数据结构课程设计

说明书

|  |  |  |  |
| --- | --- | --- | --- |
| **学生姓名:** | **朱继熠** | **学 号：** | **1607104139** |
| **学生姓名:** | **梁亚亚** | **学 号：** | **1607104117** |
| **学生姓名:** | **李静涵** | **学 号：** | **1607104104** |
| **学生姓名:** | **涂大祥** | **学 号：** | **1607104143** |
| **学 院:** | **大数据学院** | | |
| **专 业:** | **数字媒体技术** | | |
| **题 目:** | **校园导游系统咨询** | | |
| **指导教师** | **蔺素珍 李玲** | | |

2018年1月5日

目录

[1 设计目的 1](#_Toc7901)

[2 设计内容 1](#_Toc29720)

[3 模块分析 3](#_Toc5023)

[4 数据结构 3](#_Toc32681)

[5 功能模块详细设计 4](#_Toc22810)

[5.1 全局变量及结构体 4](#_Toc15309)

[5.1.1 详细设计思想 6](#_Toc12178)

[5.1.2 核心代码 7](#_Toc1492)

[5.2 图的建立模块 13](#_Toc7882)

[5.2.1 详细设计思想 13](#_Toc12133)

[5.2.2 核心代码 16](#_Toc7952)

[5.3 邻接矩阵转置邻接表模块 1](#_Toc4044)9

[5.3.1 详细设计思想 1](#_Toc20941)9

[5.3.2 核心代码 2](#_Toc5338)0

[5.4 两景点最短路径模块 21](#_Toc8054)

[5.4.1 详细设计思想 21](#_Toc30602)

[5.4.2 核心代码 25](#_Toc6919)

[5.5 两景点所有路径模块 2](#_Toc31878)8

[5.5.1 详细设计思想 2](#_Toc4308)8

[5.5.2 核心代码 29](#_Toc4855)

[5.6 多景点最佳路径模块 31](#_Toc7700)

[5.6.1 详细设计思想 31](#_Toc22887)

[5.6.2 核心代码 32](#_Toc23976)

[5.7 图的关节点模块 33](#_Toc356)

[5.7.1 详细设计思想 33](#_Toc19236)

[5.7.2 核心代码 35](#_Toc15291)

[5.8 主函数模块 36](#_Toc24394)

[5.8.1 详细设计思想 36](#_Toc5985)

[5.8.2 核心代码 37](#_Toc7804)

[6 运行结果 46](#_Toc4088)

[7 总结 5](#_Toc10312)0

[参考文献 51](#_Toc21884)

1. **设计目的**

《数据结构》课程主要介绍最常用的数据结构，阐明各种数据结构内在的逻辑关系，讨论其在计算机中的存储表示，以及在其上进行各种运算时的实现算法，并对算法的效率进行简单的分析和讨论。该课程让我们进一步巩固了数据结构的相关知识，提高了我们解决实际问题的能力，通过该课程的设计，我们掌握了综合运用数据结构相关知识的，进行数据结构课程设计要达到以下目的：

(1)了解并掌握数据结构与算法的设计方法，具备初步的独立分析和设计能力；

(2)初步掌握软件开发过程的问题分析、系统设计、程序编码、测试等基本方法和技能；

(3)提高综合运用所学的理论知识和方法独立分析和解决问题的能力；

训练用系统的观点和软件开发一般规范进行软件开发，培养软件工作者所应具备的科学的工作方法和作风。

1. **设计内容**

设计一个校园导游程序，为来访的客人提供各种信息查询服务与旅游路线定制等功能

1. 设计中北大学的校园平面图，所含景点不少于10个，以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息，存储在文本文档中，并制定相应的文件读取算法按照格式录入程序。各景点之间的邻接关系与自定义权值同样通过文档读取，为了方便后续的深度遍历求关节点，需要设计一个邻接表转换算法，边的权值作为入度边存放在衍生邻接表子节点的数值域。
2. 为来访客人提供图中任意景点相关信息的查询，即采用图形交互界面得到用户想要访问的景点序号，并将该值传入已录入信息的结构体数组中，直接打印输出。抑或通过字符串匹配函数，采用顺序检索的方式逐一进行比对。
3. 调用EGE外部图形库，即可以将高清图片显示在控制台应用程序中，用户可以根据更加形象先进的显示方式直观地观察到各个景点的序号与连接方式，并按照菜单显示的要求键入关键值选择功能模块。系统自动识别用户输入的信息，进行相应的字符串序号匹配，智能剔除不符合要求的键值，通过设计的数组类型转换，适配其他功能函数参数的传入。二级主界面模块采用循环的方式，并在每个子功能结束后的全局参数进行必要的重设，使用户可以在不关闭程序的前提下不断使用。输出模块取代了C自带的格式化输出函数，采用自由度较强的外部函数高度定制输出的样式与颜色，采用必要的像素函数及视窗函数不断对输出数据的显示进行调整，始终保持视觉优化。
4. 为来访客人提供图中任意景点的问路查询，与任意两相景点之间的一条最短的简单路径。先采用Dijkstra算法处理数组的路径向量与距离向量，得到以用户当前所在起点向除起点以外其他点的路径向量与距离向量，鉴于距离向量是存储最短路径中前驱节点的特殊性，恰好与栈的后进先出特性相耦合，因此采用栈的方式不断的将路径向量中的前驱节点循环录入，最终以出栈的方式依次倒序输出，即可显示出目标路径。与Dijsktra算法相似，Floyd算法在对路径与距离向量的处理上增加了一个维度，因此具有更高的灵活性，在处理任意景点的问题上较前者具有更少的时间复杂度。输出路径方式类似，较Dijsktra算法的一维处理方式，只需要将二维路径向量中的第一个维度固定为用户传入的起始点，并进行循环迭代入栈，采用相同方式输出即可。
5. 提供图中任意景点问路查询，即求任意两个景点之间的所有路径。受无向连通图的DFS深度优先遍历算法思想的启发，进行基于起始点的遍历。由于采用递归思想，因此需要单独定义全局计数器，全局状态变量数组，全局路径存储数组。核心思想是：从起始点出发进行递归的深度遍历，如果遇到终点那么全局计时器加一，存储在全局路径数组内，并继续遍历到结束，因此逐层回滚，在处于层数较高的递归函数中，将特定顶点的全局访问状态置零，实现回滚后的新一轮不完全遍历，求出完整的结果。因深度遍历算法的固有缺陷，输出时只需要控制路径的终点是用户指定终点即停止输出，并进行下一路径的新一轮输出，以此类推，获得完整的路径结果。
6. 提供校园图中多个景点的最佳访问路线查询，即求途经这多个景点的最佳路径。多个景点的处理思想是相邻节点两两按照Floyd的要求输出。用户指定多个景点，出于难度和可行性考虑，系统默认为已确定顺序，也就意味着用户想要按照自己规划的顺序制定景点的路径。首先，通过进行格式矫正将用户输入的信息标准化为整形数组存储起来，然后读取数组的长度，也就是景点的个数。读入后，从顺序较后的景点至顺序较前的景点相邻两景点依次进行基于Floyd路径数组与距离数组的调用。采用全局栈的方式，将整个路径全部入栈存储，并最终出栈输出得到目的效果。
7. 判断图的连通性，即求图的关节点，依然采用衍生邻接表的存储结构，无向图深度遍历思想，采用Tarjan算法：对根节点，若其有两棵或以上的子树，则为关节点；对非叶子节点，若其字数的节点均没有指向它子树的回边，说明删除后，根节点与它的子树不再连通，那么该节点为割点。定义全局数组，存储符合条件的这两类点，最后输出即为所有的关节点。
8. **模块分析**

工作任务：

1. 朱继熠：主函数模块，图的深度遍历求关节点，图形交互界面的适配与全局算法优化。
2. 梁亚亚：基于DFS深度遍历的两景点所有路径模块，迭代Floyd求途经多个景点最佳路径模块与控制全局栈的路径输出。
3. 李静涵：Dijkstra与Floyd方法求解两景点最佳路径，路径向量与距离向量创建与输出，景点详细信息的检索与输出。
4. 涂大祥：邻接矩阵与景点信息文档的读取，控制邻接矩阵转邻接表。
5. **数据结构**

存储结构：线性存储

逻辑结构：线性结构与链式结构

链式存储的优点：

1. 空间上  
   顺序比链式节约空间。
2. 存储操作上  
   顺序支持随机存取，方便操作  
   3.插入和删除上  
   顺序表的插入要执行更大的空间复杂度，包括一个从表头索引以及索引后的元素后移，而链表是索引后，插入就完成了

线性结构的优点：

1）同一线性表中元素具有相同特性.  
2）相邻数据元素之间存在序偶关系.  
（即,除第一个元素外,其他每一个元素有且仅有一个直接前驱；除最后一个元素外,其他每一个元素有且仅有一个直接后继.）  
3）元素在线性表中的“下标”唯一地确定该元素在表中的相对位置.

1. **功能模块详细设计**

**5.1 全局变量及结构体**

int st\_All[M];//用于存放多个景点的floyd遍历顺序

int top\_all;//保存递归算法得出栈顶的结果

int visit[M];//用于标记在深度遍历求关节点的访问情况

int dfs\_seq[M];//记录节点u在深度遍历的顺序

int low\_anc[M]; 记录节点u或u的子树通过非父子边追溯到最早的祖先节点（即DFS次序号最小）

int parent[M];//记录节点u的父节点（深度遍历前驱）

int v[Maxi][Maxi];//用于记录递归中所有路径

int visiting[Maxi];//记录所有路径求解过程中节点访问情况

int cnt;//用于记录所有路径的数量

typedef int SerialNum;//

typedef char SpotName;//

typedef char SpotDetail;//景点信息类型

typedef struct

{

SerialNum No;/\*景点序号\*/

SpotName name[NameSpace];/\*景点名称\*/

SpotDetail detail[NameSpace];/\*景点详细信息，介绍\*/

}Spot;/\*景点的详细结构体\*/

typedef Spot SpotMenu;

typedef struct

{

SpotMenu place[NameSpace];

int capacity;/\*景点数量\*/

}SpotList;/\*这是一个存储景点的线性表\*/

typedef struct ANode

{

int adjvex;

struct ANode \*nextarc;

int value;//

}ArcNode;//链表节点

typedef int Vertex;

typedef struct Vnode

{

Vertex data;//头结点序号

ArcNode \*firstarc;//头结点指向的第一个节点

}VNode;//头结点类型

typedef VNode AdjList[M];

typedef struct

{

AdjList adjlist;//头结点数组

int n,e;//节点数量和边数

}ALGraph;//邻接表

typedef char vertextype[M];

typedef int edgetype;

typedef struct

{

int adjvex;

vertextype data;

}VType;//邻接矩阵节点类型

typedef struct

{

VType vexs[M];

edgetype edges[M][M];

int n,e;

}Mgraph;//邻接矩阵类型

5.1.1 详细设计思想

设计中北大学的校园平面图，所含景点不少于10个，以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息，存储在文本文档中，并制定相应的文件读取算法按照格式录入程序。各景点之间的邻接关系与自定义权值同样通过文档读取，为了方便后续的深度遍历求关节点，需要设计一个邻接表转换算法，边的权值作为入度边存放在衍生邻接表子节点的数值域。如图5.1

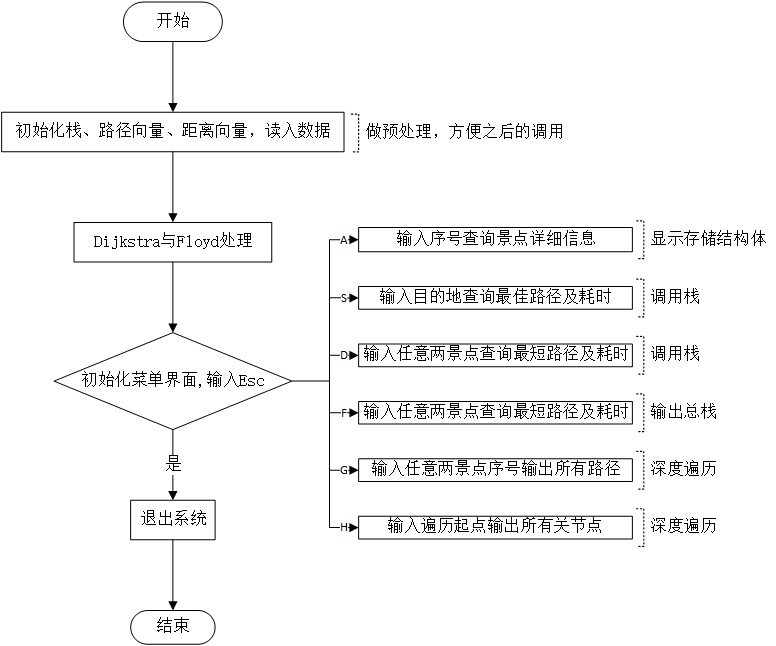
****

图5.1全局变量与结构体图

5.1.2 核心代码

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

#include "Scenespot.h"

#include "Ljjz.h"

#include "EgeTest.h"

#include "Arti.h”

int main()

{

int path\_dij[M];//Dijkstra路径向量

int dist\_dij[M];//Dijkstra距离向量

int path\_flo[M][M];//Floyd路径向量

int dist\_flo[M][M];//Floyd距离向量

int stack\_dij[M];//Dijkstra输出栈

int stack\_flo[M];//Floyd输出栈

int stack\_All[M];//多Floyd输出总栈

SpotList sp;//文件信息结构体

readFile(&sp);//读取景点信息

Mgraph g;//邻接矩阵

ALGraph G;//邻接表

creat(&g,0);//创建邻接矩阵

MatToList(&g,&G);//邻接矩阵转邻接表

dijkstra(g,0,path\_dij,dist\_dij);//Dijkstra处理

floyd(g,path\_flo,dist\_flo);//Floyd处理

char getK;//获取按键常量

int x,y,i,j;

while(getK!=key\_esc)

{

if(getK=='a')

{

char str[NameSpace];

inputbox\_getline("请输入", "序号", str, NameSpace);

x=readnumber(str);

sprintf(str,"%d %s %s",sp.place[x].No,sp.place[x].name,sp.place[x].detail); outtextxy(0,0,str);

}

char str1[NameSpace];

char str2[NameSpace];

inputbox\_getline("请输入", "序号", str1, NameSpace);

x=readnumber(str1);

sprintf(str1,"需要花费时间：%d",dist\_dij[x]);

outtextxy(0,0,str1);

print\_gpd(g,x,path\_dij,dist\_dij,stack\_dij);//获取输出栈

int counter=0,top\_dij=StTop(stack\_dij);//获取栈顶

while(top\_dij>0)

{

sprintf(str2,"%d->",stack\_dij[top\_dij--]);//控制格式

outtextxy(30+(counter++)\*70,30,str2);//输出

}

sprintf(str2,"%d",stack\_dij[top\_dij--]);

outtextxy(30+(counter++)\*70,30,str2);

}

else if(getK=='d')

{

char str1[NameSpace];

char str2[NameSpace];

char str3[NameSpace];

inputbox\_getline("请输入", "序号1", str1, NameSpace);

inputbox\_getline("请输入", "序号2", str2, NameSpace);

x=readnumber(str1);//读入用户数据

y=readnumber(str2);

sprintf(str1,"需要花费时间：%d",dist\_flo[x][y]);//距离

outtextxy(0,0,str1);//输出到屏幕上

print\_floyd(g,x,y,path\_flo,dist\_flo,stack\_flo);//得到Floyd输出栈

int counter=0,top\_flo=StTop(stack\_flo);

while(top\_flo>0)

{

sprintf(str3,"%d->",stack\_flo[top\_flo--]);

outtextxy(30+(counter++)\*70,30,str3);

}

sprintf(str3,"%d",stack\_flo[top\_flo--]);

outtextxy(30+(counter++)\*70,30,str3);

}

else if(getK=='f')

{

char str1[NameSpace];

char str2[NameSpace];

int iti[NameSpace];

int AllNum;

int top\_All;

int counter=0;

inputbox\_getline("请输入", "空格隔开", str1, NameSpace);

AllNum=LongNum(str1,iti);

sprintf(str1,"需要花费时间：%d",OverFloyd(g,iti,AllNum,path\_flo,dist\_flo,stack\_All));//处理获取总栈

outtextxy(0,0,str1);

top\_All=StTop(stack\_All);

printf("%5d",top\_All);

while(top\_All>0)

{

sprintf(str2,"%d->",stack\_All[top\_All--]);

outtextxy(30+(counter++)\*70,30,str2);

}

sprintf(str2,"%d",stack\_All[top\_All]);

outtextxy(30+(counter++)\*70,30,str2);

stack\_All[0]='\0';/\*全局变量初始化\*/

}

else if(getK=='h')

{

char str1[NameSpace];

char str2[NameSpace];

int x;

i=0;

inputbox\_getline("你选择从哪里遍历关节点", "回车确认", str2, NameSpace);

x=readnumber(str2);

arti(&G,x);

while(i<anti\_cnt)

{

sprintf(str1,"%d",anti\_array[i++]);

outtextxy(30+(i)\*70,30,str1);

}

anti\_array[0]='\0';

}

else if(getK=='g')

{

char str1[NameSpace];

char str2[NameSpace];

char str3[NameSpace];

inputbox\_getline("请输入", "序号1", str1, NameSpace);

inputbox\_getline("请输入", "序号2", str2, NameSpace);

x=readnumber(str1);

y=readnumber(str2);

SearchAllpath(g,x,y);

i=0;j=0;

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

{

for(j=0;v[i][j]!=y;j++)

{

if(10+i\*20<750)

{

sprintf(str3,"%d->",v[i][j]);

outtextxy(10+j\*40,10+i\*20,str3);

}

else

{

sprintf(str3,"%d->",v[i][j]);

outtextxy(693+j\*40,i\*20-730,str3);

}

}

if(10+i\*20<750)

{

sprintf(str3,"%d",v[i][j]);

outtextxy(10+j\*40,10+i\*20,str3);

}

else

{

sprintf(str3,"%d",v[i][j]);

outtextxy(693+j\*40,i\*20-730,str3);

}

}

}

getK=getch();

delimage(NUC);

cleardevice();

return 0;

}

**5.2 图的建立模块**

### 5.2.1 详细设计思想

## 图的建立主要是通过指针打开文件，然后用循环从文件中依次读取顶点数，边数，顶点，边，最后建立无向图邻接矩阵。如图5.2



图5.2无向邻接矩阵图

景点信息的读入主要是用文件指针打开文件，然后利用循环依次读取景点的序号，名称，以及景点的介绍及详细信息，最后关闭文件。如图5.2



图5.2读取景点信息图

### 5.2.2 核心代码

### void creat(Mgraph \*g, int c)

### {

### int i,j,k,w;//定义整形变量i,j,k,w（声明变量）

### FILE \*rf;//定义一个返回FILE结构体的指针rf

### rf=fopen("Ljjz.txt","r");//打开文件Ljjz.txt,打开方式为只读

### if(rf)

### {

### fscanf(rf,"%d%d\n",&g->n,&g->e);//读入顶点数，边数

### for(i=0;i<g->n;i++)//i<g->n:i小于取出的g所指向的结构体中包含的数据n

### {

### fscanf(rf,"%d",&g->vexs[i].adjvex);//读入顶点值

### }

### for(i=0;i<g->n;i++)//初始化邻接矩阵

### for(j=0;j<g->n;j++)

### if(i==j)

### g->edges[i][j]=0;

### else g->edges[i][j]=FINITY;

### for(k=0;k<g->e;k++)//读入边

### {

### fscanf(rf,"%d%d%d",&i,&j,&w);

### g->edges[i][j]=w;

### if(c==0) g->edges[j][i]=w；//建立无向图邻接矩阵

### }

### }

### else g->n=0;

### }

### void readFile(SpotList \*sp)

### {

### FILE \*rf;

### if((rf=fopen("info.txt","r"))==NULL)

### printf("NONE");/\*读取文件失败了！！！\*/

### else

### {

### fscanf(rf,"%d",&sp->capacity);/\*首先读取景点数量\*/

### int i=0;

### while(i<sp->capacity)/\*根据上面读到的大小\*/

### {

### fscanf(rf,"%d\n",&sp->place[i].No);

### fgets(sp->place[i].name,NameSpace,rf);

### fgets(sp->place[i].detail,NameSpace,rf);

### i++;

### }

### fclose(rf);

### }

### }

## 5.3 邻接矩阵转置邻接表模块

### 5.3.1 详细设计思想

## 邻接矩阵转置邻接表主要是首先要给邻接表中所有头结点置初值，然后在邻接表中存在元素的情况下，利用头插法，将新建立的节点插入到链表中。如图5.3



图5.3邻接矩阵转置邻接表图

### 5.3.2 核心代码

void MatToList(Mgraph \*g, ALGraph \*G)

{

int i,j;

ArcNode \*p;

for(i=0;i<g->n;i++)//给邻接表中所有头结点的指针置初值

G->adjlist[i].firstarc=NULL;

for(i=0;i<g->n;i++)//检查邻接矩阵中每个元素

for(j=g->n-1;j>=0;j--)

if((g->edges[i][j]!=0)&&(g->edges[i][j]!=FINITY))//存在一条边（邻接矩阵的当前元素不为0）

{

p=(ArcNode\*)malloc(sizeof(ArcNode));//创建一个节点\*p

p->adjvex=j;

p->value=g->edges[i][j];

p->nextarc=G->adjlist[i].firstarc;/\*头插法\*///采用头插法插入\*p

G->adjlist[i].firstarc=p;

}

G->n=g->n;

G->e=g->e;

}

5.4  两景点最短路径模块

5.4.1 详细设计思想

用dijstra算法算单源最短路径的算法思想为：令G = （V，E）为一个带权有向网，把图中的顶点集合V分成两组：已求出最短路径的顶点集合S（初始时S中只有源节点，以后每求得一条最短路径，就将它对应的顶点加入到集合S中，直到全部顶点都加入到S中）；未确定最短路径的顶点集合V-S。在加入过程中，总保持从源节点v到S中各顶点的最短路径长度不大于从源节点v到V-S中任何顶点的最短路径长度。如图5.4

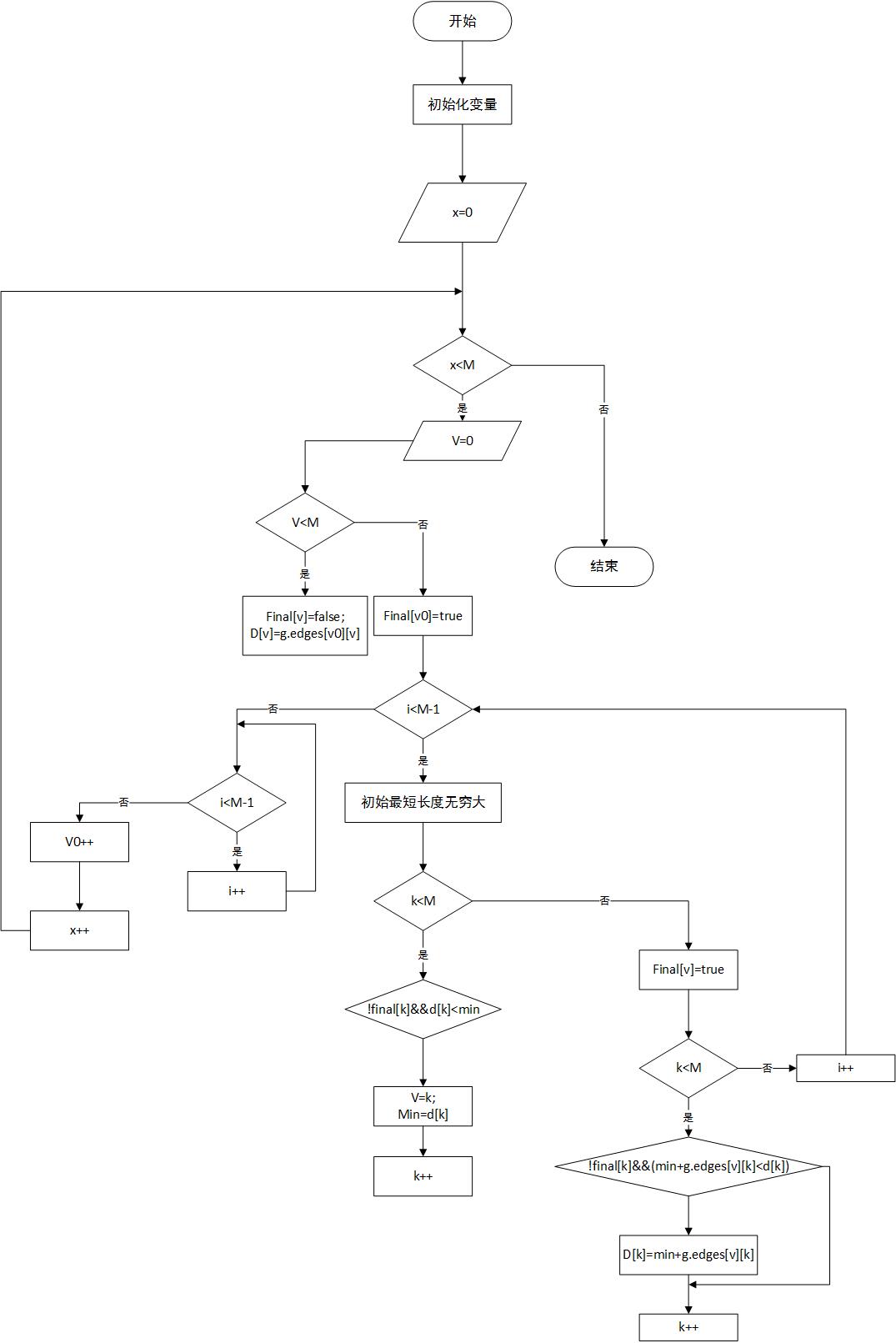


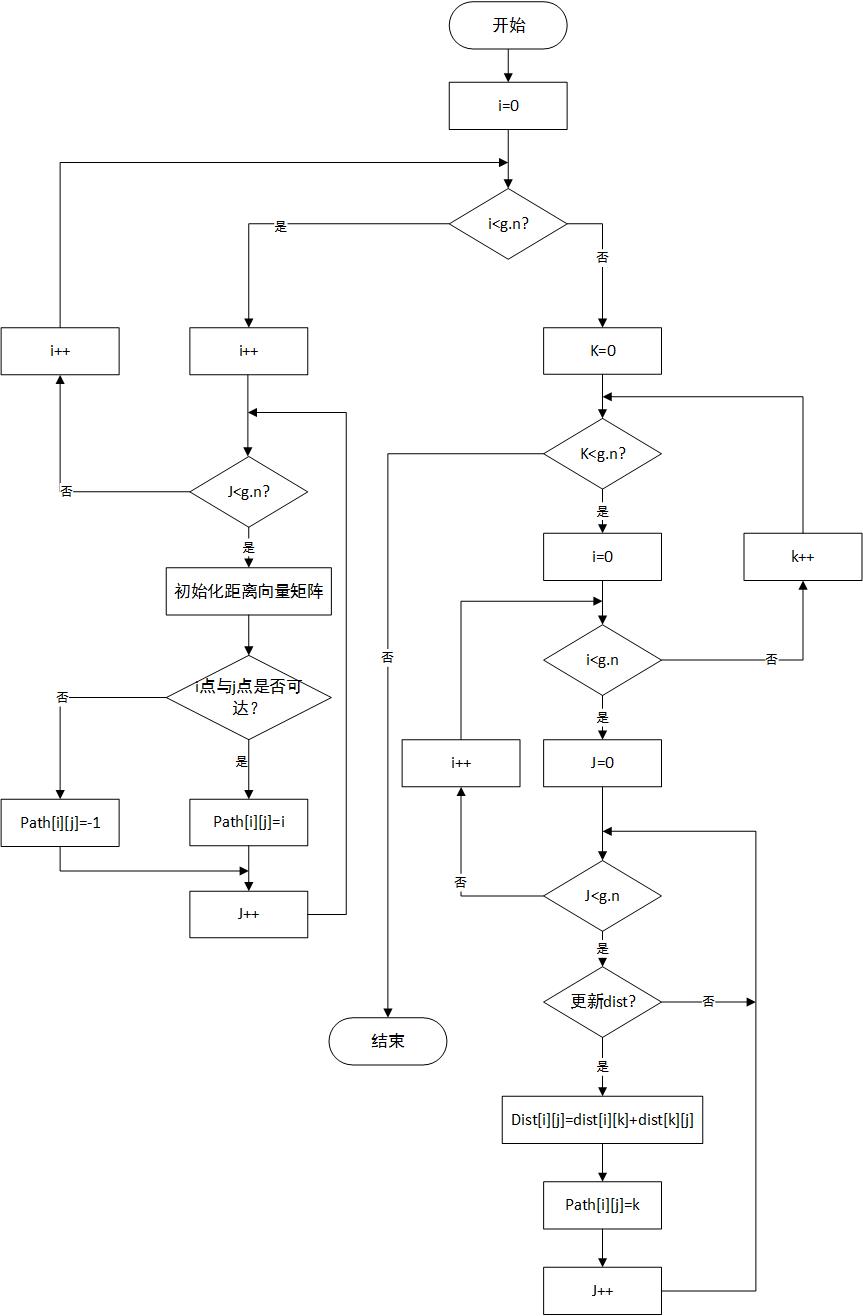
图5.4用dijstra求最短路径图

用floyed算法通过一个图的权值矩阵求出它的每两点间的最短路径矩阵。具体算法思想如下：

1．从任意一条单边路径开始。所有两点之间的耗时是边的权，如果两点之间没有边相连，则权为无穷大。

2．对于每一对顶点u和v，看看是否存在一个顶点w使得比从u到w到v比已知路径短。如果是就更新它。

3．把图用邻接矩阵G表示出来如果从Vi到Vj有路可达，则G[i][j]=d，d表示该路的长度；否则G[i][j]=无穷大。定义一个矩阵D用来记录所插入点的信息，D[i][j]表示从vi到vj需要经过的点，初始化D[i][j]=j。把各个顶点插入图中，比较插点后的距离与原来的距离，G[i][j]=min(G[i][j],G[i][k]+G[k][j]),如果G[i][j]的值变小，则D[i][j]=k.在G中包含有两点之间最短道路的信息，而在D中则包含了最短通路径的信息。如图5.4.1.2



5.4用Floyd求最短路径图

### 5.4.2 核心代码

typedef int path\_d[M];

typedef int dist\_d[M];

typedef int path\_f[M][M];

typedef int dist\_f[M][M];

typedef int st\_d[M];

typedef int st\_f[M];

void floyd(Mgraph g,path\_f p,dist\_f d)

{

int i,j,k;

for(i=0;i<g.n;i++)/\*初始化矩阵\*/

for(j=0;j<g.n;j++)

{

d[i][j]=g.edges[i][j];

if(i!=j&&d[i][j]<FINITY)

p[i][j]=i;

else p[i][j]=-1;

}

for(k=0;k<g.n;k++)

{

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

for(j=0;j<g.n;j++)

if(d[i][j]>d[i][k]+d[k][j])/\*新找到一条更短路径则更新\*/

{

d[i][j]=d[i][k]+d[k][j];/\*取小的值\*/

p[i][j]=k;/\*改变路径\*/

}

}

}

void dijkstra(Mgraph g,int v0,path\_d p,dist\_d d)

{

boolean final[M];

int i,k,v,min;

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

{

final[v]=FALSE;/\*全部顶点初始化为未知最短路径状态\*/

d[v]=g.edges[v0][v];/\*找到与v0有连线的顶点加上权值\*/

if(d[v]<FINITY&&d[v]!=0)

p[v]=v0;else p[v]=-1;

}

final[v0]=TRUE;d[v0]=0;

for(i=1;i<g.n;i++)

{

min=FINITY;

for(k=0;k<g.n;++k)/\*寻找V-S中离v0最近的顶点\*/

if(!final[k]&&d[k]<min)

{

v=k;min=d[k];/\*下标为k的顶点离v0更近\*/

}

// printf("\n%c---%d\n",g.vexs[v],min);

if(min==FINITY) return;

final[v]=TRUE;/\*将下标为k的顶点加入集合S，当前找到的最短路径的顶点标记为true\*/

for(k=0;k<g.n;++k)/\*若经过v0顶点的路径比当前路径的长度短\*/

if(!final[k]&&(min+g.edges[v][k]<d[k]))

{

d[k]=min+g.edges[v][k];

p[k]=v;

}

}

}

void print\_gpd(Mgraph g,int fn,path\_d p,dist\_d d,st\_d s)

{

int pre,top=-1;/\*定义一个栈st并初始化空栈\*/

// printf("%5d",d[fn]);

s[++top]=fn;

pre=p[fn];

while(pre!=-1)

{

s[++top]=pre;

pre=p[pre];

}

// while(top>=0)/\*从第i个顶点开始向前搜索最短路径上的顶点\*/

// printf("%5d",s[top--]);

}

void print\_floyd(Mgraph g,int bgn,int fnl,path\_f p,dist\_f d, st\_f s)

{

int pre,top=-1;

// printf("%5d",d[bgn][fnl]);

s[++top]=fnl;

pre=p[bgn][fnl];

while(pre!=-1)

{

s[++top]=pre;

pre=p[bgn][pre];

}

// while(top>=0)

// printf("%5d",s[top--]);

}

5.5 两景点所有路径模块

5.5.1 详细设计思想

提供图中任意景点问路查询，即求任意两个景点之间的所有路径。受无向连通图的DFS深度优先遍历算法思想的启发，进行基于起始点的遍历。由于采用递归思想，因此需要单独定义全局计数器，全局状态变量数组，全局路径存储数组。核心思想是：从起始点出发进行递归的深度遍历，如果遇到终点那么全局计时器加一，存储在全局路径数组内，并继续遍历到结束，因此逐层回滚，在处于层数较高的递归函数中，将特定顶点的全局访问状态置零，实现回滚后的新一轮不完全遍历，求出完整的结果。因深度遍历算法的固有缺陷，输出时只需要控制路径的终点是用户指定终点即停止输出，并进行下一路径的新一轮输出，以此类推，获得完整的路径结果。如图5.5

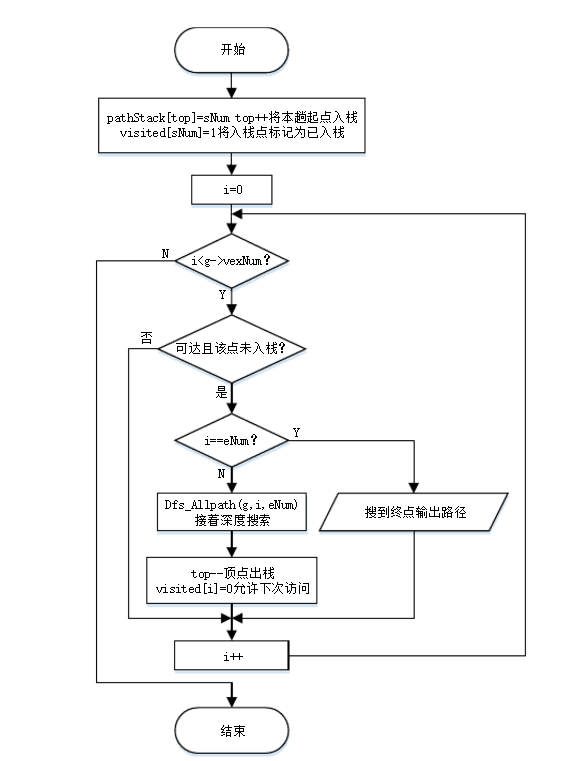


图5.5求所有路径图

### 5.5.2 核心代码

int v[Maxi][Maxi],visiting[Maxi];

int cnt; //计数器

void pathl(Mgraph \*g,int i,int j,int k) /\*确定路径上第k+1个顶点的序号\*/

{

int s;

if(v[cnt][k]==j)/\*找到一条路径\*/

{

cnt++;

for(s=0;s<k;s++)/\*输出一条路径\*/

{

v[cnt][s]=g->vexs[v[cnt-1][s]].adjvex;

}

v[cnt][s]=g->vexs[v[cnt-1][s]].adjvex;

/\*路径的条数值加1\*/

}

s=0;

while(s<g->n)

{

if(s!=i)/\*保证找到的是简单路径\*/

{

if(g->edges[v[cnt][k]][s]!=FINITY&&visiting[s]==0) /\*当vk与vs之间有边存在且vs未被访问过\*/

{

visiting[s]=1;/\*置访问标志位为1,即已访问的\*/

v[cnt][k+1]=s;/\*将顶点s加入到v数组中\*/

pathl(g,i,j,k+1);/\*递归调用之\*/

visiting[s]=0;/\*重置访问标志位为0，即未访问的，以便该顶点能被重新使用\*/

}

}

s++;

}

}

void SearchAllpath(Mgraph g,int i,int j)/\*查询两个景点间的所有路径\*/

{

int k;

v[0][0]=i;

cnt=0;/\*初始化路径的条数\*/

for(k=0;k<g.n;k++)

{

visiting[k]=0;/\*初始化各顶点的访问标志位，即都为未访问过的\*/

}

pathl(&g,i,j,0);/\*通过调用path函数，找到从vi到vj的所有路径并输出\*//\*用来输出两个景点间的所有路径\*/

}

## 5.6 多景点最佳路径模块

5.6.1 详细设计思想

提供校园图中多个景点的最佳访问路线查询，即求途经这多个景点的最佳路径。多个景点的处理思想是相邻节点两两按照Floyd的要求输出。用户指定多个景点，出于难度和可行性考虑，系统默认为已确定顺序，也就意味着用户想要按照自己规划的顺序制定景点的路径。首先，通过进行格式矫正将用户输入的信息标准化为整形数组存储起来，然后读取数组的长度，也就是景点的个数。读入后，从顺序较后的景点至顺序较前的景点相邻两景点依次进行基于Floyd路径数组与距离数组的调用。采用全局栈的方式，将整个路径全部入栈存储，并最终出栈输出得到目的效果。如图5.6

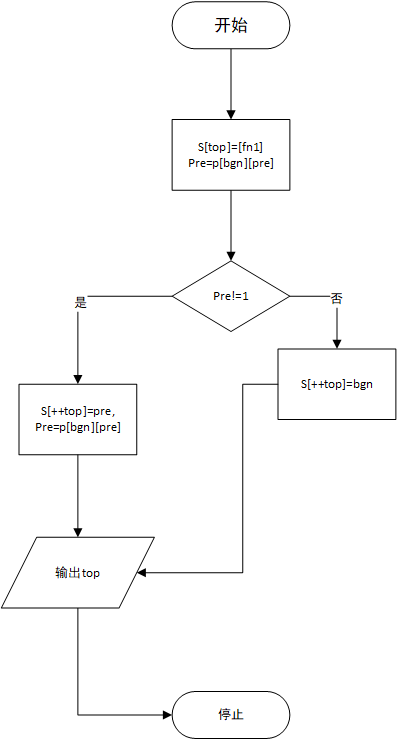
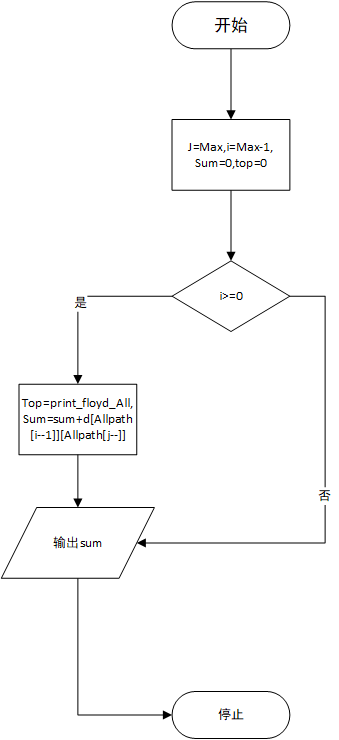


图5.6求最佳路径图

### 5.6.2 核心代码

int st\_All[M];

int top\_all;

int print\_floyd\_All(Mgraph g,int bgn,int fnl,path\_f p,dist\_f d,int top,st\_f s)

{

int pre;

s[top]=fnl;

pre=p[bgn][fnl];

while(pre!=-1)

{

s[++top]=pre;

pre=p[bgn][pre];

}

s[++top]=bgn;

return top;

}

int OverFloyd(Mgraph g,int AllPath[M],int Max,path\_f p,dist\_f d,st\_f s)

{

int j=Max;

int i=Max-1;

int sum=0,top=0;

while(i>=0)

{

top=print\_floyd\_All(g,AllPath[i],AllPath[j],p,d,top,s);

sum=sum+d[AllPath[i--]][AllPath[j--]];

}

return sum;

}

**5.****7 图的关节点模块**

5.7.1 详细设计思想

判断图的连通性，即求图的关节点，依然采用衍生邻接表的存储结构，无向图深度遍历思想，采用Tarjan算法：对根节点，若其有两棵或以上的子树，则为关节点；对非叶子节点，若其字数的节点均没有指向它子树的回边，说明删除后，根节点与它的子树不再连通，那么该节点为割点。定义全局数组，存储符合条件的这两类点，最后输出即为所有的关节点。如图5.7

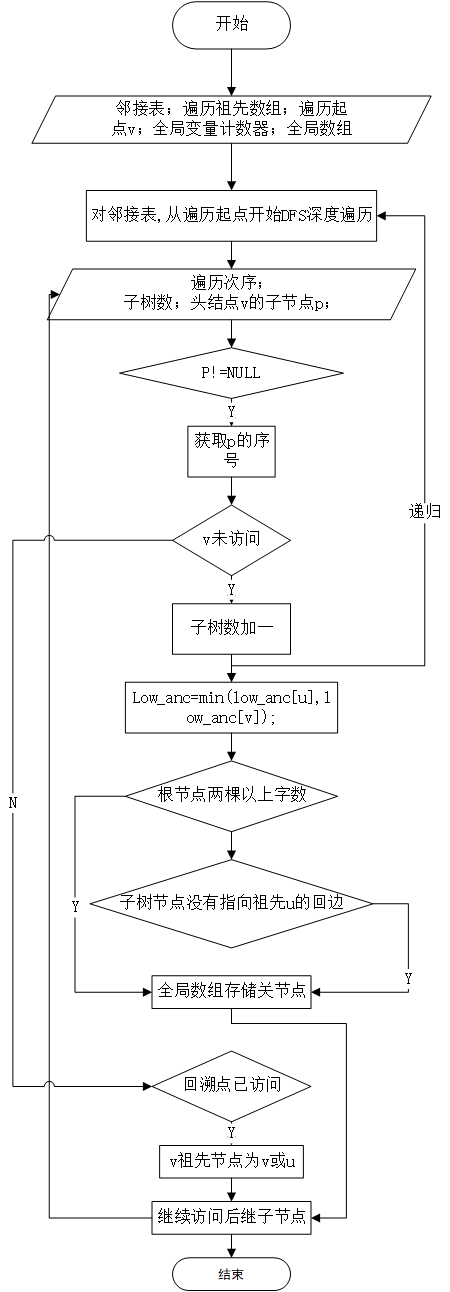
****

图5.7图的关节点图

### 5.7.2 核心代码

void dfs\_sqc(ALGraph \*g,int u)/\*第二个参数选择遍历起点\*/

{

int v;

//记录dfs遍历次序

//记录节点u的子树数

int children = 0;

ArcNode \*p = g->adjlist[u].firstarc;

visit[u] = 1;

//初始化dfn与low

dfs\_seq[u]=low\_anc[u] = counter++;

for(; p != NULL; p = p->nextarc)

{

v = p->adjvex;

//节点v未被访问，则(u,v)为树边

if(!visit[v])

{

children++;

parent[v] = u;

dfs\_sqc(g,v);

low\_anc[u] = min(low\_anc[u], low\_anc[v]);

//case (1)

if(parent[u] == -1 && children > 1)

{

anti\_array[anti\_cnt++]=u;

}

//case (2)

if(parent[u] != -1 && low\_anc[v] >= dfs\_seq[u])

{

anti\_array[anti\_cnt++]=u;

}

}

//节点v已访问，则(u,v)为回边

else if(v != parent[u])

{

low\_anc[u] = min(low\_anc[u], dfs\_seq[v]);

}

}

}

void arti(ALGraph\* g,int v)

{

// int i;

parent[0]=-1;

dfs\_sqc(g,v);

}

**5.8 主函数模块**

5.8.1 详细设计思想

设计中北大学的校园平面图，所含景点不少于10个，以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息，存储在文本文档中，并制定相应的文件读取算法按照格式录入程序。各景点之间的邻接关系与自定义权值同样通过文档读取，为了方便后续的深度遍历求关节点，需要设计一个邻接表转换算法，边的权值作为入度边存放在衍生邻接表子节点的数值域。如图5.8

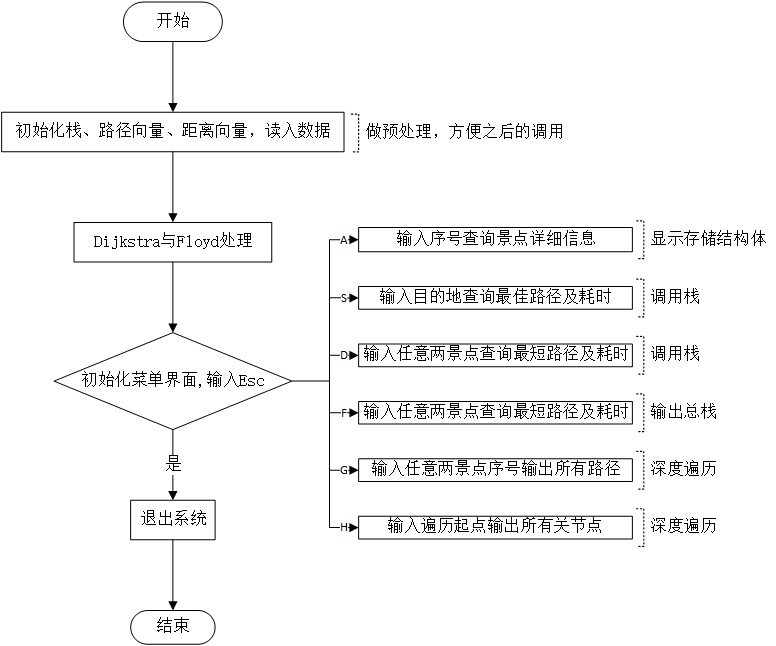
****

图5.8主函数图

5.8.2 核心代码

# #include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

#include "Scenespot.h"

#include "Ljjz.h"

#include "EgeTest.h"

#include "Arti.h"

int main()

{

int path\_dij[M];

int dist\_dij[M];

int path\_flo[M][M];

int dist\_flo[M][M];

int stack\_dij[M];

int stack\_flo[M];

int stack\_All[M];

SpotList sp;

readFile(&sp);

Mgraph g;

ALGraph G;

creat(&g,0);

MatToList(&g,&G);

dijkstra(g,0,path\_dij,dist\_dij);

floyd(g,path\_flo,dist\_flo);

setinitmode(0,0,0);

initgraph(1366, 768);

setcolor(EGERGB(0x0, 0xFF, 0x0));

setfont(80, 0, "黑体");

setbkmode(TRANSPARENT);

outtextxy(523, 150, "中北大学");

setcolor(EGERGB(0xff, 0x0, 0xff));

setfont(60, 0, "黑体");

outtextxy(503, 400, "校园导游系统");

setcolor(EGERGB(0x40, 0xe0, 0xd0));

setfont(40, 0, "黑体");

outtextxy(563, 680, "按任意键继续");

getch();

cleardevice();

char getK;

int x,y,i,j;

while(getK!=key\_esc)

{

PIMAGE NUC = newimage();

getimage(NUC, "NUC.jpg");

putimage(10, 10, NUC);

setcolor(EGERGB(0xEE,0x82,0xEE));

setfont(40, 0, "黑体");

char str[NameSpace];

for(i=0;i<sp.capacity/2;i++)

{

sprintf(str,"%d %s",sp.place[i].No,sp.place[i].name);

outtextxy(700, 20+i\*40, str);

}

for(;i<sp.capacity;i++)

{

sprintf(str,"%d %s",sp.place[i].No,sp.place[i].name);

outtextxy(900, 20+(i-sp.capacity/2)\*40, str);

}

setcolor(EGERGB(0x0,0xFF,0x0));

setfont(20, 0, "宋体");

outtextxy(1200, 720, "按Esc退出");

i=0;

setcolor(EGERGB(0x00, 0xFF, 0xFF));

setfont(40, 0, "微软雅黑");

setbkmode(TRANSPARENT);

outtextxy(200, 500, "A: 查询景点详细信息");

outtextxy(650, 500, "S: 查询当前位置到任意景点路径及耗时");

outtextxy(200, 550, "D: 任意两景点路径及耗时");

outtextxy(650, 550, "F: 定制路径及耗时");

outtextxy(200, 600, "G: 两个景点所有路径查询");

outtextxy(650, 600, "H: 看一看关节点");

getK=getch();

if(getK=='a')

{

setviewport(30, 650, 1300, 700, 1);

char str[NameSpace];

inputbox\_getline("请输入", "序号", str, NameSpace);

x=readnumber(str);

sprintf(str,"%d %s %s",sp.place[x].No,sp.place[x].name,sp.place[x].detail);

setcolor(EGERGB(0x87,0xCE,0xFA));

setfont(30, 0, "宋体");

outtextxy(0,0,str);

setviewport(0, 0, getwidth(), getheight(), 1);

}

else if(getK=='s')

{

setviewport(50, 650, 1300, 750, 1);

char str1[NameSpace];

char str2[NameSpace];

inputbox\_getline("请输入", "序号", str1, NameSpace);

x=readnumber(str1);

sprintf(str1,"需要花费时间：%d",dist\_dij[x]);

setcolor(EGERGB(0x87,0xCE,0xFA));

setfont(30, 0, "宋体");

outtextxy(0,0,str1);

print\_gpd(g,x,path\_dij,dist\_dij,stack\_dij);

int counter=0,top\_dij=StTop(stack\_dij);

while(top\_dij>0)

{

sprintf(str2,"%d->",stack\_dij[top\_dij--]);

outtextxy(30+(counter++)\*70,30,str2);

}

sprintf(str2,"%d",stack\_dij[top\_dij--]);

outtextxy(30+(counter++)\*70,30,str2);

setviewport(0, 0, getwidth(), getheight(), 1);

}

else if(getK=='d')

{

setviewport(50, 650, 1300, 750, 1);

char str1[NameSpace];

char str2[NameSpace];

char str3[NameSpace];

inputbox\_getline("请输入", "序号1", str1, NameSpace);

inputbox\_getline("请输入", "序号2", str2, NameSpace);

x=readnumber(str1);

y=readnumber(str2);

sprintf(str1,"需要花费时间：%d",dist\_flo[x][y]);

setcolor(EGERGB(0x87,0xCE,0xFA));

setfont(30, 0, "宋体");

outtextxy(0,0,str1);

print\_floyd(g,x,y,path\_flo,dist\_flo,stack\_flo);

int counter=0,top\_flo=StTop(stack\_flo);

while(top\_flo>0)

{

sprintf(str3,"%d->",stack\_flo[top\_flo--]);

outtextxy(30+(counter++)\*70,30,str3);

}

sprintf(str3,"%d",stack\_flo[top\_flo--]);

outtextxy(30+(counter++)\*70,30,str3);

setviewport(0, 0, getwidth(), getheight(), 1);

}

else if(getK=='f')

{

setviewport(50, 650, 1300, 750, 1);

char str1[NameSpace];

char str2[NameSpace];

int iti[NameSpace];

int AllNum;

int top\_All;

int counter=0;

inputbox\_getline("请输入", "空格隔开", str1, NameSpace);

AllNum=LongNum(str1,iti);

sprintf(str1,"需要花费时间：%d",OverFloyd(g,iti,AllNum,path\_flo,dist\_flo,stack\_All));

setcolor(EGERGB(0x87,0xCE,0xFA));

setfont(30, 0, "宋体");

outtextxy(0,0,str1);

top\_All=StTop(stack\_All);

printf("%5d",top\_All);

while(top\_All>0)

{

sprintf(str2,"%d->",stack\_All[top\_All--]);

outtextxy(30+(counter++)\*70,30,str2);

}

sprintf(str2,"%d",stack\_All[top\_All]);

outtextxy(30+(counter++)\*70,30,str2);

stack\_All[0]='\0';/\*全局变量初始化\*/

setviewport(0, 0, getwidth(), getheight(), 1);

}

else if(getK=='h')

{

setviewport(50, 650, 1300, 750, 1);

char str1[NameSpace];

char str2[NameSpace];

int x;

i=0;

inputbox\_getline("你选择从哪里遍历关节点", "回车确认", str2, NameSpace);

x=readnumber(str2);

arti(&G,x);

setcolor(EGERGB(0x87,0xCE,0xFA));

setfont(30, 0, "宋体");

while(i<anti\_cnt)

{

sprintf(str1,"%d",anti\_array[i++]);

outtextxy(30+(i)\*70,30,str1);

}

anti\_array[0]='\0';

setviewport(0, 0, getwidth(), getheight(), 1);

}

else if(getK=='g')

{

char str1[NameSpace];

char str2[NameSpace];

char str3[NameSpace];

inputbox\_getline("请输入", "序号1", str1, NameSpace);

inputbox\_getline("请输入", "序号2", str2, NameSpace);

x=readnumber(str1);

y=readnumber(str2);

cleardevice();

SearchAllpath(g,x,y);

setcolor(EGERGB(0x87,0xCE,0xFA));

setfont(15, 0, "黑体");

i=0;j=0;

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

{

for(j=0;v[i][j]!=y;j++)

{

if(10+i\*20<750)

{

sprintf(str3,"%d->",v[i][j]);

outtextxy(10+j\*40,10+i\*20,str3);

}

else

{

sprintf(str3,"%d->",v[i][j]);

outtextxy(693+j\*40,i\*20-730,str3);

}

}

if(10+i\*20<750)

{

sprintf(str3,"%d",v[i][j]);

outtextxy(10+j\*40,10+i\*20,str3);

}

else

{

sprintf(str3,"%d",v[i][j]);

outtextxy(693+j\*40,i\*20-730,str3);

}

}

}

getK=getch();

delimage(NUC);

cleardevice();

}

setcolor(EGERGB(0x0, 0xFF, 0x0));

setfont(80, 0, "黑体");

setbkmode(TRANSPARENT);

outtextxy(523, 150, "感谢使用");

setcolor(EGERGB(0x0, 0xFF, 0xFF));

setfont(50, 0, "黑体");

outtextxy(450, 400, "组长： 朱继熠");

outtextxy(450, 480, "组员： 梁亚亚");

outtextxy(450, 560, "组员： 李静涵");

outtextxy(450, 640, "组员： 涂大祥");

getch();

closegraph();

return 0;

}

1. **运行结果**

****

图6.1 界面图

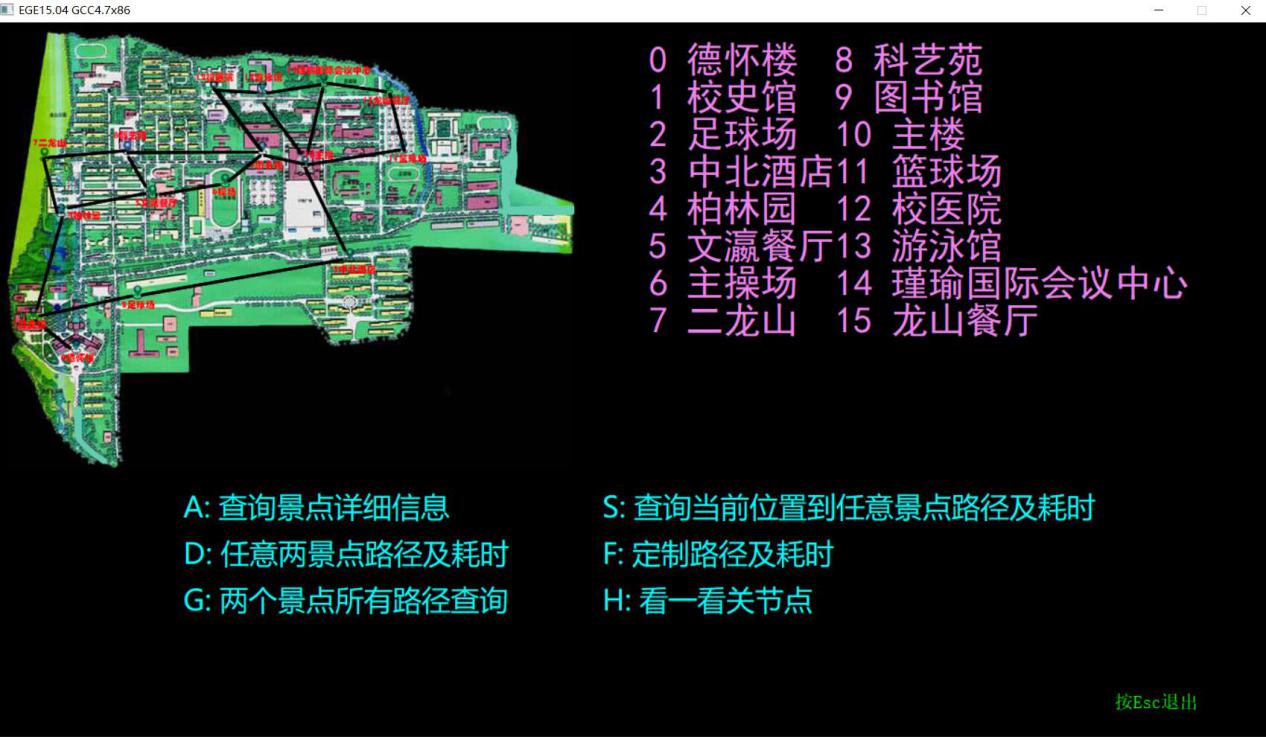
****

图6.2 菜单图

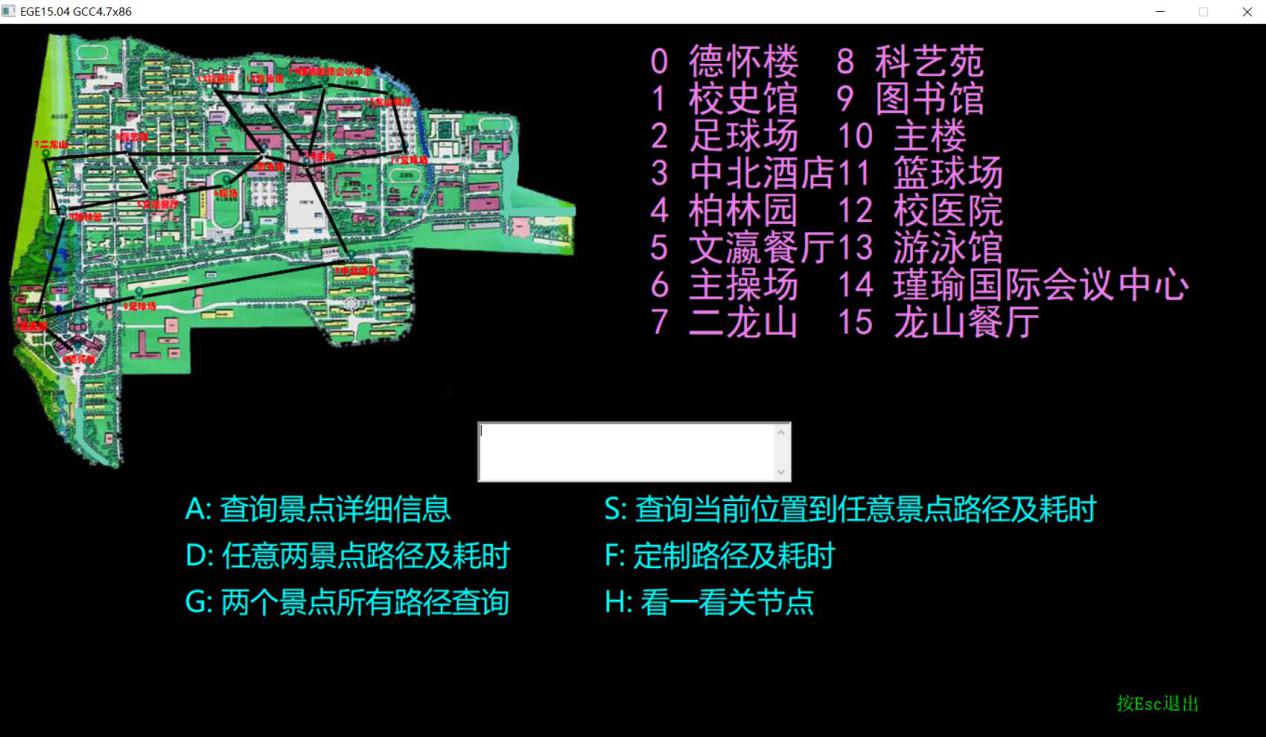
****

图6.3 操作图

****

图6.4 单源最短距离图

****

图6.5 两景点最短路径图

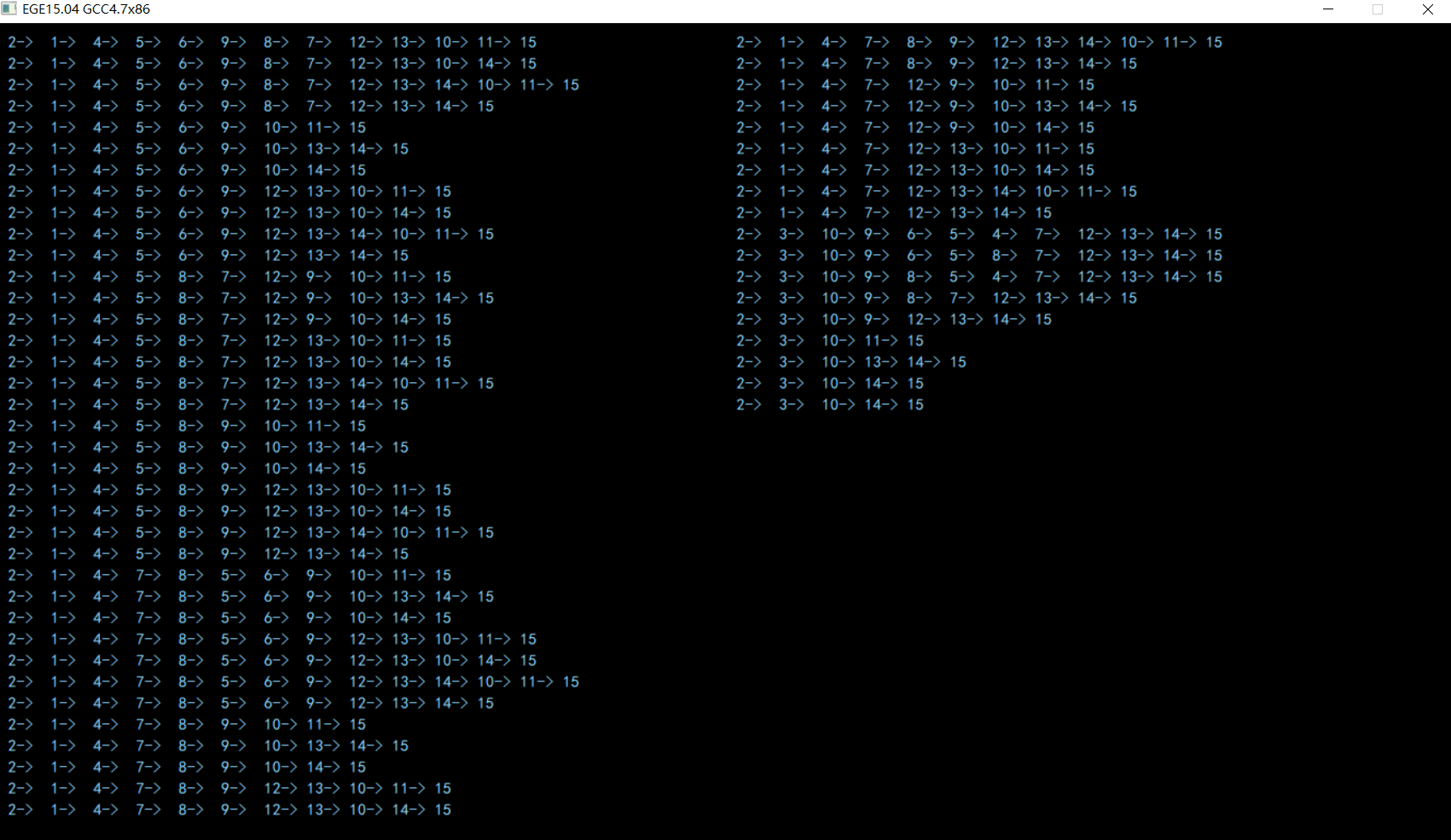
****

图6.6 两景点所有路径图

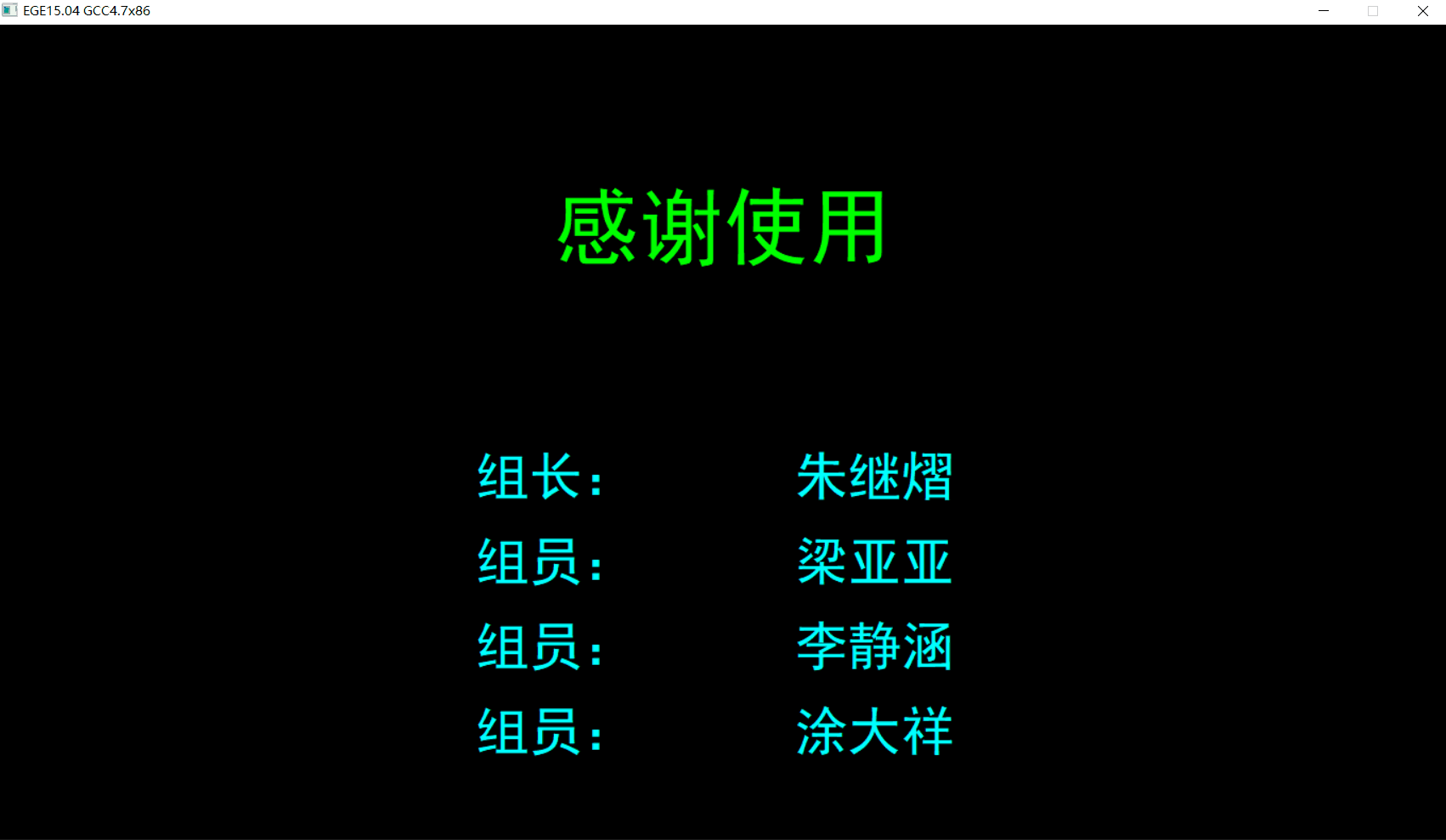
****

图6.7 结束图

1. **总结**

为期两周的数据结构课程设计在紧张刺激中结束了，在这次课程设计中，我们小组中的每一个成员都取得了巨大的进步，并且收获了大量的感悟，相信通过此次课设，每个人都能提升自己在这一方面的能力。  
当然，此次课程设计中，我们小组也遇到了不少的问题，首先，便是小组成员分工的问题。每一个小组中，总有能力高低之分，对于算法，程序等的理解程度也不尽相同，因此，如何合理的安排分工，是小组开始课程设计的第一难题。如果安排不合理，轻则拖沓课设进度，拉慢节奏，而重则可能无法按时完成课设，影响成绩。因此，在紧张的讨论与交流后，以及再综合组员的能力及擅长之后，我们小组做出了相对合理的分工，虽然可能人存在一些不足，但是相比毫无章法，没有组织的进行课设，确实增进不少。对于组内能力较弱，基础 较差的同学，我们选择让其完成一些难度相对较低，但仍具有思考价值的内容，而对于比较熟悉的同学，则大多进行稍微具备难度的工作，总的来说，这样的分配使整个课设项目进行的较为顺利。  
课设的过程中，遇到的问题不在少数，无论是部分同学能力的不足导致的课设进展缓慢，例如一个打开文件或者是头插法的指针，可能对于能力较强的同学来说，很快就可以搞定，而对于基础不是很好的同学，则需要大量的时间去理解概念，然后尝试去运用。又或者是某些不注意的bug或是错误拖拉节奏，可以说，小组内每天都要面临各种各样的难题，当然最大的难题，还是每天早上都要早早的起床，在凛冽的冬风里走向实验室。作为小组内可以说是能力最为突出的人，组长可以说是最忙碌的人了，每天都要不停的帮助组员们解决各种各样的难题，与此同时还要将自己手里的任务完成，工作量十分大。但是组长有时也会忙不过来，因此有些时候有些问题就需要组员自己去解决，但是由于一个班级内不同的小组所做的课设也不同，因此很多时候都需要组员自己在网上或者图书馆找寻资料，来完善自己不懂的或者是不足的地方，这无疑又拖慢了课设的进度。虽然过程十分艰辛，但最后在全体组员的努力下，课程设计还是按时完成了任务。  
此次课设中，我们小组也存在着一些不足，比如缺少了一些创新，没有把自己学到的知识灵活的运用，为自己的课题增添光彩。除此之外，课程设计书格式也存在着不足，有很多需要改进的地方，然后就是可能有些组员在课程设计的过程中存在着一点点的懒惰行为，但在组长和组员的监督下，整个小组内的风气总的来说还是非常良好的。  
在这次数据结构课程设计中，我们小组的每个人都学到了不少的知识，对于我们来说，这次课设不仅是一次作业，更是一次历练和挑战，也希望在将来的课程设计里，同学们能够取得更大的进展。  
最后希望所有的同学课程设计顺利完成。

# 参考文献

[1] 李云清，杨庆红.数据结构（C语言版）.北京：人民邮电出版社，2004.

[2] 严蔚敏,吴伟民.数据结构（C语言版）.北京：清华大学出版.1997.

[3] 苏光奎,李春葆.数据结构导学.北京:清华大学出版.2002.

[4] 周海英，马巧梅，靳雁霞.数据结构与算法设计.北京：国防工业出版社，2007.

[5] 张海藩. 软件工程导论. 北京：清华大学出版社.2003.