# 2017数据结构课程设计报告

**【题目】图的操作和应用之景区信息管理系统**

**(1) 读文件创建图**

输入：从Vex.txt文件中读取景点信息，从Edge.txt文件中读取道路信息。

处理：根据读取的景区信息创建景区景点图。

**(2) 查询景点**

输入：想要查询的景点的编号。

处理：根据输入的景点编号，查询该景点及相邻景点的信息。

输出：

① 景点名字

② 景点介绍

③ 相邻景区的名字

④ 到达相邻景区的路径长度

**(3) 旅游景点导航**

输入：起始景点的编号。

处理：使用深度优先搜索(DFS)算法，查询以该景点为起点，无回路游览整个景区的路线。

输出：所有符合要求的导航路线。

**(4) 搜索最短路径**

输入：

① 起始景点的编号

② 终点的编号。

处理：使用迪杰斯特拉(Dijkstra)算法，求得从起始景点到终点之间的最短路径，计算路径总长度。

输出：

① 最短路线

② 路径总长度

**(5) 铺设电路规划**

处理：根据景区景点图使用普里姆(Prim)算法构造最小生成树，设计出一套铺设线路最短，但能满足每个景点都能通电的方案。

输出：

① 需要铺设电路的道路

② 每条道路铺设电路的长度

③ 铺设电路的总长度

**(6) 修改图保存文件**

插入、删除、修改顶点、边的信息，注意顶点和边的关系，之后保存文件，重新读取文件建立图的存储结构并显示。

重点注意顶点和边的关系，考虑边是否重复？顶点是否存在？……

**菜单及结构体定义代码**

#define MAX\_SIZE 1000//最大可存顶点数

#define INFINITY 99999//无穷大

struct Vertex//顶点

{

int tag;//标记此顶点是否存在

string name,info;

};

struct ArcNode

{

int n;

ArcNode \*next;

};

struct VNode//邻接表

{

Vertex vex;

ArcNode \*next;

}AdjList[MAX\_SIZE];

struct Graph//图

{

int arc[MAX\_SIZE][MAX\_SIZE];//每个顶点与其余各个顶点之间边长

int vexnum;//目前顶点总数

int edgnum;//目前此图边总数

int maxvexno; //目前此图顶点最大编号

}G;

int created=0;//表示未创建图

int main()//主函数

{

int tag,m,n;

while(1)

{

tag=0;

Menu();

cin>>n;

switch(n)

{

case 1:CreatMap();break;

case 2:{cout<<"请输入景区编号："<<endl;cin>>n;SearchVex(n);}break;

case 3:{cout<<"请输入景区编号："<<endl;cin>>n;Navigation\_DFS(n);}break;

case 4:{cout<<"请输入起止景区编号："<<endl;cin>>m>>n;ShortestPath\_DIJ(m,n);}break;

case 5:WirePath\_Prim();break;

case 6:Edit();break;

case 7:tag=1;break;

default:cout<<"更多功能正在开发，敬请期待……"<<endl;

}

if(tag)

break;

}

return 0;

}

void Menu()//主菜单

{

printf("---------------景区管理系统功能菜单---------------\n");

printf("1-----创建图\n");

printf("2-----查询景点信息\n");

printf("3-----旅游景点导航\n");

printf("4-----搜索最短路径\n");

printf("5-----铺设电路规划\n");

printf("6-----修改图\n");

printf("7-----退出主菜单\n");

printf("请输入功能序号执行功能：\n");

}

**1、数据格式**

Vex.txt中第一行景点总数，之后每一行一个景点信息，包括景点编号、名称、介绍。

Edge.txt中每一行一条道路信息。包括景点编号、景点编号、道路距离。

**2、数据结构（读文件创图）**

将文件中的数据读入内存，建立图的邻接表并输出图的邻接表。

实现该功能代码如下：

void CreatMap()//导入文件数据创建图，并输出邻接表

{

int m,t,i,j;

//读取Vex.txt文件的景点信息

fstream f1("./Vex.txt",ios::in);

f1>>G.vexnum;

t=0;//记录已读取景点信息个数

for(i=0;t<G.vexnum;i++)

{

f1>>m;

while(i!=m)

{

AdjList[i].vex.tag=0;//编号为i的景点不存在，标记为0

i++;

}

AdjList[m].vex.tag=1; //景点存在，标记为1

t++;

f1>>AdjList[m].vex.name>>AdjList[m].vex.info;//从vex.txt文件中读取景点信息

}

G.maxvexno=m;//记录景点的最大编号

f1.close();

//初始化各路径距离

for(i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

for(j=0;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

G.arc[i][j]=INFINITY;//初始化各景点间距离为INFINITY

//读取Edge.txt的路径信息

fstream f2("./Edge.txt",ios::in);

G.edgnum=0;

while(f2>>i>>j)

{

f2>>G.arc[i][j];

G.arc[j][i]=G.arc[i][j];//从Edge.txt文件中读取景点间距离，若文件中未出现两景点间距离则保持初始的INFINITY

G.edgnum++;

}

f2.close();

created=1;//表示已经创建图

//创建邻接表

CreatAdjList();

//展示邻接表

cout<<"邻接表如下："<<endl;//格式为V1->V2->V3

for(i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

{

cout<<'V'<<i;

ArcNode \*an=AdjList[i].next;

while(an)

{

cout<<"->V"<<an->n;

an=an->next;

}

cout<<endl;

}

}

void CreatAdjList()//建立邻接表

{

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

{

ArcNode \*a;

AdjList[i].next=NULL;

for(int j=G.maxvexno;j>=0;j--)

if(AdjList[j].vex.tag)

if(G.arc[i][j]<INFINITY)

{

a=new ArcNode;

a->n=j;

a->next=AdjList[i].next;

AdjList[i].next=a;

}

}

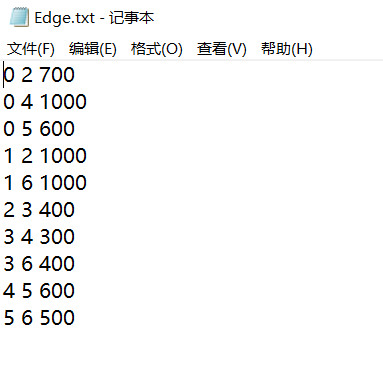
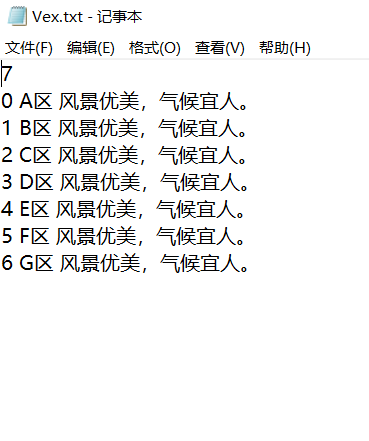
}

数据格式：

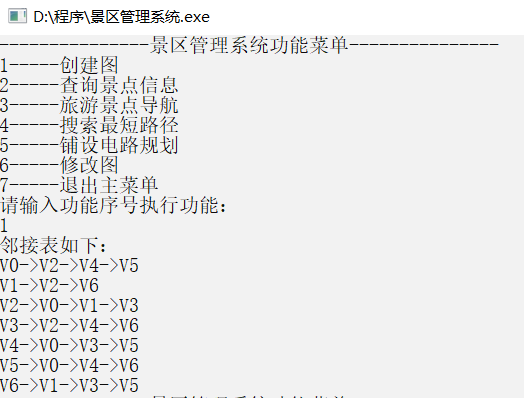
|  |  |  |
| --- | --- | --- |
| 编号 | 名字 | 介绍 |
| 0 | A区 | 风景优美，气候宜人。 |
| 1 | B区 | 风景优美，气候宜人。 |
| 2 | C区 | 风景优美，气候宜人。 |
| 3 | D区 | 风景优美，气候宜人。 |
| 4 | E区 | 风景优美，气候宜人。 |
| 5 | F区 | 风景优美，气候宜人。 |
| 6 | G区 | 风景优美，气候宜人。 |

|  |  |  |
| --- | --- | --- |
| 景点1 | 景点2 | 距离(m) |
| A | C | 700 |
| A | E | 1000 |
| A | F | 600 |
| B | C | 1000 |
| B | G | 1000 |
| C | D | 400 |
| D | E | 300 |
| D | G | 400 |
| E | F | 600 |
| F | G | 500 |

文件截图：



运行截图：



**3、查询景点**

输入：想要查询的景点的编号。

处理：根据输入的景点编号，查询该景点及相邻景点的信息。

输出：

① 景点名字

② 景点介绍

③ 相邻景区的名字

④ 到达相邻景区的路径长度

算法：根据输入编号直接用数组找到景点信息输出，再用邻接表找到与其相连的景点并输出信息。

实现该功能代码如下：

void SearchVex(int n)//查找编号为n的景点

{

if(created)//图已被创建

{

if(n<0||n>G.maxvexno||AdjList[n].vex.tag==0)//编号n不在文件读入的编号范围内或其标记为0

cout<<"该景点不存在！"<<endl;

else

{//输出景点编号、名称和景点信息，以及相邻的各个景点编号、名称和相距距离

cout<<n<" "<<AdjList[n].vex.name<<" "<<AdjList[n].vex.info<<endl;

cout<<"相邻景点："<<endl;

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

if(G.arc[i][n]<INFINITY)

cout<<i<<" "<<AdjList[i].vex.name<<" "<<G.arc[i][n]<<endl;

}

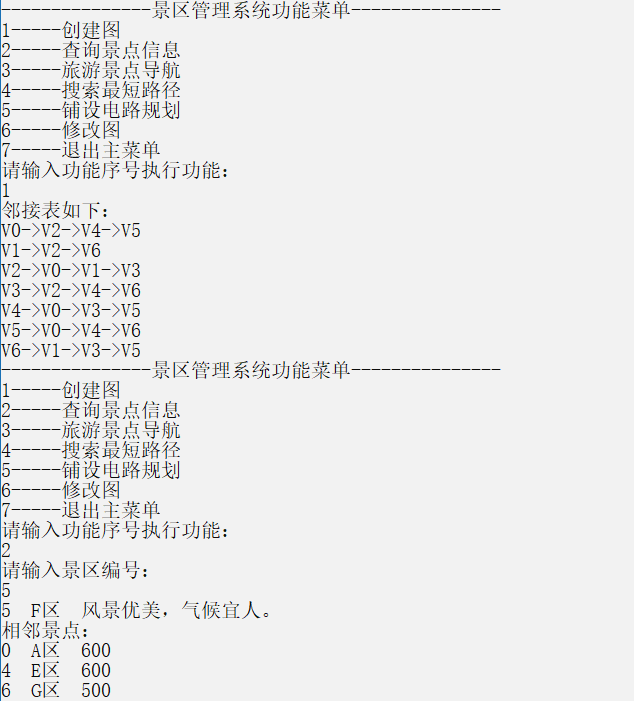
}

else//图未被创建

cout<<"尚未读取文件创建图！"<<endl;

}

运行截图：



**7、修改图保存文件**

插入、删除、修改顶点、边的信息，注意顶点和边的关系，之后保存文件，重新读取文件建立图的存储结构并显示。

核心操作：首先要保证添加、删除后仍为连通图（因此添加景点也要添加一条边，删除顶点也要删除相连的边，代码中用IsConnected函数来判断是否连通），否则操作不能执行。其次在操作执行后都要将数据重新读入文件。最后要再一次读取文件创建图。

实现该功能代码如下：

void Edit()//修改操作菜单

{

int m;

if(created)

{

while(1)

{

int tag=0;

cout<<"-----编辑操作菜单-----"<<endl;

cout<<"-----<1>添加-----"<<endl;

cout<<"-----<2>删除-----"<<endl;

cout<<"-----<3>修改-----"<<endl;

cout<<"-----<4>退出本菜单-----";

cout<<"输入编号执行相应操作："<<endl;

cin>>m;

switch(m)

{

case 1:Add();break;

case 2:Delete();break;

case 3:Revise();break;

case 4:tag=1;break;

default:cout<<"更多功能正在开发中，敬请期待……" <<endl;

}

if(tag)

break;

}

}

else

cout<<"尚未读取文件创建图！"<<endl;

}

int count;//记录已访问顶点数

int IsConnected()//判断是否为连通图

{

count=0;

int \*Visited=new int[G.maxvexno+1]; //标记是否已访问过

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

Visited[i]=0;//初始化

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

{

Connect\_DFS(i,Visited);

break;

}

if(count==G.vexnum)

return 1;//该图连通，返回1

else

return 0; //该图不连通，返回0

}

void Connect\_DFS(int i,int \*Visited)//DFS搜索

{

ArcNode \*an=AdjList[i].next;

Visited[i]=1;//标记已访问过

count++;//并入已访问过的点集

cout<<count<<endl;

while(an)//搜索与该顶点相连的每个顶点

{

while(Visited[an->n]||AdjList[an->n].vex.tag==0)//已访问过就找下一个顶点

{

an=an->next;

if(an==NULL)break;

}

if(an)

{

Connect\_DFS(an->n,Visited);//对此顶点进行DFS搜索

an=an->next;

}

}

}

void Add()//添加操作菜单

{

int n,tag;

while(1)

{

tag=0;

cout<<"Vex.txt内容读取："<<endl;

ShowVexTxt();

cout<<"Edge.txt内容读取："<<endl;

ShowEdgeTxt();

cout<<"--------添加选项------"<<endl;

cout<<"-----<1>景点-----" <<endl;

cout<<"-----<2>路径-----"<<endl;

cout<<"-----<3>退出当前菜单-----"<<endl;

cout<<"输入功能序号执行相应功能："<<endl;

cin>>n;

switch(n)

{

case 1:AddVex();break;

case 2:AddEdge();break;

case 3:tag=1;break;

default:cout<<"更多功能正在开发，敬请期待……"<<endl;

}

if(tag)

break;

}

}

void AddVex()//添加景点

{

int no,start,end,dis,maxno;

string name,info;

if(G.vexnum<MAX\_SIZE)

{

cout<<"请输入添加的景点及路径：（第一行：景点编号 名称 信息 第二行：景点编号 景点编号 距离）"<<endl;

cin>>no>>name>>info;

cin>>start>>end>>dis;

if(no<0)

cout<<"景点编号不合法！添加失败！"<<endl;

else

{

if(no>G.maxvexno||AdjList[no].vex.tag==0)

{//初始化该景点相关信息

AdjList[no].vex.tag=1;

AdjList[no].vex.name=name;

AdjList[no].vex.info=info;

G.vexnum++;

maxno=G.maxvexno;

if(no>G.maxvexno)//更改最大编号，并将不存在的顶点标记

{

for(int i=G.maxvexno+1;i<no;i++)

AdjList[i].vex.tag=0;

G.maxvexno=no;

}

for(int i=0;i<=G.maxvexno;i++)//初始化no与其他顶点的距离

if(AdjList[i].vex.tag)

{

G.arc[i][no]=INFINITY;

G.arc[no][i]=INFINITY;

}

//添加道路

if(start<0||start>G.maxvexno||AdjList[start].vex.tag==0||end<0||end>G.maxvexno||AdjList[end].vex.tag==0)

cout<<"景点不存在！添加失败！"<<endl;

else if(dis<=0||dis>=INFINITY)

cout<<"该路径距离超限！添加失败！"<<endl;

else if(G.arc[start][end]<INFINITY)

cout<<"该路径已记录！添加失败！"<<endl;

else

{

G.arc[start][end]=dis;

G.arc[end][start]=dis;

G.edgnum++;

CreatAdjList();//重建邻接表

if(IsConnected())//添加后仍是连通图

{

fstream f3("./Vex.txt",ios::out);

fstream f4("./Edge.txt",ios::out);

//读入Vex文件

f3<<G.vexnum<<endl;

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

f3<<i<<' '<<AdjList[i].vex.name<<' '<<AdjList[i].vex.info<<endl;

f3.close();

//读入Edge文件

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

for(int j=i;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

if(G.arc[i][j]<INFINITY)

f4<<i<<' '<<j<<' '<<G.arc[i][j]<<endl;;

f4.close();

cout<<"添加景点成功！"<<endl;

}

else//否则添加失败

{//更改为添加前的值

AdjList[no].vex.tag=0;

G.vexnum--;

G.maxvexno=maxno;

G.arc[start][end]=INFINITY;

G.arc[end][start]=INFINITY;

G.edgnum--;

cout<<"添加后为非连通图！添加失败！"<<endl;

}

//重新创建图

CreatMap();

}

}

else

cout<<"该景点已存在！添加失败！"<<endl;

}

}

else

cout<<"景点存储数量已达上限，无法再添加！"<<endl;

}

void AddEdge()//添加道路

{

int start,end,dis;

//添加道路

cout<<"请输入添加的路径：（景点编号 景点编号 距离）"<<endl;

cin>>start>>end>>dis;

if(start<0||start>G.maxvexno||AdjList[start].vex.tag==0||end<0||end>G.maxvexno||AdjList[end].vex.tag==0)

cout<<"景点不存在！添加失败！"<<endl;

else if(dis<=0||dis>=INFINITY)

cout<<"该路径距离超限！添加失败！"<<endl;

else if(G.arc[start][end]<INFINITY)

cout<<"该路径已记录！添加失败！"<<endl;

else

{

G.arc[start][end]=dis;

G.arc[end][start]=dis;

G.edgnum++;

CreatAdjList();//重建邻接表

if(IsConnected())//添加后仍是连通图

{

fstream f5("./Edge.txt",ios::out);

//读入Edge文件

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

for(int j=i;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

if(G.arc[i][j]<INFINITY)

f5<<i<<' '<<j<<' '<<G.arc[i][j]<<endl;

f5.close();

cout<<"添加路径成功！"<<endl;

}

Else//添加后不是连通图

{//恢复原值

G.arc[start][end]=INFINITY;

G.arc[end][start]=INFINITY;

G.edgnum--;

cout<<"添加后为非连通图！添加失败！"<<endl;

}

CreatMap();//重新创建图

}

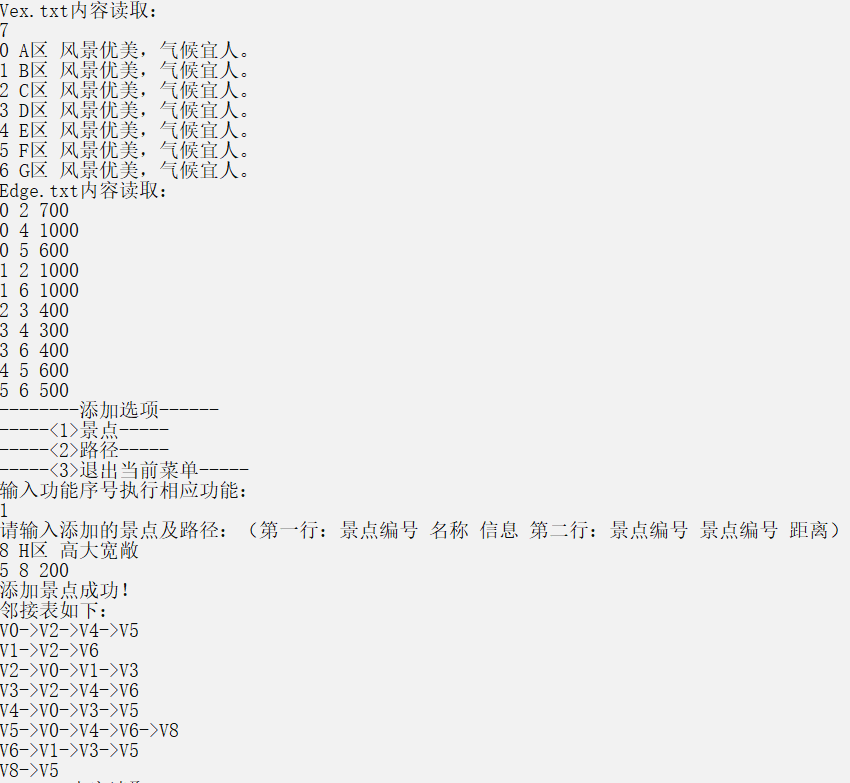
}

添加景点输入：

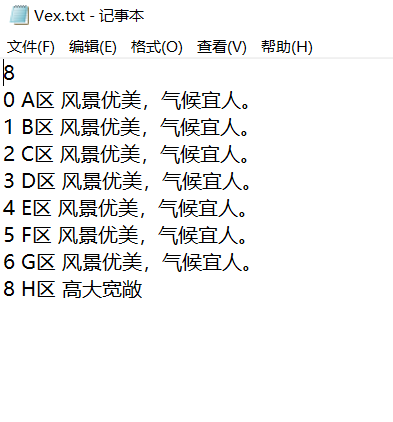
8 H区 高大宽敞

5 8 200

运行截图：



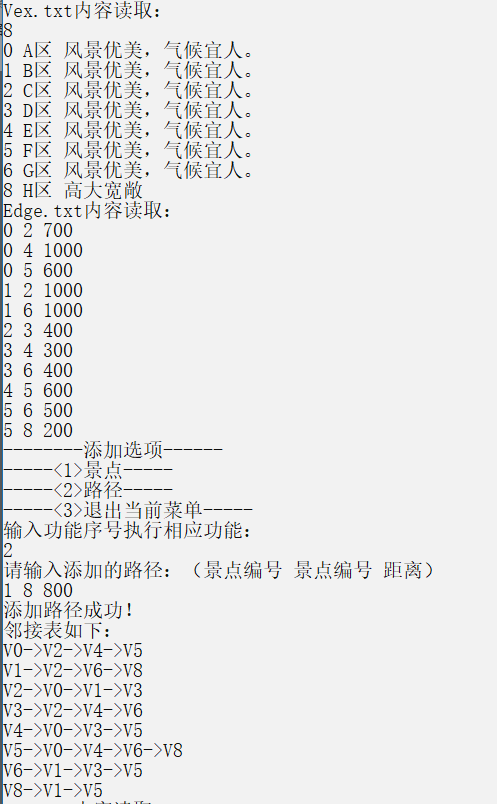
文件截图：



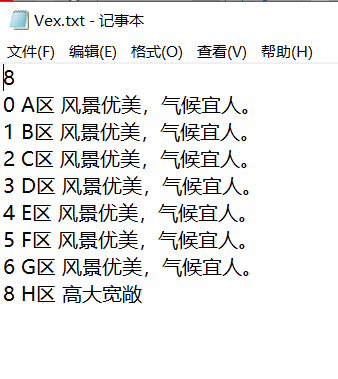
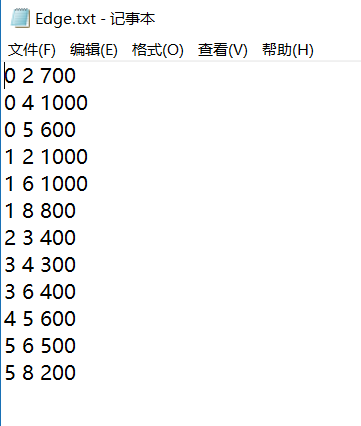
添加道路输入：

1 8 800

运行截图：



文件截图：



void Revise()//修改操作菜单

{

int n,tag;

while(1)

{

tag=0;

cout<<"Vex.txt内容读取："<<endl;

ShowVexTxt();

cout<<"Edge.txt内容读取："<<endl;

ShowEdgeTxt();

cout<<"--------修改选项------"<<endl;

cout<<"-----<1>景点-----" <<endl;

cout<<"-----<2>路径-----"<<endl;

cout<<"-----<3>退出当前菜单-----"<<endl;

cout<<"请输入功能序号执行相应功能："<<endl;

cin>>n;

switch(n)

{

case 1:ReviseVex();break;

case 2:ReviseEdge();break;

case 3:tag=1;break;

default:cout<<"更多功能正在开发，敬请期待……"<<endl;

}

if(tag)

break;

}

}

void ReviseEdge()//修改道路距离

{

//修改路径信息

int m,n,start,end,dis;

cout<<"请输入要修改路径距离的路径个数：（不得超过"<<G.edgnum<<"个）"<<endl;

cin>>n;

while(n<0||n>G.edgnum)

{

cout<<"已超过已存在路径个数！请重新输入："<<endl;

cin>>n;

}

m=n;

cout<<"请逐行输入路径新信息：（格式：景点编号 景点编号 距离）"<<endl;

while(n)

{

cin>>start>>end>>dis;

if(start<0||start>G.maxvexno||AdjList[start].vex.tag==0||end<0||end>G.maxvexno||AdjList[end].vex.tag==0)

cout<<"景点不存在！请重新输入修改信息："<<endl;

else if(dis<=0||dis>=INFINITY)

cout<<"该路径距离超限！请重新输入修改信息："<<endl;

else if(G.arc[start][end]==INFINITY)

cout<<"该路径不存在！请重新输入修改信息："<<endl;

else

{

G.arc[start][end]=dis;

G.arc[end][start]=dis;

n--;

}

}

fstream f7("./Edge.txt",ios::out);

//读入文件

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

for(int j=i;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

if(G.arc[i][j]<INFINITY)

f7<<i<<' '<<j<<' '<<G.arc[i][j]<<endl;

f7.close();

cout<<m<<"条路径信息已成功修改！"<<endl;

CreatMap();//重新创建图

}

void ReviseVex()//修改景点信息

{

int m,n,no;

string name,info;

//修改景点信息

cout<<"请输入要修改的景点信息的景点个数：（不得超过"<<G.vexnum<<"个）"<<endl;

cin>>n;

while(n<0||n>G.vexnum)

{

cout<<"已超过目前已存在景点个数！请重新输入："<<endl;

cin>>n;

}

m=n;

cout<<"请逐行输入景点新信息：（格式：编号 景点名 介绍）"<<endl;

while(n)

{

cin>>no>>name>>info;

if(no>=0&&no<G.maxvexno&&AdjList[no].vex.tag)

{

AdjList[no].vex.name=name;

AdjList[no].vex.info=info;

n--;

}

else

cout<<"该景点不存在！请重输修改信息："<<endl;

}

//读入文件

fstream f6("./Vex.txt",ios::out);

f6<<G.vexnum<<endl;

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

f6<<i<<' '<<AdjList[i].vex.name<<' '<<AdjList[i].vex.info<<endl;

f6.close();

cout<<m<<"条景点信息已成功修改！"<<endl;

CreatMap();//重新创建图

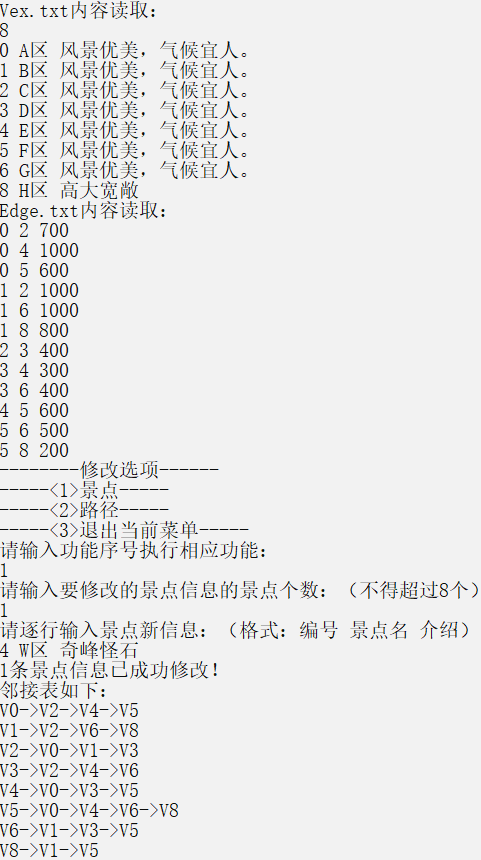
}

修改景点输入：

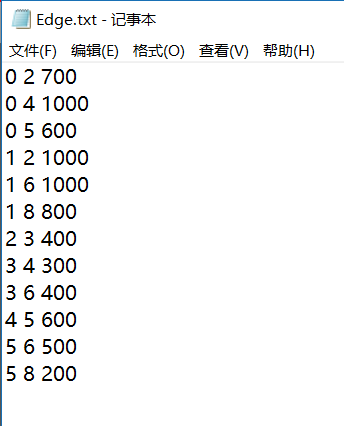
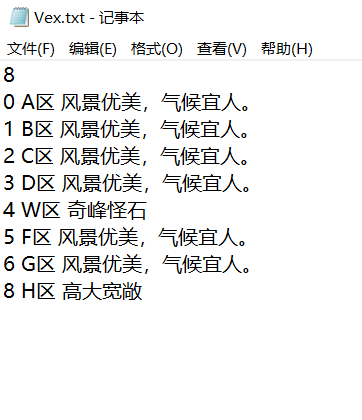
1

4 W区 奇峰怪石

运行结果：



文件截图：

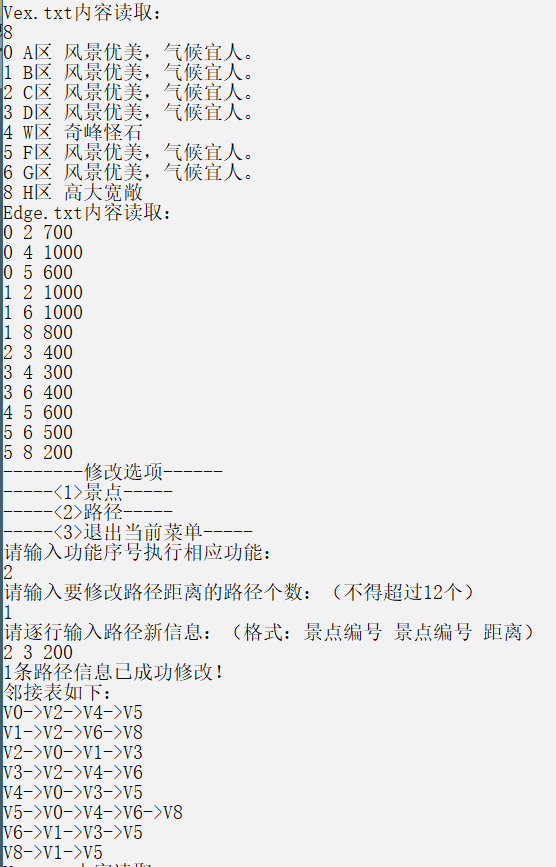


修改道路输入：

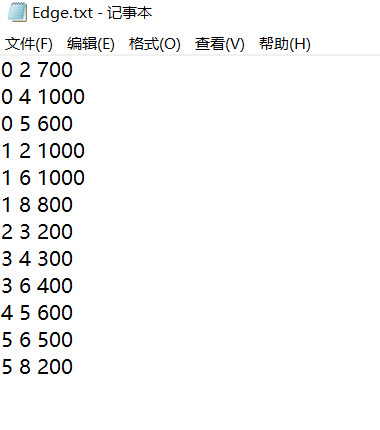
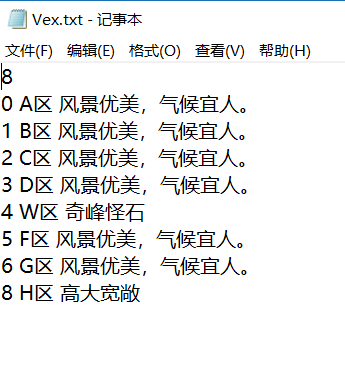
1

2 3 200

运行截图：



文件截图：



void Delete()//删除操作菜单

{

int n,tag;

while(1)

{

tag=0;

cout<<"Vex.txt内容读取："<<endl;

ShowVexTxt();

cout<<"Edge.txt内容读取："<<endl;

ShowEdgeTxt();

cout<<"--------删除选项------"<<endl;

cout<<"-----<1>景点-----" <<endl;

cout<<"-----<2>路径-----"<<endl;

cout<<"-----<3>退出当前菜单-----"<<endl;

cout<<"输入功能序号执行相应功能："<<endl;

cin>>n;

switch(n)

{

case 1:DeleteVex();break;

case 2:DeleteEdge();break;

case 3:tag=1;break;

default:cout<<"更多功能正在开发，敬请期待……"<<endl;

}

if(tag)

break;

}

}

void DeleteVex()//删除景点

{

int no,maxno;

cout<<"请输入要删除的编号："<<endl;

cin>>no;

if(no<0||no>G.maxvexno||AdjList[no].vex.tag==0)

cout<<"景点不存在！删除失败！"<<endl;

else

{

G.vexnum--;

AdjList[no].vex.tag=0;//修改标记

maxno=G.maxvexno;

if(no==G.maxvexno)//更新最大编号

for(int i=G.maxvexno-1;i>=0;i--)

if(AdjList[i].vex.tag)

{

G.maxvexno=i;

break;

}

CreatAdjList();//重建邻接表

if(IsConnected())

{//删除后连通

fstream f8("./Vex.txt",ios::out);

//读入Vex文件

f8<<G.vexnum<<endl;

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

f8<<i<<' '<<AdjList[i].vex.name<<' '<<AdjList[i].vex.info<<endl;

f8.close();

fstream f9("./Edge.txt",ios::out);

//更新删除顶点后的路径并读入Edge文件

G.edgnum=0;

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

for(int j=i;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

if(G.arc[i][j]<INFINITY)

{

f9<<i<<' '<<j<<' '<<G.arc[i][j]<<endl;

G.edgnum++;

}

f9.close();

cout<<"成功删除景点！"<<endl;

}

else

{//不连通则恢复原值

G.vexnum++;

AdjList[no].vex.tag=1;//修改标记

G.maxvexno=maxno;

cout<<"删除后为非连通图！删除失败！"<<endl;

}

CreatMap();//重新创建图

}

}

void DeleteEdge()//删除道路

{

int n,m,dis,start,end;

//删除路径信息

cout<<"请输入要删除的路径个数：（不得超过"<<G.edgnum-G.vexnum+1<<"个）"<<endl;

cin>>n;

while(n<0||n>G.edgnum-G.vexnum+1)

{

cout<<"已超过可删除路径数限制！请重新输入："<<endl;

cin>>n;

}

m=n;

cout<<"请逐行输入路径连接的两景点：（格式：景点编号 景点编号）"<<endl;

while(n)

{

cin>>start>>end;

if(start<0||start>G.maxvexno||AdjList[start].vex.tag==0||end<0||end>G.maxvexno||AdjList[end].vex.tag==0)

cout<<"景点不存在！请重新输入删除信息："<<endl;

else if(G.arc[start][end]==INFINITY)

cout<<"该路径不存在！请重新输入删除信息："<<endl;

else

{

dis=G.arc[start][end];

G.arc[start][end]=INFINITY;

G.arc[end][start]=INFINITY;

G.edgnum--;

n--;

CreatAdjList();//重建邻接表

if(IsConnected())

{//删除后仍连通

fstream f10("./Edge.txt",ios::out);

//读入文件

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

for(int j=i;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

if(G.arc[i][j]<INFINITY)

f10<<i<<' '<<j<<' '<<G.arc[i][j]<<endl;

f10.close();

cout<<"成功删除路径！"<<endl;

}

else

{//不连通则恢复原值

G.arc[start][end]=dis;

G.arc[end][start]=dis;

G.edgnum++;

cout<<"删除后为非连通图！删除失败！"<<endl;

}

}

}

cout<<m<<"条路径删除完毕！"<<endl;

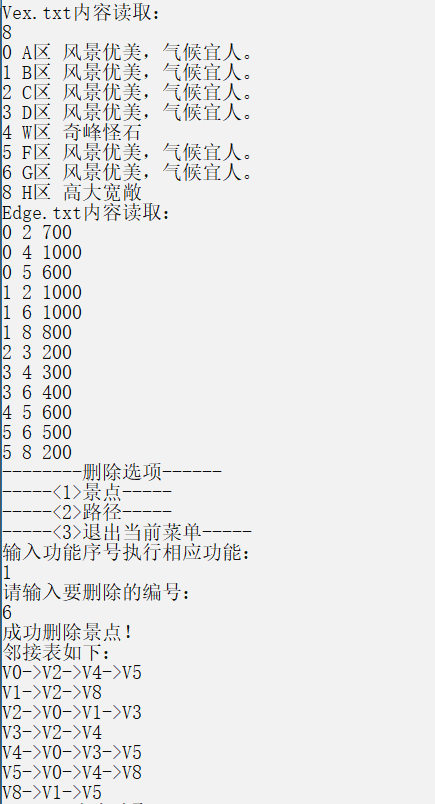
CreatMap();//重新创建图

}

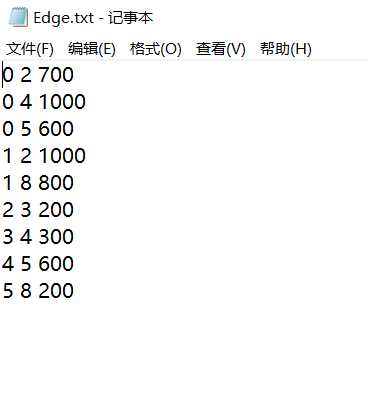
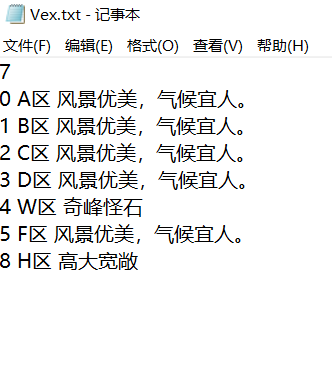
删除景点输入：

6

运行截图：



文件截图：

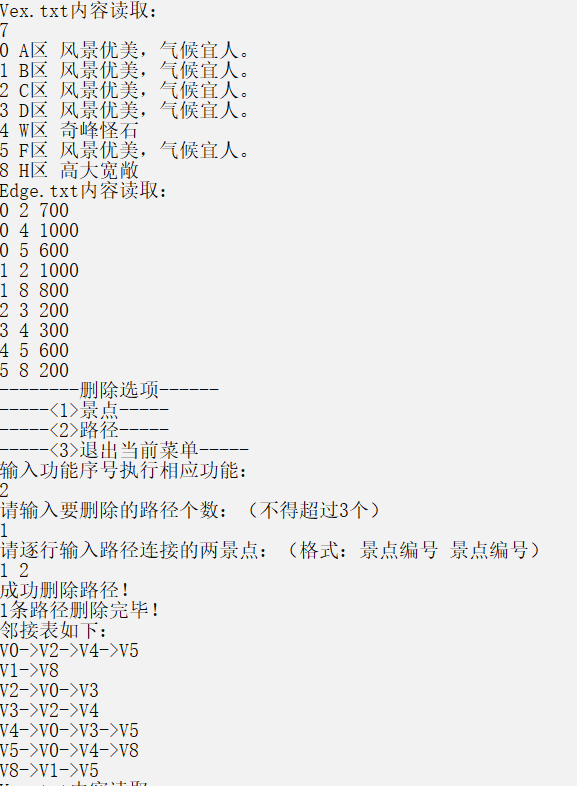


删除道路输入：

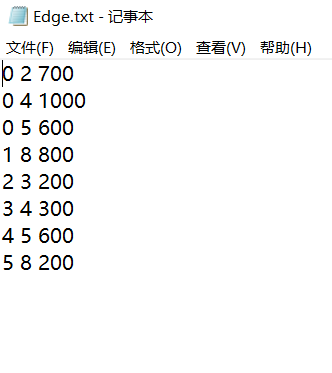
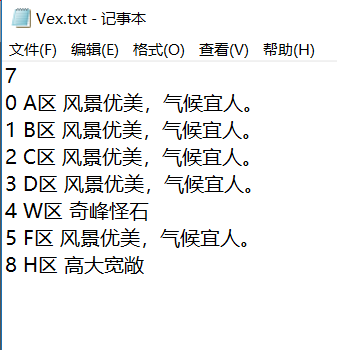
1

1 2

运行截图：



文件截图：



**4、旅游景点导航**

输入：起始景点的编号。

处理：使用深度优先搜索(DFS)算法，查询以该景点为起点，无回路游览整个景区的路线。

输出：所有符合要求的导航路线。

实现该功能代码如下：

void Navigation\_DFS(int n)//DFS算法从编号为n的景点开始无回路遍历全图

{

if(created)//图已被创建

{

if(n<0||n>G.maxvexno||AdjList[n].vex.tag==0)//编号n不在文件读入的编号范围内或顶点标记为0

cout<<"该景点不存在！"<<endl;

else

{

int count=1;//count记录本条路径已搜索过的顶点

int \*BVisited=new int[G.maxvexno+1];//横向标记

int \*DVisited=new int[G.maxvexno+1];//纵向标记

int \*no=new int[G.vexnum];//记录路径上的顶点编号

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

{

DVisited[i]=0;//复制前面纵向搜索的标记

BVisited[i]=0;//横向均未搜索过

}

no[0]=n;//记录起点

DVisited[n]=1;//标记纵向已访问过

ArcNode \*an=AdjList[n].next;

while(an)//横向搜索与该顶点相连的每个顶点

{

while(BVisited[an->n])//横向或纵向已访问过就找下一个顶点

{

an=an->next;

if(an=NULL)break;

}

if(an)

{

DFS(an->n,count,DVisited,no);//对此顶点进行DFS搜索

BVisited[an->n]=1;//标记此顶点横向已访问过

an=an->next;

}

}

}

}

else//图未被创建

cout<<"尚未读取文件创建图！"<<endl;

}

void DFS(int n,int count,int \*a,int \*no)//DFS搜索路径

{

int \*DVisited=new int[G.maxvexno+1];//纵向标记

int \*BVisited=new int[G.maxvexno+1];//横向标记

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

{

DVisited[i]=a[i];//复制前面纵向搜索的标记

BVisited[i]=0;//横向均未搜索过

}

no[count++]=n;

DVisited[n]=1;//纵向标记已访问过

ArcNode \*an=AdjList[n].next;

while(an)//横向搜索与该顶点相连的每个顶点

{

while(BVisited[an->n]||DVisited[an->n])//横向或纵向已访问过就找下一个顶点

{

an=an->next;

if(an==NULL)break;

}

if(an)

{

DFS(an->n,count,DVisited,no);//对此顶点进行DFS搜索

BVisited[an->n]=1;//标记此顶点横向已访问过

an=an->next;

}

}

if(count==G.vexnum)//如果此路径上顶点数与总顶点数相同，说明该路径无回路

{

for(int i=0;i<count-1;i++)//输出路径

cout<<AdjList[no[i]].vex.name<<"-->";

cout<<AdjList[count-1].vex.name<<endl;

}

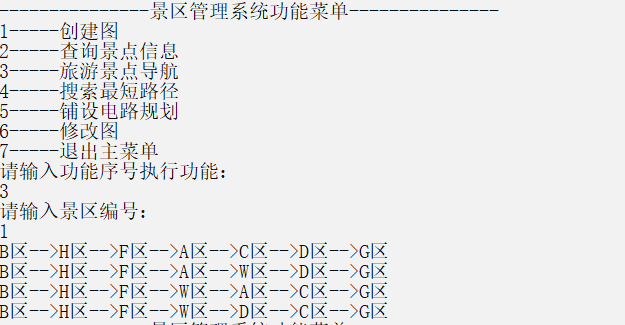
}

算法：对DFS进行改良，对一个景点连接的所有景点都进行搜索，每一个都可以搜索出一条新路（用两个数组，一个数组用来标记本条路上访问过的景点，另一个数组标记搜索连接的景点中已搜索过的景点），以此递归，最终如果本条路上的景点数等于景点总数，说明该路无回路。

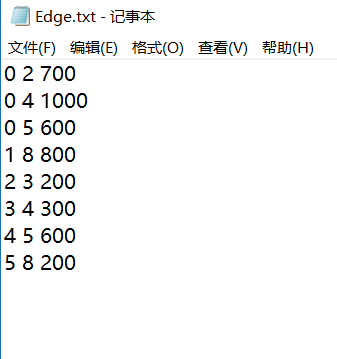
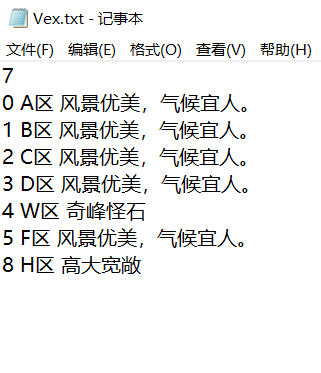
输入：

1

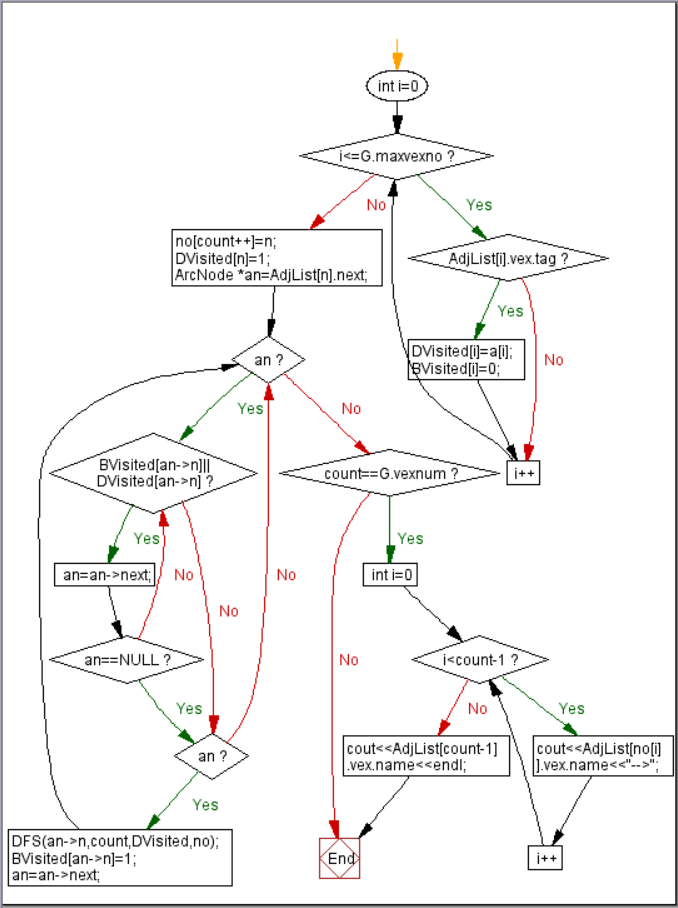
运行截图：



文件截图：



DFS核心代码流程图：



**5、搜索最短路径**

输入：

1. 起始景点的编号
2. 终点的编号。

处理：使用迪杰斯特拉(Dijkstra)算法，求得从起始景点到终点之间的最短路径，计算路径总长度。

输出：

1. 最短路线
2. 路径总长度

算法：先用Dijkstra算法，然后根据两个景点编号输出其最短路径。优点是可以找到任意两点间最短路径，缺点是遍历的节点太多，效率很低。

实现该功能代码：

void ShortestPath\_DIJ(int m,int n)//Dijkstra算法求编号m景区到编号n景区的最短路线

{

if(created)

{

if(m<0||m>G.maxvexno||AdjList[m].vex.tag==0||n<0||n>G.maxvexno||AdjList[n].vex.tag==0)

cout<<"景点不存在"<<endl;

else

{

int \*Pre=new int[G.maxvexno+1];//记录当前顶点的前一顶点

int \*Dis=new int[G.maxvexno+1];//起点到各个顶点的当前最短距离

int \*final=new int[G.maxvexno+1];// 记录是否已找到起点到各个顶点的最短路径

for(int i=0;i<=G.maxvexno;i++)//初始化

if(AdjList[i].vex.tag)

{

final[i]=0;//表示起点到各个顶点的最短距离均未找到

Dis[i]=G.arc[m][i];//初始化起点到各个顶点的最短距离，与起点相连的就是连线的距离，否则就是INFINITY

if(Dis[i]<INFINITY)

Pre[i]=m;//最短距离小于INFINITY，表明与起点相连，则其前一顶点为起点

else

Pre[i]=-1;//否则置为-1，表示无前一顶点

}

Dis[m]=0;//起点到自身距离为0

final[m]=1;//起点到起点已找到最短路径

for(int i=0;i<=G.maxvexno;i++)

if(AdjList[i].vex.tag)

{

int min=INFINITY;//记录离起点最近的距离

int v=-1;//记录离起点最近的点

for(int j=1;j<=G.maxvexno;j++)//从未找到最短路径的顶点中找到离起点最近的顶点

if(AdjList[j].vex.tag)

if(!final[j]&&Dis[j]<min)

{

min=Dis[j];

v=j;

}

final[v]=1;//起点到该点已找到最短距离

for(int j=0;j<=G.maxvexno;j++)

if(AdjList[j].vex.tag)

if(!final[j]&&(min+G.arc[v][j]<Dis[j]))//更新起点到各点的最短距离

{

Dis[j]=min+G.arc[v][j];

Pre[j]=v;//记录前一顶点

}

}

cout<<"从"<<AdjList[m].vex.name<<"至"<<AdjList[n].vex.name;

if(Dis[n]<INFINITY)//倒序输出路径

{

cout<<"最短路径总长度为"<<Dis[n]<<" 最短路径(倒序)如下："<<endl;

cout<<AdjList[n].vex.name;

int v=Pre[n];

while(v!=m)

{

cout<<"<--"<<AdjList[v].vex.name;

v=Pre[v];

}

cout<<"<--"<<AdjList[m].vex.name<<endl;

}

else

cout<<"不通！"<<endl;

}

}

else//图未被创建

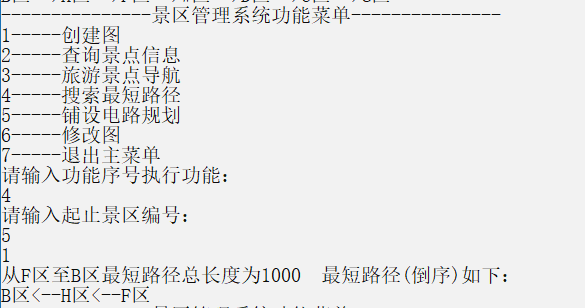
cout<<"尚未读取文件创建图！"<<endl;

}

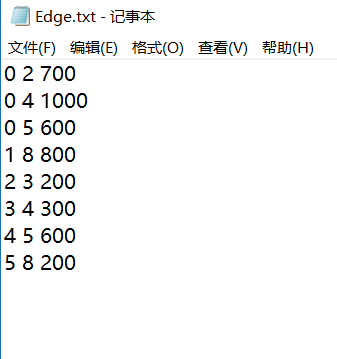
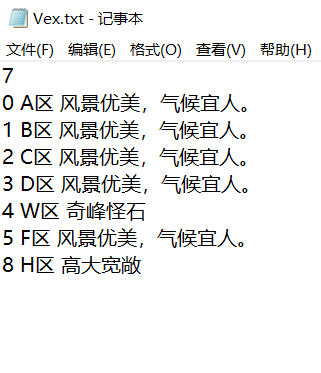
输入：

5 1

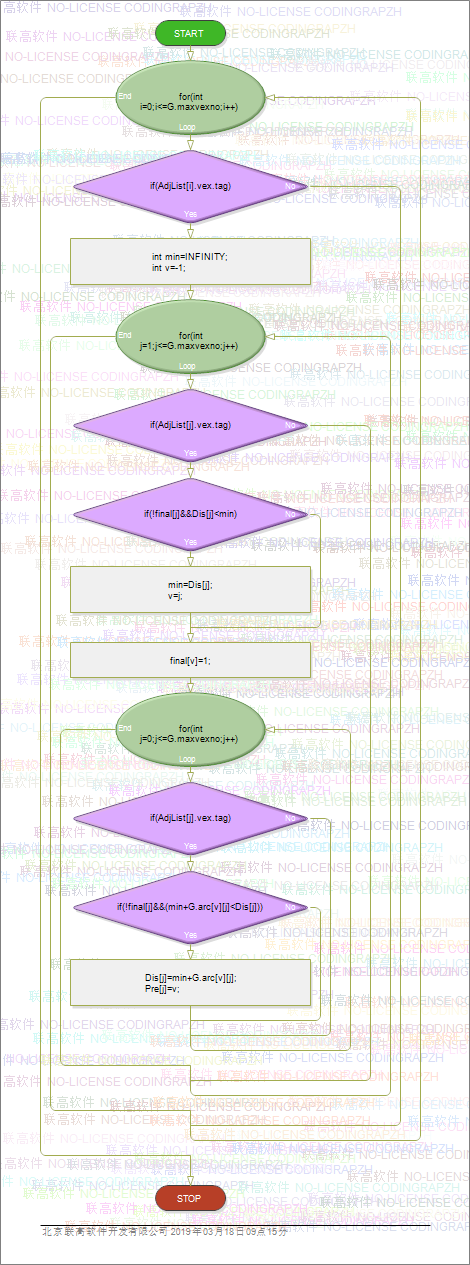
运行截图：



文件截图：



核心代码流程图：



**6、铺设电路规划**

处理：根据景区景点图使用普里姆(Prim)算法构造最小生成树，设计出一套铺设线路最短，但能满足每个景点都能通电的方案。

输出：

1. 需要铺设电路的道路
2. 每条道路铺设电路的长度
3. 铺设电路的总长度

算法：Prim算法，适用于边数多的图

实现该功能代码如下：

void WirePath\_Prim()//Prim算法求铺设最短电路

{

struct

{

int vexno;

int lowcost;

}closedge[MAX\_SIZE];//记录当前已经确定的点集到某顶点i的最短距离及对应顶点编号

int sum=0;//总长度

int t;//记录一个顶点编号，以此顶点为起点

for(int i=0;i<=G.maxvexno;i++)//找t

if(AdjList[i].vex.tag)

{

t=i;

break;

}

for(int i=0;i<=G.maxvexno;i++)//初始化已确定点集仅含t

if(AdjList[i].vex.tag)

{

closedge[i].vexno=t;

closedge[i].lowcost=G.arc[t][i];

}

closedge[t].lowcost=0;//表示该顶点属于已确定点集

for(int i=1;i<=G.vexnum;i++)//将其余G.vexnum-1个顶点并入已确定点集中

{

int vex2=-1,vex1=-1,min=INFINITY;

for(int j=0;j<=G.maxvexno;j++)//找与已确定点集距离最近的顶点

if(AdjList[j].vex.tag)

if(closedge[j].lowcost>0&&closedge[j].lowcost<INFINITY)

if(closedge[j].lowcost<min)

{

min=closedge[j].lowcost;

vex2=j;

vex1=closedge[j].vexno;//与该顶点相连的已确定点集中的某一顶点

}

if(vex1==-1&&vex2==-1)//搜索结束

break;

cout<<AdjList[vex1].vex.name<<"-->"<<AdjList[vex2].vex.name<<" "<<min<<endl;//输出最短距离及连接的两点

closedge[vex2].lowcost=0;//将该顶点并入已确定点集

sum+=min;//总长度更新

for(int j=0;j<=G.maxvexno;j++)//更新已确定点集到各顶点的最短距离

if(G.arc[vex2][j]<closedge[j].lowcost)

{

closedge[j].vexno=vex2;

closedge[j].lowcost=G.arc[vex2][j];

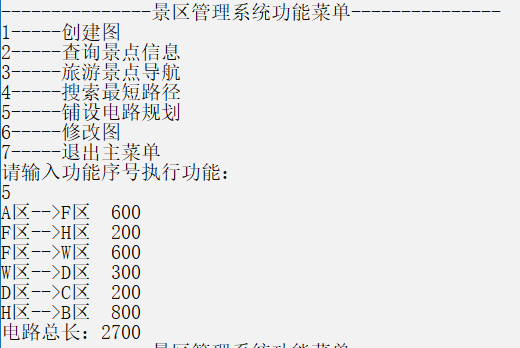
}

}

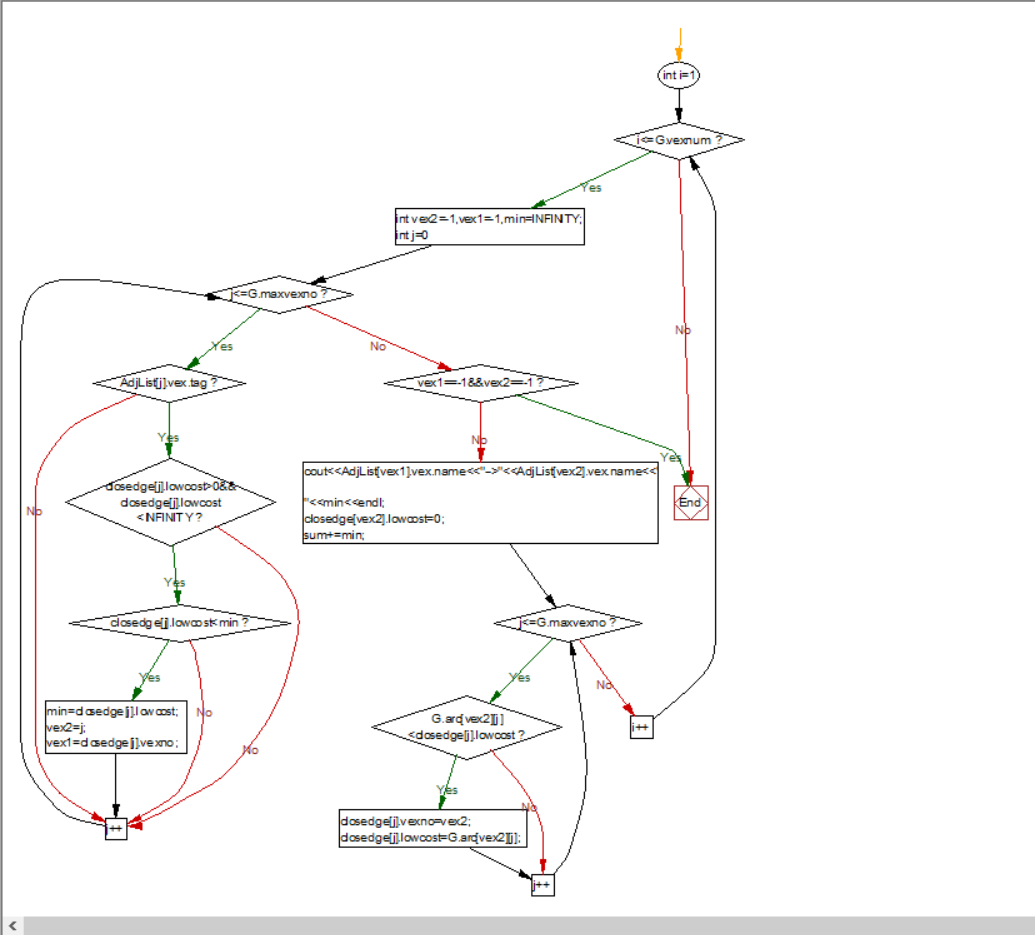
cout<<"电路总长："<<sum<<endl;//输出电路总长

}。

运行结果截图：



核心代码流程图：



**8.设计总结**

首先对DFS、迪杰斯特拉、Prim算法都再次深入理解了一遍，而且能更加熟练的运用和修改后适用于新问题，同时也对算法的利弊有所体会。

再一个就是学会了使用文件的读取和读写流，文件数据修改后应该重新写入文件，以及重新创建相应的图。

这个过程中，一个很重要的问题还是指针的熟练掌握和运用，对于所创建的数据结构必要时要在数据的末尾指针赋空，这样检索遍历时不会混乱。

在刚开始程序写完运行的时候很容易出现烫烫烫，就是因为传值没有传过来，数据没有初始化等原因。

另外，通过设计这个程序，让我对于程序的整体性、容错性以及尽可能让软件输入简洁明了有了更深的认识，尽可能让用户输入正确的数据，这样才能保证程序不会因为用户输错数据类型而崩溃；其次还有用户输入的数据必须有效，这就需要代码去识别，如果无效必须进行处理，否则也会引起崩溃。

主要考察的修改操作这一块，要求很强的逻辑性，必须处理好修改成功与否与文件是否重新读写的关系。因为整个程序是在连通图的大前提下，所以无论哪一步操作都要注意操作后图是否连通，如果连通方可修改，否则不能执行操作。修改后还要对应重新读写文件创建新图。

总之，在写程序时要全面考虑，不管要考虑需求，还要对程序的容错性最大限度地考虑到，保证程序在用户输错数据的情况下依然能够正常运行。