数据结构

课程设计报告

题 目： 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172

学生姓名： 范鑫

指导教师： 杜晓昕

2019 年12 月26日

数据结构课程设计评分表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 班级 | 软件172 | | | 姓名 | 范鑫 | | 指导教师 | 杜晓昕 | |
| 题目：校园导游查询系统 | | | | | | | | | |
| 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

随着现代生活节奏的加快，人们外出旅行的时间越来越多。考虑到游客不可能对所有景点都有所了解，因此可能无法找到游玩景点最省时，最高效的路径，而人工导游成本又过高，故使用C和C++，基于《数据结构》中图的相关算法与HGE引擎窗口界面开发了“齐齐哈尔大学导游咨询系统”。

此课程设计的主要内容是校园导游系统。所谓系统其实也不尽然，只不过是个小小的提示，为来访的客人提供各种信息查询服务。现在大多数的学校由于不断的扩张，这也就使得学校不得不建立的更大。这也就为人们拜访学校造成了极大的不便。人们往往不熟悉学校，找个东西，或某处带来了极大的不便，往往要花很多时间在这一方面。然而要是有一个学校导游系统这将给游客带来极大的方便，使人们一下就能了解到这个学校的大致情况。

主要的功能包括：

①查看学校的全景图。

②各个景点的简介。

③学校主要景点的分析。

④查看某一景点到其他所有景点的最短路径。

⑤查询任意连个景点之间的最短路径。

2. 数据结构描述

2.1 图

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。在计算机科学中，图是最灵活的数据结构之一，很多问题都可以使用图模型进行建模求解，例如系统工程、化学分析、统计力学、遗传学、人工智能等领域。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。因为道路与道路之间是多种相同的关系，即多对多的关系。并且二者之间没有特特定的方向，所以本系统是利用数据结构中无向图设计出来的。因为道路网多是稀疏网，故采用邻接多重表作存储结构。

typedef struct arcell

{

int adj;

//权值。

}arcell,adjmatrix[MaxVertexNum][MaxVertexNum];

//图的邻接矩阵类型。

typedef struct vexsinfo

//顶点信息。

{

int position;

//景点的编号。

char name[32];

//景点的名称。

char introduction[256];

//景点的介绍。

}vexsinfo;

typedef struct mgraph

//图结构信息。

{

vexsinfo vexs[MaxVertexNum];

//顶点向量(数组)。

adjmatrix arcs;

//邻接矩阵。

int vexnum,arcnum;

//分别指定顶点数和边数。

}mgraph;

2.2 邻接矩阵存储

邻接[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)（Adjacency Matrix）是表示顶点之间相邻关系的矩阵。设G=(V,E)是一个图，其中V={v1,v2,…,vn}。G的邻接矩阵是一个具有下列性质的n阶方阵：

1. 对[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)而言，邻接矩阵一定是对称的，而且主对角线一定为零（在此仅讨论无向简单图），副对角线不一定为0，[有向图](https://baike.baidu.com/item/%E6%9C%89%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)则不一定如此。
2. 在无向图中，任一顶点i的度为第i列（或第i行）所有非零元素的个数，在有向图中顶点i的出度为第i行所有非零元素的个数，而入度为第i列所有非零元素的个数。
3. 用邻接矩阵表示图，很容易确定图中任意两个顶点是否有边相连。用邻接矩阵法表示图共需要n^2个空间，由于无向图的邻接矩阵一定具有[对称关系](https://baike.baidu.com/item/%E5%AF%B9%E7%A7%B0%E5%85%B3%E7%B3%BB" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)，所以扣除对角线为零外，仅需要存储上三角形或下三角形的数据即可，因此仅需要n（n-1）/2个空间。

[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)的邻接[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)一定是对称的，而有向图的邻接矩阵不一定对称。因此，用邻接矩阵来表示一个具有n个顶点的有向图时需要n^2个单元来存储邻接矩阵；对有n个顶点的无向图则只存入上（下）三角阵中剔除了左上右下对角线上的0元素后剩余的元素，故只需1+2+...+(n-1)=n(n-1)/2个单元。无向图邻接矩阵的第i行（或第i列）非零元素的个数正好是第i个顶点的度。

有向图邻接矩阵中第i行非零元素的个数为第i个顶点的出度，第i列非零元素的个数为第i个顶点的入度，第i个顶点的度为第i行与第i列非零元素个数之和。

需要注意到的是：

①在简单应用中，可直接用二维[数组](https://baike.baidu.com/item/%E6%95%B0%E7%BB%84" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)作为图的邻接[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)(顶点表及顶点数等均可省略)。

② 当邻接矩阵中的元素仅表示相应的边是否存在时，EdgeTyPe可定义为值为0和1的[枚举类型](https://baike.baidu.com/item/%E6%9E%9A%E4%B8%BE%E7%B1%BB%E5%9E%8B" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)。

③[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)的邻接矩阵是[对称矩阵](https://baike.baidu.com/item/%E5%AF%B9%E7%A7%B0%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)，对规模特大的邻接矩阵可压缩存储。

④邻接矩阵表示法的空间复杂度S(n)=0(n 2 )。

⑤建立无向网络的算法。

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。

int creatgragh(mgraph &c)

{

int i,j,m,n;

int v0,v1;

int distance;

printf("请输入图的顶点数和边数: \n");

scanf("%d %d",&c.vexnum ,&c.arcnum );

printf("下面请输入景点的信息：\n");

for(i=0;i<c.vexnum ;i++)

{

printf("请输入景点的编号：");

scanf("%d",c.vexs[i].position );

printf("\n请输入景点的名称：");

scanf("%s",c.vexs[i].introduction );

}

for(i=0;i<c.arcnum ;i++)

for(j=0;j<c.arcnum ;j++)

c.arcs[i][j].adj =Infinity;

printf("下面请输入图的边的信息：\n");

for(i=1;i<=c.arcnum ;i++)

{

printf("\n第%d条边的起点 终点 长度为：",i);

scanf("%d %d %d",&v0,&v1,&distance);

m=locatevex(c,v0);

n=locatevex(c,v1);

if(m>=0 && n>=0)

{

c.arcs[m][n].adj =distance;

c.arcs[n][m].adj =c.arcs[m][n].adj ;

}

}

return 1;

}//creatgragh

3. 算法描述

3.1 系统功能模块结构

本系统主要有八大功能模块分别是：

1. 学校景点介绍
2. 查看浏览路线
3. 查询景点间的最短路径
4. 景点信息查询
5. 更改图信息
6. 查询景点间可行路径
7. 打印邻接矩阵
8. 退出

通过系统功能结构图可以更能清楚的描述出来。其功能结构图如图3-1所示。

图3-1 系统功能结构图

当我们想要更加实际的了解一个程序的算法过程的时候，我们就要依据程序的流程图来给我们一个比较实际的过程，从流程图当中能够更加清楚整个程序实现的过程是怎样的。

本系统中在主界面中选择编号进入不同的功能，例如：选择1就是学校景点介绍，若是想要其他功能，直接输出编号即可，如果使用完本系统，想要退出，可以选择8，退出本系统。其流程图如图3-2所示。



图3-2 系统流程图

3.2 查看浏览路线模块

1．功能

当用户选择该功能，系统根据用户输入的起始景点及目的景点编号，查询任意两个景点之间的最短路线路径及距离。

1. 算法描述

算法步骤如下：

1.迪杰斯特拉算法，求从顶点v0到其余顶点的最短路经及其带权长度d[v]。

2.若p[v][w]为1，则w是从v0到v的最短路经上的顶点。

3. final[v]类型用于设置访问标志。

迪杰斯特拉算法解释：迪杰斯特拉最最朴素的思想就是按长度递增的次序产生最短路径。即每次对所有可见点的路径长度进行排序后，选择一条最短的路径，这条路径就是对应顶点到源点的最短路径。算法描述如下：

1）令arcs表示弧上的权值。若弧不存在，则置arcs为∞（在本程序中为MAXCOST）。S为已找到的从v出发的的终点的集合，初始状态为空集。那么，从v出发到图上其余各顶点Vi可能达到的长度的初值为D=arcs[Locate Vex(G,Vi)],Vi∈V。

2）选择，使得D[j]=Min{ D|Vi∈V-S}。

3）修改从v出发的到集合V-S中任一顶点Vk的最短路径长度。

3．具体语言实现

void shortestpath\_dij(mgraph c)

{

int v,w,i,min,t=0,x,flag=1,v0;

int final[35],d[35],p[35][35];

printf("\n请输入一个起始景点的编号：");

scanf("%d",&v0);

printf("\n\n");

while(v0<0||v0>c.vexnum)

{

printf("\n你所输入的景点编号不存在\n");

printf("请重新输入：");

scanf("%d",&v0);

}//while

d[v0]=0;

final[v0]=1;

for(i=1;i<c.vexnum ;i++)

{

min=Infinity;

final[v]=1;

//v的访问标志设置为1，v属于s集。

for(v=0;v<c.vexnum ;v++)

//输出v0 到其它顶点v 的最短路径。

{

if(v!=v0)

printf("%s",c.vexs[v0].name);

printf("---->%s",c.vexs[v].name);

printf("\n总路线长为%d米\n\n",d[v]);

}//for

}

//shortestpath\_dij

3.3 查询景点间最短距离模块

1．功能

当用户选择该功能，系统根据用户输入的起始景点及目的景点编号，查询任意两个景点之间的最短路线路径及距离。

2．算法描述

floyd算法求各对顶点v和w间的最短路经及其带权长度的d[v][w]。若p[v][w][u]==1；则u是v到w的当前求得的最短路经上的顶点。

floyd算法解释：Floyd算法又称为插点法，是一种利用[动态规划](https://baike.baidu.com/item/%E5%8A%A8%E6%80%81%E8%A7%84%E5%88%92/529408" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)的思想寻找给定的[加权图](https://baike.baidu.com/item/%E5%8A%A0%E6%9D%83%E5%9B%BE/10579361" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)中多源点之间[最短路径](https://baike.baidu.com/item/%E6%9C%80%E7%9F%AD%E8%B7%AF%E5%BE%84/6334920" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)的算法，与Dijkstra算法类似。Floydl算法是一种在具有正或负边缘权重（但没有负周期）的加权图中找到最短路径的算法。算法的单个执行将找到所有顶点对之间的最短路径的长度（加权）。 虽然它不返回路径本身的细节，但是可以通过对算法的简单修改来重建路径。 该算法的版本也可用于查找关系R的传递闭包，或（与Schulze投票系统相关）在加权图中所有顶点对之间的最宽路径。

通过一个图的权值[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)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" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)矩阵。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](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。

采用松弛技术（[松弛操作](https://baike.baidu.com/item/%E6%9D%BE%E5%BC%9B%E6%93%8D%E4%BD%9C" \t "https://baike.baidu.com/item/Floyd%E7%AE%97%E6%B3%95/_blank)），对在i和j之间的所有其他点进行一次松弛。所以时间复杂度为O(n^3)。

3．具体语言实现

void shortestpath\_floyd(mgraph c)

{

int i,j,k,d[35][35],p[35][35][35];

int v,u,w;

for(v=0;v<c.vexnum ;v++)

{

for(w=0;w<c.vexnum ;w++)

{

d[v][w]=c.arcs[v][w].adj;

}//endfor

for(u=0;u<c.vexnum ;u++)

// 求v至w的最短路径及距离。对任意顶点u。

//试探其是否为v至w最短路径上的顶点。

{

for(v=0;v<c.vexnum ;v++)

for(w=0;w<c.vexnum ;w++)

if(d[v][u]+d[u][w]<d[v][w])

//从 v 经 u 到 w 的一条路径更短。

{

d[v][w]=d[v][u]+d[u][w];

for(i=0;i<c.vexnum ;i++)

p[v][w][i]=p[v][u][i]||p[u][w][i];

}

}//endfor

printf("\n请输入出发点和目的地编号：");

scanf("%d%d",&k,&j);

printf("\n\n");

//输出出发点和目的地编号。

printf("%s",c.vexs[k].name );

for(u=0;u<c.vexnum ;u++)

if(p[k][j][u] && k!=u && j!=u)

printf("--->%s",c.vexs[u].name );

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

printf("\n\n\n总长为%d米\n\n\n",d[k][j]);

}//shortestpath\_floyd

3.4 查询景点间可行路模块

1．功能

查询景点间所有可行路径。

2．算法描述

由于是无向网，如果网中的边数很多，任意两个景点间的所有路径也会有很多，但很多路径是无实际意义的（有近路，为什么去走远路呢？）所有，本算法只求得的两景点间所有可行路径中，现只输出路径长度不超过8个景点的路线。对[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)而言，邻接矩阵一定是对称的，而且主对角线一定为零（在此仅讨论无向简单图），副对角线不一定为0，[有向图](https://baike.baidu.com/item/%E6%9C%89%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)则不一定如此。

在无向图中，任一顶点i的度为第i列（或第i行）所有非零元素的个数，在有向图中顶点i的出度为第i行所有非零元素的个数，而入度为第i列所有非零元素的个数。

[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)的邻接[矩阵](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E7%9F%A9%E9%98%B5/_blank)一定是对称的，而有向图的邻接矩阵不一定对称。因此，用邻接矩阵来表示一个具有n个顶点的有向图时需要n^2个单元来存储邻接矩阵；对有n个顶点的无向图则只存入上（下）三角阵中剔除了左上右下对角线上的0元素后剩余的元素，故只需1+2+...+(n-1)=n(n-1)/2个单元。无向图邻接矩阵的第i行（或第i列）非零元素的个数正好是第i个顶点的度。

3．具体语言实现

void path(mgraph c, int m,int n,int k)

{ int s,x=0;

int t=k+1;

//t 记载路径上下一个中间顶点在d[]数组中的下标。

if(d[k]==n && k<8)

//d[k]存储路径顶点。若d[k]是终点n且景点个数<=8，则输出该路径。

{

for(s=0;s<k;s++)

printf("%s--->",c.vexs[d[s]].name);

printf("%s",c.vexs[d[s]].name);

printf("\n\n");

}

else

{

s=0;

}//endelse

}//endpath

int allpath(mgraph c)

{

int k,i,j,m,n;

printf("\n\n请输入你要查询的两个景点编号:\n\n");

scanf("%d%d",&i,&j);

printf("\n\n");

m=locatevex(c,i);

d[0]=m;

//存储路径起点m (int d[]数组是全局变量)。

for(k=0;k<c.vexnum;k++)

Visited[k]=0;

Visited[m]=1;

//第m个顶点访问标志设置为1。

path(c,m,n,0);

return 1;

}

4. 时间复杂度分析

考虑道路网多是稀疏网，故采用邻接多重表作存储结构，其空间复杂度为O（e），此时的时间复杂度也为O（e）。构建邻接多重表的时间复杂度为O（n+e），输出路径的时间复杂度为O（n）。由此，本导游程序的时间复杂度为O（n+e）。

邻接多重表解释：邻接多重表是[无向图](https://baike.baidu.com/item/%E6%97%A0%E5%90%91%E5%9B%BE/1680427" \t "https://baike.baidu.com/item/%E9%82%BB%E6%8E%A5%E5%A4%9A%E9%87%8D%E8%A1%A8/_blank)的一种存储方式。邻接多重表是邻接表的改进，它把边的两个顶点存放在边表结点中，所有依附于同一个顶点的边串联在同一链表中，由于每条边依附于两个顶点，则每个边表结点同时链接在两个链表中。在此存储结构中，图的顶点信息存放在顶点数组中，数组元素有两个域：data域，存放与顶点相关的信息；firstedge域，指向一个单链表，此单链表存储所有依附于该顶点的边的信息。这些单链表的一个表结点对应一条边，表结点有六个域：mark为标志域，用来标记该边是否被访问过；ivex和jvex分别存放该边两个顶点在图中的位置；info域存放该边相关的信息，实际上就是弧的权值，对于无向图，info域可省略; ilink指向下一条依附于顶点ivex的边对应的表结点;jlink指向下一条依附于顶点jvex的边对应的表结点。

邻接表类似于树的孩子链表法顺序存储是指图中的顶点信息用一个一维数组来存储，一个顶点数组元素是一个顶点结点。顶点结点有两个域，一个是数据域(data)，存放与顶点相关的信息;一个是指针域( firestar)存放该顶点的邻接表的第一个结点的地址。顶点的邻接表是把所有邻接于某结点的顶点构成一个表，它采用链式存储结构。邻接表中的每个结点保存的是与该顶点相关的边或弧的信息，它有两个域，一个是邻接顶点域 adnex)，存放邻接顶点的信息，实际上就是邻接顶点在顶点数组中的序号:另一个是指针域( next arc)，存放下一个邻接顶点的结点的地址。

5. 系统测试

各子功能测试运行结果如下。

5.1学校景点介绍

在主菜单下，用户输入1并回车，系统会给出学校景点的介绍。运行结果的截图如图5-1所示。

不足之处：线路的编排受景点编号的影响，可能有些不合理。

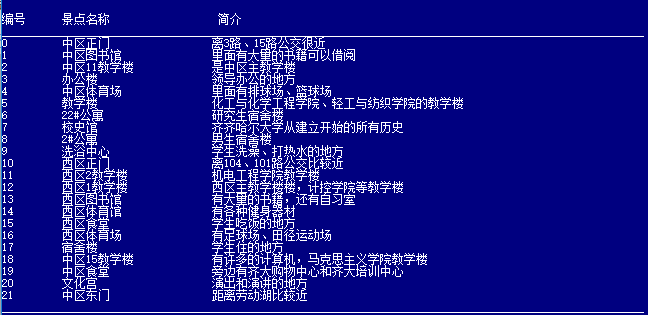


图5-1 齐齐哈尔大学中西校区景点名称及简介截图

5.2查看浏览路线

在主菜单下，用户输入2并回车，根据屏幕提示输入一个景点编号4并回车，系统会给出景点4到其余21个景点的最短浏览线路及最短距离。运行结果的截图如图5-2所示。

不足之处：线路的编排受景点编号的影响，可能有些不合理。

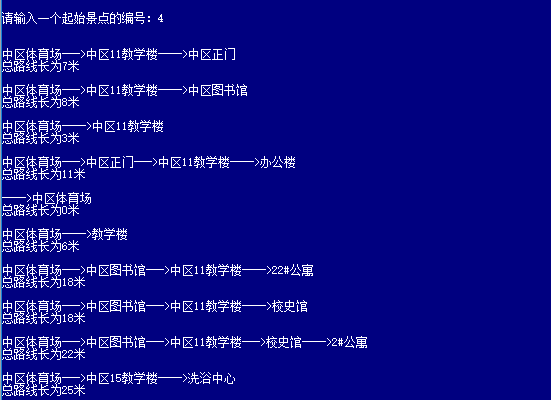


图5-2 从一个景点出发的浏览线路图截图

5.3查询景点间最短路径

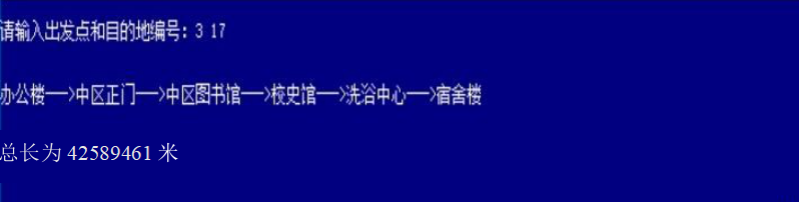


图5-3 错误运行界面

经排查后，发现其函数当中的循环结构的初始值和临界值设置为0和小于节点个数，即for (k=0; k<G.vexnum; k++)，与之前定义不同导致错误的发生，将其改为for (k=1; k<=G.vexnum; k++)后程序正常运行。

在主菜单下，用户输入3并回车，根据屏幕提示输入一个出发景点编号及目的地点编号：3 17并回车后，运行结果如图5-4所示。

不足之处：线路的编排受景点编号的影响，可能有些不合理。

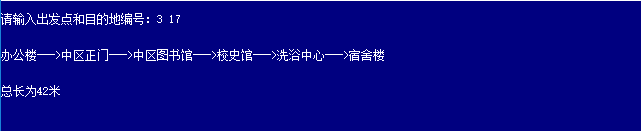


图5-4 任意两个景点之间的最短浏览路线图截图

5.4景点信息查询

在主菜单下，用户输入4并回车，根据屏幕提示输入一个要查询的景点编号20并回车后，运行结果如图5-5所示。

不足之处：线路的编排受景点编号的影响，可能有些不合理。



图5-5 景点查询信息截图

5.5 更改图的信息

在主菜单下，用户输入5后出现二级菜单界面，运行结果如图5-6所示。再进一步做选择，可以实现图的相关操作。

不足之处：线路的编排受景点编号的影响，可能有些不合理。

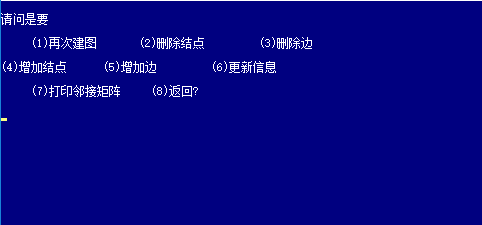


图5-6更改图的信息截图

5.6查询景点间可行路径

本算法在求得的两景点间所有可能路径中，限制只能输出路径长度不超过8个景点的路线。在主菜单下，用户输入6并回车，根据屏幕提示输入要查询的两个景点编号：5 17并回车后，运行结果如图5-7所示。本功能有递归函数实现，所以当图中的边数过多时，可能造成死循环而得不到正确的结果。

不足之处：线路的编排受景点编号的影响，可能有些不合理。

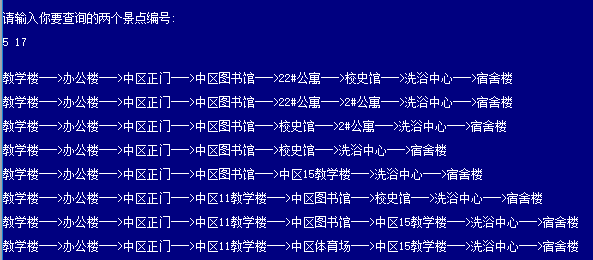


图5-7 查询景点间的可行路径截图

5.7打印邻接矩阵

在主菜单下，用户输入7并回车，即可输出图的邻接矩阵的值，如图5-8所示。

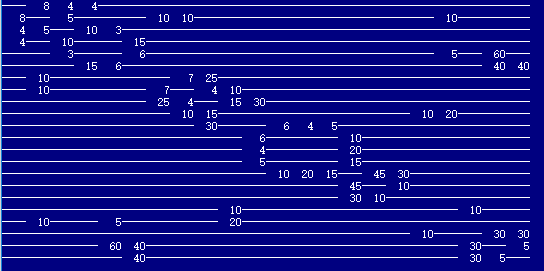


图5-8 图的邻接矩阵截图

1. 总结

本系统主要是为了使游客快速的简单查询本校的信息和各路段，可以方便游客的使用。本系统是利用数据结构中图的应用设计出来的。通过邻接多重表作存储结构，存储学校的各种信息。经过这此课程设计，我对程序中的算法的概念理解的更加透彻。算法是程序中必不可少的部分。它是指解题方案的准确而完整的描述，是一系列解决问题的清晰指令，算法代表着用系统的方法描述解决问题的策略机制。也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。同时，在选择算法时必须考虑算法的时间复杂度和空间复杂度这样才能让程序正常，高校的运行。

在系统设计时也碰到了很多问题，比如在设计InitGraph()函数时，首先我想到的是使用数组来保存信息，但发现这种设计无法方便的让程序中各个独立算法访问存储的景点信息，编写代码时十分复杂，后来，在出查阅相关书籍和阅读了网上其它相关算法后，我选择了使用结构体数组来保存信息，这种方式顺利地将信息保存到了图中，同时其它函数模块也可以方便的访问这部分信息，达到了我想要实现的功能。

《数据结构》在计算机科学中是一门综合性的专业基础课，数据结构的研究不仅涉及计算机的硬件（特别是编码理论、存储装置和存取方法）的研究范围，而且和计算机软件的研究有着更密切的关系，无论是编译程序还是操作系统，都涉及到数据元素在存储器中的分配问题，在研究信息检索时也必须考虑如何组织数据，以便查找和存取数据元素更为方面。因此，可以认为数据结构介于数学、计算机硬件和计算机软件三者之间的一门核心课程。在计算机科学中，数据结构不仅是一般程序设计（特别是非数值计算的程序设计）的基础，而且是设计和实现编译程序、操作系统、数据库系统以及其他系统程序和大型应用程序的重要基础。