#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<malloc.h>

#include<string.h>

#define MAX\_VERTEX\_NUM 30

#define MAX\_NAME 20

typedef char VertexType[MAX\_NAME];

//定义边节点

typedef struct EBox{

bool mark;

int ivex,jvex;

struct EBox \*ilink,\*jlink;

}EBox;

//定义顶点节点

typedef struct VexBox{

VertexType data;

EBox \*firstedge;

}VexBox;

//定义表示图结构的结构体

typedef struct{

VexBox adjmulist[MAX\_VERTEX\_NUM];

int vexnum,edgenum;

}AMLGraph;

//根据数据域内容返回数组下标

int LocateVexBox(VertexType c,AMLGraph \*g)

{

int i;

for(i=0;i<(\*g).vexnum;i++)

{

if(strcmp(c,(\*g).adjmulist[i].data)==0)

return i;

}

return -1;

}

//创建图的函数

void CreateGraph(AMLGraph \*g)

{

int i,vi,vj;

VertexType va,vb;

EBox \*p;

printf("请输入图的顶点数：\n");

scanf("%d",&(\*g).vexnum);

printf("请输入图的边数：\n");

scanf("%d",&(\*g).edgenum);

printf("请输入图的%d个顶点：\n",(\*g).vexnum);

for(i=0;i<(\*g).vexnum;i++)

{

scanf("%s",&(\*g).adjmulist[i].data);

(\*g).adjmulist[i].firstedge=NULL;

}

printf("请输入图的%d条边,顶点之间用空格隔开：\n",(\*g).edgenum);

for(i=0;i<(\*g).edgenum;i++)

{

scanf("%s%s%\*c",&va,&vb);

p=(EBox \*)malloc(sizeof(EBox)); //创建边结点

p->mark=false; //将边标记为未访问

vi=LocateVexBox(va,g);

vj=LocateVexBox(vb,g);

p->ivex=vi;

p->jvex=vj;

p->ilink=(\*g).adjmulist[vi].firstedge;

(\*g).adjmulist[vi].firstedge=p;

p->jlink=(\*g).adjmulist[vj].firstedge;

(\*g).adjmulist[vj].firstedge=p;

}

}

int FirstAdjvex(int i,AMLGraph \*g) //获取顶点的第一条边

{

if(i<0)

return -1;

if((\*g).adjmulist[i].firstedge)

if((\*g).adjmulist[i].firstedge->ivex==i)

return (\*g).adjmulist[i].firstedge->jvex;

else return (\*g).adjmulist[i].firstedge->ivex;

else return -1;

}

int NextAdjvex(int i,int j,AMLGraph \*g) //获取顶点i的相对于j的邻接顶点

{

EBox \*p;

if(i<0||j<0)

return -1;

p=(\*g).adjmulist[i].firstedge;

while(p)

if(p->ivex==i&&p->jvex!=j)

p=p->ilink;

else if(p->jvex==i&&p->ivex!=j)

p=p->jlink;

else break;

if(p&&p->ivex==i&&p->jvex==j)

{

p=p->ilink;

if(p&&p->ivex==i)

return p->jvex;

else if(p&&p->jvex==i)

return p->ivex;

}

if(p&&p->ivex==j&&p->jvex==i)

{

p=p->jlink;

if(p&&p->ivex==i)

return p->jvex;

else if(p&&p->jvex==i)

return p->ivex;

}

return -1;

}

int visited[MAX\_VERTEX\_NUM];

//从下标为v的顶点出发递归的深度优先遍历图

void DFS(AMLGraph \*g,int v)

{

int w;

visited[v]=true;

printf("%s ",(\*g).adjmulist[v].data);

for(w=FirstAdjvex(v,g);w>=0;w=NextAdjvex(v,w,g))

if(!visited[w])

DFS(g,w);

}

//对图作深度优先遍历

void DFSTraverse(AMLGraph \*g)

{

int v;

for(v=0;v<(\*g).vexnum;v++)

visited[v]=false;

printf("深度遍历图的结果为：\n");

for(v=0;v<(\*g).vexnum;v++)

if(!visited[v])

DFS(g,v);

printf("\n");

}

//该函数实现图的所有边的输出

void AllEdgenum(AMLGraph \*g)

{

int v,i,j;

EBox \*p;

for(v=0;v<(\*g).vexnum;v++)

visited[v]=false;

printf("该图所有的边为：\n");

for(v=0;v<(\*g).vexnum;v++)

{

p=(\*g).adjmulist[v].firstedge;

while(p)

{

if(!(\*p).mark)

{

i=(\*p).ivex;

j=(\*p).jvex;

printf("%s-%s ",(\*g).adjmulist[i].data,(\*g).adjmulist[j].data);

(\*p).mark=true;

}

p=(\*p).ivex==v?(\*p).ilink:(\*p).jlink;

}

}

printf("\n");

}

int main()

{

AMLGraph g;

CreateGraph(&g);

DFSTraverse(&g);

AllEdgenum(&g);

return 0;

}

//测试数据

/\*

a b c d e

a b

b e

a d

d c

c e

b c

\*/

