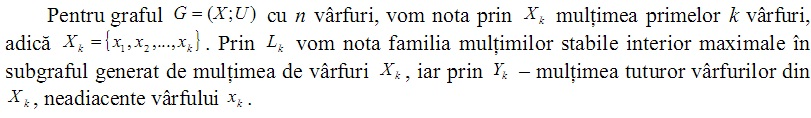
Laborator 4

1. Formularea problemei

Să se conceapă un algoritm care să determine toate multimile interior stabile maximale a unui graf dat prin matricea sa de adiacență.

1. Dezvoltarea algoritmului

Algoritmul presupune urmatorii pași simpli:



1.Citirea matricei de adiacenta a grafului.

2.Pentru fiecare multime de varfuri  vom afla (familia multimilor stabile interior maximale in subgraful generat de multimea de varfuri ).

3.Aflarea  – multimea tututor varfurilor din , neadiacente varfului .

4.Afisarea multimilor interior stabile maximale.

1. Descrierea și demonstrarea algoritmului

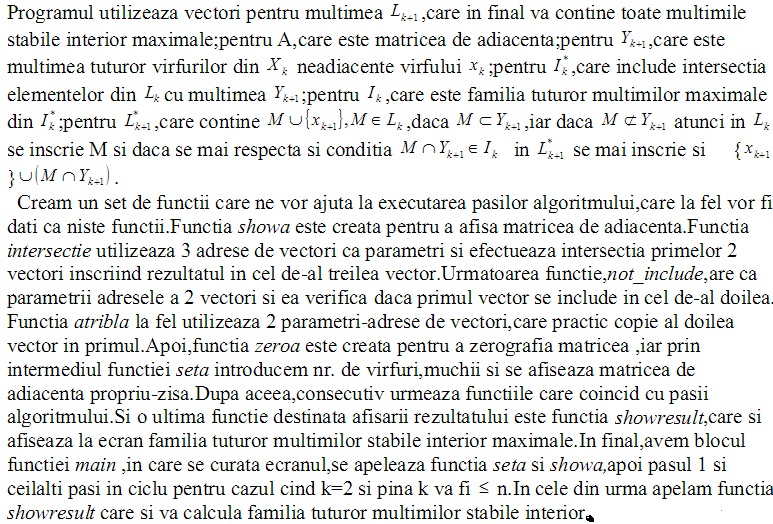
O submultime de vârfuri S \subset \!\, X a unui graf G = (X;U) se

numeste interior stabila daca oricare doua vârfuri x, y \in \!\, S nu sunt

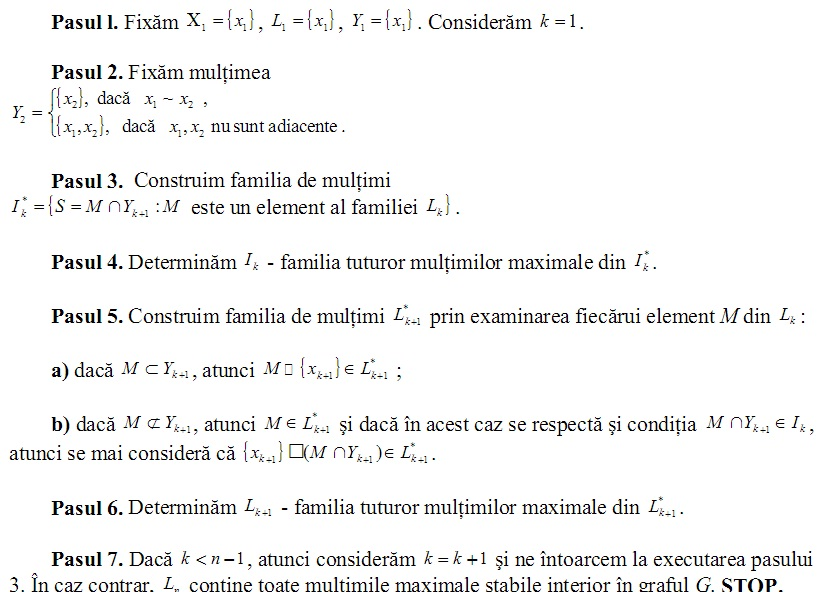
adiacente în G . Multimea interior stabila S se numeste maximala,

daca în G nu exista o alta multime interior stabila A astfel încât

S\subset \!\, A.



Descriem în continuare algoritmul în pseudocod:



1. Codul sursă

#include<cstdio>

**using** **namespace** std**;**

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 vid este insemnata prin -1}}

**{**

c**[**0**]** **=**1**;**

c**[**1**]** **=** **-**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 atrib**(**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**; //** Zerografiaza matricea de incidenta

**}**

//Initializeaza matricea de incidenta

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**;**

**}**

**}**

//Pasii algoritmului

//Pasul 1 ----------

void pasul1**()**

**{**

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 pasul2**()**

**{**

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 pasul3**()**

**{**

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 atunciare loc atribuirea

**{**

nis**++;**

atrib**(**Is**[**i**],**c**);**

**}**

**}**

**}**

//Pasul 4 -----------

void pasul4**()**

**{**

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**++;**

atrib**(**I**[**ni**],**Is**[**i**]);**

**}**

u1**=**1**;**

**}**

**}**

//Pasul 5 -----------

void pasul5**()**

**{**

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**++;**

atrib**(**Ls**[**nls**],**L**[**i**]);**

Ls**[**nls**][**0**]++;**

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

**}**

**else**

**{**

nls**++;**

atrib**(**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**++;**

atrib**(**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 pasul6**()**

**{**

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**++;**

atrib**(**L**[**nl**],**Ls**[**i**]);**

**}**

**else** **if(**nls**==**1**)**

**{**

nl**++;**

atrib**(**L**[**nl**],**Ls**[**i**]);**

**}**

u1**=**1**;**

**}**

**}**

//Afisarea rezultatului

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"**);**

**}**

**}**

//MAIN

int main**()**

**{**

freopen**(**"mult.in"**,**"r"**,**stdin**);**

freopen**(**"mult.out"**,**"w"**,**stdout**);**

seta**();**

// showa();

pasul1**();**

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

**{**

pasul2**();**

pasul3**();**

pasul4**();**

pasul5**();**

pasul6**();**

**}**

showresult**();**

**return** 0**;**

**}**

1. Date de test

|  |  |
| --- | --- |
| Date de intrare – mult.in | Date de ieșire – mis.out |
| 0 1 0 0 0  1 0 1 0 0  0 1 0 1 1  0 0 1 0 0  0 0 1 0 0 | Multimile stabile interior maximale sunt:  S={ 1 3}  S={ 1 4 5}  S={ 2 4 5} |
| 8  0 1 0 1 0 0 0 0  1 0 1 1 0 0 0 0  0 1 0 1 0 0 0 0  1 1 1 0 1 1 0 0  0 0 0 1 0 0 1 0  0 0 0 1 0 0 1 0  0 0 0 0 1 1 0 1  0 0 0 0 0 0 1 0 | Multimile stabile interior maximale sunt:  S={ 1 3 5 6 8}  S={ 1 3 7}  S={ 2 5 6 8}  S={ 2 7} |