

**学期实训报告（数据结构）**

|  |  |
| --- | --- |
| 题 目 | **图 的 应 用** |

|  |  |
| --- | --- |
| 班 级 | **1 7 0 6** |
| 姓名 | **王洁 杨雨婷 郑术娟** |
| 学号 | **2017115050613** |
|  | **2017115050614** |
|  | **2017115050615** |
| 专业名称 | **物联网工程** |
| 指导老师 | **刘进军** |

2019 年 1 月 2 日

目录

[1需求分析 3](#_Toc534361150)

[1.1目标 3](#_Toc534361151)

[1.2工作任务 3](#_Toc534361152)

[1.2.1景点数据 4](#_Toc534361153)

[1.2.2系统功能 8](#_Toc534361154)

[2 概要设计 9](#_Toc534361155)

[3详细设计 10](#_Toc534361156)

[3.1基本数据类型及操作 10](#_Toc534361157)

[3.1.1数据类型定义 10](#_Toc534361158)

[3.1.2主要函数模块流程图 11](#_Toc534361159)

[3.2函数的调用关系 12](#_Toc534361160)

[4调试分析 12](#_Toc534361161)

[5用户手册 15](#_Toc534361162)

[6测试数据 16](#_Toc534361163)

[6.1创建景区景点图 16](#_Toc534361164)

[6.2查询景点信息 17](#_Toc534361165)

[6.3旅游景点导航 18](#_Toc534361166)

[6.4搜索最短路径 18](#_Toc534361167)

[6.5铺设电路规划 19](#_Toc534361168)

[7学期实训总结 19](#_Toc534361169)

[8附录 21](#_Toc534361170)

# 1需求分析

## 1.1目标

开发和学习“景区信息管理系统”，达到如下目标：

(1）掌握图的定义和图的存储结构

(2）掌握图的创建方法

(3）掌握图的两种遍历方法

(4）理解迪杰斯特拉（Dijkstra）算法

(5）理解最小生成树的概念和普里姆（Prim）算法

(6）掌握文件操作

(7）使用C/C什语言，利用图的数据结构，开发景区信息管理系统。

## 1.2工作任务

现有一个景区，景区里面有若干个景点，如图1-1所示，景点之间满足以下条件：

条件一：某些景点之间铺设了道路（相邻）

条件二：这些道路都是可以双向行驶的（无向图）

条件二：从任意一个景点出发都可以游览整个景区（连通图）

开发景区信息管理系统，对景区的信息进行管理。使用图的数据结构来保存景区景点信息，为用户提供创建图、查询景点信息、旅游景点导航、搜索最短路径、铺设电路规划等功能。

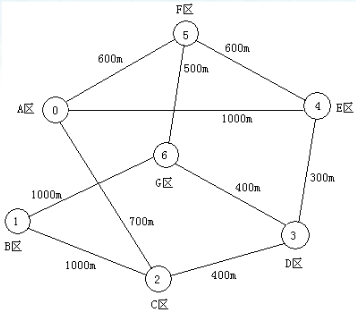


图1-1 景区示意图

### 1.2.1景点数据

景区的数据包括景点信息和景点 之间的道路信息。分别由两个文本文件存储。Vex.txt文件用来存储景点信息；Edge.txt文件用来存储道路信息。

景点信息：景点编号，名字和介绍。如表1-1所示。

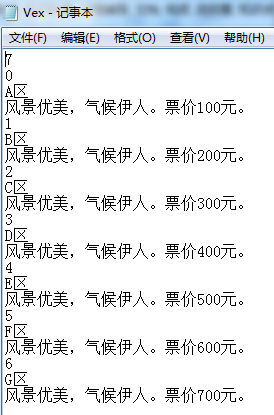
表1-1 景点信息表

|  |  |  |
| --- | --- | --- |
| 编号 | 名字 | 介绍 |
| 0 | A区 | … |
| 1 | B区 | … |
| 2 | C区 | … |
| 3 | D区 | … |
| 4 | E区 | … |
| 5 | F区 | … |
| 6 | G区 | … |

Vex.txt文件第一行：记录景区的景点个数。从第2行开始，3行记录一个景点信息。如表1-2，图1-2所示。

表1-2

|  |
| --- |
| 景点个数 |
| 景点编号  景点名字  景点介绍 |
| 景点编号  景点名字  景点介绍 |
| … |



1-2景区信息图

道路信息：景点1，景点2，两个景点之间的距离，如表1-3所示。

表1-3

|  |  |  |
| --- | --- | --- |
| 景点1 | 景点 | 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 |

Edge.txt文件中，每l行记录l条道路信息。格式为：“景点l的编号 景点2的编号 道路的长度”.（每个字段使用空格符分割）

若文件中没有某两个景点的信息，就表示这两个景点之间没有直接的路径。

文件格式及示例如表1-4，图1-3所示：

表1-4

|  |
| --- |
| 景点1的编号 景点2的编号 道路的长度 |
| 景点1的编号 景点2的编号 道路的长度 |
| 景点1的编号 景点2的编号 道路的长度 |
| … |

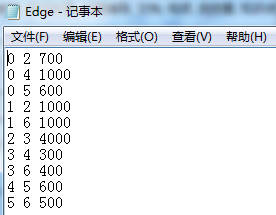


图1-3

### 1.2.2系统功能

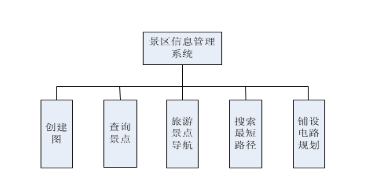


图1-4

1. 创建图

创建成功，依次输出：顶点数目，顶点编号，顶点名字，边两端的顶点编号，边的权值；

创建失败，输出失败的提示信息。

1. 查询景点

输入：想要查询的景点编号

输出：景点名字，景点介绍，相邻景区的名字，到达相邻景区的路径和长度。

1. 旅游景点导航

输入：起始景点的编号

输出：所有符合要求的导航路线。

1. 搜索最短路径

输入：起始景点的编号，终点的编号

输出：最短路线，路径总长度。

1. 铺设电路规划

输入：景区的所有景点信息和道路信息

输出：需要铺设电路的道路，每条道路铺设电路的长度，铺设电路的总长度。

# 2 概要设计

Graph.h:

int CreatGraph(Graph &graph);

int GetSpotInfo(Graph &graph);

int TravelPath(Graph &graph);

int FindShortPath(Graph &graph);

int DesignPath(Graph &graph);

Tourism.h:

void Init(Graph &graph);

bool InsertVex(Vex vex,Graph &graph);

bool InsertEdge(Edge edge,Graph &graph);

Vex GetVex(int v,Graph &graph);

int FindEdge(int v,Edge aEdge[],Graph &graph);

int GetVexNum(Graph graph);

void DFS(int v,bool aVisited[],int &nIndex,PathList &pList,Graph &graph);

//调用DFS（）函数，得到深度优先搜索遍历的结果

void DFSTraverse(int nVex,PathList &pList,Graph &graph);

void Prim(Edge aPath[],Graph &graph);

# 3详细设计

## 3.1基本数据类型及操作

### 3.1.1数据类型定义

① struct Vex{

int num; //景点编号

char name[20]; //景点名字

char desc[1024]; //景点介绍

};

②typedef struct Edge{

int vex1; //边的第一个顶点

int vex2; //边的第二个顶点

int weight; //权值

}Edge;

③typedef struct Graph

{

int m\_aAdjMatrix[20][20]; //邻接矩阵

Vex m\_aVex[20]; //顶点信息数组¨¦

int m\_nVexNum; //当前图的顶点个数

}Graph;

④typedef struct Path

{

int vex[20];//保存一条路径

Path\* next;

}\*PathList;

### 3.1.2主要函数模块流程图

操作页面

用户输入操作编号

判断输入正误

错误

搜索最短路径

铺设电路规划

旅游景点导航

查询景点信息

创建景区景点图

输出两个景点之间铺设的电路，输出电路的总长度

输入起始点终点编号

输入起始点编号

输出景点描述、周边景区和到周边景区的距离

显示顶点创建情况，边的创建情况

输出最短路线，输出最短距离

输出由该景点出发的所有导游路线

图3-1（主要函数流程图）

## 3.2函数的调用关系

Main

CreatGraph GetSpotInfo TravelPath FindShortPath DesignPath

Init GetVex GetVexNum DFSTraverse Prim

图3-2（函数调用关系图）

# 4调试分析

4.1问题1：在从文件中读取信息到Vex，Edge结构体中时，容易读取不正确报错

分析：采用格式化顺序读取信息不容易导致出错。

Vex：fscanf(fp1,"%d\n%s\n%s\n",&vex.num,&vex.name,&vex.desc)!=-1

Edge：while(fscanf(fp2,"%d\t%d\t%d\n",&edge.vex1,&edge.vex2,&edge.weight)!=-1)

4.2 问题2：输出路径时，可以输出，但是错的。

分析：图的遍历用了邻接矩阵来存储信息，邻接矩阵的前一个变量应该是最开始已经访问过的顶点，而不是由循环产生的变量。

算法如下：

else

{

for(int i=0;i<graph.m\_nVexNum;i++)

{

if((!aVisited[i])&&(graph.m\_aAdjMatrix[v][i]>0))

{

DFS(i,aVisited,nIndex,pList,graph);

aVisited[i]=false;

nIndex--;

}

}

}

4.3 问题3：计算最短路径出错，得不到预想结果

分析：利用迪杰斯特拉算法时要注意每当归并一个景点的时候就需要更新一次它的distance[],也就是最短路径，用于下一次循环归并最短路径相对应的顶。

算法如下：

for(int j=0;j<graph.m\_nVexNum;j++){

if((s[j]==0)&&(graph.m\_aAdjMatrix[u][j]+distance[u]<distance[j])

&&(graph.m\_aAdjMatrix[u][j]<MaxWeight)&&graph.m\_aAdjMatrix[u][j]>0||((distance[j]==0)&&graph.m\_aAdjMatrix[u][j]>0)){

distance[j]=graph.m\_aAdjMatrix[u][j]+distance[u];

path[j]=u;

}

4.4 问题4：输出最短路径的顺序发现经过了整个图

分析：图的生成过程是顺序生成最短路径，只有当生成到目的节点时才能知道它的路径，而且寻找的方法是通过path知道最短路径的上一个节点，再同过上一个节点知道他上一个节点，以此类推得出整个路径。

算法如下：

int t=end,n=0;

char ch[20][10];

while(t!=start){

t=graph.m\_aVex[path[t]].num;

strcpy(ch[n],graph.m\_aVex[t].name);

n++;

}

此处由于逆序找到最短路径，而输出时要正序输出，所以就用一个缓冲数组暂存路径的name，后再正序输出。

注：特别要注意的t取出了graph.m\_aVex[path[t]].num; 之后就不能再使用graph.m\_aVex[path[t]].num; 而是要用t。

输出算法：

printf("最短路线为a： ");

printf("%s",graph.m\_aVex[start].name);

for(int j=n-2;j>=0;j--){

printf("->%s",ch[j]);

}

printf("->%s",graph.m\_aVex[end].name );

4.5 问题5：在实现铺设电路规划的过程中，最初，输不出两个顶点之间的铺设电路

分析：GetVexNum（）函数中return 老是出现断点，一直以为是DesignPath函数错了，后面才发现，在Prim算法中，0顶点加入集合中，aVisited[0]=false应换为aVisited[0]=true; 不然直接跳出for循环。

# 5用户手册

5.1用户进入操作页面后，按照上面的提示选择操作编号（0-5）。

5.2如果用户选择操作编号1 创建景区景点图

则显示顶点创建情况 0-A区 1-B区 2-C区 3-D区 4-E区 5-F区 6-G区

边的创建情况为： <v0,v2>700 <v0,v4>1000 <v0,v5>600 <v1,v2>1000 <v1,v6>1000

<v2,v3>400 <v3,v4>300 <v3,v6>400 <v4,v5>600 <v5,v6>500

5.3如果用户选择操作2 查询景点信息，则给出ABCDEFG 区的相应编号

输入查询景点编号，输出该区景点描述及其周边景区和到周边景区的距离。

5.4如果用户选择操作3 旅游景区导航，则给出ABCDEFG 区的相应编号。

输入起始点编号 则输出由该景点出发的所有导游路线。

5.5如果用户选择操作4 搜索最短路径，则给出ABCDEFG 区的相应编号

输入起始点编号，输入终点编号，则给出最短路线，输出最短距离。

5.6如果用户选择操作5 铺设电路规划

输出两个景点之间铺设的电路，输出电路的总长度。

# 6测试数据

## 6.1创建景区景点图

如图6-1所示

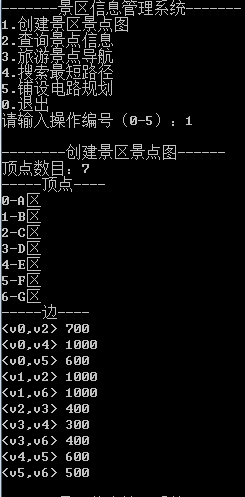


图6-1

## 6.2查询景点信息

如图6-2所示

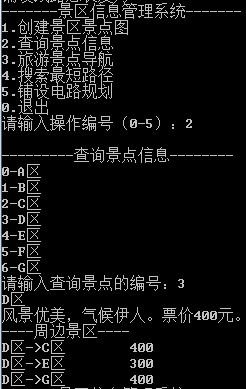


图6-2

## 6.3旅游景点导航

如图6-2所示

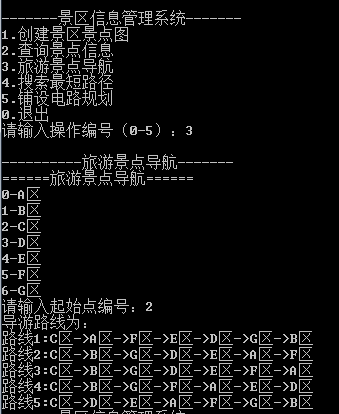


图6-3

## 6.4搜索最短路径

如图6-2所示

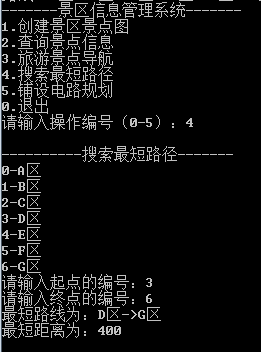


图6-4

## 6.5铺设电路规划

如图6-2所示

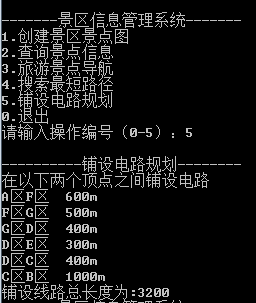


图6-5

# 7学期实训总结

郑术娟：

1. 通过实验加深了自己对图的学习。对图的遍历等原理有了更深刻的认识。
2. 创建景区景点图，先初始化图，打开文本文档，读取文件信息，创建景点，判断是否越界。
3. 在Prim算法中，是创建一个集合，将选出的具有最小权值的边的顶点加入到集合中，不断重复，直至生成最小生成树。集合里面放的就是最小生成树顶点的集合。用另外一个集合存放最小生成树边的集合。用3个for循环，找出最短的边，用集合内的一个顶点，去寻找与集合外边最小的定点，最后得到最小生成树。
4. Dijkstra算法，寻找最短路径，设置两个数组用于存放两个顶点的集合，一个用于存放已找到最短路径的顶点，另一个用于存放还未找到最短路径的顶点。寻找由一个顶点到另一个顶点的最短路径，如有A->C->E->D的路径小于A->D的路径，则最短路径行更新，回溯的时候计算路径的距离。
5. DepthFSearch算法，图的深度遍历。

杨雨婷：

1.图的应用有三大主要应用：图的两种生成与遍历，迪杰斯特拉（Dijkstra） 寻找最短路径，小生成树的概念和普里姆（Prim）。

2.图的生成与遍历主要是递归调用

3. 迪杰斯特拉（Dijkstra）算法实现主要注意三个动态数组的变化。

int \*path=(int \*)malloc(sizeof(int)\*graph.m\_nVexNum);

//最短路径

int \*distance=(int \*)malloc(sizeof(int)\*graph.m\_nVexNum);

//最短距离

int \*s=(int \*)malloc(sizeof(int)\*graph.m\_nVexNum);

//标志

首先是对三个动态数组初始化。初始化后用一个大的for循环用于控制循环找出起点到其他各个最小路径。For循环内具体实现时，第一步是根据distance数组找出最短路径后将其找到的边对应的点归并到s数组中。下一步，也是最重要的就是要修改从start到其他顶点的最短路径。完成后就完成了寻找路径的第一步。

4. 普里姆（Prim））算法实现主要注意三个for循环的控制。首先用一个大的for循环控制要生成的道路。For循环内嵌套的第二个for循环用于依次取集合内的点。第三个for循环内的第三个for循环用于控制依次寻找集合外的点，最终找出，集合内的点到集合外的点的最短路边后生成一条路径。

王洁：

1. 正确打开txt文件应该用fsacnf().
2. 图的遍历有深度遍历和广度遍历两种，深度遍历相当于树的前序遍历，先把一个点作为顶点，标记为已访问，再来判断剩下的点是否访问，一直找到没有被访问的点。若此时该点既没有被访问，又是该点的邻接节点，就开辟新空间让头指针指向它，把原来的那个顶点保存在数组中，该点作为新的顶点，继续重复去找一样的点，知道没有，然后再依次输出保存在数组中的点即可。
3. 再用if(nVexNum==graph.m\_nVexNum)去找下一条可行的路径。

# 8附录

8.1 Graph.h

#ifndef \_HEADERNAME\_H

#define \_HEADERNAME\_H

struct Vex{

int num; //景点编号

char name[20]; //景点名字

char desc[1024]; //景区介绍

};

typedef struct Edge{

int vex1; //边的第一个顶点

int vex2; //边的第二个顶点

int weight; //权值

}Edge;

typedef struct Graph

{

int m\_aAdjMatrix[20][20]; //邻接矩阵

Vex m\_aVex[20]; //顶点信息数组

int m\_nVexNum; //当前图的定点个数

}Graph;

typedef struct Path

{

int vex[20];//保存一条路径

Path \*next;//下一条路径

}\*PathList;

#endif

void Init(Graph &graph);

bool InsertVex(Vex vex,Graph &graph);

bool InsertEdge(Edge edge,Graph &graph);

Vex GetVex(int v,Graph &graph);

int FindEdge(int v,Edge aEdge[],Graph &graph);

int GetVexNum(Graph graph);

void DFS(int v,bool aVisited[],int &nIndex,PathList &pList,Graph &graph);

//调用DFS（）函数，得到深度优先搜索遍历的结果

void DFSTraverse(int nVex,PathList &pList,Graph &graph);

void Prim(Edge aPath[],Graph &graph);

8. 2 Tourism.h

#include "Graph.h"

class CTourism

{

private:

Graph m\_Graph;//Graph对象，用于存储景区景点图

};

int CreatGraph(Graph &graph);

int GetSpotInfo(Graph &graph);

int TravelPath(Graph &graph);

int FindShortPath(Graph &graph);

i*nt D*esignPath(Graph &graph);

8. 3main.cpp

#include <stdio.h>

#include <stdlib.h>

#include "Tourism.h"

#include "Graph.h"

int main(){

Graph graph;

int sel=1;

while(sel){

//循环输出菜单

printf("-------景区信息管理系统-------\n");

printf("1.创建景区景点图\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("0.退出\n");

printf("请输入操作编号（0-5）：");

scanf("%d",&sel);

switch(sel){

//跟据选项sel进行相应的操作

case 1:{

printf("\n--------创建景区景点图------\n");

CreatGraph(graph);

break;

}

case 2:{

printf("\n---------查询景点信息--------\n");

GetSpotInfo(graph);

break;

}

case 3:{

printf("\n----------旅游景点导航-------\n");

TravelPath(graph);

break;

}

case 4:{

printf("\n----------搜索最短路径-------\n");

FindShortPath(graph);

break;

}

case 5:{

printf("\n----------铺设电路规划--------\n");

DesignPath(graph);

break;

}

case 0:{

printf("退出系统，欢迎下次使用！\n");

break;

}

}

}

system("pause");

return 0;

}

8. 4 Graph.cpp

#include <stdio.h>

#include "Graph.h"

#include <stdlib.h>

#define MAX\_VERTEX\_NUM 20

void Init(Graph &graph){ //图的初始化

int i=0,j=0;

for(i=0;i<20;i++){

for(j=0;j<20;j++){

graph.m\_aAdjMatrix[i][j]=0;

}

}

graph.m\_nVexNum=0;

}

bool InsertVex(Vex vex,Graph &graph){

if(graph.m\_nVexNum==MAX\_VERTEX\_NUM){

printf("景点已满！\n");

return false;

}

graph.m\_aVex[graph.m\_nVexNum]=vex;

graph.m\_nVexNum++;

return true;

}

bool InsertEdge(Edge edge,Graph &graph){

if(edge.vex1<0||edge.vex2<0||edge.vex1>=graph.m\_nVexNum||edge.vex2>=graph.m\_nVexNum){

printf("graph.m\_nVexNum:%d\n",graph.m\_nVexNum);

printf("下标越界！\n");

return false;

}

graph.m\_aAdjMatrix[edge.vex1][edge.vex2]=edge.weight;

graph.m\_aAdjMatrix[edge.vex2][edge.vex1]=edge.weight;

return true;

}

Vex GetVex(int v,Graph &graph){

if(v>=graph.m\_nVexNum){

printf("没有此景点\n");

}

return graph.m\_aVex[v];

}

int FindEdge(int v,Edge aEdge[],Graph &graph){

int k=0;

for(int i=0;i<graph.m\_nVexNum;i++){

//得到边的信息

if(graph.m\_aAdjMatrix[v][i]!=0){

printf("%s->%s\t%d\n",graph.m\_aVex[v].name,graph.m\_aVex[i].name,graph.m\_aAdjMatrix[v][i]);

k++;

}

}

return k;//返回边的条数

}

int GetVexNum(Graph graph){

return graph.m\_nVexNum;

}

//使用深度优先搜索算法遍历图

void DFS(int v,bool aVisited[],int &nIndex,PathList &pList,Graph &graph)

{

aVisited[v]=true;//改为已访问

pList->vex[nIndex++]=v;//访问顶点nVex

int nVexNum=0;

for(int i=0;i<graph.m\_nVexNum;i++)

{

if(aVisited[i])

{

nVexNum++;

}

}

if(nVexNum==graph.m\_nVexNum)

{

pList->next=(PathList)malloc(sizeof(Path));

for(int i=0;i<graph.m\_nVexNum;i++)

{

pList->next->vex[i]=pList->vex[i];

}

pList=pList->next;

pList->next=NULL;

}

else

{

for(int i=0;i<graph.m\_nVexNum;i++)

{

if((!aVisited[i])&&(graph.m\_aAdjMatrix[nVexNum][i]>0))

{

DFS(i,aVisited,nIndex,pList,graph);

aVisited[i]=false;

nIndex--;

}

}

}

}

//调用DFS（）函数，得到深度优先搜索遍历的结果

void DFSTraverse(int nVex,PathList &pList,Graph &graph)

{

int nIndex=0;

bool aVisited[20]={false};

DFS(nVex,aVisited,nIndex,pList,graph);

}

//Prim算法

void Prim(Edge aPath[],Graph &graph)

{

//判断某顶点是否在最小生成树的集合里

bool aVisited[MAX\_VERTEX\_NUM];

for(int i=0;i<MAX\_VERTEX\_NUM;i++)

{

aVisited[i]=false;

}

aVisited[0]=true; //0顶点加入集合中

int min ;

int nVex1,nVex2;

for(int k=0;k<graph.m\_nVexNum;k++)

{

min = 0x7FFFFFFF;

for(int i=0;i<graph.m\_nVexNum;i++)

{

if(aVisited[i]) //从集合中取一个点

{

for(int j=0;j<graph.m\_nVexNum;j++)

{

if(!aVisited[j]) //从不在集合中的顶点中取出一个点

{

if((graph.m\_aAdjMatrix[i][j]<min)&&(graph.m\_aAdjMatrix[i][j]!=0))

{

nVex1=i;

nVex2=j;

min=graph.m\_aAdjMatrix[i][j]; //找出最短的边

}

}

}

}

}

//保存最短边的两个顶点

aPath[k].vex1=nVex1;

aPath[k].vex2=nVex2;

aPath[k].weight=graph.m\_aAdjMatrix[nVex1][nVex2];

//将两个顶点加入集合

aVisited[nVex1]=true;

aVisited[nVex2]=true;

}

}

8. 5 Tourism.cpp

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <string.h>

#include "Graph.h"

#include"Tourism.h"

#define MAX\_VERTEX\_NUM 20

//生成图

int CreatGraph(Graph &graph){

Vex vex;

Edge edge;

int i,k;

int ch;

char buf[1024];

int n=0,m;

FILE \*fp1,\*fp2;

Init(graph);

//生成顶点

fp1=fopen("Vex.txt","r");

if(fp1==NULL){

printf("打开失败！\n");

return 0;

}

if(fscanf(fp1,"%d",&ch)!=-1)

n=ch; //顶点个数

else {

printf("文件读取失败\n");

return 0;

}

//初始化Vex[n]

printf("顶点数目：%d\n",n);

printf("-----顶点----\n");

i=0;

while(fscanf(fp1,"%d\n%s\n%s\n",&vex.num,&vex.name,&vex.desc)!=-1)

{

if(!InsertVex(vex,graph)){

printf("新增节点失败！\n");

return 0;

}

printf("%d-%s\n",vex.num,vex.name);

i++;

}

fclose(fp1);

//生成边

i=0;

fp2=fopen("Edge.txt","rb");

if(fp2==NULL){

printf("打开失败！\n");

return 0;

}

printf("-----边----\n");

while(fscanf(fp2,"%d\t%d\t%d\n",&edge.vex1,&edge.vex2,&edge.weight)!=-1)

{

if(!InsertEdge(edge,graph)){

printf("新增节点失败！\n");

return 0;

}

printf("<v%d,v%d> %d\n",edge.vex1,edge.vex2,edge.weight);

}

fclose(fp2);

printf("\n");

return 1;

}

int GetSpotInfo(Graph &graph){

//printf("graph.m\_nVexNum：%d\n",GetVexNum(graph));

for(int i=0;i<GetVexNum(graph);i++){

printf("%d-%s\n",graph.m\_aVex[i].num,graph.m\_aVex[i].name);

}

int n=0;

printf("请输入查询景点的编号：");

scanf("%d",&n);

Vex find;

find=GetVex(n,graph);

printf("%s\n%s\n",find.name,find.desc);

//周边景区

Edge fedge[20];

int k; //边的条数

printf("----周边景区----\n");

k=FindEdge(n,fedge,graph);

return 1;

}

int TravelPath(Graph &graph)

{

Graph m\_Graph;

printf("======旅游景点导航======\n");

//获得景点数

int nVexNum=/\*m\_Graph.\*/GetVexNum(graph);

//列出所有景点编号的名称

for(int i=0;i<nVexNum;i++)

{

Vex svex=/\*m\_Graph.\*/GetVex(i,graph);

printf("%d",svex.num);

printf("-");

printf("%s\n",svex.name);

}

//输入景点编号

int nVex;

printf("请输入起始点编号：");

scanf("%d",&nVex);

if(nVex<0||nVex>nVexNum+1)

{

printf("输入错误！\n");

return 0;

}

PathList pList=(PathList)malloc(sizeof(Path));

PathList pHead=pList;

/\*m\_Graph.\*/DFSTraverse(nVex,pList,graph);

printf("导游路线为：\n");

int i=1;

pList=pHead;

while(pList->next)

{

Vex svex=/\*m\_Graph.\*/GetVex(pList->vex[0],graph);

printf("路线%d:",i);

i++;

printf("%s",svex.name);

for(int j=1;j<nVexNum;j++)

{

svex=/\*m\_Graph.\*/GetVex(pList->vex[j],graph);

printf("->%s",svex.name);

}

printf("\n");

Path \*temp=pList;

pList=pList->next;

free(temp);

}

free(pList);

pList=NULL;

pHead=NULL;

return 1;

}

int FindShortPath(Graph &graph){

for(int i=0;i<GetVexNum(graph);i++){

printf("%d-%s\n",graph.m\_aVex[i].num,graph.m\_aVex[i].name);

}

int start=0,end=0;

int \*path=(int \*)malloc(sizeof(int)\*graph.m\_nVexNum); //最短路径

int \*distance=(int \*)malloc(sizeof(int)\*graph.m\_nVexNum); //最短距离

int \*s=(int \*)malloc(sizeof(int)\*graph.m\_nVexNum); //标志

int MaxWeight=100000;

int u=0;

int minDis;

printf("请输入起点的编号：");

scanf("%d",&start);

printf("请输入终点的编号：");

scanf("%d",&end);

//初始化

for(int i=0;i<graph.m\_nVexNum;i++){

distance[i]=graph.m\_aAdjMatrix[start][i];

s[i]=0;

if(i!=start&&distance[i]<MaxWeight){

path[i]=start;

}

else{

path[i]=-1;

}

}

s[start]=1; //标记顶点start已从集合T加入到集合s当中

printf("最短路线为：");

printf("%s",graph.m\_aVex[start].name);

//for循环控制寻找路径

for(int i=1;i<graph.m\_nVexNum;i++){

minDis=MaxWeight;

//for循环控制在集合外面寻找最短边

for(int j=0;j<graph.m\_nVexNum;j++){

if((s[j]==0)&&(distance[j]<minDis)&&(distance[j]>0)){

u=j;

minDis=distance[j];

}

}

//当前已不存在路径时算法结束

if(minDis==MaxWeight){

printf("为非连通图！\n");

return 0;

}

s[u]=1; //T->S

//修改从start到其他顶点的最短路径

//for循环控制集合外面其他顶点到集合内的最短路的更新

for(int j=0;j<graph.m\_nVexNum;j++){

if((s[j]==0)&&(graph.m\_aAdjMatrix[u][j]+distance[u]<distance[j])&&

(graph.m\_aAdjMatrix[u][j]<MaxWeight)&&graph.m\_aAdjMatrix[u][j]>0

||((distance[j]==0)&&graph.m\_aAdjMatrix[u][j]>0)){

distance[j]=graph.m\_aAdjMatrix[u][j]+distance[u];

path[j]=u;

}

}

}

int t=end,n=0;

char ch[20][10];

while(t!=start){

t=graph.m\_aVex[path[t]].num;

strcpy(ch[n],graph.m\_aVex[t].name);

//注意此处的t将graph.m\_aVex[path[t]]的num取走，所以不可以直接用graph.m\_aVex[path[t]]了

n++;

}

for(int j=n-2;j>=0;j--){

printf("->%s",ch[j]);

}

printf("->%s",graph.m\_aVex[end].name );

printf("\n最短距离为：%d\n\n",distance[end]);

return 1;

}

int DesignPath(Graph &graph){

Edge aPath[MAX\_VERTEX\_NUM]; //定义边结构体数组用来保存铺设的线路

Prim(aPath,graph);

int nVexNum=GetVexNum(graph);

if(nVexNum==0)

{

printf("请先创建图!\n");

return 0;

}

int nAllLength=0;

printf("在以下两个顶点之间铺设电路\n");

int i;

for(int i=0;i<nVexNum-1;i++)

{

Vex sVex1=graph.m\_aVex[aPath[i].vex1];

Vex sVex2=graph.m\_aVex[aPath[i].vex2];

printf("%s%s\t%dm\n",sVex1.name,sVex2.name,aPath[i].weight);

nAllLength+=aPath[i].weight;

}

printf("铺设线路总长度为:%d\n",nAllLength);

return 1;

}