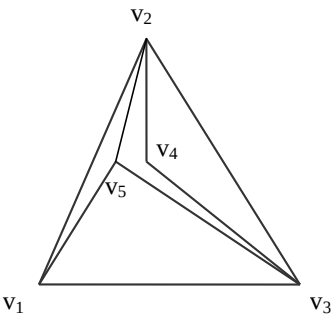
Seminar 11

Conectivitate, drumuri in graf

1. **Pentru graful,**



Γ_1: v_1, v_2, v_3, v_2, v_5, v_3, v_4 este un v_1- v_4 drum care nu este proces;(drum in care se repeta din arce)

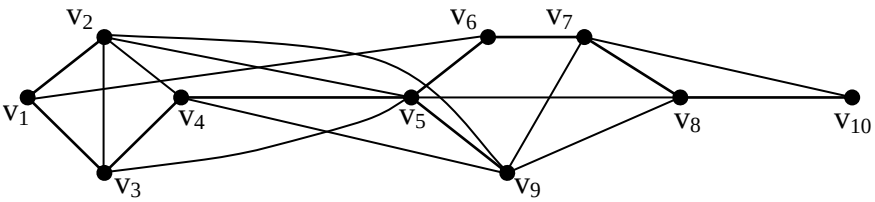
Γ_2: v_1, v_2, v_5, v_1, v_3, v_4 este un v_1- v_4 proces care nu este drum elementar;(drum in care se repeta noduri)

Γ_3: v_1, v_3, v_4 este un v_1- v_4 drum elementar;(drum in care nu se repeta noduri)

Subgraf = cand se sterg dintr-un graf niste noduri si arcele lor.

Graf partial = cand se sterg dintr-un graf arcele.s

2. **În graful,**

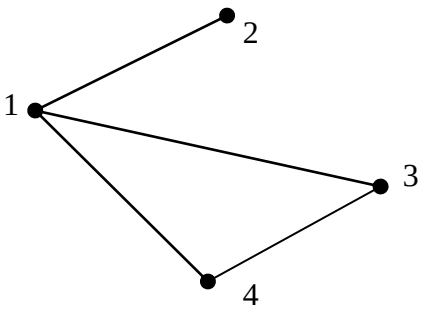


dacă Γ: v_1, v_2, v_4, v_5, v_3, v_1, v_2, v_5, v_6, v_7, v_8, v_9, v_5, v_9, v_8, v_{10}, atunci

Γ_1: v_1, v_2, v_5, v_9, v_8, v_{10}

Γ_2: v_1, v_2, v_4, v_5, v_9, v_8, v_{10} sunt v_1-v_{10} subdrumuri elementare.

3. **Pentru graful,**



Determinați matricea existenței drumurilor.

Răspuns:

Matricea existenţei drumurilor; algoritmul Roy-Warshal

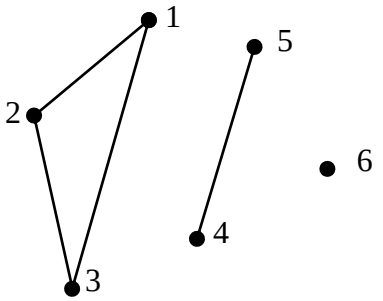
Matricea $M = \overline{A}^{(1)} \oplus \overline{A}^{(2)} \oplus \mathrm{K} \oplus \overline{A}^{(n-1)}$ se numeşte *matricea existenţei drumurilor* în graful *G*. Semnificaţia componentelor matricei *M* este:

$$\forall 1 \le i, j \le n, \quad m_{ij} = \begin{cases} 0, \text{dacă nu există } v_i - v_j \text{ drum în } G \\ 1, \text{altfel} \end{cases}$$

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad \overline{A}^2 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad \overline{A}^3 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Observaţie Calculul matricei existenţei drumurilor permite verificarea dacă un graf dat este conex. Graful este conex dacă şi numai dacă toate componentele matricei *M* sunt egale cu 1.

4. **Care sunt componentele conexe ale grafului:**

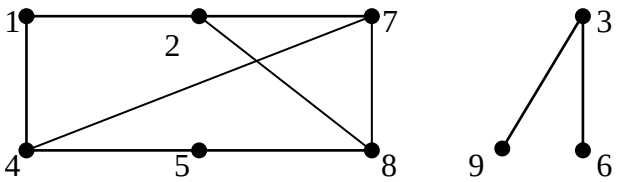


Răspuns: C1={1,2,3}, C2={4,5}, C3={6}.

Observaţii

Un graf este conex dacă şi numai dacă numărul componentelor sale conexe este 1.

5. **Pentru graful,**



Determinați componenta conexă care conține vârful 1.

Răspuns:

Determinarea componentei conexe care conţine un vârf v_0 dat poate fi realizată pe baza următorului algoritm.

Pentru $G=(V,E)$, $|V| = n$, $n \geq 1$ şi $v_0 \in V$, paşii algoritmului sînt:

Pas1: $V_0=\{v_0\}$; $E_0=\Phi$; i=0;

Pas 2: repetă *Pas 3* pînă cînd $V_i=V_{i-1}$ şi $E_i=E_{i-1}$

Pas 3: i=i+1;

$V_i = V_{i-1} \cup \{v / v \in V, \exists u \in V_{i-1}, uv \in E\}$;

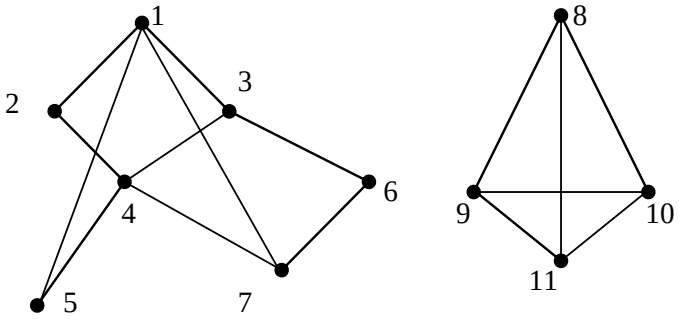
$E_i = E_{i-1} \cup \{e / e \in E, \exists u \in V_{i-1}, u \text{ incident cu } e\}$;

Ieşirea este $G_i=(V_i,E_i)$, componenta conexă din care face parte v_0.

Aplicarea algoritmului descris pentru $v_0=1$, determină următoarea evoluţie:

I	V_i	E_i
i=0	{1}	Ø
i=1	{1,2,4}	{(1,2),(1,4)}
i=2	{1,2,4,7,8,5}	{(1,2),(1,4),(2,7),(2,8),(7,8),(4,5),(4,7),(5,8)}

6. Fie graful,



Câte component conexe are graful?

Răspuns: Graful are 2 componente conexe.

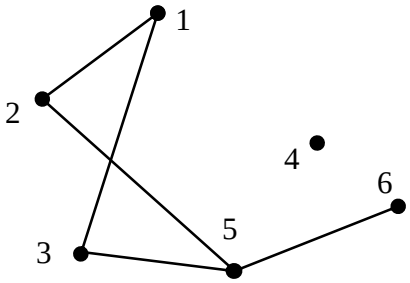
7. Pentru graful de mai sus, determinaţi componenta conexă care conţine vârful 5.

Răspuns:

Aplicarea algoritmului descris pentru $v_0=5$, determină următoarea evoluţie:

I	V_i	E_i
i=0	{5}	Ø
i=1	{1,4,5}	{(1,5),(4,5)}
i=2	{1,2,3,4,5,7}	{(1,5),(4,5),(1,2),(1,3),(2,4),(3,4),(1,7),(4,7)}
i=3	{1,2,3,4,5,6,7}	{(1,5),(4,5),(1,2),(1,3),(2,4),(3,4),(1,7),(4,7),(3,6),(3,7)}

8. **Fie graful,**



Care este matricea existenţei drumurilor?

Răspuns:

Matricea existenţei drumurilor este $A = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$

9. **Scrieți un subprogram care calculează matricea existenței drumurilor într-un graf neponderat (algoritmul Roy-Warshall), precum și programul apelator.**

```
#include <stdio.h>
#include <malloc.h>

//I: matricea de adiacenta, nr. vârfuri
//E: matricea existentei drumurilor, dimensiune
int** Roy_Warshall(int** a, int n, int *dim)
{
	int i, j, k;
	int** m;

	*dim = n;
	m = (int**)malloc(n * sizeof(int*));
	for (i = 0; i < n; i++)
		m[i] = (int*)malloc(n * sizeof(int));

	//se pune in matricea drumurilor valorile din matricea de adiacenta
	for (i = 0; i < n; i++)
		for (j = 0; j < n; j++)
			m[i][j] = a[i][j];

	//ROW_WARSHALL
	for (i = 0; i < n; i++)
		for (j = 0; j < n; j++)
			if (m[i][j] == 1)
				for (k = 0; k < n; k++)
					if (m[j][k] == 1)
						m[i][k] = m[k][i] = 1;
	return m;
}

//preia graful din fisierul text
```

```
void preia_graf(char *nume, int ***a, int *n)
{
	int i, j, m, u, v;
	FILE *f = fopen(nume, "rt");
	fscanf(f, "%i", n);

	int **mw = (int **)malloc(*n * sizeof(int*));
	for (i = 0; i < *n; i++)
		mw[i] = (int *)malloc(*n * sizeof(int));

	fscanf(f, "%i", &m);
	for (i = 0; i < *n; i++)
		for (j = 0; j < *n; j++)
			mw[i][j] = 0;

	for (i = 0; i < m; i++)
	{
		fscanf(f, "%i", &u);
		fscanf(f, "%i", &v);
		mw[u - 1][v - 1] = mw[v - 1][u - 1] = 1;
	}

	fclose(f);
	*a = mw;
}


void main()
{
	//a - matricea de adiacenta; m_dr - matricea drumurilor
	int **a, **m_dr, l;
	int n, i, j;

	//citire graf din fisier
	char numef[20];
	printf("Introduceti numele fisierului care contine graful:");
	scanf("%s", numef);

	//Reminder:
	//pointerul - este o variabila ce contine ADRESA unei zone din memorie, iar la ACEA zona
de memorie este o valoare.
	//&: extrage adresa unei variabile
	//*: referirea continutului zonei de memorie indicate de pointer(indirectare).
	//Ex: int x=10; int *y=&x; --> Pentru a avea acces la valoarea 10 trebuie sa folosim
indirectarea (*y);

	preia_graf(numef, &a, &n);//elementele trimise aici ca input o sa modifice in functie

	//Functia Roy_Warshall returneaza o matrice. Ea va fi asignata variabilei m_dr
	//O alta varianta ar fi fost sa trimitem matricea m_dr ca input a functiei ca sa nu mai
returnam nimic :)
	m_dr = Roy_Warshall(a, n, &l);

	//afisarea matricii drumurilor
	for (i = 0; i < l; i++)
	{
		for (j = 0; j < l; j++) printf("%3i ", m_dr[i][j]);
		printf("\n");
	}

	//eliberare memorie matrici
	for (i = 0; i < n; i++) free(a[i]);
	free(a);

	for (i = 0; i < l; i++) free(m_dr[i]);
	free(m_dr);
```

```
}
```

Pentru graful preluat din fișierul graf3.txt (fisierul text va fi adaugat in directorul proiectului), anume:

```
11 11
1 2
1 3
2 4
2 6
3 4
3 5
4 5
5 6
7 8
7 9
8 9
```

matricea existenței drumurilor este:

```
1     1     1     1     1     1     0     0     0     0     0
1     1     1     1     1     1     0     0     0     0     0
1     1     1     1     1     1     0     0     0     0     0
1     1     1     1     1     1     0     0     0     0     0
1     1     1     1     1     1     0     0     0     0     0
1     1     1     1     1     1     0     0     0     0     0
0     0     0     0     0     0     1     1     1     0     0
0     0     0     0     0     0     1     1     1     0     0
0     0     0     0     0     0     1     1     1     0     0
0     0     0     0     0     0     0     0     0     0     0
0     0     0     0     0     0     0     0     0     0     0
```