

**课 程 设 计 报 告 书**

**实验课程名称 《程序设计课程设计》**

**开 课 学 院 网络空间安全学院**

**指导老师姓名 郑秋华**

**小组成员姓名学号：**

**吴开洧 18041831**

**王邦耀 18201128**

**蔡祖润 18011608**

**2019年12月12日**

选题名称：景区旅游管理系统

1. 需求分析

景区旅游信息管理系统的需求背景：在旅游景区，经常会遇到游客打听从一个景点到另一个景点的最短路径和最短距离，这类游客不喜欢按照导游图的线路来游览，而是挑选自己感兴趣的景点游览。

所以景区旅游管理系统包括如下功能：

1、文件导入景点与道路  
　 2、输出景区景点图  
　　3、输出导游线路图；  
　　4、判断导游线路图有无回路；  
　　5、求两个景点间的最短路径和最短距离；  
　　6、输出道路修建规划图。  
　　主程序用菜单选项供用户选择功能模块。

二、总体设计

用nodes文件输入景点信息，用routes文件输入景点连通情况。

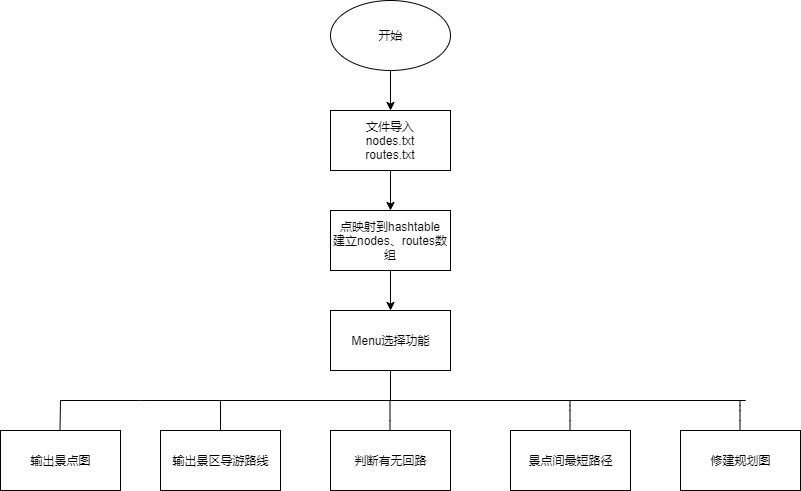
1、输出景点图，用ege库实现

2、输出导游路线图，采用深度优先遍历

3、判断有无回路，采用拓扑排序

4、求两个景点的最短路径和最短距离，采用弗洛伊德算法

5、输出道路修建规划图，即最小生成树，采用prim算法



三、详细设计

//2' 输出图形化景点分布

int PrintfGraph(PGraph pgraph){

initgraph(1000,600); //初始化，显示一个窗口

for(int i=0;i<sizeof(nodes)/sizeof(node);i++){

outtextxy(nodes[i].x,nodes[i].y,nodes[i].point\_name); //输出景点名称

}

for(int i=0;i<sizeof(routes)/sizeof(route);i++){

line(nodes[routes[i].src].x,nodes[routes[i].src].y,nodes[routes[i].dst].x,nodes[routes[i].dst].y);

}

getch(); // 暂停一下等待用户按键

closegraph(); // 关闭图形界面

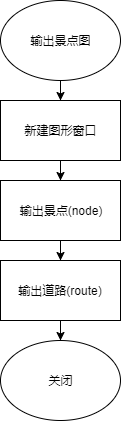
printf("按任意键返回主菜单\n");

printf("\n");

system("pause");

return SUCCESS;

}



//3、输出景区导游路线(深度优先)

int TraverseGraph(PGraph pgraph){

int V,i,j;

np=0;

for(i=0;i<pgraph->VexNum;i++)

Visited[i]=0;

printf("输出景区导游线路图：\n");

printf("请输入景点出发位置编号：\n");

scanf("%d",&V);

printf("景点导游路线为：\n");

DFS(pgraph,V-1);

printf("\n\n");

printf("按任意键返回主菜单\n");

system("pause");

return SUCCESS;

}

//深度优先函数

void DFS(PGraph pgraph,int V){

int j;

np++;

printf("%s ",pgraph->Vexs[V]);

Visited[V]=1;

for(j=0;j<pgraph->VexNum;j++)

{

if((pgraph->Arcs[V][j]!=INF&&pgraph->Arcs[V][j]!=0)&&Visited[j]!=1)

{

DFS(pgraph,j);

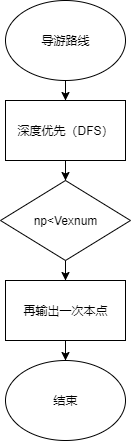
if(pgraph->VexNum>np)

printf("%s ",pgraph->Vexs[V]);

}

}

}



//4、判断导游线路有无回路（拓扑排序）

int TopologicalSort(PGraph pgraph)

{

int i,j;

printf("判断导游线路有无回路：\n\n");

//定义入度数组,记录每个顶点的入度，初始化为0

int Indegree[MAX\_VERTEX\_NUM]={0};

//定义桟、并初始化桟

STACK Stack;

if(InitStack(&Stack,pgraph->VexNum)==FAILURE)

{

printf("警告：创建桟失败！！！\n");

printf("按任意键回主菜单\n");

system("pause");

return FAILURE;

}

//求顶点的入度

for(j=0;j<pgraph->VexNum;j++)

{

for(i=0;i<pgraph->VexNum;i++)

if(pgraph->Arcs[i][j]!=0&&pgraph->Arcs[i][j]!=INF)

Indegree[j]++;

//入度为0的点入栈

if(Indegree[j]==0)

PushStack(&Stack,j);

}

int Count=0,k;

while(StackLen(&Stack))

{

PopStack(&Stack,&k);

Count++;

//对出栈的顶点所指向的点的入度减一，并将新入度为0的点入栈

for(j=0;j<pgraph->VexNum;j++)

{

if(pgraph->Arcs[k][j]!=0&&pgraph->Arcs[k][j]!=INF)

if((--Indegree[j])==0)

PushStack(&Stack,j);

}

}

//判断是否有回路

if(Count==pgraph->VexNum)

printf("导游线路图无回路\n");

else

{

printf("导游线路图有回路\n");

}

printf("\n按任意键回主菜单\n");

system("pause");

return SUCCESS;

}

//初始化栈，为拓扑排序用

int InitStack(STACK\*PStack,int num)

{

PStack->base=(int\*)malloc(num\*sizeof(int));

PStack->top=PStack->base;

PStack->stacksize=num;

return SUCCESS;

}

//入栈

int PushStack(STACK\*PStack,int j)

{

if(PStack->top-PStack->base==PStack->stacksize)

return FAILURE;

\*(PStack->top++)=j;

return SUCCESS;

}

//栈的长度

int StackLen(STACK\*PStack)

{

return PStack->top-PStack->base;

}

//出栈

int PopStack(STACK\*PStack,int\*Pk)

{

if(PStack->top==PStack->base)

return FAILURE;

\*Pk=\*(--(PStack->top));

return SUCCESS;

}

//5、两个景点的最短路径（弗洛伊德）

int MinShortPath(PGraph pgraph)

{

printf("景点间的最短路径：\n\n");

int PathMatrix[pgraph->VexNum][pgraph->VexNum];//路径矩阵

int DisMatrix[pgraph->VexNum][pgraph->VexNum]; //距离矩阵

int i,j,k;

//初始化两矩阵

for(i=0;i<pgraph->VexNum;i++)

{

for(j=0;j<pgraph->VexNum;j++)

{

DisMatrix[i][j]=pgraph->Arcs[i][j];

if(DisMatrix[i][j]!=INF&&DisMatrix[i][j]!=0)

PathMatrix[i][j]=i;

else

PathMatrix[i][j]=-1;

}

}

//更新两矩阵

for(k=0;k<pgraph->VexNum;k++)

for(i=0;i<pgraph->VexNum;i++)

for(j=0;j<pgraph->VexNum;j++)

if(DisMatrix[i][j]>DisMatrix[i][k]+DisMatrix[k][j])

{

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

PathMatrix[i][j]=k;

}

//准备输出两景点的最短路径

int Stav=-1,Endv=-1;

printf("请输入起点和终点编号：\n");

scanf("%d%d",&Stav,&Endv);

Stav--;

Endv--;//用户输入时是以正常数字顺序输入，而数组是从0开始，故减1

// 判断起始点名称和终点名称是否存在

if(Stav==-1||Endv==-1)

return FAILURE;

STACK Stack;

if(InitStack(&Stack,pgraph->VexNum)==FAILURE)

{

printf("警告：创建桟失败！！！ \n");

printf("按任意键回主菜单\n");

system("pause");

}

printf("\n两景点最短距离为%d\n",DisMatrix[Stav][Endv]);

while(Endv!=-1)

{

PushStack(&Stack,Endv);

Endv=PathMatrix[Stav][Endv];

}

// 将所有路径出桟、并输出

printf("两景点最短路线为：\n");

while(1)

{

PopStack(&Stack,&Stav);

printf("%s ",pgraph->Vexs[Stav]);

if(!StackLen(&Stack))

break;

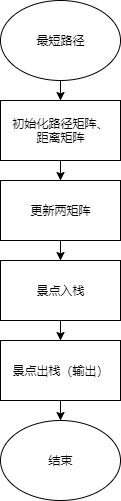
}

printf("\n\n按任意键返回主菜单\n");

system("pause");

return SUCCESS;

}



//6、输出道路修建规划图（最小生成树 prim算法）

MiniSpanTree\_Prim(PGraph pgraph)

{

int lowcost[MAX\_VERTEX\_NUM],closest[MAX\_VERTEX\_NUM];

int i,j,k,min,v=0;//(v=0:从第一个顶点开始构建生成树）

printf("输出道路修建规划图:\n");

//初始化lowcost数组和closest数组

for(i=0;i<pgraph->VexNum;i++)

{

lowcost[i]=pgraph->Arcs[v][i];

closest[i]=v;

}

//开始加点并更新lowcost和closest数组

for(i=1;i<pgraph->VexNum;i++)

{

min=INF;

for(j=0;j<pgraph->VexNum;j++)

{

if(lowcost[j]!=0&&lowcost[j]<min)

{

min=lowcost[j];

k=j;

}

}

if(min<INF){

printf("%s->%s\n",pgraph->Vexs[closest[k]],pgraph->Vexs[k]);

}

lowcost[k]=0;

//更新两数组

for(j=0;j<pgraph->VexNum;j++)

{

if(pgraph->Arcs[k][j]!=0&&pgraph->Arcs[k][j]<lowcost[j])

{

lowcost[j]=pgraph->Arcs[k][j];

closest[j]=k;

}

}

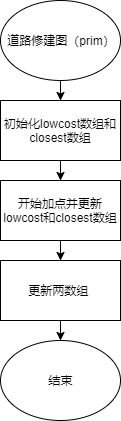
}

printf("\n\n按任意键返回主菜单\n");

system("pause");

return SUCCESS;

}

****

//显示功能

void Frame(){

printf("景区旅游信息管理系统功能：\n");

printf("1、创建景区景点分布图\n");

printf("2、输出景区景点分布图\n");

printf("3、输出景区导游路线\n");

printf("4、判断导游线路图有无回路\n");

printf("5、景点之间的最短路径\n");

printf("6、输出道路修建规划图\n");

printf("请选择相应的功能\n");

}

//主菜单

int Menu(){

Graph graph;

int select;

while(1){

system("cls"); //清屏

Frame();

scanf("%d",&select);

system("cls");

switch(select)

{

case 1:

CreateGraph(&graph);

break;

case 2:

PrintGraph(&graph);

break;

case 3:

TraverseGraph(&graph);

break;

case 4:

TopologicalSort(&graph);

break;

case 5:

MinShortPath(&graph);

break;

case 6:

MiniSpanTree\_Prim(&graph);

break;

default:

break;

}

}

}

int main()

{

Menu();

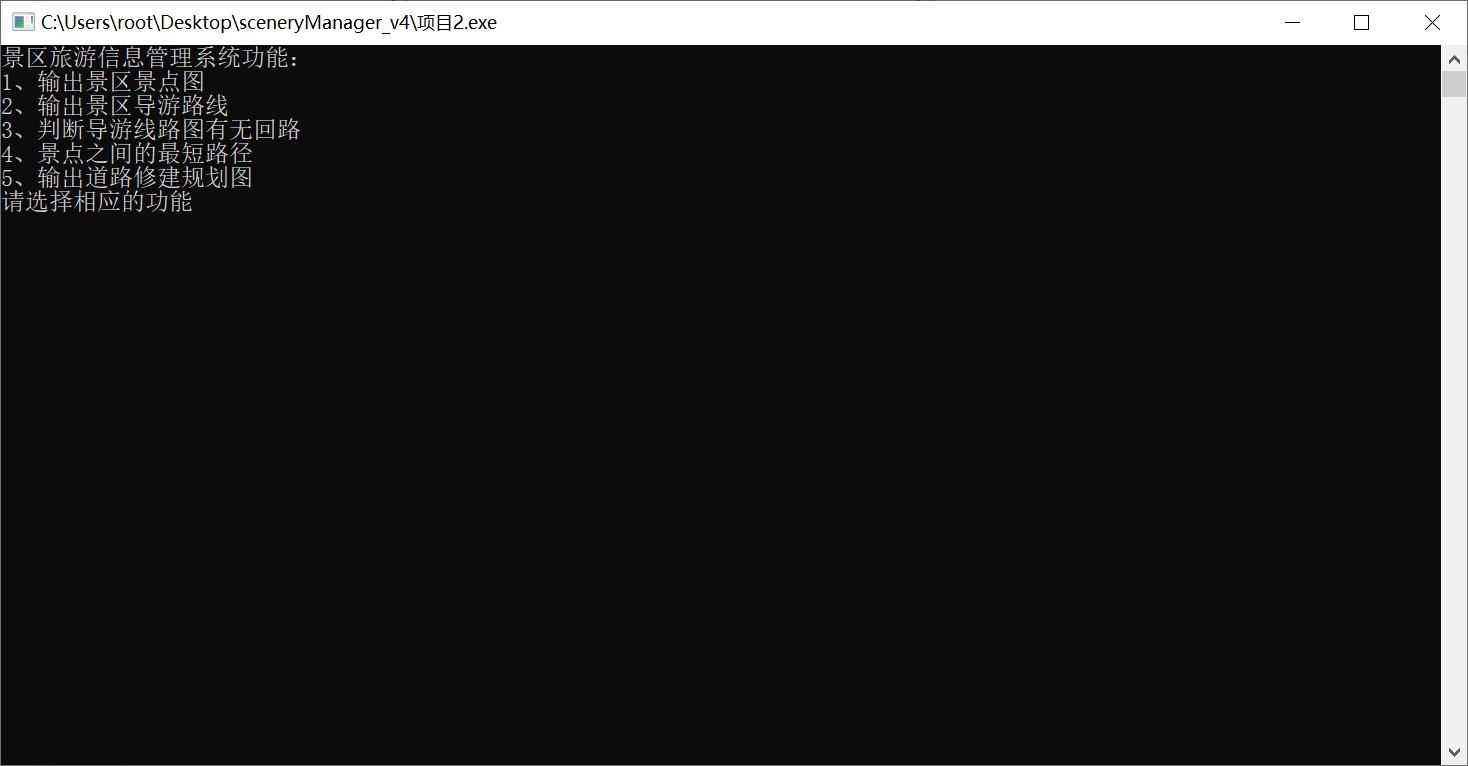
return 0;

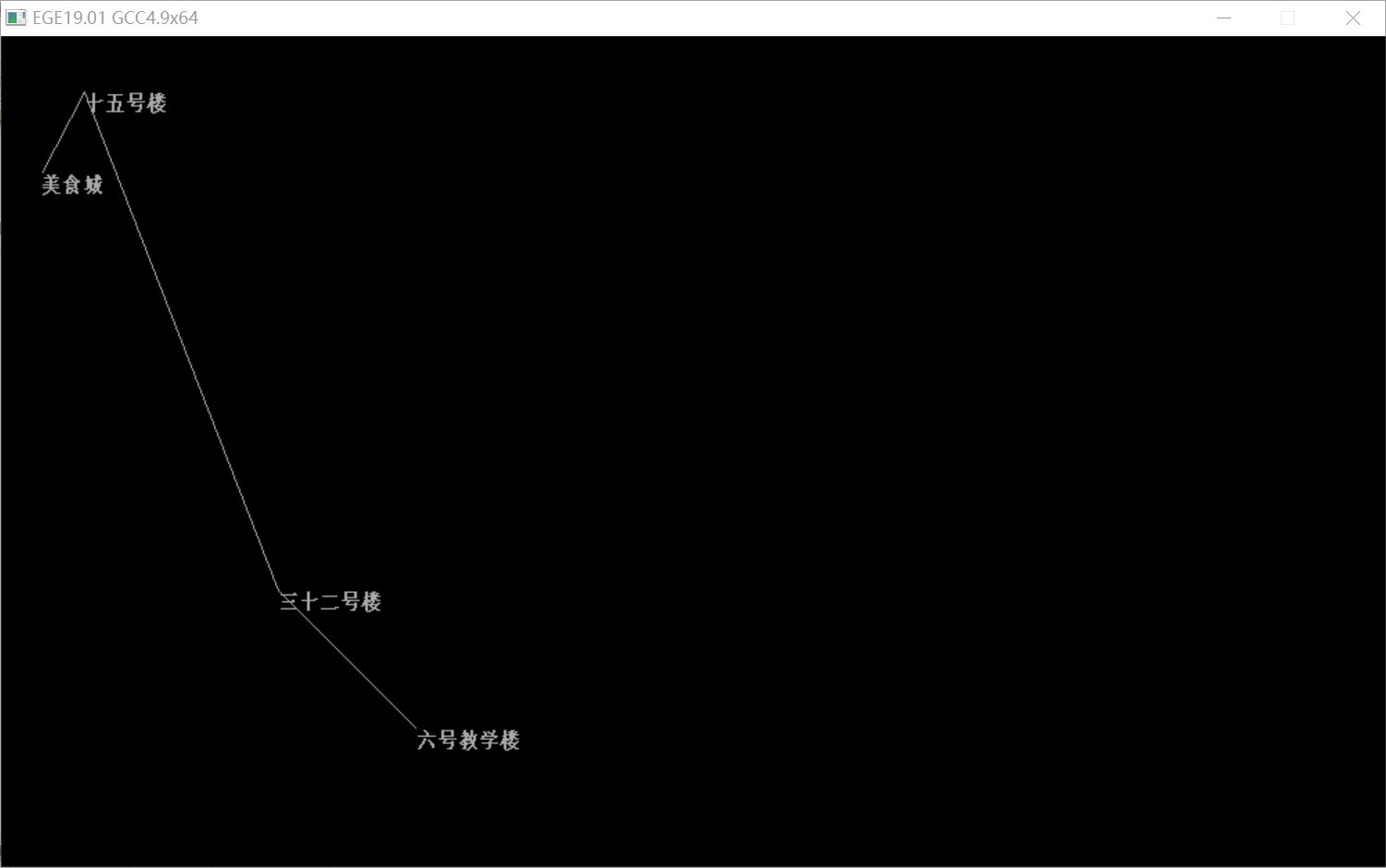
}

四、系统测试与分析

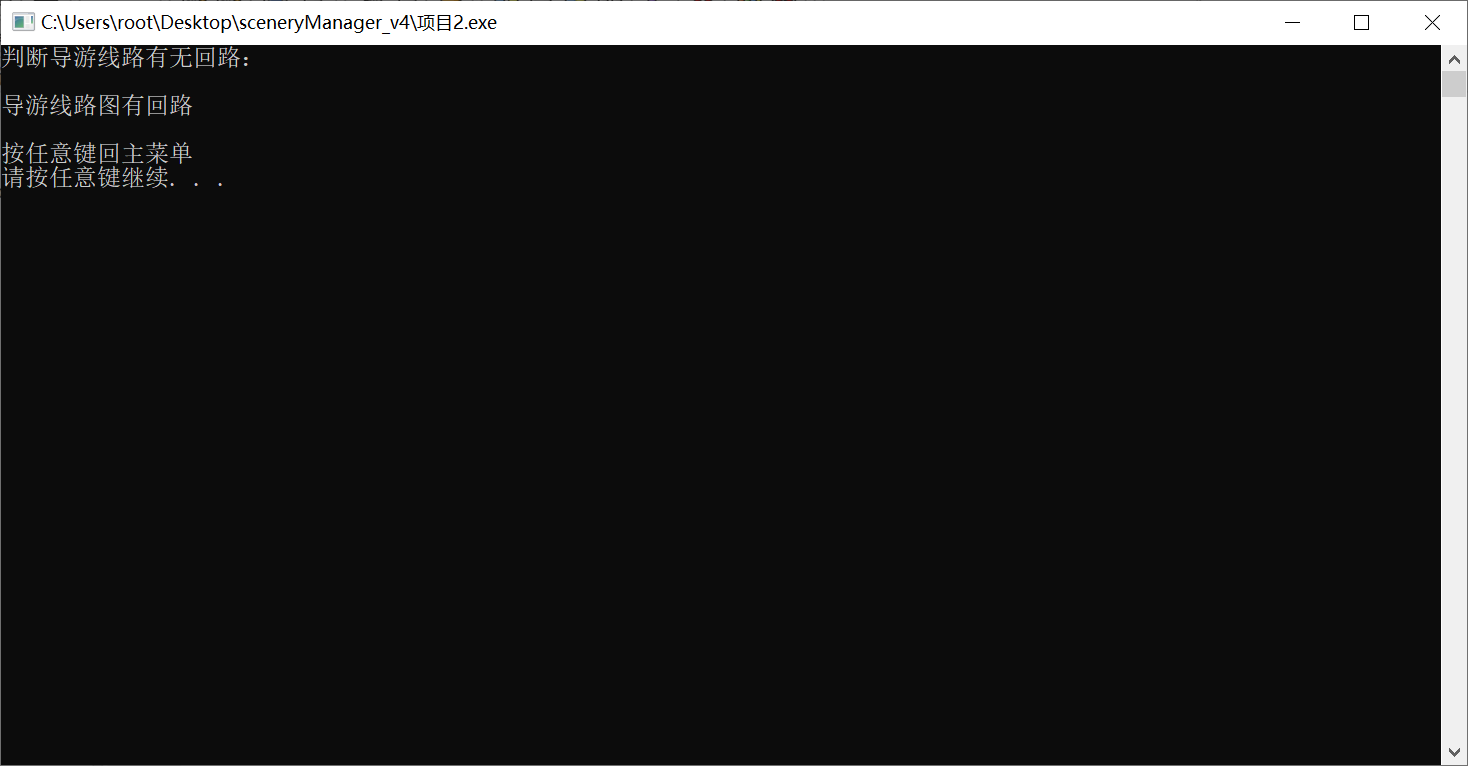


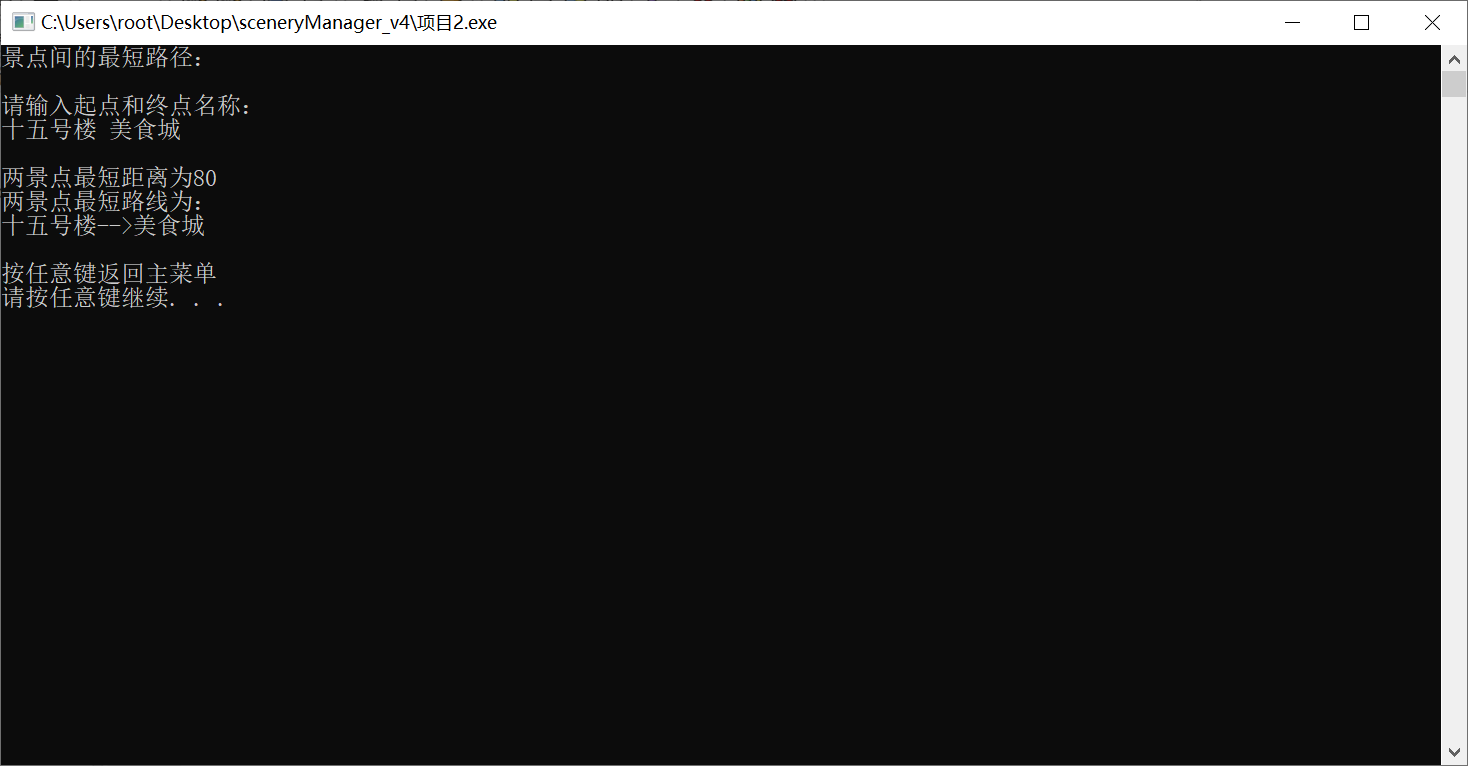


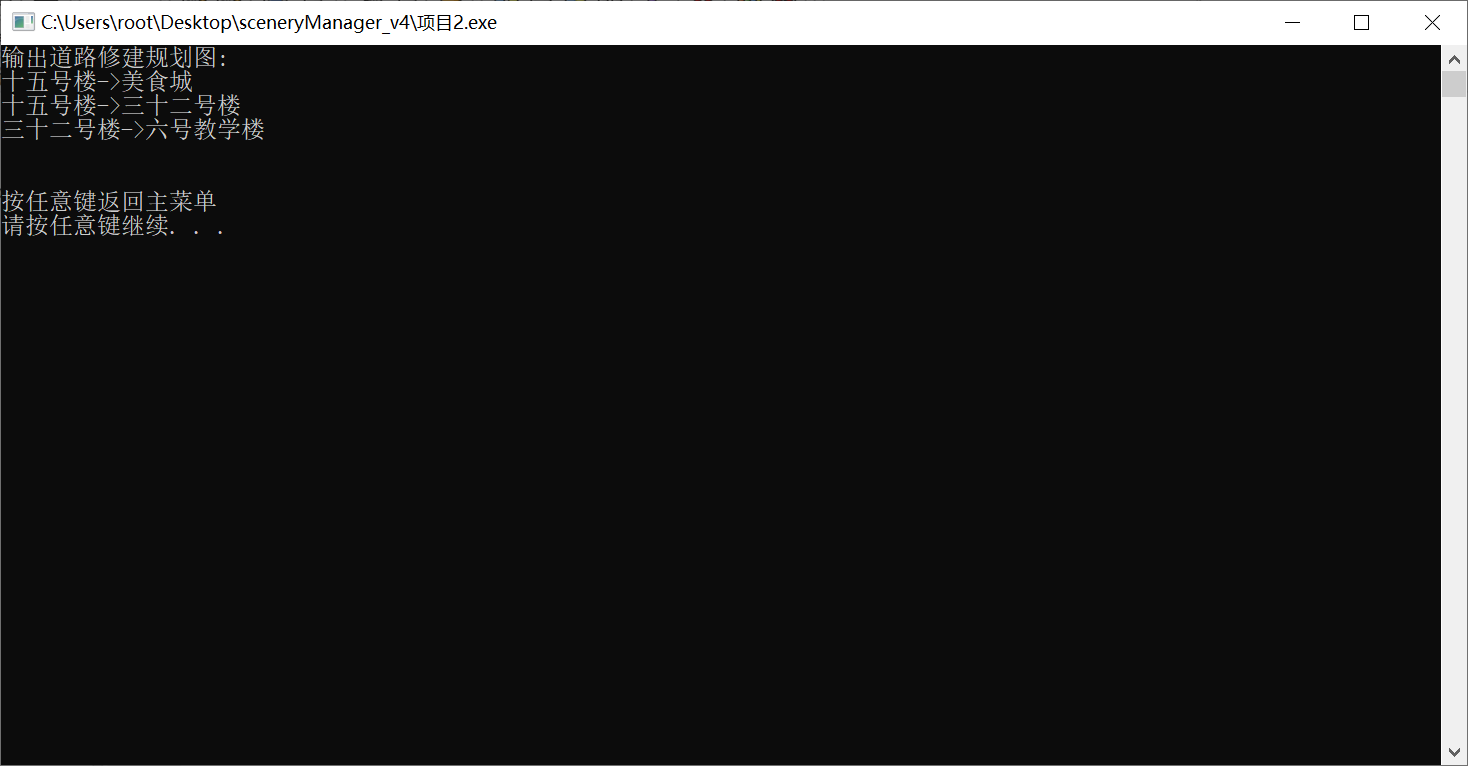












五、结论与心得

蔡祖润：主要负责创建景区景点分布图、输出景区景点分布图、输出景区导游路线模块。

在设计输出景区导游路线时，遇到了一个问题，就是如何在深度优先回溯时再次输出回溯的点。这个需求源于从中间点出发的旅游路线，会再次经过该中间点，需要再次输出该点，但一般的深度优先是只会每个点输出一次。查阅了很多资料无果，最后灵机一动，如果定义一个变量来记录已经访问过顶点的数量，如果小于总顶点数量，则回溯时再次输出该顶点。因为如果不是从中间点，也就是不需要重复输出的情况，新定义的变量在回溯完后是会等于总顶点数的。

在设计中新学习到的一些知识点：

1、定义一个字符串数组：char a[][],后面的[]表示字符串的最长长度，前一个则表示字符串的个数。

2、若想要显示的结果一闪而过，直接跳回主菜单，应用system(“pause”)

实现清屏操作用system(“cls”)

3、在深度优先时有个地方很巧，若回溯时想显示回溯的顶点，应在定义一个变量来记录已经显示的顶点，若次数小于总顶点数，则显示

4、另外，若要实现选择访问完后，再继续访问别的函数，则要用while（1）！！

5、一些优秀的习惯，在定义失败时或超过限制都要提醒并返回FALSE。

吴开洧：主要负责输出道路修建规划图、主程序用菜单选项供用户选择功能模块。

在景区建设中，道路建设是其中一个重要内容。道路建设首先要保证能连通所有景点，但又要花最小的代价，可以通过求最小生成树来解决这个问题。本任务中假设修建道路的代价只与它的里程相关。采用prim算法实现最小生成树从而输出最小连通图，主要是在对于权值更新的理解花了很多时间，有U和V两个点集，U中存放已找到的点，V中存放未连通的点，每次找到最小权值后将V中的该点放入U中，这样就能实现最小连通图。

主菜单界面要向使用者说明使用方法，要先定义图的变量，并采用switch语句将各个函数调用。首次创建时提示使用者首先创建景点图才能使用其他功能。

收获的知识点：

1.若想要显示的结果一闪而过，直接跳回主菜单，应用system(“pause”)

每次操作结束后，实现清屏操作用system(“cls”）。

2.若用户操作不当，输入有误，应当输出提示语报错。

王邦耀: 负责判断导游图是否存在回路，输出两点间的最短的行走路线。

输出最短路径的时候复习一哈弗洛伊德算法，有一点有趣的东西。使用二维数组记录从（起点、终点）到中转点的映射关系，通过while循环从末尾输出，体现了弗洛伊德算法的本质，递推+贪心。倒序入栈，再逐个出栈。完成路径的输出。