**UNIVERSITATEA DE STAT DIN MOLDOVA**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALITATEA INFORMATICA**

**Pavlovschi Cătălin**

**Dare de seama**

***Lucrare de laborator nr.2:***

***Algoritmica Grafurilor***

**Lucrare de laborator nr.2**

**Varianta 1**

* Determinați toate mulțimile stabile interior într-un graf neorientat prin algoritmul lui Bednarek și Taulbee.

#include<stdio.h>

#include<conio.h>

int a[50][50];

int n=5,m=4;

int dim=50;

int L [50][50], nl;

int Ls [50][50], nls;

int I [50][50], ni;

int Is [50][50], nis;

int y[50],c[50];

int k;

void showa (){

int i,j;

printf("\nMatricea de adiacenta a grafului este:\n\n");

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

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

printf("%d",a[i][j]);

printf("\n");

}

printf("\n");

}

void intersectie(int \*a, int \*b, int \*c){

int i,j,k;

c[0]=0;

for (i=1;i<=a[0];i++)

for (j=1;j<=b[0];j++)

if (a[i]==b[j]){

c[0]++;

c[c[0]]=a[i];

}

if(c[0]==0){ //intersectia este multime vida

c[0]=1;

c[1]=-1; //multimea vida este insemnata prin -1

}

}

int not\_include(int \*a,int \*b){

int i,j,c=0,l;

for (i=1;i<=a[0];i++)

for (j=1;j<=b[0];j++)

if (a[i]==b[j])

c++;

if (a[0]==c) l=0; else l=1;

if (a[0]==0) l=1;

return l;

}

void atribuire(int \*a, int \*b){ //al doilea se copie in primul

int i;

for(i=0;i<=b[0];i++) //b[0]-nr elemnt din b

a[i]=b[i];

}

void zeroa(){

int i,j;

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

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

a[i][j]=0;

}

void seta(){

int i;

int x1,x2; //extremitatile muchiei

zeroa();

printf("Introdu nr. de virfuri:n=");

scanf("%i",&n);

printf("Introdu nr. de muchii :m=");

scanf("%i",&m);

printf("\n");

for (i=1;i<=m;i++){

printf("\nIntroduceti extremitatile muchiei %d:",i);

scanf("%d%d",&x1,&x2);

a[x1][x2]=1;

a[x2][x1]=1;

}

}

///--Pasul 1 --

void p1(){

L[1][0]=1; // pe linia 1 se contine un element

L[1][1]=1; // acest element este 1

nl=1; // nr de linii utilizate este 1

}

///--Pasul 2 --

void p2(){

int i;

y[0]=0;

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

if (a[k][i]==0){

y[0]++;

y[y[0]]=i;

}

}

///--Pasul 3 --

void p3(){

int i;

nis=0;

for(i=1;i<=nl;i++){

intersectie(L[i],y,c);

if(c[0]>0&&c[1]!=-1){ //daca in c este cel putin un element si acest element nu e multimea vida atunci are loc atribuireuirea

nis++;

atribuire(Is[i],c);

}

}

}

///--Pasul 4 --

void p4(){

int i,j,u,u1=1;

ni=0;

for (i=1;i<=nis;i++){

for (j=1;j<=nis;j++){

u=not\_include(Is[i],Is[j]);

if((u==0)&&(i!=j)) //daca are loc conditia din if atunci isi schimba valoarea

u1=0;

}

if (u1&&(nis>0)){

ni++;

atribuire(I[ni],Is[i]);

}

u1=1;

}

}

///--Pasul 5 --

void p5(){

int i,j;

int u,v,v1=1;

nls=0;

for(i=1;i<=nl;i++){

u=not\_include(L[i],y);

if (u==0){

nls++;

atribuire(Ls[nls],L[i]);

Ls[nls][0]++;

Ls[nls][Ls[nls][0]]=k;

}

else{

nls++;

atribuire(Ls[nls],L[i]);

intersectie(L[i],y,c);

v1=1;

if(c[1]==-1)v1=0;

for (j=1;j<=ni;j++){

v=not\_include(c,I[j]);

if(v==0)

v1=0; //intersectea este mult. vida

}

if (v1==0){

nls++;

atribuire(Ls[nls],c);

if(c[0]==1&&c[1]==-1){

Ls[nls][Ls[nls][0]]=k;

}

else{

Ls[nls][0]++;

Ls[nls][Ls[nls][0]]=k;

}

}

}

}

}

///--Pasul 6 --

void p6(){

int i,j;

int u,u1=1;

nl=0;

for (i=1;i<=nls;i++){

for (j=1;j<=nls;j++){

u=not\_include(Ls[i],Ls[j]);

if((u==0)&&(i!=j))

u1=0;

}

if (u1){

nl++;

atribuire(L[nl],Ls[i]);

}

else

if(nls==1){

nl++;

atribuire(L[nl],Ls[i]);

}

u1=1;

}

}

void showresult(){

int i,j;

printf("Multimile stabile interior maximale sunt:\n\n");

for (i=1;i<=nl;i++){

printf("S={");

for(j=1;j<=L[i][0];j++)

printf("%2.d",L[i][j]);

printf("}\n");

}

}

int main(){

seta();

showa();

p1();

for(k=2;k<=n;k++){

p2();

p3();

p4();

p5();

p6();

}

showresult();

getch();

}

**Pasul 1.** Fixăm , , . Considerăm .

**Pasul 2.** Fixăm mulţimea



**Pasul 3.**  Construim familia de mulţimi

 este un element al familiei .

**Pasul 4.** Determinăm  - familia tuturor mulţimilor maximale din .

**Pasul 5.** Construim familia de mulţimi  prin examinarea fiecărui element *M* din :

**a)** dacă , atunci  ;

**b)** dacă , atunci  şi dacă în acest caz se respectă şi condiţia , atunci se mai consideră că .

**Pasul 6.** Determinăm  - familia tuturor mulţimilor maximale din .

**Pasul 7.** Dacă , atunci considerăm  şi ne întoarcem la executarea ***pasului 3***. În caz contrar,  conţine toate mulţimile maximale stabile interior în graful *G*.