**实验六**

题目：图及其应用——图的构造与遍历

班级：17级计科一班 姓名：元雨暄 学号：1725111037 完成日期：2018.12.5

**一、需求分析**

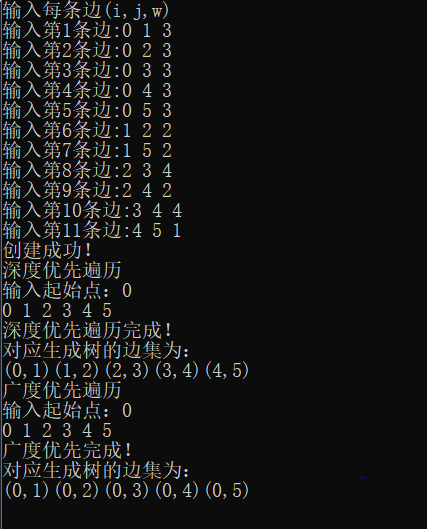
(1)**输入形式**： 按一条边的两个顶点一个权重的形式（i,j,w）来输入每一条边。

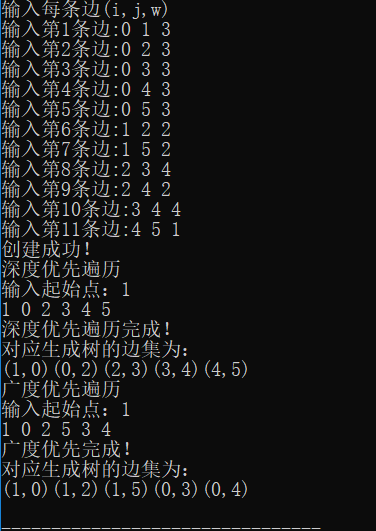
**输入值的范围**：整数

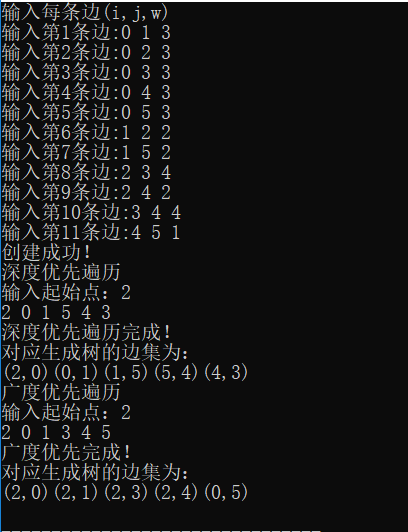
(2)**输出的形式**： 输出每个点的序号（整数型）

(3)**功能**：对题目中的图，能够在用户输入任意起始点的情况下，进行深度优先遍历和广度优先遍历并输出遍历顺序。

(4)**测试数据**:







**二、概要设计**

(1)抽象数据定义

struct queue

{

int data[6];

int front,rear;

};

struct ArcNode

{

int weight;

int adjvex;//邻接点

ArcNode \*next; //指向下一条边

};

struct VNode

{

int data;

ArcNode \*firstedge;

};

struct Graph

{

int vexnum,arcnum; //顶点个数和边数

VNode \*adjlist; //邻接表数组

};

struct EdgeArr

{

int start,end;

};

(2)主程序流程

int main()

{

Graph g;

int v;

CreateGraph(g);

cout<<"深度优先遍历\n输入起始点：";

cin>>v;

DFSTraverse(g,v);

cout<<"\n深度优先遍历完成！\n对应生成树的边集为："<<endl;

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

cout<<"("<<Darr[i].start<<","<<Darr[i].end<<")";

cout<<"\n广度优先遍历\n输入起始点：";

cin>>v;

BFSTraverse(g,v);

cout<<"\n广度优先完成！\n对应生成树的边集为："<<endl;

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

cout<<"("<<Barr[i].start<<","<<Barr[i].end<<")";

cout<<endl;

return 0;

}

(3)各程序模块之间的调用

mian函数调用CreateGraph函数，DFSTraverse函数，BFSTraverse函数

DFSTraverse函数 DFSTraverse函数 BFSTraverse函数

主程序模块

**三、详细设计**

(1)数据类型的定义

bool visited[6]={false,false,false,false,false,false};

struct queue //创建一个队列用来辅助广度优先遍历函数

{

int data[6];

int front,rear;

};

struct ArcNode

{

int weight;

int adjvex;//邻接点

ArcNode \*next; //指向下一条边

};

struct VNode

{

int data;

ArcNode \*firstedge;

};

struct Graph

{

int vexnum,arcnum; //顶点个数和边数

VNode \*adjlist; //邻接表数组

};

struct EdgeArr

{

int start,end;

};

EdgeArr Darr[20]; //用来输出生成树边集

EdgeArr Barr[20];

int d=0,b=0;

(2)操作的伪码算法

void CreateGraph(Graph &G) //创建一个图

{

int i;

int v1,v2,w;

G.vexnum=6;

G.arcnum=11;

G.adjlist=new VNode[G.vexnum];

ArcNode \*e,\*p,\*q;

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

{

G.adjlist[i].data=i;

G.adjlist[i].firstedge=NULL;

}

cout<<"输入每条边(i,j,w)\n";

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

{

cout<<"输入第"<<i+1<<"条边:";

cin>>v1>>v2>>w; //输入两个顶点和权重

if(G.adjlist[v1].firstedge==NULL) //如果第一个顶点没有弧边

{

e=new ArcNode; //new一个新的弧边

e->adjvex=v2;

e->weight=w;

e->next=NULL;

G.adjlist[v1].firstedge=e;

}

Else ////如果第一个顶点有弧边

{

p=G.adjlist[v1].firstedge;

while(p->next)

{

p=p->next;

}

e=new ArcNode;

e->adjvex=v2;

e->weight=w;

e->next=NULL;

p->next=e;

}

if(G.adjlist[v2].firstedge==NULL) //如果第二个顶点没有弧边

{

e=new ArcNode;

e->adjvex=v1;

e->weight=w;

e->next=NULL;

G.adjlist[v2].firstedge=e;

}

else //如果第二个顶点有弧边

{

p=G.adjlist[v2].firstedge;

while(p->next)

{

p=p->next;

}

e=new ArcNode;

e->adjvex=v1;

e->weight=w;

e->next=NULL;

p->next=e;

}

}

cout<<"创建成功！"<<endl;

}

void DFSTraverse(Graph &G,int v) //深度优先

{

cout<<G.adjlist[v].data<<" ";

visited[v]=true;

ArcNode \*arc;

arc=G.adjlist[v].firstedge;

while(arc)

{

if(!visited[arc->adjvex]) //如果输入的点的邻接点没有被访问过 则访问它

{

Darr[d].start=v;

Darr[d].end=arc->adjvex;

d++;

DFSTraverse(G,arc->adjvex);

}

arc=arc->next;

}

}

void BFSTraverse(Graph &G,int v) //广度优先

{

queue q;

bool visit[G.vexnum];

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

visit[i]=false;

q.front=q.rear=-1;

cout<<G.adjlist[v].data<<" ";

visit[v]=true;

q.data[++q.rear]=v;

while(q.rear!=q.front)

{

v=q.data[++q.front];

ArcNode \*p=G.adjlist[v].firstedge;

while(p!=NULL)

{

int m=p->adjvex;

if(!visit[m])

{

Barr[b].start=v;

Barr[b].end=m;

b++;

cout<<G.adjlist[m].data<<" ";

visit[m]=true;

q.data[++q.rear]=m;

}

p=p->next;

}

}

}

(3)函数调用关系图

main函数()

CreateGraph()DFSTraverse()BFSTraverrese()

**四、调试分析**

(1)问题：对广度优先遍历的算法不够熟练，理解不清楚

解决:查看书中的算法结合资料理解

(2)算法的时空分析

时间复杂度：

设n为元素个数，m为边数

CreateGraph(): O(m)

DFSTraverse(): O(n)

BFSTraverse(): O(n)

空间复杂度：

CreateGraph(): O(2\*m)

DFSTraverse(): O(n)

BFSTraverse(): O(n)

**五、用户使用说明**

当弹出界面时，用户可以根据界面的提示，输入（顶点，顶点，权重）形式的弧，当建图完成后，进入深度优先遍历，输入任意起始点，可得到相应顶点序列和生成树的边集，然后进入广度优先遍历，输入任意起始点，可得到相应顶点序列和生成树的边集，然后结束程序。

**六、测试结果**

