数据结构

课程设计报告

题 目： 齐齐哈尔大学校园导游咨询系统

学 院： 计算机与控制工程学院

专业班级： 软件191

学生姓名： 刘士华

指导教师： 王一萍

2021 年12月8日

数据结构课程设计评分表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 班级 | 软件191 | | 姓名 | 刘士华 | 指导教师 | 王一萍 | |
| 题目：齐齐哈尔大学校园导游咨询系统 | | | | | | | |
| 评分标准 | | | | | | | |
| 评分标准 | | 分数权重 | 评分的依据 | | | | 得分 |
| A | | C | |
| 选题 | | 10 | 选题符合大纲要求，题目较新颖，工作量大 | | 选题基本符合大纲要求，工作量适中 | |  |
| 工作态度 | | 10 | 态度端正，能主动认真完成各个环节的工作，不迟到早退，出勤好。 | | 能够完成各环节基本工作，出勤较好。 | |  |
| 存储结构、算法描述 | | 20 | 能正确选择存储结构，定义准确，算法流程图或编程语言描述的算法准确无误 | | 能正确选择存储结构，算法流程图或编程语言描述的算法基本准确 | |  |
| 独立解决问题的能力 | | 10 | 具有独立分析、解决问题能力，有一定的创造性，能够独立完成软件的设计与调试工作，程序结构清晰，逻辑严谨，功能完善。 | | 有一定的分析、解决问题能力。能够在老师指导下完成软件的设计与调试工作，程序功能较完善。 | |  |
| 答辩问题回答 | | 20 | 能准确回答老师提出的问题 | | 能基本准确回答老师提出的问题 | |  |
| 程序运行情况 | | 10 | 程序运行正确、界面清晰，测试数据设计合理。 | | 程序运行正确、界面较清晰，能给出合适的测试数据。 | |  |
| 课程设计报告 | | 20 | 格式规范，层次清晰，设计思想明确，解决问题方法合理，体会深刻。 | | 格式较规范，设计思想基本明确，解决问题方法较合理。 | |  |
| 总分 | | | | | | |  |
| 指导教师（签字）： | | | | | | | |

注：介于A和C之间为B级，低于C为D级和E级。按各项指标打分后，总分在90～100为优，80～89为良，70～79为中，60～69为及格，60分以下不及格。

# 1 问题描述

随着学校的发展，来到齐大的新同学越来越多 ，但是对于第一次来到齐大的学生，不熟悉学校，上课或者去某个地方可能会绕远路，该系统可以帮助新同学规划一条最合适的路径，此外学校的导游咨询系统可也以很快地帮助他们熟悉学校、了解学校。该系统也是实现该目的而设计。

本设计使用邻接矩阵法构造无向图，针对用户的实际需求，将学校的建筑代号、建筑名称、建筑简介等信息放入到图的顶点当中，将各建筑间的距离信息存放在弧当中。利用弗洛伊德算法求出两个建筑之间的最短路径；用相应的函数来查找建筑，并显示出它的信息，简介。在界面显示适量的提示信息，保证用户能够较为容易的上手该系统。

本系统将齐齐哈尔大学中校区主要建筑模拟成一个无向图，用户只需观察模拟建筑图根据要求输入两个建筑的编号即可根据用户选择来显示距离最短的路线，并可以根据现实情况发布公告或者提醒某建筑已关闭及关闭理由，但不能对用户输入的信息进行存储。当用户操作完毕需要退出时，需提供退出选项，便于用户退出系统。

# 2 数据结构描述

## 2.1 数据结构和存储结构

### 2.1.1 数据结构

图是由顶点的集合和边的集合组成的，建筑可以看做图的顶点，建筑之间的距离可以看做是图的边，这样就将地图转化成图，然后可以通过解决图的问题解决实际问题。

图是计算机中比较重要的一种数据结构，在很多方面应用比较广泛比如：计算机网络中的路由算法，地图上搜索最短路径，搜索引擎等方面。图是比树更一般，更复杂的非线性数据结构。图结构中结点之间的关系是任意的，每个元素都可以和其他任何元素相关，即元素之间是多对多的关系。

本系统中，为了描述各个建筑以及其之间的路径关系，采用图的数据结构，结点代表建筑，结点间的边代表建筑间的路径，边的权值代表路径的长度。

typedef struct { /\*校园建筑图结构体\*/

vertextype vexs[M]; /\*顶点信息域\*/

edgtype edge[M][M]; /\*邻接矩阵\*/

int vexNum, edgNum; /\*顶点数和边数\*/

}mgraphtype;

### 2.1.2 邻接矩阵存储

本系统采用图的邻接矩阵存储结构。图的邻接矩阵表示法也成为数组表示法。它采用两个数组来表示图：一个是用于存储定点信息的一维数组，另一个是用于存储图中顶点之间关联关系的二维数组，这个关联关系数组被称为邻接矩阵。采用邻接矩阵存储法表示图，很便于实现图的一些基本操作，比如查询，判断，插入，删除等操作。本系统会经常进行这些操作，所以本系统采用图的邻接矩阵存储结构。

typedef struct {

int num; /\*建筑代号\*/

char name[20]; /\*建筑名称\*/

char intro[200]; /\*建筑简介\*/

}vertextype;

typedef int edgtype; /\*权值类型\*/

typedef struct { /\*校园建筑图结构体\*/

vertextype vexs[M]; /\*顶点信息域\*/

edgtype edge[M][M]; /\*邻接矩阵\*/

int vexNum, edgNum; /\*顶点数和边数\*/

}mgraphtype;

## 2.2 ADT

（1）弗洛伊德算法

弗洛伊德算法是通过一个图的权值[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5)求出它的每两点间的[最短路径](https://baike.baidu.com/item/%E6%9C%80%E7%9F%AD%E8%B7%AF%E5%BE%84)矩阵。弗洛伊德算法(Floyd)计算图中各个顶点之间的最短路径，迪杰斯特拉算法通过选定的被访问项点，求出从出发访问顶点到其他项点的最短路径;弗洛伊德算法中每个顶点都是出发访问点，所以需要将每一个顶点看做被访问顶点，求出从每一个顶点到其他项点的最短路径。

从图的带权[邻接矩阵](https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)A=[a(i,j)] n×n开始，[递归](https://baike.baidu.com/item/%E9%80%92%E5%BD%92" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)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)；又用由D(1)构造出D(2)；……；最后构造出矩阵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](https://baike.baidu.com/item/%E8%B7%9D%E7%A6%BB%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)，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。（2）

（2）ADT

ADT，即抽象数据类型，是一个实现包括储存数据元素的存储结构以及实现基本操作的算法。在这个数据抽象的思想中，数据类型的定义和它的实现是分开的。以下为本系统的ADT。

ADT system{

数据对象{ vexNum，edgNum，vexs[M]，edge[M][M] }

基本操作{

int menu(); /\*主菜单\*/

void Create\_Map(mgraphtype \*g); /\*从文件读取信息建立图\*/

void Print\_Map(); /\*显示校园建筑地图\*/

int Judge\_Input(int num); /\*判断输入的编号是否合理\*/

void Search\_Location(mgraphtype \*g);/\*建筑信息查询\*/

void ShortPath(mgraphtype \*g); /\*求建筑间最短路径\*/

void Floyd\_Print(mgraphtype \*g,int sNum,int eNum); /\*递归打印两点间最短路径\*/

void Shortpath\_Print(mgraphtype \*g);/\*输出并打印两点间的最短路径\*/

void Dfs\_Print(mgraphtype \*g,int sNum,int eNum);/\*深度优先遍历查询两建筑间所有路径\*/

void Allpath\_Print(mgraphtype \*g); /\*查询两顶点间的所有路径并打印\*/

void BestPath(mgraphtype \*g); /\*多顶点间求最佳路径\*/

void System\_Exit(int \*q); /\*退出系统\*/

} ADT system

# 3 算法描述

## 3.1 系统功能

面对一个复杂而又庞大的系统，我们无法通过一张图表就把系统所有元素之间的关系描述清楚，这时就要将系统按一定的原则分解成若干个子系统。将本系统的功能进行分解，本系统一共包括六个部分：显示校园建筑地图，建筑信息查询，建筑间路径查询，多建筑间访问路线查询，公告发布与查询，建筑状态管理。

功能结构图就是按照功能的从属关系画成的图表，图中的每一个框都称为一个功能模块。功能模块可以根据具体情况分的大一点或小一点，分解得最小功能模块可以是一个程序中的每个处理过程，而较大的功能模块则可能是完成某一个任务的一组程序。

功能结构图是对硬件、软件、解决方案等进行解剖，详细描述功能列表的结构，构成，剖面的从大到小，从粗到细，从上到下等而描绘或画出来的结构图。从概念上讲，上层功能包括 (或控制)下层功能，愈上层功能愈笼统，愈下层功能愈具体。

为了使用户更加清楚本系统的功能有几个部分组成、每部分是什么、以及各有什么作用，本系统的功能结构图如图3-1所示。



图3-1功能结构图

流程图是使用图形表示算法的思路是一种极好的方法，因为千言万语不如一张图。流程图是流经一个系统的信息流、观点流或部件流的图形代表。

作为诊断工具，它能够辅助决策制定，让管理者清楚地知道，问题可能出在什么地方，从而确定出可供选择的行动方案。流程图有时也称作输入-输出图。该图直观地描述一个工作过程的具体步骤。

流程图对准确了解事情是如何进行的，以及决定应如何改进过程极有帮助。这一方法可以用于整个企业，以便直观地跟踪和图解企业的运作方式。流程图使用一些标准符号代表某些类型的动作，如决策用菱形框表示，具体活动用方框表示。但比这些符号规定更重要的，是必须清楚地描述工作过程的顺序。

流程图也可用于设计改进工作过程，具体做法是先画出事情应该怎么做，再将其与实际情况进行比较。流程图优点是形象直观，各种操作一目了然，不会产生“歧义性”，便于理解，算法出错时容易发现，并可以直接转化为程序。因此，不论作业研究过程中采用何种技术，流程图总是必不可少的一步。

本系统的操作流程图如图3-2所示。



图3-2 操作流程图

以下将抽出3个主要功能模块进行详细说明。

## 3.2查询最短路径模块

（1）功能

此模块主要完成查询并输出从选定建筑到另一建筑的最短距离，并通过递归算法输出其路径，此功能主要运用弗洛伊德算法。

（2）算法描述

算法步骤如下：

int dist[M][M]; /\*距离向量\*/

int path[M][M]; /\*路径向量\*/

1. 首先把初始化距离dist数组为图的邻接矩阵，路径数组path初始化为-1。其中对于邻接矩阵中的数首先初始化为正无穷，如果两个顶点存在边则初始化为权重。

dist[i][j] = g->edge[i][j];

if (i != j && dist[i][j] < INFINITY) path[i][j] = i;

else path[i][j] = -1; /\*-1代表当前两点不可达\*/

2）对于每一对顶点 u 和 v，看看是否存在一个顶点 w 使得从 u 到 w 再到 v 比己知的路径更短。如果是就更新它。  
状态转移方程为：若dist[i][j] >(dist[i][k] + dist[k][j]) 则dist[i][j] = dist[i][k] + dist[k][j]

（3）具体实现

/\*Floyd算法求两建筑间的一条最短的路径\*/

int dist[M][M]; /\*距离向量\*/

int path[M][M]; /\*路径向量\*/

void ShortPath(mgraphtype \*g) {

int i, j, k;

for (i = 0; i < g->vexNum; i++) /\*初始化距离向量矩阵与路径向量矩阵\*/

for (j = 0; j < g->vexNum; j++) {

dist[i][j] = g->edge[i][j];

if (i != j && dist[i][j] < INFINITY) path[i][j] = i;

else path[i][j] = -1; /\*-1代表当前两点不可达\*/

}

for (k = 0; k < g->vexNum; k++) /\*递推求解每两建筑的最短路径\*/

for (i = 0; i < g->vexNum; i++)

for (j = 0; j < g->vexNum; j++) /\*更新dist[i][j]的值\*/

if (dist[i][j] >(dist[i][k] + dist[k][j])) {

dist[i][j] = dist[i][k] + dist[k][j];

path[i][j] = k; /\*path用于记录最短路径上的经结点\*/

}

}

/\*递归实现打印两点间的最短路径\*/

void Floyd\_Print(mgraphtype \*g, int sNum, int eNum) {

if (path[sNum][eNum] == -1 || path[sNum][eNum] == eNum || path[sNum][eNum] == sNum)

return;

else {

Floyd\_Print(g, sNum, path[sNum][eNum]); /\*将中间点作为终点继续打印路径\*/

printf("%s->", g->vexs[path[sNum][eNum]].name); /\*打印中间建筑名字\*/

Floyd\_Print(g, path[sNum][eNum], eNum); /\*将中间点作为起点继续打印路径\*/

}

}

/\*输出并打印两点间的最短路径\*/

void Shortpath\_Print(mgraphtype \*g) {

int sNum, eNum; /\*起点编号，终点编号\*/

do {

printf("\n请输入起点编号：");

scanf("%d", &sNum);

} while (Judge\_Input(sNum));

do {

printf("\n请输入终点编号：");

scanf("%d", &eNum);

} while (Judge\_Input(eNum));

printf("\n%s到%s的最短距离是：%dm\n", g->vexs[--sNum].name, g->vexs[--eNum].name, dist[sNum][eNum]);

printf("\n这是最佳游览路线：");

printf("%s->", g->vexs[sNum].name); /\*输出路径上的起点\*/

Floyd\_Print(g, sNum, eNum); /\*输出路径上的中间点\*/

printf("%s\n\n", g->vexs[eNum].name); /\*输出路径上的终点\*/

}

## 3.3多建筑间求最佳路径

（1）功能

此模块主要是在包含多个建筑的情况下求最短路径，此模块利用弗洛伊德算法逐个求出两建筑间的最短距离，并输出用户选择的路径及路径长度。

（2）算法描述

算法步骤如下：

int vNum[M] = { 0 }, j = 1; /\*记录用户输入的编号信息\*/

int d = 0; /\*统计全程总长\*/

1. 通过while循环输入多个建筑，直到输入-1时结束，并调用Judge\_Input判断输入的建筑是否存在。Judge\_Input(vNum[j - 1]) if (vNum[j - 1] == -1) break;
2. 首先输入路径上的起点，然后通过弗洛伊德算法分别求得建筑间的最短距离，将距离加到d中继续下次循环，最后输出路径上的终点及整个路径长度。

（3）具体实现

/\*多建筑间求最佳路径\*/

void BestPath(mgraphtype \*g) {

int vNum[M] = { 0 }, j = 1; /\*记录用户输入的编号信息\*/

int d = 0; /\*统计全程总长\*/

printf("\n请输入你要游览的第%d个建筑的编号（输入-1结束输入）：", j);

scanf("%d", &vNum[j - 1]);

while (vNum[j - 1] != -1 && j < 12) {

while (Judge\_Input(vNum[j - 1])) {

printf("\n请输入你要游览的第%d个建筑编号：", j);

scanf("%d", &vNum[j - 1]);

}

if (vNum[j - 1] == -1) break;

printf("\n请输入你要游览的第%d个建筑编号：", ++j);

scanf("%d", &vNum[j - 1]);

}

printf("\n这是最佳访问路径：");

for (int i = 0; vNum[i] > 0 && vNum[i + 1] > 0; i++) {

printf("%s->", g->vexs[vNum[i] - 1].name); /\*输出路径上的起点\*/

Floyd\_Print(g, vNum[i] - 1, vNum[i + 1] - 1); /\*利用Floyd算法\*/

d += dist[vNum[i] - 1][vNum[i + 1] - 1];

}

printf("%s\n\n", g->vexs[vNum[j - 2] - 1].name); /\*输出路径上的终点\*/

printf("全程总长为：%dm\n\n", d);

## 3.4查询两建筑间的所有路径

（1）功能

通过深度优先遍历查询任意两点间的所有路径，给用户提供多种路线选择。

（2）算法描述

算法步骤如下：

int pathStack[M]; /\*路径栈，存储路径信息\*/

int top; /\*栈顶\*/

int visited[M]; /\*入栈标记，防止形成回路\*/

int count; /\*路径计数器\*/

1. 通过栈进行深度优先遍历，首先判断前一个入栈点可达且该点未入栈。
2. 如果深度优先遍历到了终点，就输入刚才的路径，并且统计路径长度，将最后一条路单独出去，因为enum不能入栈。
3. 如果不是终点，接着深度搜索，所有支路被访问一边后，顶点出栈，将出栈点标记为已出栈，允许下次访问。

（3）具体实现

/\*深度优先遍历查询任意两个建筑之间的所有路径\*/

int pathStack[M]; /\*路径栈，存储路径信息\*/

int top; /\*栈顶\*/

int visited[M]; /\*入栈标记，防止形成回路\*/

int count; /\*路径计数器\*/

void Dfs\_Print(mgraphtype \*g, int sNum, int eNum) {

int dis = 0; /\*用于记录路径长度\*/

pathStack[top] = sNum; /\*将本趟起点入栈\*/

top++;

visited[sNum] = 1; /\*将入栈点标记为已入栈\*/

for (int i = 0; i < g->vexNum; i++) {

if (g->edge[sNum][i] > 0 && g->edge[sNum][i] != INFINITY && !visited[i]) {

/\*表明前一个入栈点与该点可达，且该点未入栈（未被访问）\*/

if (i == eNum) { /\*如果深度遍历搜到了终点，就输出刚才的路径\*/

printf("第%d条路:", count++);

for (int j = 0; j < top; j++) {

printf("%s->", g->vexs[pathStack[j]].name);

if (j < top - 1)

dis = dis + g->edge[pathStack[j]][pathStack[j + 1]]; /\*统计路径长度\*/

}

dis = dis + g->edge[pathStack[top - 1]][eNum]; /\*最后一条路单独出来，因为enum不能入栈\*/

printf("%s\n", g->vexs[eNum].name);

printf("总长度是：%dm\n\n", dis);

}

else {

Dfs\_Print(g, i, eNum); /\*如果该点不是终点,接着深度搜索\*/

top--; /\*支路全被访问一遍后，顶点出栈\*/

visited[i] = 0; /\*将出栈点标记为已出栈，允许下次访问\*/

}

}

}

}

# 4 时间复杂度分析

本系统的主要功能是对齐齐哈尔大学中校区各建筑间路径进行查询，本设计基于结构体数组的结构，用数组表示法创建一个无向图，针对用户的实际需求，将齐齐哈尔大学中校区的主要建筑的编号、名称、介绍等信息放入到图的顶点当中，将路径长度存放在弧当中，它们的时间复杂度为T(n)=O(n)。在本系统中，求出两个建筑之间的最短路径，利用弗洛伊德算法进行查找，其时间复杂度为T(n)= O(n^3)，空间复杂度O(N2)。在求多个建筑间的最佳路径时，因其也利用了弗洛伊德算法所以其时间复杂度为T(n)=O(n^4)。

# 系统测试

## 5.1测试数据

（1）在Dis\_Map()中创建齐齐哈尔大学中校区的模拟图如图5-1所示。

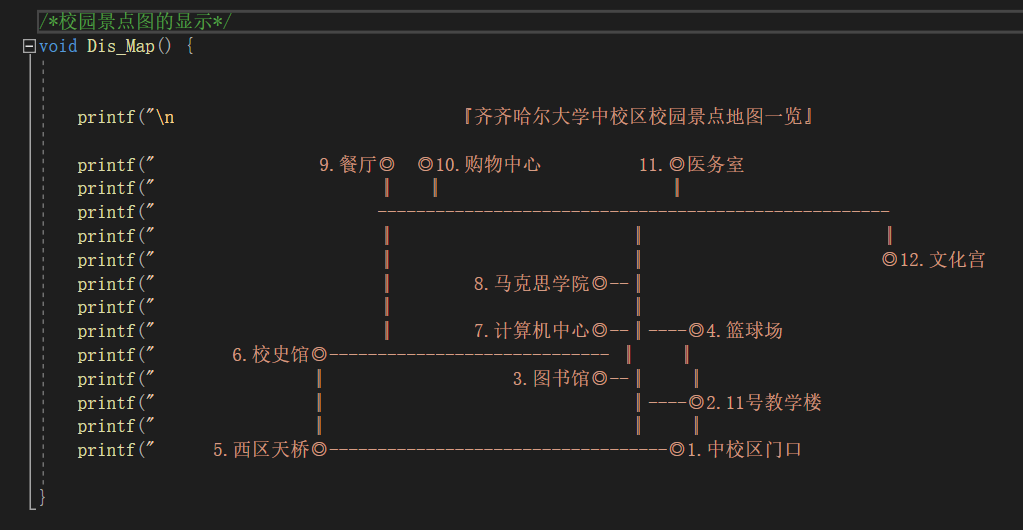


图5-1 齐齐哈尔大学中区模拟图

（2）在map.txt中填写建筑编号、建筑名称、建筑介绍如图5-2所示。

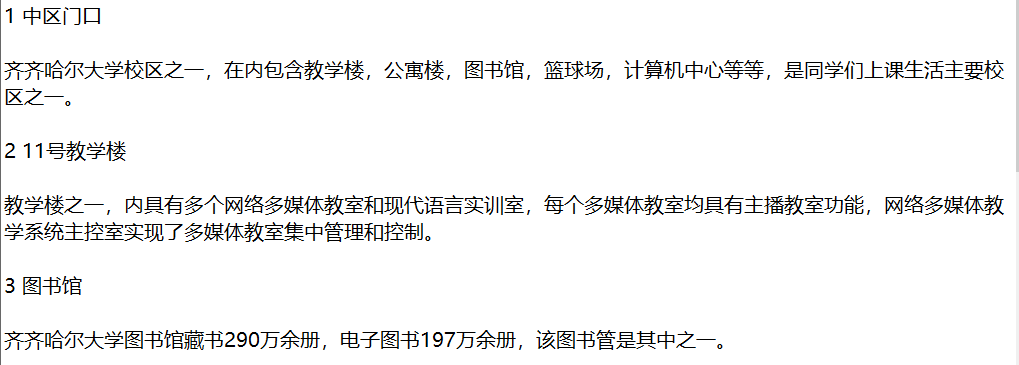


图5-2 建筑编号、名称、介绍

（3）在map.txt中创建建筑之间的路径并输入距离如图5-3所示。



图5-3 建筑之间的路径

## 5.2 测试输出结果

### 5.2.1 管理员界面

进入系统后，系统显示登陆提示选择需要的服务，如图5-4所示。

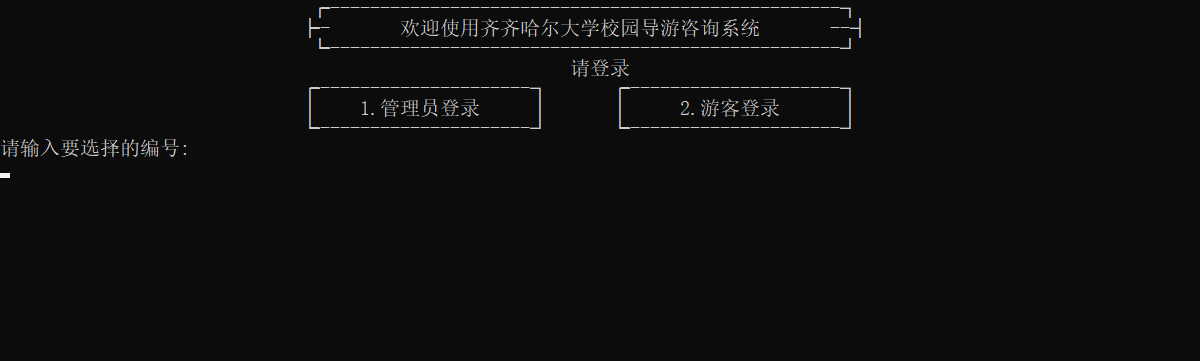


图5-4 登陆选择界面

选择1进入管理员界面，输入账号密码登陆后系统界面如图5-5所示。



图5-5 管理员功能界面

选择不同的操作界面和查询结果如图5-6、5-7所示。

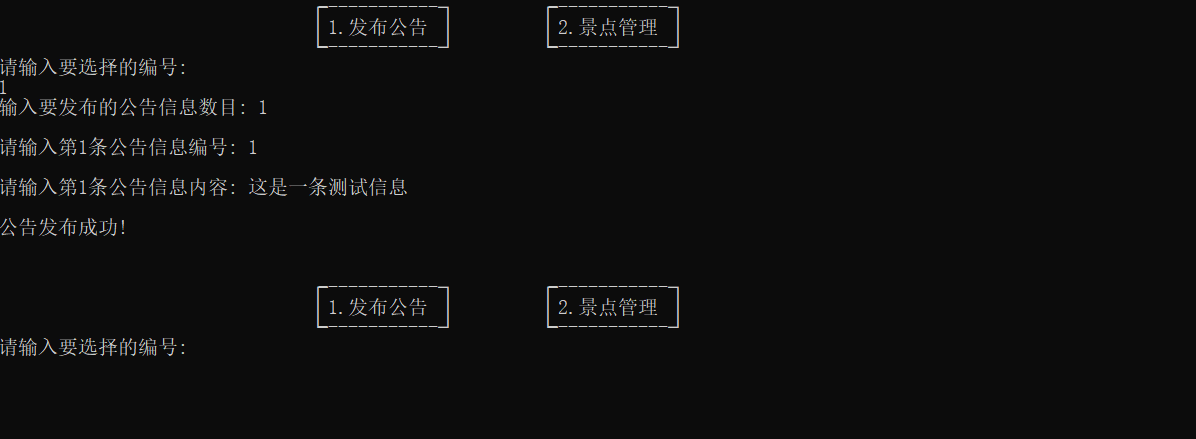


图5-6 发布公告界面

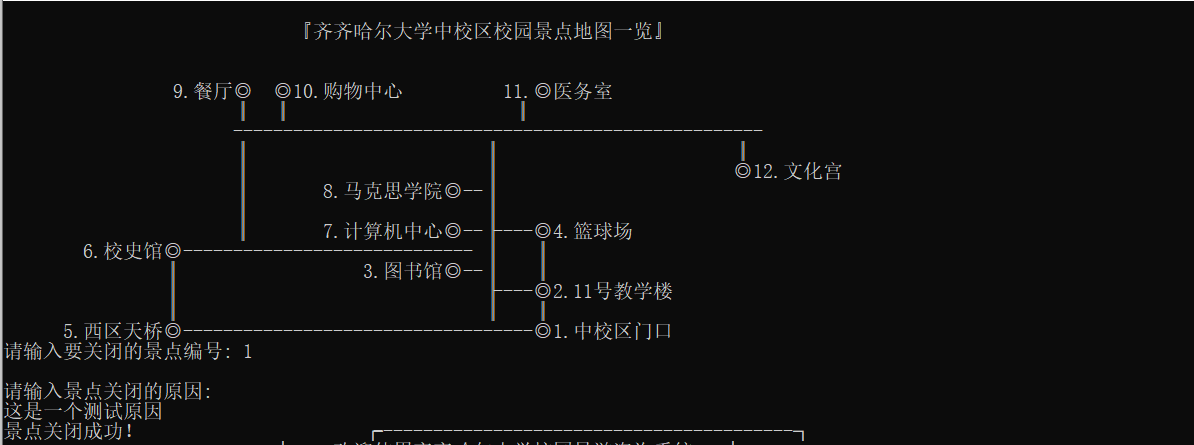


图5-7 景点管理界面

### 5.2.2 用户界面

进入系统后，系统显示登陆提示选择需要的服务，如图5-8所示。

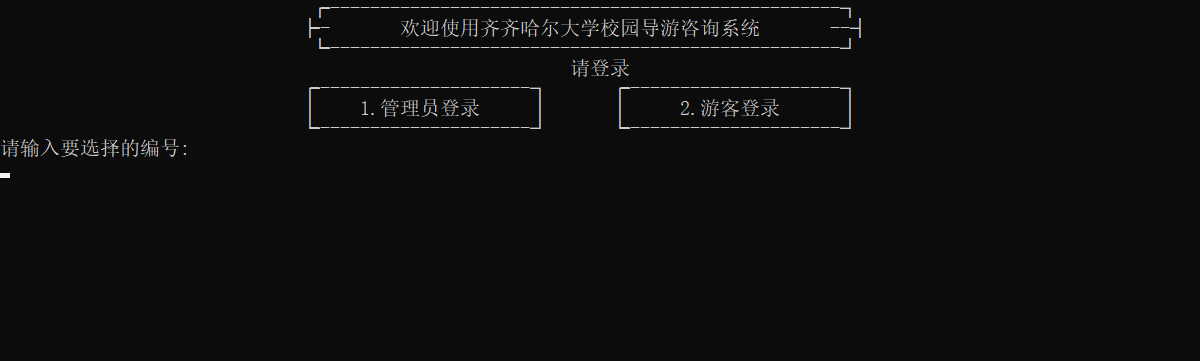


图5-8 登陆选择界面

选择2进入管理员界面，输入账号密码登陆后系统界面如图5-9所示。

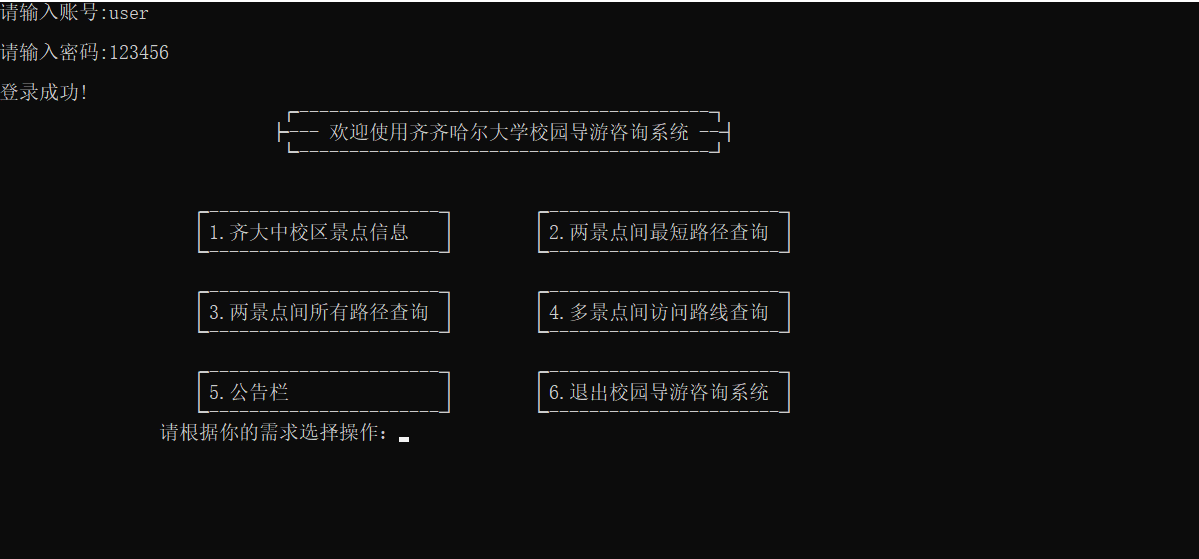


图5-9 用户功能界面

选择功能1，结果输出完毕后，摁任意键回到操作菜单界面，其操作界面和查询结果如图5-10所示。

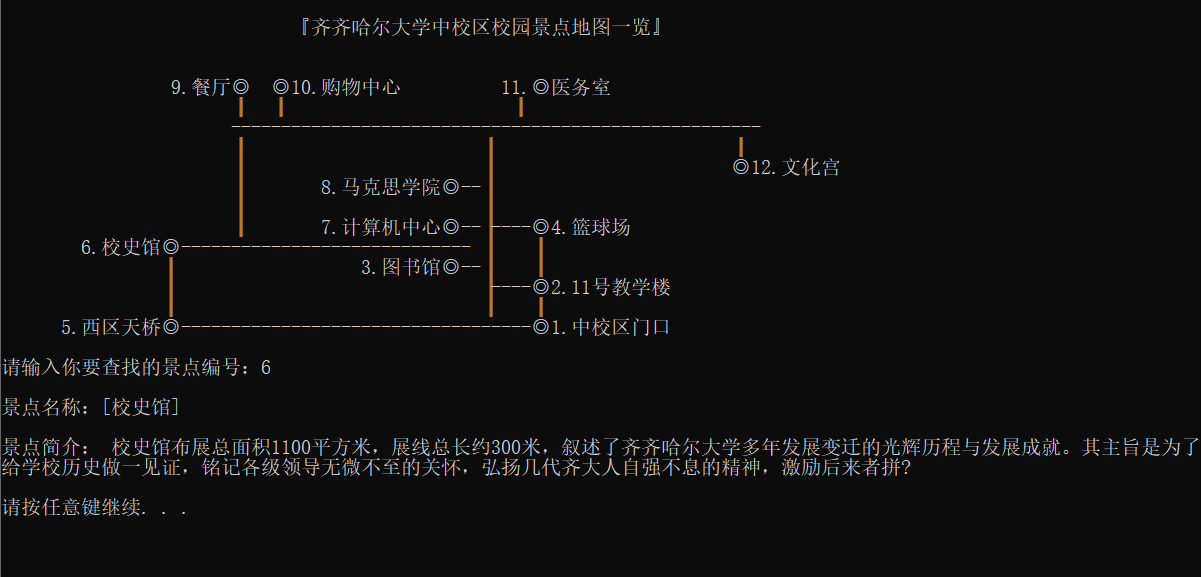


图5-10 景点信息界面

选择功能2时候曾出现如图5-11、5-12所示的情况。

经过排查求最短路径算法发现：状态转移方程为：若dist[i][j] >(dist[i][k] + dist[k][j]) 则dist[i][j] = dist[i][k] + dist[k][j]写成若dist[i][j] <(dist[i][k] + dist[k][j]) 则dist[i][j] = dist[i][k] + dist[k][j]

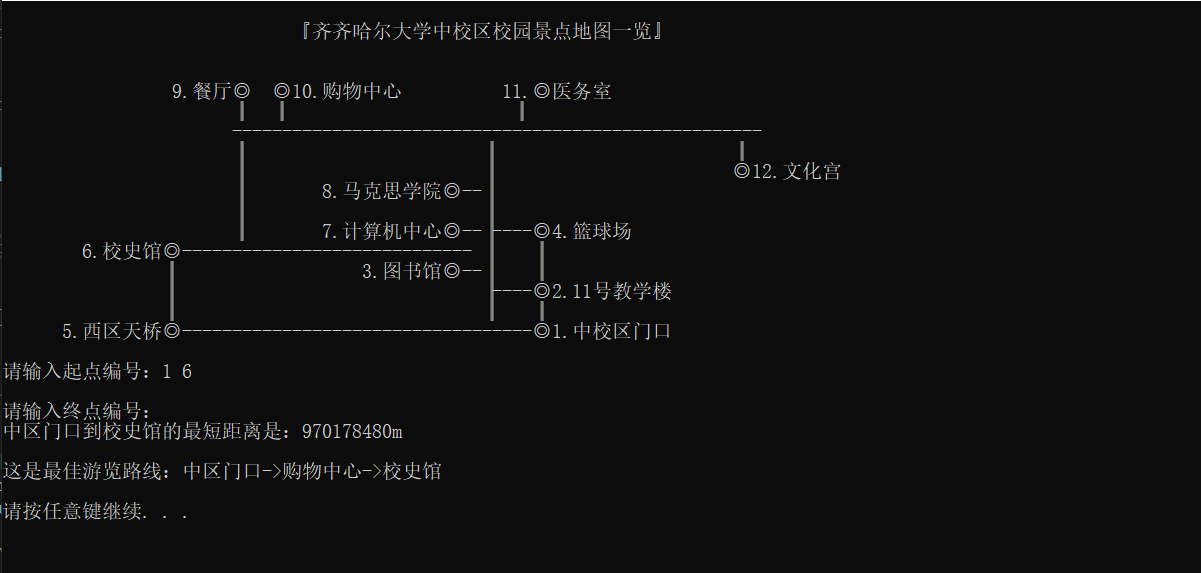


图5-11 错误界面

选择功能2，结果输出完毕后，摁任意键回到操作菜单界面，这里输入编号1 10其操作界面和查询结果如图5-12所示。

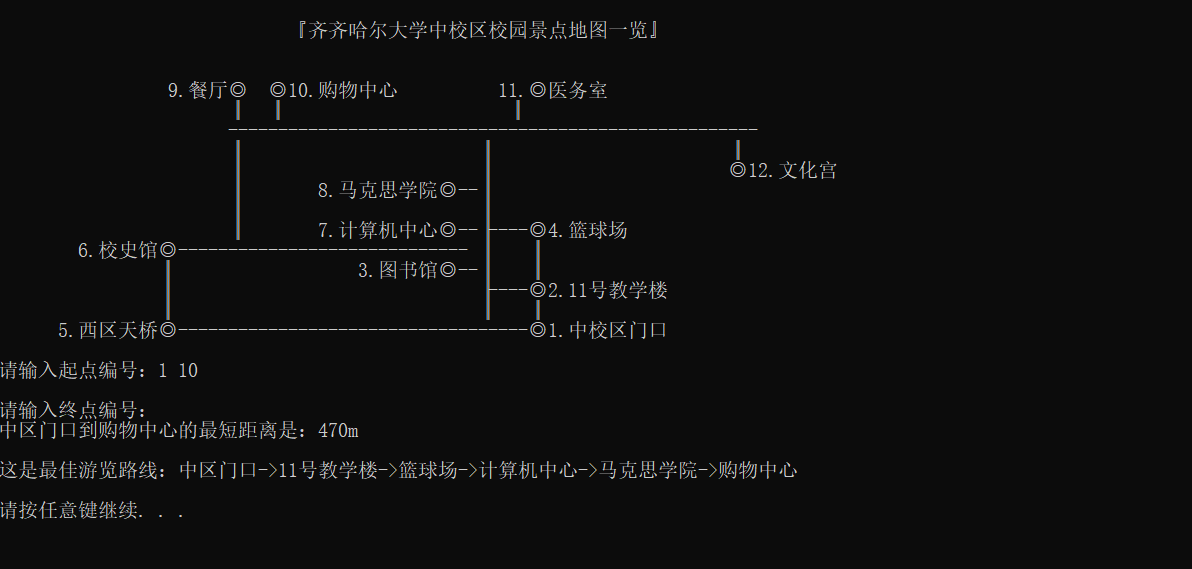


图5-12 最短路径查询界面

选择功能3，结果输出完毕后，摁任意键回到操作菜单界面，这里输入编号1 8其操作界面和查询结果如图5-13所示。

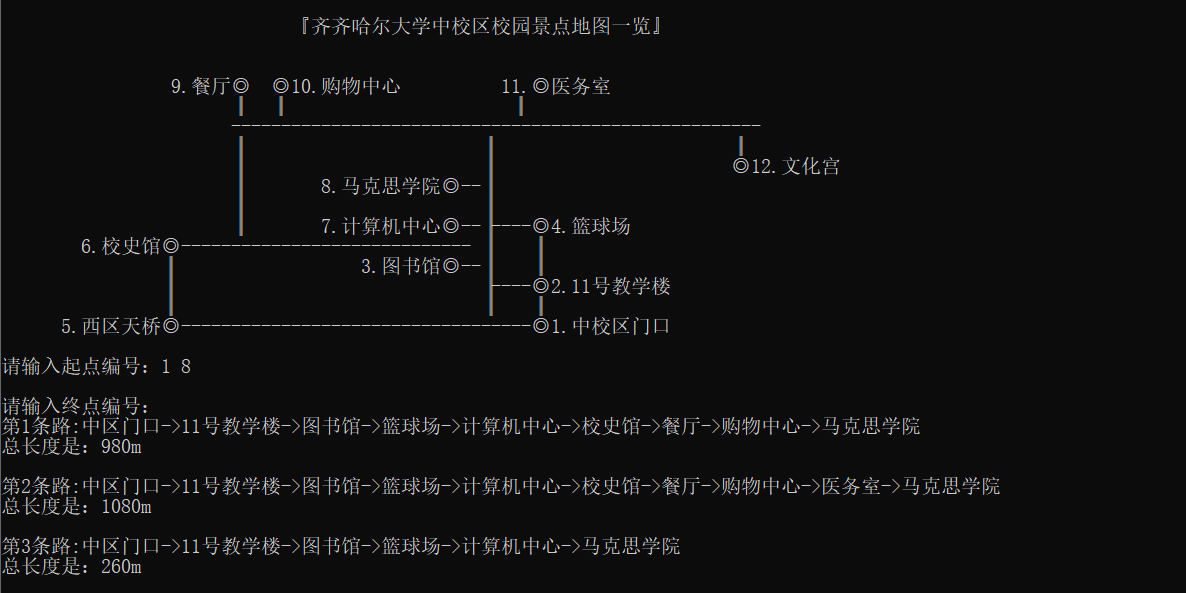


图5-13 两建筑所有路径查询界面

选择功能4，结果输出完毕后，摁任意键回到操作菜单界面，这里输入编号1 3 6 10 8 -1其操作界面和查询结果如图5-14所示。

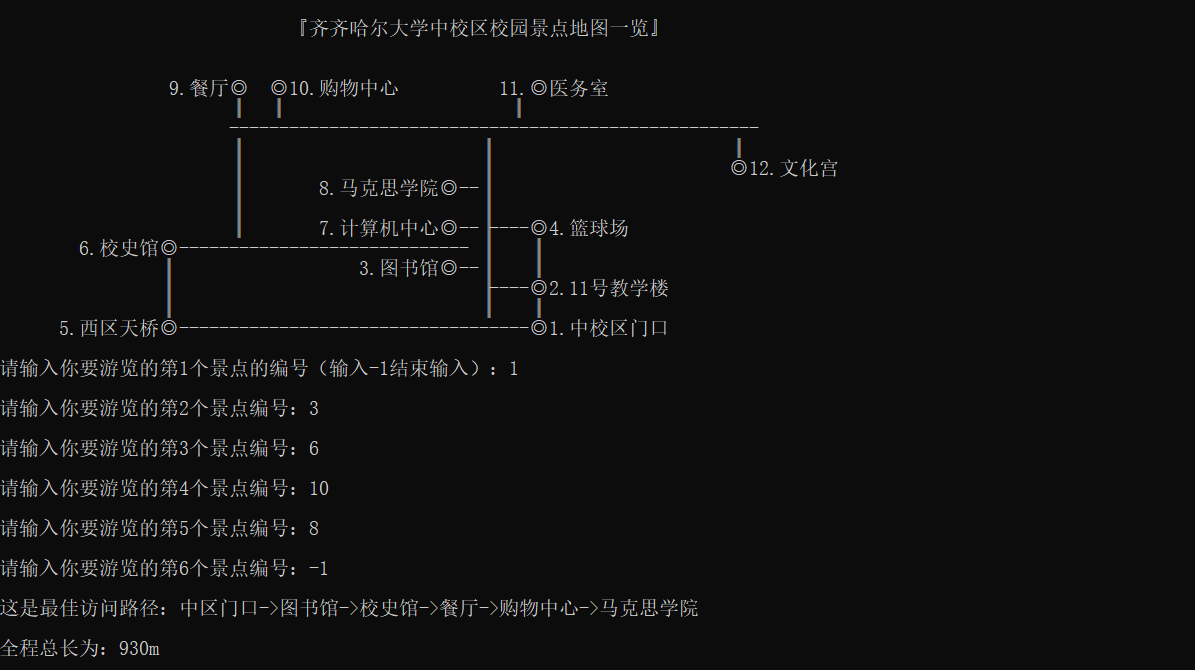


图5-14 多景点最短路径查询界面

选择功能5，结果输出完毕后，摁任意键回到操作菜单界面，其操作界面和查询结果如图5-15所示。

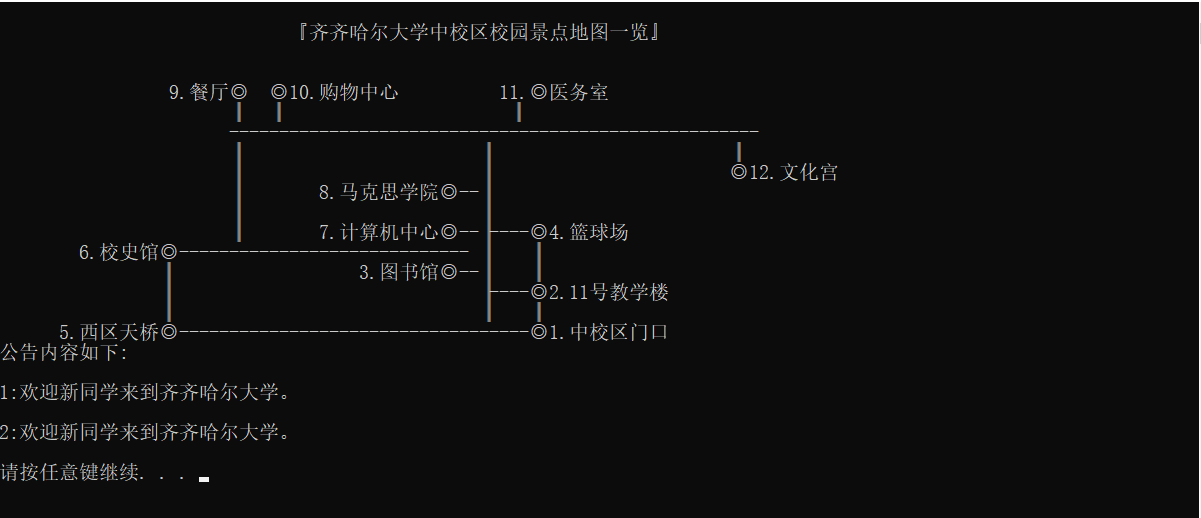


图5-15 公告界面

选择功能6，退出系统。其操作界面和查询结果如图5-16所示。



图5-16 退出系统界面

# 6 总结

本系统是为了新同学快速熟悉校园而开发的，具有建筑查询、路径查询、发布查询公告等功能。运用了学过的弗洛伊德算法来提供系统功能的算法支持；用相应的函数来查找城市，并显示出它的编号，信息，简介。

设计的初衷是为了方便同学快速了解校园，最终实现的效果虽然与自己预想的有所偏差但是大体上已经实现了目标功能，在我看来是一个偏向实用性的、有着现实意义的系统。

经过这次课程设计，让我认识到自己所掌握的知识量还是十分有限，对于基本概念的掌握较为不足，也让我认识到算法对于解决问题的重要性。我从此次课程设计中也对于弗洛伊德算法有了更进一步的认识，也让我明白了算法在学习与实现过程是存在差异的，需要通过仔细的学习算法理论以及做大量的练习才可以达到一个熟练地状态。

数据结构作为计算机一门非常重要的学科，对于整个计算机专业也是尤为重要的，它的重要性渗透在计算机专业的方方面面，我们在学习各种高级编程语言的同时不应忽视计算机基础学科的重要性，只有将基础打好，我们在学习更高级的东西时才会理解其中的内涵，又或者说，如果你将基础打好后，更深一步的学习将不再吃力。本次设计中，让我体会到了数据结构的重要性，不同的数据结构可以解决不同类型的问题。对于合适的问题采用合适的数据结构和存储结构，可以大幅度的提高解决问题的效率，也是一个合格的程序设计者的必备技能，通过这次课程设计，我认为我还要继续完善关于数据结构方面的知识，更多的练习使用不同的数据结构去解决不同的问题，提高使用各种数据结构的熟练度。

整个开发过程使我受益匪浅，也让我把在学校学习到的知识进行了实际的应用，经过了多次的调试，终于完成了本系统，同时也发现自身的不足，对于对知识的查缺补漏非常的有帮助。虽然本系统的功能不多，也存在着一些尚未解决的问题，但是还是对我起到了非常大的鼓励作用。同时此次课程设计也一定程度的锻炼了我的自学能力、分析问题能力以及解决问题的能力，对我之后的学习和工作都会有着非常大的影响。