# 一、

## 1、

#include<stdio.h>

#include<stdlib.h>

#define ERROR 0

#define OK 1

#define Overflow 2

#define Underflow 3

#define NotPresent 4

#define Duplicate 5

typedef int ElemType;

typedef struct

{

ElemType \*\*a;

int n;

int e;

ElemType noEdge;

}mGraph;

typedef int Status;

Status Init (mGraph \*mg,int nSize,ElemType noEdgeValue)

{

int i,j;

mg->n=nSize;

mg->e=0;

mg->noEdge=noEdgeValue;

mg->a=(ElemType\*\*)malloc(nSize\*sizeof(ElemType\*));

if(!mg->a)

return ERROR;

for(i=0;i<mg->n;i++)

{

mg->a [i]=(ElemType\*)malloc(nSize\*sizeof(ElemType\*));

for(j=0;j<mg->n;j++)

mg->a[i][j]=mg->noEdge;

mg->a[i][j]=0;

}

return OK;

}

void Destroy (mGraph \*mg )

{

int i;

for(i=0;i<mg->n;i++)

free(mg->a[i]);

free(mg->a);

}

Status Exist(mGraph \*mg,int u,int v)

{

if(u<0||v<0||v>mg->n-1||u==v||mg->a[u][v]==mg->noEdge)

return ERROR;

return OK;

}

Status Insert(mGraph \*mg,int u,int v,ElemType w)

{

if(u<0||v<0||v>mg->n-1||u==v) return ERROR;

if(mg->a[u][v]!=mg->noEdge)

return Duplicate;

mg->a[u][v]=w;

mg->e++;

return OK;

}

Status Remove(mGraph \*mg,int u,int v)

{

if(u<0||v<0||v>mg->n-1||u==v)

return ERROR;

if(mg->a[u][v]==mg->noEdge)

return NotPresent;

mg->a[u][v]=mg->noEdge;

mg->e--;

return OK;

}

Status Putout(mGraph \*mg)

{

int i,j;

for(i=0;i<mg->n;i++)

{ for(j=0;j<mg->n;j++)

{printf("%d",mg->a[i][j]);}

printf("\n");

}

}

void main()

{

mGraph mg;

Init (&mg,4,0);

Insert(&mg,1,0,1);

Insert(&mg,1,2,1);

Insert(&mg,2,3,1);

Insert(&mg,3,0,1);

Remove(&mg,3,0);

Putout(&mg);

//Exist(&mg,1,1);

// Destroy (&mg);

}

# 二、

## 1、

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#define MAX\_NUM 20

typedef struct ArcNode {

int adjvex;

struct ArcNode \*nextarc;

}ArcNode;

typedef int VertexType;

typedef struct VNode {

VertexType data;

ArcNode \*firstarc;

}VNode,AdjList[MAX\_NUM];

void createDgraph(AdjList g,int n){

ArcNode \*p,\*q;

int i,j;

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

{

g[i].data=i;

g[i].firstarc=NULL;

}

printf("\nEdgei->j:");

scanf("%d%d",&i,&j);

while (i!=-1)

{

p=(ArcNode \*)malloc(sizeof(ArcNode));

q=(ArcNode \*)malloc(sizeof(ArcNode));

p->adjvex=j;

p->nextarc=g[i].firstarc;

g[i].firstarc=p;

q->adjvex=i;

q->nextarc=g[j].firstarc;

g[j].firstarc=q;

printf("\nEdge i->j:");

scanf("%d%d",&i,&j);

}

}

int visited[MAX\_NUM]={0};

int count=0;

void DFS(AdjList G,int v){

ArcNode \*p;

visited[v]=1;

count++;

printf("%d ",v);

p=G[v].firstarc;

while(p!=NULL){

if(visited[p->adjvex]==0){

DFS(G,p->adjvex);

}

p=p->nextarc;

}

}

void BFS(AdjList G,int v){

ArcNode \*p;

int Qu[20],front,rear;

int visited[20]={0};

int w;

front=rear=0;

printf("%d ",v);

visited[v]=1;

rear=(rear+1)%20;

Qu[rear]=v;

while(front!=rear){

front=(front+1)%20;

w=Qu[front];

p=G[w].firstarc;

while(p){

if(visited[p->adjvex]==0){

printf("%d ",p->adjvex);

visited[p->adjvex]=1;

rear=(rear+1)%20;

Qu[rear]=p->adjvex;

}

p=p->nextarc;

}

}

}

void printDgraph(AdjList g,int n){

ArcNode \*p;

int i;

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

{

printf("\n%d: ",g[i].data);

p=g[i].firstarc;

while (p)

{

printf("->%d",p->adjvex);

p=p->nextarc;

}

}

}

int main()

{

AdjList g;

int num;

printf("Input Number of Vertex:");

scanf("%d",&num);

createDgraph(g,num);

printDgraph(g,num);

printf("\n");

printf("深度优先:");

DFS(g,num);

printf("\n");

printf("广度优先:");

BFS(g,num);

printf("\n");

if(count==num)

printf("且得该无向图连通\n");

else

printf("且得该无向图不连通\n");

return 0;

}