安 徽 新 华 学 院 实 验 报 告

姓 名 张卢 学号 1632104249 专业班级 16软件2班

课程名称 数据结构 实验名称 图的存储结构的实现及其应用

实验日期 2017.11.27 同组人员 指导老师 汪红霞 得分

**实 验 项 目 五**

**【实验名称】**图的存储结构的实现及其应用

**【实验类型】**基础性实验

**【实验目的与要求】**

* 掌握图的邻接矩阵或邻接表的存储方式；
* 实现图的一些基本运算，特别是深度遍历和广度遍历；
* 掌握以图为基础的一些常用算法。

**【实验主要仪器设备及台套数】**

一人一台安装Visual C++的电脑

**【实验学时】** 4学时

**【实验内容】**

1. 实现图的邻接矩阵或邻接表方法建立、深度遍历和广度遍历、图的基础常用算法的实现。

#include <iostream.h>

const int n=8; // 图中顶点数

const int e=10; // 图中边数

typedef int elemtype;

bool visited[n+1];

class graph

{

public:

elemtype v[n+1]; // 存放顶点信息v1,v2,…,vn，不使用v[0]存储空间

int arcs[n+1][n+1]; // 邻接矩阵

void creatadj1(graph &g); //无向图

void creatadj2(graph &g); //有向图

void creatadj3(graph &g); //无向网

void creatadj4(graph &g); //有向网

void dfs (graph g,int i);

void bfs (graph g,int i);

void print(graph &g) ; // 成员函数说明及定义

};

//建立无向图的邻接矩阵

void graph::creatadj1(graph &g)

{ int i, j,k ;

cout<<"请输入顶点信息";

for(k=1; k<=n; k++){

cin>>g.v[k];

}

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

for (j=1; j<=n; j++)

g.arcs[i][j]=0;

cout<<"请输入顶点对信息" ;

for (k=1; k<=e; k++)

{ cin>>i>>j;

g.arcs[i][j]=1;

g.arcs[j][i]=1;

}

}

//建立有向图的邻接矩阵

void graph::creatadj2(graph &g)

{ int i, j,k ;

cout<<"请输入顶点信息";

for(k=1; k<=n; k++)

cin>>g.v[k]; //输入顶点信息

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

for (j=1; j<=n; j++)

g.arcs[i][j]=0;

cout<<"请输入顶点对信息" ;

for (k=1; k<=e; k++)

{ cin>>i>>j; //输入一条弧<i,j>

g.arcs[i][j]=1;

}

}

/\*

//建立无向网的邻接矩阵

void graph::creatadj3(graph &g)

{ int i, j,k ;

float w;

cout<<"请输入顶点信息";

for(k=1; k<=n; k++)

cin>>g.v[k]; //输入顶点信息

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

for (j=1; j<=n; j++)

if (i==j) g.arcs[i][j]=0;

else g.arcs[i][j]=8; //∞代表顶点i和顶点j无边，上机时可以用一个很大的数代替(8)

cout<<"请输入顶点对信息" ;

for (k=1; k<=e; k++)

{ cin>>i>>j>>w; //输入一条边(i,j) 及权值w

g.arcs[i][j]=w;

g.arcs[j][i]=w;

}

}

//建立有向网的邻接矩阵

void graph::creatadj4(graph &g)

{

int i, j,k ;

float w;

cout<<"请输入顶点信息";

for(k=1; k<=n; k++)

cin>>g.v[k]; //输入顶点信息

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

for (j=1; j<=n; j++)

if (i==j) g.arcs[i][j]=0;

else

g.arcs[i][j]=8;

cout<<"请输入顶点对信息" ;

for (k=1; k<=e; k++) //输入e条边及权值

{

cin>>i>>j>>w; //输入一条弧<i,j> 及权值w

g.arcs[i][j]=w;

}

}

\*/

void graph::print(graph &g)

{

for(int i=1;i<=n;i++)

{ for(int j=1;j<=n;j++)

cout<<g.arcs[i][j]<<" ";

cout<<endl;

}

}

void graph::dfs(graph g,int i) //从顶点i 出发实现深度优先搜索遍历

{

int j;

cout<<g.v[i]<<' '; //输出访问顶点

visited[i]=true; //全局数组访问标记置1表示已经访问

for(j=1; j<=n; j++)

if ((g.arcs[i][j]==1)&&(!visited[j]))

dfs(g,j);

}

void graph::bfs(graph g,int i) //从顶点i出发实现图的广度优先搜索遍历

{ int q[n+1] ; //q为队列

int f,r,j ; //f、r分别为队列头、尾指针

f=r=0 ; //设置空队列

cout<<g.v[i]<<' '; //输出访问顶点

visited[i]=true ; //全局数组标记置true表示已经访问

r++; q[r]=i ; //入队列

while (f<r)

{ f++; i=q[f] ; //出队列

for (j=1; j<=n; j++)

if ((g.arcs[i][j]==1)&&(!visited[j]))

{ cout<<g.v[j]<<" " ;

visited[j]=true ;

r++; q[r]=j ;

}

}

}

void main()

{

cout<<"输入1建立无向图"<<endl;

cout<<"输入2建立有向图"<<endl;

cout<<"输入3邻接矩阵的输出"<<endl;

cout<<"输入4深度优先搜索的算法"<<endl;

cout<<"输入5广度优先搜索的算法"<<endl;

cout<<"输入0退出"<<endl;

graph g;

int a;

cout<<"请输入操作数: ";

cin>>a;int x;

while(a!=0)

{

switch(a)

{

case 1:

{cout<<"建立无向图"<<endl;

g.creatadj1(g);}break;

case 2:

{cout<<"建立有向图"<<endl;

g.creatadj2(g);}break;

case 3:

{cout<<"邻接矩阵输出为: "<<endl;

g.print(g);}break;

case 4:

{for (int i=1; i<=n;i++ )

visited[i]=false;

cout<<"请输入顶点遍历: "<<endl;

cin>>x;

cout<<"深度遍历结果为: "<<endl;

g.dfs(g,x);

cout<<endl;}break;

case 5:

{for (int i=1; i<=n;i++ )

visited[i]=false;

cout<<"请输入顶点遍历: "<<endl;

int x;

cin>>x;

cout<<"广度遍历结果为: "<<endl;

g.bfs(g,x);

cout<<endl;}break;

default:cout<<"输入有误!"<<endl;

}

cout<<"请输入操作数: ";

cin>>a;

}

cout<<"退出"<<endl;

}

**【实验心得】**

**此实验中**完成图的邻接矩阵存储结构，要求能够完成图的邻接矩阵的建立（有向图、无向图，有向网，无向网）、邻接矩阵的输出、深度优先搜索，广度优先搜索的算法，并在main方法中优化了改程序，使程序更加健壮，通过不断的深入研究，明白了，其中算法的意义，也记住了这些算法，并能够举一反三。