

**实 验 报 告**

**课程名称：** 数据结构

**实验名称：** 图及其应用

**院 （系）：** 计算机与通信工程学院

**姓 名：** 原彬贺

**学 号：** 542001020223

**专业班级：** 计算机科学与技术20-02

**指导教师：** 南姣芬 王治国

2022 **年** 5 **月** 20 **日**

实验报告正文

1. **实验目的**

掌握图的存储、构建、搜索等操作和应用，能用最短路径及其搜索等算法编制较综合性的程序，求解最优路线问题，进行程序设计、数据结构和算法设计等方面的综合训练。

**实验内容：**

1．任务：设计一个城市交通咨询模拟系统，利用该系统实现至少两种最优决策：最短路程到达、最省时到达等线路规划。

2．内容：

用户驾车出行由于出行目的的不同对道路路线选择的要求也有不同。例如，有的希望在途中的路程尽可能短，有的则可能希望路程中时间最短。为了能满足广大旅客的需求，编制一个城市交通咨询模拟系统，选取城市部分位置、道路抽象为程序所需要图的顶点和边，并以城市道路长度（路程），道路的某时段的速度等信息作为图结点中的弧信息，为旅客提供这两种最优决策的交通咨询。

**实验说明：**

1．输入和输出：

（1）输入形式：

* 构建图时，输入顶点、弧涉及的信息，包括：起始地、目的地、长度、该弧此时间段的平均速度等信息；
* 用户或者客户要输入出发地和目的地，并选择何种最优决策的路线规划。

（2）输出形式：根据用户需求输出对应信息

* 输出最短路程所需要的路线信息和最短路程；
* 输出最短时间所需要的路线信息和最短时间。

2．实验要求：

* + 实现一个简单的交互式界面，包括系统菜单、清晰的输入提示等。
  + 根据输入的交通图数据，以图形化形式把交通图显示在屏幕上。
  + 以图形化形式把最优路线显示在屏幕上。

能够上机编辑、调试出完整的程序。

3．类型定义

邻接表存储示例：

#define MAX\_VERTEX\_NUM 20 //顶点最大个数

typedef struct ArcNode{

int adjvex;

struct ArcNode \*nextarc;

int weight; //边的权值

}ArcNode; //表结点 #define VertexType int //顶点元素类型

typedef struct VNode{

VertexType data;

ArcNode \*firstarc;

}VNode, AdjList[MAX\_VERTEX\_NUM]; //

typedef struct{

AdjList vertices;

int vexnum, arcnum; //顶点的实际数，边的实际数

int kind;

}ALGraph;

邻接矩阵存储示例：

#define MAX\_VERTEX\_NUM 20 //顶点最大个数

typedef enum {DG, DN, UDG, UDN} GraphKind;

typedef struct ArcCell{

VRType adj;

int weight; //边的权值

}ArcCell; AdjMatrix[MAX\_VERTEX\_NUM] [MAX\_VERTEX\_NUM]; typedef struct{

VertexType vexs[MAX\_VERTEX\_NUM];

AdjMatrix arcs;

int vexnum, arcnum; //顶点的实际数，边的实际数

GraphKind kind;

}MGraph;

1. 主要数据类型与变量

（**必要时，可对数据类型和变量进一步解释或说明，增加可读性**）

const int inf=999999;

double mp[20][20];

int path[20][20];

int n,m;

1. 算法或程序模块

（**必要时，可对算法或程序模块进一步解释或说明，增加可读性**）

for(int i=0;i<m;i++){

cin>>Start>>End>>dis;

mp[Start][End]=dis;

}

for(int k=0;k<n;k++)//第k个点进行松弛

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

for(int j=0;j<n;j++)

if(mp[i][j]>mp[i][k]+mp[k][j])//如果能够缩短就更新距离

{

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

path[i][j]=k;//记录能松弛的点

}

1. **测试**

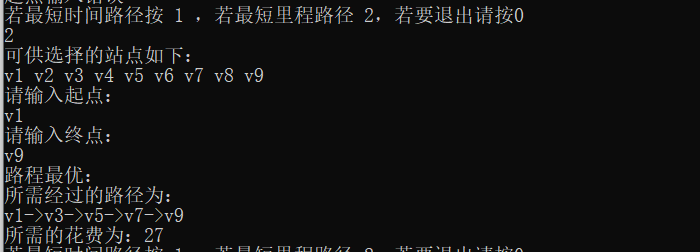
由指导书上信息进行输入

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 弧 | a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 | a11 |
| **距离**  （Km） | 12 | 8 | 6 | 5 | 8 | 7 | 5 | 6 | 10 | 6 | 8 |
| **速度**  （Km / h） | 40 | 50 | 60 | 20 | 25 | 55 | 20 | 20 | 40 | 30 | 25 |

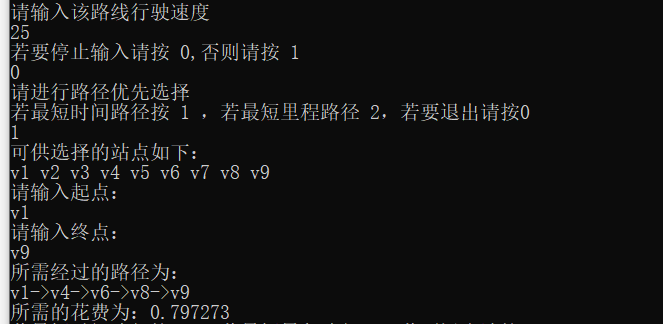


分别以从v1到v9为例：

查询最短路程



查询最短时间



1. **总结与讨论**

在本实验中，通过对使用最短路径算法，动态规划，每次将起点周围最短路径的站点就进行录入，最终得到每个站点离该起点的最短路径。通过将其最短路径判断，若其权值为 -1则其不可达，若其权值为正数则可达，通过入栈出栈将其路径打印出。这次实验对我编程能力提升很大，不单单使我熟练了最短路径算法，还让我对于大型程序有了一定的认识，知道了对整个程序进行模块化划分。对我学习有很大帮助。

**附：程序的源代码**

#include<stdio.h>

#include<iostream>

#include<vector>

#include<map>

#include<string>

#include<stack>

using namespace std;

class node{

public:

string name;

node(string i):name(i){

}

node(){

}

};

class knod{

public:

double s;

node name;

knod(node as,double sa):s(sa),name(as){

}

knod(){

string s=" ";

name=node(s);

}

};

class border{

public:

string name;

node pname;

node bname;

double time;

double velocity;

double distance;

border():name(NULL),pname(NULL),bname(NULL),distance(0),velocity(0){

}

border (string iname,node ipname,node ibname,double ti,double v):name(iname),pname(ipname),bname(ibname),distance(ti),velocity(v){

time=distance/velocity;

}

};

void maue(vector<border> &data);

void times(map<string,vector<border>> &tdata);

void output(map<string,knod>& target,string end,string begin);

void distances ( map<string,vector<border> > & tdata);

void stander(vector<border> &data,map<string,vector<border>> &tdata);

void toget(map<string,vector<border> > &tdata, map < string,knod>& target);

void test();

map<string,knod> ::iterator mine(map<string,knod> &target);

int main()

{

test();

return 0;

}

void test()

{

vector<border> data;

maue(data);

return ;

}

void maue(vector<border> &data)

{

int se;

double v,s;

string iname,pname,bname;

cout<<"欢迎进入地图寻路"<<endl;

cout<<"请进行地图的创建"<<endl;

cout<<"若要输入请按 1 ，若要退出系统请按 0"<<endl;

cin>>se;

if(!se)

{

return ;

}

while(se!=0)

{

cout<<"请输入该路线姓名"<<endl;

cin>>iname;

cout<<"请输入该路线起点姓名"<<endl;

cin>>pname;

cout<<"请输入该路线终点姓名"<<endl;

cin>>bname;

cout<<"请输入该路线所需路程"<<endl;

cin>>s;

cout<<"请输入该路线行驶速度"<<endl;

cin>>v;

data.push\_back(border(iname,node(pname),node(bname),s,v));

cout<<"若要停止输入请按 0,否则请按 1"<<endl;

cin.get();

if(cin.get()=='0')

{

se=0;

}

}

vector<border> ::iterator itr=data.begin();

map<string,vector<border>> tdata;

stander(data,tdata);

cout<<"请进行路径优先选择"<<endl;

cout<<"若最短时间路径按 1 ，若最短里程路径 2，若要退出请按0"<<endl;

cin>>se;

while(se)

{

if(se==1)

{

times(tdata);

}

else if(se==2)

{

distances(tdata);

}

else if(se==0)

{

return ;

}

else {

cout<<"请重新输入"<<endl;

}

cout<<"若最短时间路径按 1 ，若最短里程路径 2，若要退出请按0"<<endl;

cin>>se;

}

}

void stander(vector<border> &data,map<string,vector<border>> &tdata)

{

vector<border>::iterator itr=data.begin();

map<string,vector<border>>::iterator its;

while(itr!=data.end())

{

its=tdata.find(itr->pname.name);

if(its==tdata.end())

{

vector<border> dis;

dis.push\_back(\*itr);

tdata.insert(make\_pair(itr->pname.name,dis));

}

else{

its->second.push\_back(\*itr);

}

its=tdata.find(itr->bname.name);

if(its==tdata.end())

{

vector<border> dis;

dis.push\_back(\*itr);

tdata.insert(make\_pair(itr->bname.name,dis));

}

itr++;

}

return ;

}

void times(map<string,vector<border>> &tdata)

{

map<string,knod> target,targod;

toget(tdata,target);

string be,end;

map<string,knod> ::iterator itr,itc;

vector<border> ::iterator itz;

cout<<"请输入起点："<<endl;

cin>>be;

itr=target.find(be);

if(itr==target.end())

{

cout<<"起点输入错误"<<endl;

return;

}

else

{

(itr->second).s=0;

}

cout<<"请输入终点："<<endl;

cin>>end;

itr=target.find(end);

if(itr==target.end())

{

cout<<"终点输入错误"<<endl;

return;

}

if(be==end)

{

cout<<"其所耗费路程为 0"<<endl;

}

itr=mine(target);

map<string,vector<border>> ::iterator dwa;

while(!target.empty()&&itr!=target.end())

{

dwa=tdata.find(itr->first);

itz=(dwa->second).begin();

while(itz!=(tdata.find(itr->first)->second).end())

{

itc=target.find(itz->bname.name);

if(itc->second.s<0||itc->second.s>itr->second.s+itz->time)

{

itc->second.s=itr->second.s+itz->time;

itc->second.name.name=itr->first;

}

itz++;

}

targod.insert(\*itr);

if(itr->first==end)

{

break;

}

target.erase(itr);

itr=mine(target);

}

output(targod,end,be);

}

void output(map<string,knod>& target,string end,string begin)

{

double sa=0;

stack<string> que;

string ddi;

map<string,knod>::iterator iit=target.find(end);

while(iit->first!=begin&& iit!=target.end())

{

que.push(iit->first);

iit=target.find(iit->second.name.name);

}

que.push(begin);

if(iit==target.end())

{

cout<<"两站无法到达"<<endl;

}

else{

cout<<"所需经过的路径为："<<endl;

while(!que.empty())

{

ddi=que.top();

cout<<ddi;

que.pop();

if(!que.empty())

{

cout<<"->";

}

else

{

cout<<endl;

}

}

map<string,knod>::iterator iit=target.find(end);

cout<<"所需的花费为："<<iit->second.s<<endl;

return ;

}

}

void distances(map<string,vector<border> > &tdata)

{

map<string,knod> target,targod;

toget(tdata,target);

string be,end;

map<string,knod> ::iterator itr,itc;

vector<border> ::iterator itz;

cout<<"请输入起点："<<endl;

cin>>be;

itr=target.find(be);

if(itr==target.end())

{

cout<<"起点输入错误"<<endl;

return;

}

else

{

(itr->second).s=0;

}

cout<<"请输入终点："<<endl;

cin>>end;

itr=target.find(end);

if(itr==target.end())

{

cout<<"终点输入错误"<<endl;

return;

}

if(be==end)

{

cout<<"其所耗费路程为 0"<<endl;

}

itr=mine(target);

map<string,vector<border>> ::iterator dwa;

while(!target.empty()&&itr!=target.end())

{

dwa=tdata.find(itr->first);

itz=(dwa->second).begin();

while(itz!=(tdata.find(itr->first)->second).end())

{

itc=target.find(itz->bname.name);

if(itc->second.s<0||itc->second.s>itr->second.s+itz->distance)

{

itc->second.s=itr->second.s+itz->distance;

itc->second.name.name=itr->first;

}

itz++;

}

targod.insert(\*itr);

if(itr->first==end)

{

break;

}

target.erase(itr);

itr=mine(target);

}

cout<<"路程最优："<<endl;

output(targod,end,be);

return ;

}

map<string,knod> ::iterator mine(map<string,knod> &target)

{

map<string,knod> ::iterator itr=target.begin();

map<string,knod> ::iterator its=target.end();

double min=-1;

while(itr!=target.end())

{

if(itr->second.s>-1)

{

if(min <0)

{

min=itr->second.s;

its=itr;

}

else if(itr->second.s<min)

{

min=itr->second.s;

its=itr;

}

}

itr++;

}

return its;

}

void toget(map<string,vector<border> > &tdata,map<string,knod>& target)

{

map<string,vector<border>>::iterator itt=tdata.begin();

vector<border> ::iterator ire;

string s=" ";

while(itt!=tdata.end())

{

target.insert(make\_pair(itt->first,knod(s,-1)));

itt++;

}

cout<<"可供选择的站点如下："<<endl;

for(auto em:target)

{

cout<<em.first<<" ";

}

return ;

}