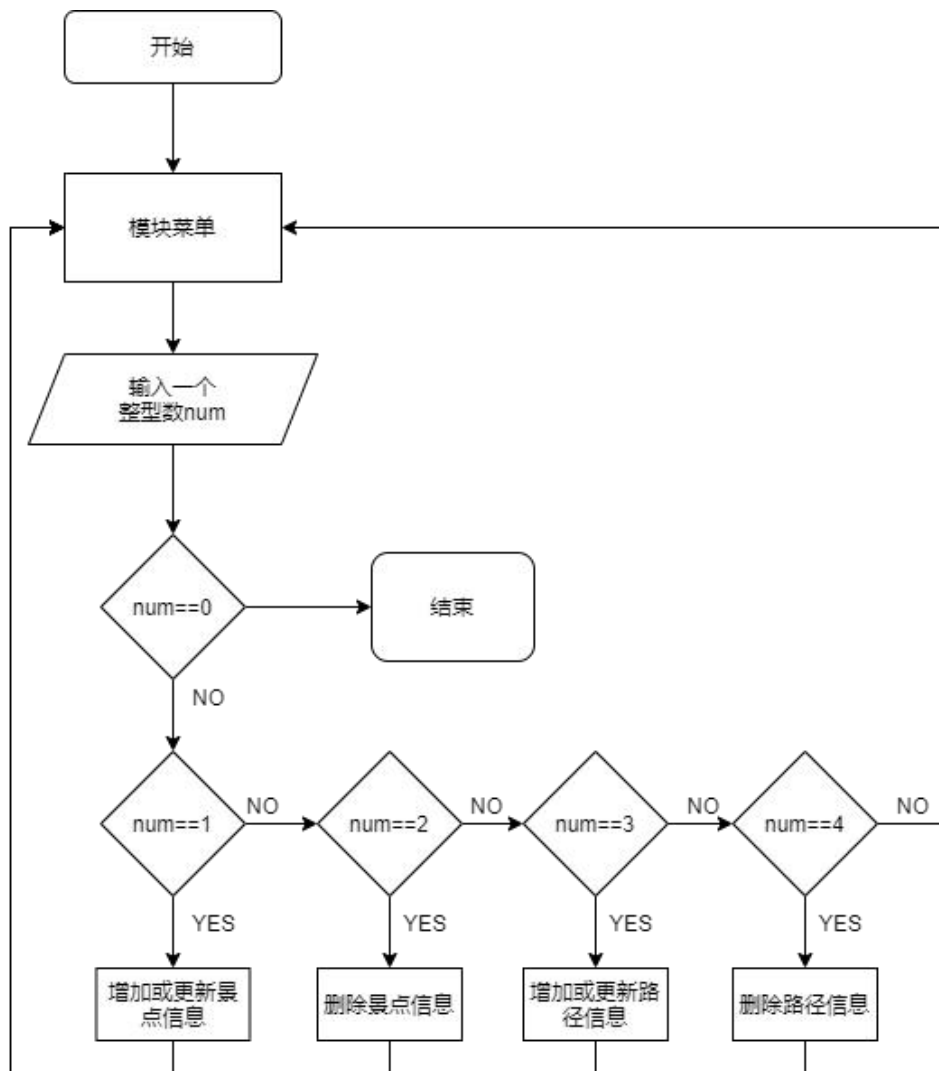


图 4.5 删除或增加更新模块流程

## 5 系统测试与运行结果

### 5.1 调试及调试分析

#### (1) 使用某一功能无结果问题

问题描述：当使用该模块时，并没有输出结果。

解决方法：修改该模块代码，添加部分参数的赋值代码语句。

#### (2) 添加新的景点和路径信息但求最短路径时报错问题

问题描述：当使用者添加一个新的景点，以及与新的景点有关的路径。再次使用求与新景点有关的最短路径功能，程序报错导致无法再运行程序。

解决方法：增加邻接矩阵的初始赋值范围，全部赋值为宏定义的无穷大，满足弗洛伊德算法要求，程序便能正常运行。

### 5.2 测试用例

#### (1) 功能 1 查询景点信息测试用例 1:

```
请输入要查询的景点的编号：
输入0则退出

1
1: 艺术馆
位于体育馆西侧, 设计艺术学院(一到四楼), 计算机学院(五楼)在此办公。
```

图 5.1 测试结果 1

#### (2) 功能 1 查询景点信息测试用例 2:

```
请输入要查询的景点的编号：
输入0则退出

2
2: 体育馆
主要有羽毛球馆、篮球馆、游泳馆、健身房、台球馆、乒乓球馆等。
```

图 5.2 测试结果 2

#### (3) 功能 2 最短路径查询测试用例 1:

```
请输入起点的景点（编号形式）：
1
请输入起点的终点（编号形式）：
4
从艺术馆到图书馆的最短路径是：290米 艺术馆->机械馆->图书馆
```

图 5.3 测试结果 3

(4) 功能 2 最短路径查询测试用例 2:

```
请输入起点的景点（编号形式）：
1
请输入起点的终点（编号形式）：
7
从艺术馆到经管楼的最短路径是：30米 艺术馆->经管楼
```

图 5.4 测试结果 4

(5) 功能 3 所有路径查询测试用例 1:

```
请输入起点的景点（编号形式）：
1
请输入起点的终点（编号形式）：
4
第1条路:艺术馆->体育馆->图书馆
总长度是：440m

第2条路:艺术馆->机械馆->图书馆
总长度是：290m

第3条路:艺术馆->图书馆
总长度是：300m

第4条路:艺术馆->经管楼->体育馆->图书馆
总长度是：850m
```

图 5.5 测试结果 5

(6) 功能 3 所有路径查询测试用例 2:

```
请输入起点的景点（编号形式）：
1
请输入起点的终点（编号形式）：
7
第1条路:艺术馆->体育馆->经管楼
总长度是：420m

第2条路:艺术馆->机械馆->图书馆->体育馆->经管楼
总长度是：1110m

第3条路:艺术馆->图书馆->体育馆->经管楼
总长度是：1120m

第4条路:艺术馆->经管楼
总长度是：30m
```

图 5.6 测试结果 6

(7) 功能 4 增加或更新景点信息测试用例 1:

```
请依次输入要增加信息的景点的编号，景点名称，景点介绍：
注意：如果景点为新增的景点编号需要在原来基础上递增！！！！
11 青阳湖 大鹅锦鲤聚集地
```

图 5.7 预期结果 7

```
{vertex=11 name=0x00a1d120 "青阳湖" vertex_message=0x00a1d134 "大
```

图 5.8 测试结果 7

(8) 功能 4 增加或更新景点信息测试用例 2:

请依次输入要增加信息的景点的编号，景点名称，景点介绍：  
注意：如果景点为新增的景点编号需要在原来基础上递增！！！！  
5 食堂 学生吃饭地点

图 5.9 预期结果 8

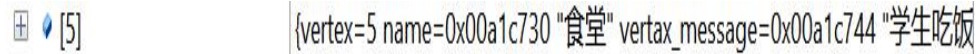
 {vertex=5 name=0x00a1c730 "食堂" vertex\_message=0x00a1c744 "学生吃饭"

图 5.10 测试结果 8

(9) 功能 4 增加或更新路径信息测试用例 1:

请输入增加路径的详细信息及其长度（用景点编号表示）：  
例如： 1 2  
          200  
1 11  
70

图 5.11 预期结果 9

(10) 功能 4 增加或更新路径信息测试用例 2:

请输入增加路径的详细信息及其长度（用景点编号表示）：  
例如： 1 2  
          200  
1 5  
80

图 5.12 预期结果 10

[1]	0x00a1f114
[0]	-858993460
[1]	9999999
[2]	9999999
[3]	210
[4]	9999999
[5]	80
[6]	9999999
[7]	30
[8]	9999999
[9]	9999999
[10]	9999999
[11]	70

图 5.13 测试结果 9、10

(11) 功能 4 删除景点信息测试用例 1:

请输入要删除信息的景点的编号：  
7

图 5.14 预期结果 11

(12) 功能 4 删除景点信息测试用例 2:

请输入要删除信息的景点的编号:  
8

图 5.15 预期结果 12

☐ [7]	{vertex=7 name=0x00a1ca80 "0" vertax_message=0x00a1ca94 "0" }
☐ [8]	{vertex=8 name=0x00a1cc28 "0" vertax_message=0x00a1cc3c "0" }

图 5.16 测试结果 11、12

(13) 功能 4 删除路径信息测试用例 1:

请输入删除的路径（用景点编号表示）：  
1 2

图 5.17 预期结果 13

(14) 功能 4 删除路径信息测试用例 2:

请输入删除的路径（用景点编号表示）：  
1 4

图 5.18 预期结果 14

☐ [1]	0x00a1f114
☐ [0]	-858993460
☐ [1]	9999999
☐ [2]	9999999
☐ [3]	210
☐ [4]	9999999
☐ [5]	80
☐ [6]	9999999
☐ [7]	30
☐ [8]	9999999
☐ [9]	9999999
☐ [10]	9999999
☐ [11]	70

图 5.16 测试结果 13、14



## 6 总 结

本次课程设计让我对图的存储结构以及图的有关算法有了更深的理解，进一步学习了深度优先遍历算法和弗洛伊德算法，弗洛伊德算法在离散数学的学习中也曾接触过，加强了我对离散数学知识的理解，也加深了在数据结构课堂中，对老师所讲授的图的有关知识的理解。加强了我编程时对栈的基本操作：入栈、出栈以及栈的清空等知识的理解，同时也更能了解图的存储结构的不同形式即邻接表和邻接矩阵的区别及其优缺点，同时我也在在调试程序的过程中发现了自己的一些错误，在程序菜单的布置中，分模块的过程中都出现了一些比较粗心的错误，经过调试对程序进行了更正，使得程序能够正常运行起来，最终实现了本次课程设计所要求的内容。本次数据结构课程设计使我受益匪浅，虽然现在编程能力不够强大以及掌握的算法个数还不是很多，但是我一定继续努力学习，加强自己的编程能力，同时也感谢老师对这次课设的指导。

## 参考文献

- [1] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 2012
- [2] (美)萨尼著, 王立柱, 刘志红等译. 数据结构、算法与应用: C++语言描述. 北京: 机械工业出版社, 2015
- [3] 钦铭. C 语言程序设计. 北京: 高等教育出版社, 2015
- [4] 百度百科. 深度优先搜索. [DB/OL]. <https://baike.baidu.com/item/%E6%B7%B1%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2/5224976?fr=aladdin>
- [5] 百度地图. [DB/OL]. <https://map.baidu.com/@13741313.13,5104005.77,12z>

## 附 录

### (程序清单)

```
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
using namespace std;
#include<string.h>
#include<algorithm>
#include<stack>
#define INF 9999999
#define MAX 30
int dist[MAX][MAX];///距离
int path[MAX][MAX];///路径
int Stack[MAX];///路径栈
int top;///栈顶
int counts;///记录路径数
typedef struct VexNode//景点信息结构体
{
    int vertex;///景点编号
    char name[20];///景点名称
    char vertax_message[400];///景点介绍
}VerNode;
typedef struct maps//景点图的结构体
{
    VerNode v[MAX];
    int vexnum;///点数
    int arcnum;///边数
    int edgs[MAX][MAX];///邻接矩阵
}AdjList;
int visited[MAX];
void create_vertex(AdjList &GL);///初始化校园景点及景点信息函数
void Init_Creat_maps(AdjList & GL);///初始化校园景点图函数
void Dis_menu();///展示程序菜单
void Dis_map();///展示校园景点图
void add_message(AdjList &GL);///更新或添加或删除校园景点及路径信息函数
void find_message(AdjList GL);///查询景点信息函数
void Floyd(AdjList GL);///弗洛伊德求最短路径
void Floyd_print(int s, int e,AdjList GL);///打印最短路径
void Dfs_allpaths(int s,int e,AdjList GL);///查询两景点之间所有路径函数
```

```

int main()
{
    AdjList graph;
    int key;
    int start, end;
    create_vertex(graph);
    Init_Creat_maps(graph);
    while(1)
    {
        Dis_menu();
        cin>>key;
        if(key==1)
        {
            Dis_map();
            find_message(graph);
        }
        if(key==2)
        {
            printf("请输入起点的景点（编号形式）：\n");
            scanf("%d",&start);
            printf("请输入起点的终点（编号形式）：\n");
            scanf("%d",&end);
            Floyd(graph);
            printf("从%s到%s的最短路径是： %d 米", graph.v[start].name, graph.v[end].name, dist[start][end]);
            printf("%s->", graph.v[start].name);
            Floyd_print(start, end, graph);
            printf("%s\n", graph.v[end].name);
        }
        if(key==3)
        {
            printf("请输入起点的景点（编号形式）：\n");
            scanf("%d",&start);
            printf("请输入起点的终点（编号形式）：\n");
            scanf("%d",&end);
            counts=0;
            top=0;
            Dfs_allpaths(start, end, graph);
        }
        if(key==4)
        {
            add_message(graph);
        }
        if(key==5)
    }
}

```

```

        {
            printf("使用结束！");
            break;
        }

        if(key>5||key<1)
        {
            printf("输入有误！请重新输入！");
            continue;
        }
    }

    return 0;
}

void create_vertex(AdjList &GL)
{
    //预置初始时景点图的景点信息
    GL.v[1].vertex=1;
    strcpy(GL.v[1].name,"艺术馆");
    strcpy(GL.v[1].vertax_message,"位于体育馆西侧,设计艺术学院(一到四楼),计算机学院(五楼)在此办公。");
    GL.v[2].vertex=2;
    strcpy(GL.v[2].name,"体育馆");
    strcpy(GL.v[2].vertax_message,"主要有羽毛球馆、篮球馆、游泳馆、健身房、台球馆、乒乓球馆等。");
    GL.v[3].vertex=3;
    strcpy(GL.v[3].name,"机械馆");
    strcpy(GL.v[3].vertax_message,"位于图书馆西侧,机电工程学院,自动化学院,材料科学与工程学院在此办公。");
    GL.v[4].vertex=4;
    strcpy(GL.v[4].name,"图书馆");
    strcpy(GL.v[4].vertax_message,"位于学校展翼大道音乐喷泉后面,是书刊查阅、借阅、资料索引、下载、读书学习的集中地。");
    GL.v[5].vertex=5;
    strcpy(GL.v[5].name,"大学生活动中心");
    strcpy(GL.v[5].vertax_message,"学生处、招生就业处、校团委、校史馆、校学生会联合会、校社团联合会、校大学生艺术团、网络思政中心、爱心联合会在此办公。");
    GL.v[6].vertex=6;
    strcpy(GL.v[6].name,"行政楼");
    strcpy(GL.v[6].vertax_message,"位于学校正门,是学校部分机关部门、直属附属单位办公场所,涉及到学生事务的部门有教务处、计财处等。");
    GL.v[7].vertex=7;
    strcpy(GL.v[7].name,"经管楼");
    strcpy(GL.v[7].vertax_message,"位于行政楼北侧。");
    GL.v[8].vertex=8;

```

```

strcpy(GL.v[8].name,"逸夫馆");
strcpy(GL.v[8].vertax_message,"位于学校西北角。");
GL.v[9].vertex=9;
strcpy(GL.v[9].name,"航空宇航馆");
strcpy(GL.v[9].vertax_message,"三航办公。");
GL.v[10].vertex=10;
strcpy(GL.v[10].name,"蓝天剧场");
strcpy(GL.v[10].vertax_message,"学校文艺活动举办场所。");
}
void Init_Creat_maps(AdjList & GL)
{
    //初始景点图
    int i,j;
    GL.vexnum=10;///10 个景点
    GL.arcnum=8;///8 条双向路径
    for(i=1; i<=MAX; i++) ///初始化邻接矩阵
    {
        for(j=1; j<=MAX; j++)
        {
            GL.edgs[i][j]=INF;
        }
    }
    GL.edgs[1][3]=GL.edgs[3][1]=210;
    GL.edgs[1][4]=GL.edgs[4][1]=300;
    GL.edgs[2][7]=GL.edgs[7][2]=400;
    GL.edgs[2][4]=GL.edgs[4][2]=420;
    GL.edgs[3][4]=GL.edgs[4][3]=80;
    GL.edgs[4][5]=GL.edgs[5][4]=780;
    GL.edgs[1][7]=GL.edgs[7][1]=30;
    GL.edgs[1][2]=GL.edgs[2][1]=20;
}
void Dis_menu()
{
    printf("\n");
    printf("          *****欢迎使用沈航导游咨询系统*****\n\n");
    printf("          ***** 1. 沈航景点信息查询 *****\n");
    printf("          ***** 2. 两景点之间最短路查询 *****\n");
    printf("          ***** 3. 两景点间所有路径查询 *****\n");
    printf("          ***** 4. 景点和路径信息的删除或增加(更新) *****\n");
    printf("          ***** 5. 退出系统 *****\n");
    printf("          *****\n");
    return ;
}

```

```

}
void Dis_map()
{
    printf("\n                                *沈航校园景点地图一览*\n\n");
    printf("
12 北区      \n");
    printf("                                ①艺术馆      ②体育馆
◎      \n");
    printf("                                ◎      ◎-----|
\n");
    printf("                                |-----|      |
\n");
    printf("                                |      |      |
\n");
    printf("                                ③机械馆      ④图书馆  ⑩蓝天剧场|
\n");
    printf("                                ◎      ◎      ◎
◎ ⑤大学生活动中心      \n");
    printf("                                ⑧逸夫楼◎-----|-----
|-----|-----|      \n");
    printf("                                |      |      ◎
-----|      \n");
    printf("                                |      |      11 青阳湖
|      \n");
    printf("                                |      ◎⑨航空宇航馆
|      \n");
    printf("                                ◎⑦经管楼      |
|      \n");
    printf("                                |-----|
|      \n");
    printf("                                |      |----- 13
南区      \n");
    printf("                                |
\n");
    printf("                                ◎⑥行政楼
\n\n");
}

void add_message(AdjList &GL)
{
    int num, p, q, num1, num2, p1, q1;
    VerNode *t;
    t=GL.v;
    while(1)

```

```

    {
        num=-1;
printf("*****\n");
printf("输入 '1' 增加（更新）景点信息，输入 '2' 增加（更新）路径信息，
输入 '3' 删除景点信息，输入 '4' 删除路径信息：\n");
printf("输入 '0' 则退出\n");
printf("*****\n");
        cin>>num;
        if(num==0)
        {
            break;
        }
        if(num==1)
        {
            printf("请依次输入要增加信息的景点的编号，景点名称，景点介绍：\n");
            printf("注意：如果景点为新增的景点编号需要在原来基础上递增！！！！\n");
            scanf("%d",&num1);
            if(num1>GL.vexnum)
            {
                GL.vexnum+=1;
            }
            GL.v[num1].vertex=num1;
            cin>>GL.v[num1].name;
            cin>>GL.v[num1].vertax_message;
            continue;
        }
        if(num==2)
        {
            GL.arcnum+=1;
            printf("请输入增加路径的详细信息及其长度（用景点编号表示）：\n");
            cout<<"例如： 1 2 "<<endl;
            cout<<"      200      "<<endl;
            scanf("%d %d",&p,&q);
            cin>>GL.edgs[p][q]; //输入长度
            GL.edgs[q][p]=GL.edgs[p][q]; //路径为双向路径
            continue;
        }
        if(num==3)
        {
            printf("请输入要删除信息的景点的编号：\n");
            scanf("%d",&num2);

```



```

        strcpy(GL.v[num2].name, "0");//置 0 代表删除
        strcpy(GL.v[num2].vertax_message, "0");//置 0 代表删除
        continue;
    }
    if(num==4)
    {
        printf("请输入删除的路径（用景点编号表示）： \n");
        scanf("%d %d", &p1, &q1);
        GL.edgs[p1][q1]=INF;
        GL.edgs[q1][p1]=INF;
        continue;
    }
}

void find_message(AdjList GL)
{
    int number;
    printf("景点图可查询的景点有： \n");
    for(int i=1; i<=GL.vexnum; i++)
    {
        printf("%d:%s\n", GL.v[i].vertex, GL.v[i].name);
    }
    while(1)
    {
        printf("请输入要查询的景点的编号： \n");
        printf("输入 0 则退出\n\n");
        cin>>number;
        if(number==0)
        {
            break;
        }
        else if(number>GL.vexnum || number<0)
        {
            printf("输入有误，请重新输入： \n");
            continue;
        }
        else
        {
            printf("%d: %s\n", GL.v[number].vertex, GL.v[number].name);
            printf("%s\n", GL.v[number].vertax_message);
        }
    }
}

```

```

void Floyd(AdjList GL) //弗洛伊德
{
    int i, j, k;
    for(i=1; i<=GL.vexnum; i++) //初始化距离与路径矩阵
    {
        for(j=1; j<=GL.vexnum; j++)
        {
            dist[i][j]=GL.edgs[i][j];
            if(i!=j&&dist[i][j]<INF)
            {
                path[i][j]=i;
            }
            else
            {
                path[i][j]=-1;//-1 代表不可达
            }
        }
    }
    //printf("%d\n", GL.);
    for(k=1; k<=GL.vexnum; k++)
    {
        for(i=1; i<=GL.vexnum; i++)
        {
            for(j=1; j<=GL.vexnum; j++)
            {
                if(dist[i][j]>(dist[i][k]+dist[k][j]))
                {
                    dist[i][j]=dist[i][k]+dist[k][j]; //更新
                    path[i][j]=k; //path 用于记录最短路径上的结点
                }
            }
        }
    }
    return ;
}

void Floyd_print(int s, int e, AdjList GL)
{
    if(path[s][e]==-1 || path[s][e]==e || path[s][e]==s) //递归终止条件
    {
        return;
    }
    else
    {

```

```

        Floyd_print(s, path[s][e], GL); //将中间点作为终点继续打印路径
        printf("%s->", GL.v[path[s][e]].name); //打印中间景点名字
        Floyd_print(path[s][e], e, GL); //将中间点作为起点继续打印路径
    }
}

void Dfs_allpaths(int s, int e, AdjList GL)
{
    int dis=0;
    int i, j;
    Stack[top]=s;
    top++; //起点入栈
    visited[s]=1; //标记入栈
    for(i=1; i<=GL.vexnum; i++)
    {
        if(GL.edgs[s][i]>0 && GL.edgs[s][i]!=INF && !visited[i])
        {
            //表明两点可达且未被访问
            if(i==e) //DFS 到了终点，打印路径
            {
                printf("第%d 条路:", ++counts);
                for(j=0; j<top; j++)
                {
                    printf("%s->", GL.v[Stack[j]].name);
                    if(j<top-1) //统计路径长度
                    {
                        dis=dis+GL.edgs[Stack[j]][Stack[j+1]];
                    }
                }
                dis=dis+GL.edgs[Stack[top-1]][e];
                printf("%s\n", GL.v[e].name); //打印终点
                printf("总长度是: %dm\n\n", dis);
            }
            else //不是终点接着 DFS
            {
                Dfs_allpaths(i, e, GL);
                top--; //支路全被访问一遍，顶点出栈
                visited[i]=0; //出栈点标记为已出栈，允许下次访问
            }
        }
    }
}

```