UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Structuri de Date și Algoritmi

Sergiu Corlat, 2021

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Structuri de Date și Algoritmi

Lecția 2:

- Structuri de date neliniare – **grafuri**
  - Definiții
  - Moduri de reprezentare
  - Operații pe grafuri
  - Complexitatea operațiilor
  - Aplicații

Sergiu Corlat, 2021

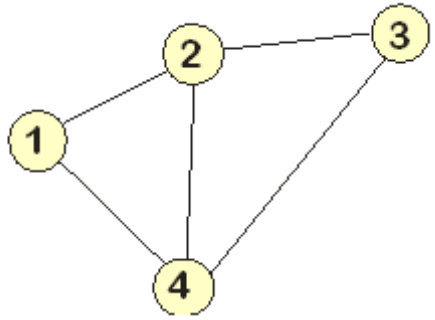
UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Definiții:

Definiții:

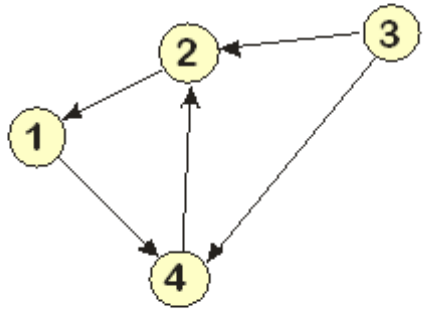
Def. Graf neorientat (graf) – o pereche arbitrară $G=(V,E)$

$E \subseteq \{\{u,v\} : u,v \in V \,\&\, u \neq v\}$



$V = \{1,2,3,4\}$, $E = \{\{1,2\},\ \{2,3\},\ \{3,4\},\ \{1,4\},\ \{2,4\}\}$

Def. Graf orientat (graf) – o pereche arbitrară $G=(V,E)$, în care $E \subseteq V \times V$



$V = \{1,2,3,4\}$, $E = \{(1,4),\ (2,1),\ (3,2),\ (3,4),\ (4,2)\}$.

$\boldsymbol{V}$ formează mulțimea vârfurilor grafului, $\boldsymbol{E}$ – mulțimea muchiilor.

Muchia (u,v) este *incidentă* vârfurilor u,v, iar acestea sunt *adiacente* muchiei.

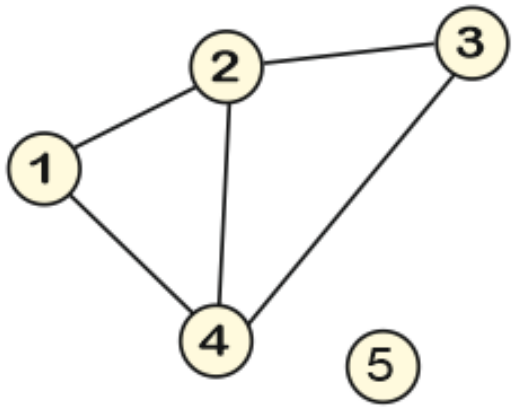
Definiții:

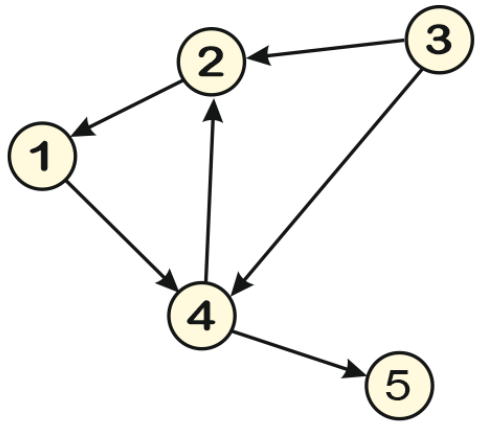
Def. *Gradul* vârfului v, $d(v)$ este numărul de muchii, incidente acestuia. Un vârf este *izolat*, dacă gradul lui este 0.

Def. Mulțimea de *vecini* ai vârfului v_i, $\Gamma(v_i)$ este formată din vârfurile adiacente la v_i. Într-un graf orientat mulțimea vecinilor este formată din două componente distincte: $\Gamma(v_i) = \Gamma^+(v_i) \cup \Gamma^-(v_i)$.

$\Gamma^+(v_i)$ - vârfurile adiacente din arcele cu originea în v_i.

$\Gamma^-(v_i)$ - vârfurile adiacente din arcele care se termină în v_i.



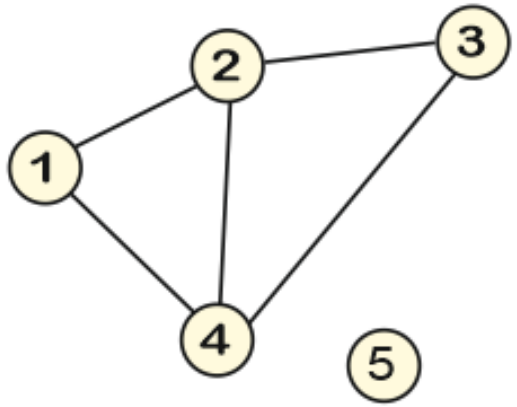


Definiții:

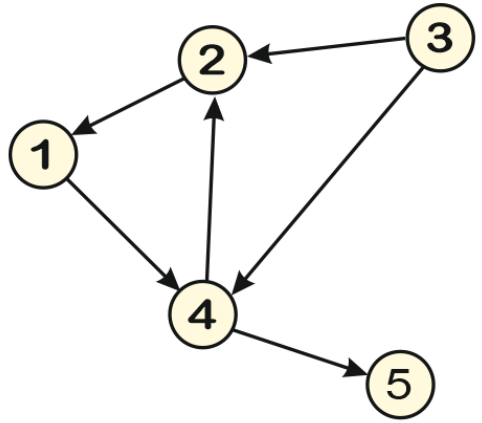
Def. *Cale* în graf este o consecutivitate de vârfuri $v_1, v_2, ...v_k$ astfel încât $\forall i = 1,...,k-1$ vârfurile v_i, v_{i+1} sunt adiacente. Dacă toate vârfurile $v_1, v_2, ...v_k$ sunt distincte, calea se numeşte *elementară*. Dacă $v_1 = v_k$ atunci $v_1, v_2, ...v_k$ formează un *ciclu* în G. Ciclul este *elementar*, dacă $v_1, v_2, ...v_{k-1}$ sunt distincte.

În grafurile orientate noțiunea de cale este substituită prin *lanț*.

Def. Lanțul este o consecutivitate de muchii (arce) luate astfel, încât vârful arcului (i) coincide cu originea arcului (i+1).



secvența de vârfuri
1,2,4 - cale;
secvența
1,2,4,1 – ciclu



secvența de arce
(3, 4) (4, 2) (2, 1) - lanț;
secvența de arce
(1, 4) (4, 2) (2, 1) – ciclu.

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

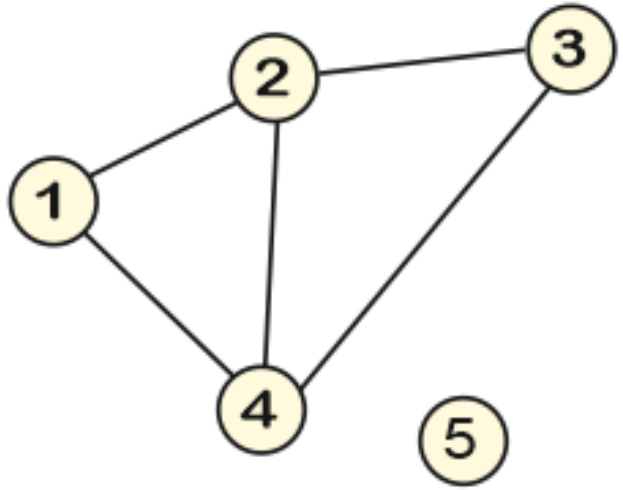
Metode de reprezentare:

Matricea de incidență

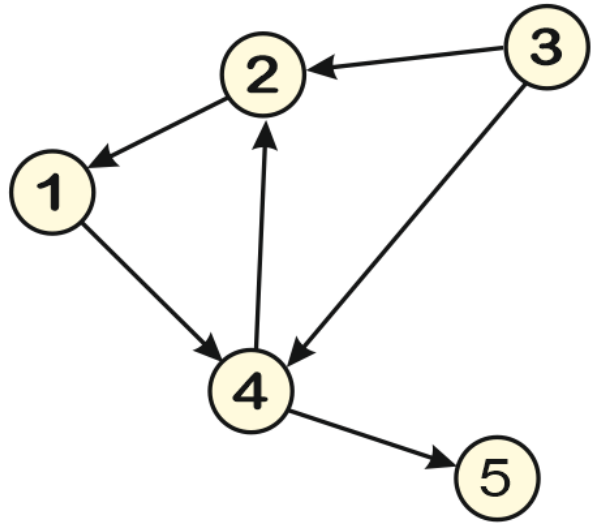
Structura de date clasică pentru reprezentarea unui graf $G=(V,E)$ este considerată **matricea de incidență**. Este realizată prin un tablou bidimensional cu N linii ($N=|V|$) şi M coloane ($M=|E|$).

Fiecare muchie (u,v) este descrisă într-o coloană a tabloului. Elementele coloanei sunt egale cu 1 pentru liniile care corespund vârfurilor u şi v, 0 – pentru celelalte. În cazul grafului orientat vârful din care începe arcul este marcat cu -1, vârful final – cu +1.

Exemple



	(1, 2)	(1, 4)	(2, 3)	(2, 4)	(3, 4)
1	1	1	0	0	0
2	1	0	1	1	0
3	0	0	1	0	1
4	0	1	0	1	1
5	0	0	0	0	0

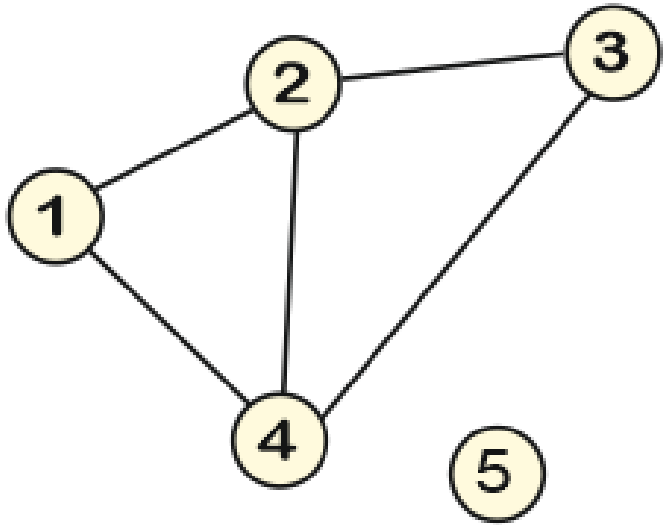


	(1, 4)	(4, 2)	(4, 5)	(2, 1)	(3, 4)	(3, 2)
1	-1	0	0	+1	0	0
2	0	+1	0	-1	0	+1
3	0	0	0	0	-1	-1
4	+1	-1	-1	0	+1	0
5	0	0	+1	0	0	0

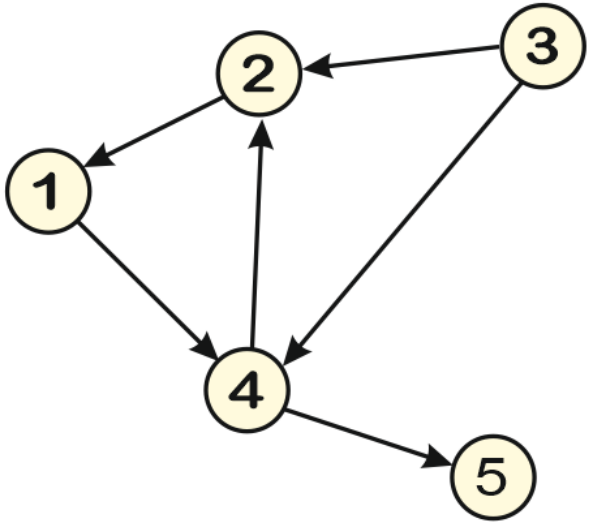
Matricea de adiacență

O altă reprezentare matriceală a grafului $G=(V,E)$ este **matricea de adiacență**. Matricea de adiacență este şi ea realizată prin un tablou bidimensional, cu N linii şi N coloane, în care elementul cu indicii (i,j) este egal cu 1 dacă există muchia care uneşte vârful v_i cu vârful v_j şi 0 – în caz contrar. Datele despre muchia (v_i,v_j) se dublează în elementele tabloului cu indicii (i,j) şi (j,i) În grafurile orientate, pentru arcul (v_i,v_j) primeşte valoarea 1 doar elementul (i,j) al tabloului.

Exemple



	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	1	0
4	1	1	1	0	0
5	0	0	0	0	0



	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	0	1	0	1	0
4	0	1	0	0	1
5	0	0	0	0	0

Reprezentări liniare:

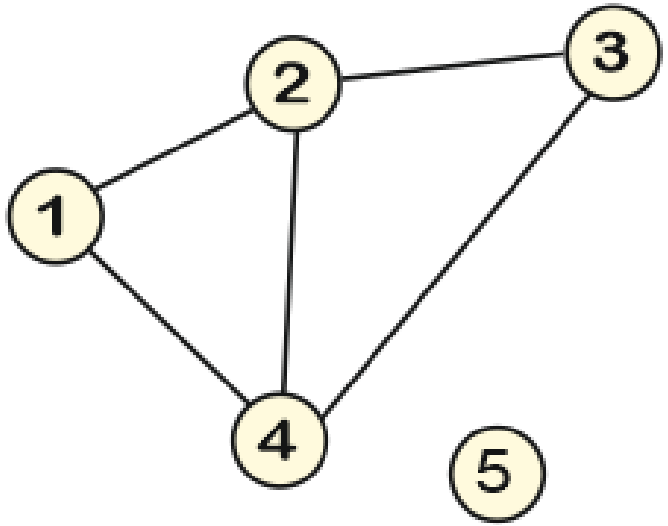
Lista de muchii

O altă categorie de reprezentări ale grafului o formează reprezentările prin liste. Cea mai simplă reprezentare este **lista de muchii**. Fie graful $G=(V,E)$ și ($N=|V|$, $M=|E|$).

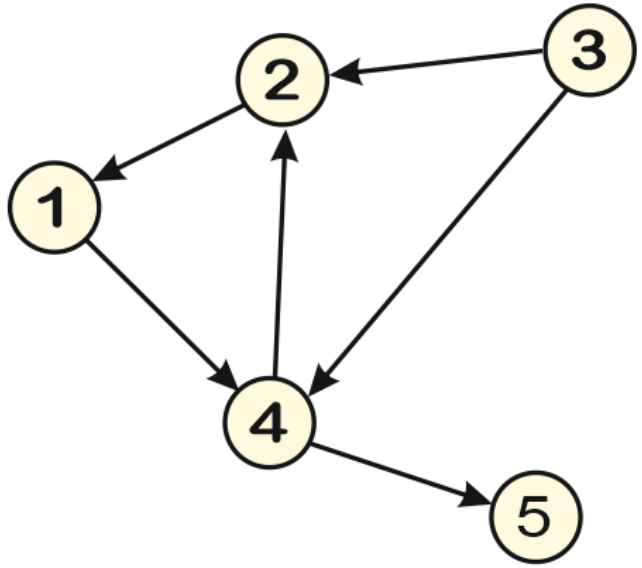
Lista de muchii conține M perechi de forma (v_i, v_j), fiecare pereche reprezentând o muchie din graf, descrisă prin vârfurile care o formează.

Într-un graf orientat perechea descrie un arc, începutul lui fiind determinat de primul indice din pereche.

Exemple



Lista de muchii:
(1,2)
(1,4)
(2,3)
(2,4)
(3,4)
(5,5)



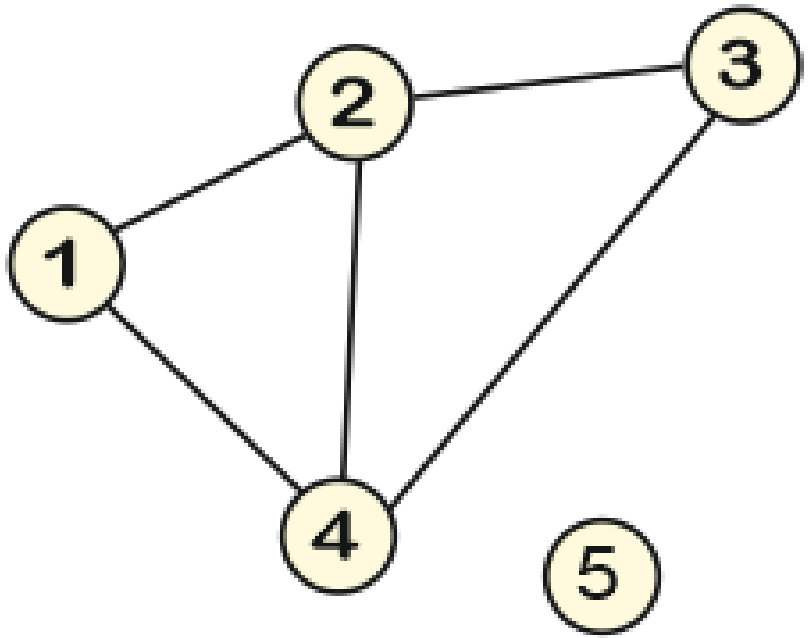
Lista de muchii:
(2,1)
(1,4)
(4,5)
(4,2)
(3,4)
(3,2)

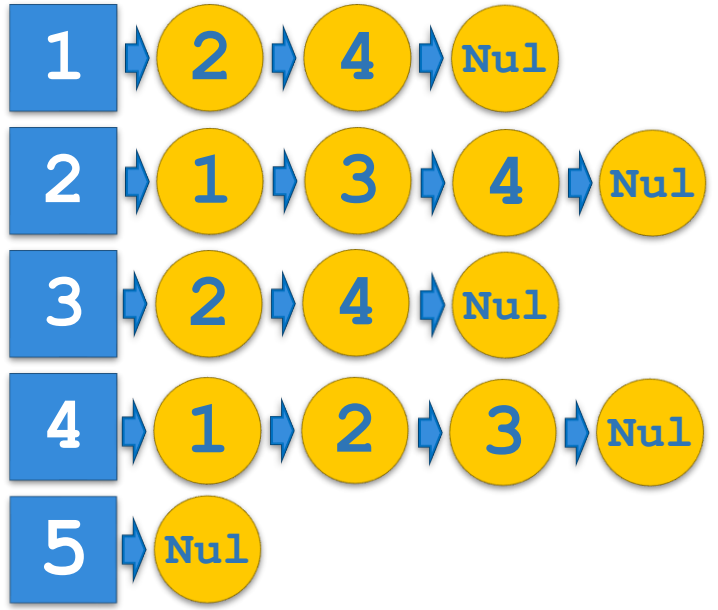
Lista de adiacență

(Listele de vecini)

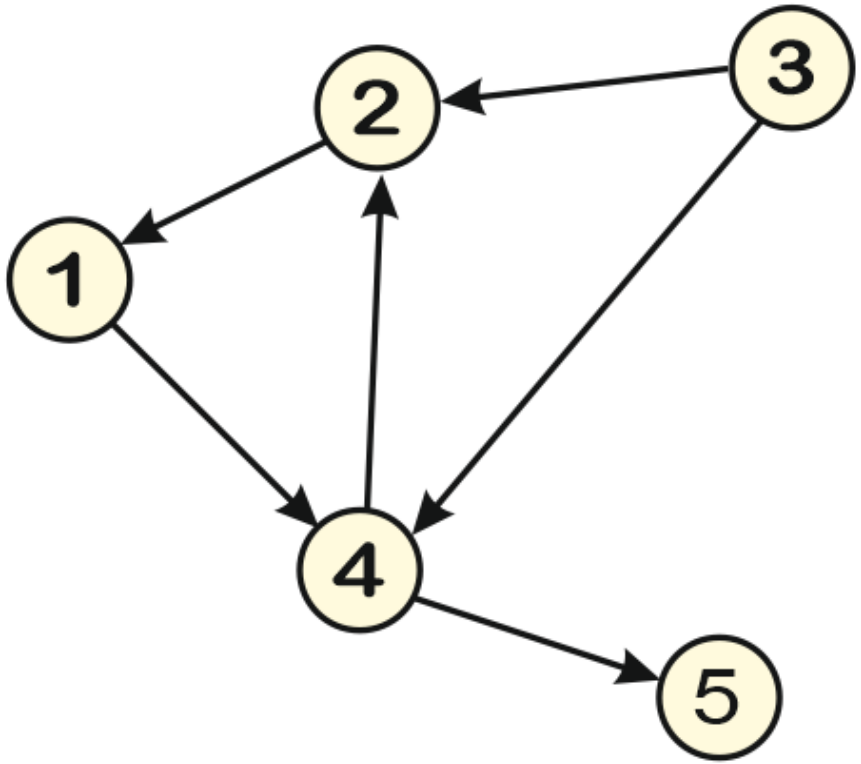
Încă o structură eficientă este **lista de adiacență.** Pentru fiecare nod $v \in V$ ea va conține o listă unidirecțională alocată dinamic, cu toate vârfurile $u : \exists (v,u) \in E$. Referințele către începutul fiecărei liste pot fi păstrate într-un tablou unidimensional. Elementul cu indicele i din tablou va conține referința către lista de vârfuri adiacente vârfului v_i din graf. Pentru grafurile neorientate descrierea fiecărei muchii se dublează.

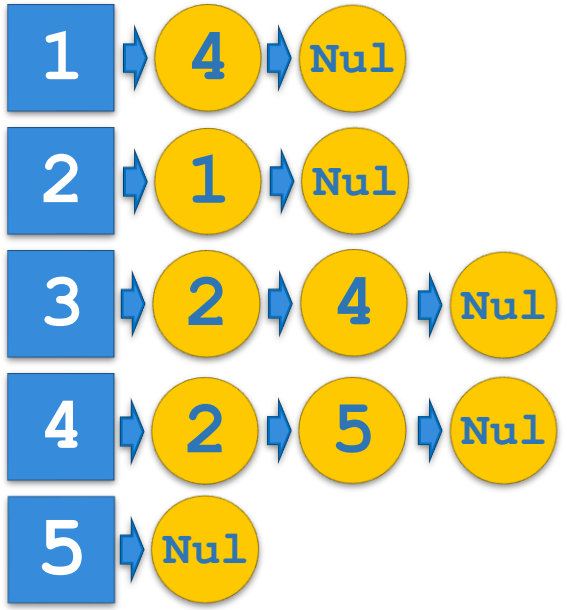
Exemple





Exemple





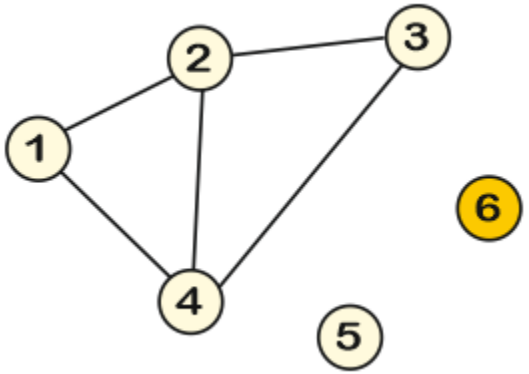
Operații pe grafuri

Operații elementare:

- Adăugare vârf
- Lichidare vârf
- Adăugare muchie
- Lichidare muchie
- Divizare (dublare) vârf
- Contracție vârfuri
- Contracție muchii

G(V, E)

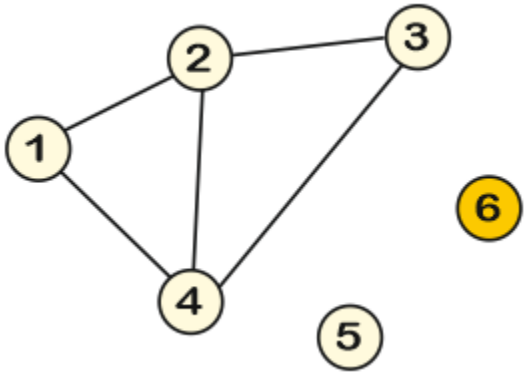
Adăugare vârf



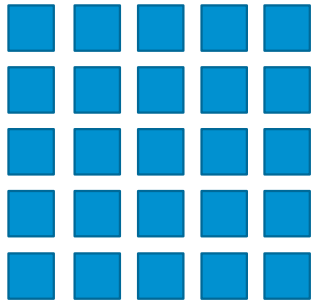
Numărul estimat de operații				
	Matricea de incidență	Matricea de adiacență	Lista de muchii	Lista de vecini
Cu dublare a structurii de date	$O(\|V\|\|E\|)$	$O(\|V\|^2)$	$O(\|E\|)$	$O(\|V\|^2)$
Cu actualizare a structurii curente	$O(1)$	$O(1)$	$O(1)$ / acces direct $O(\|E\|)$ / acces secvențial	$O(1)$

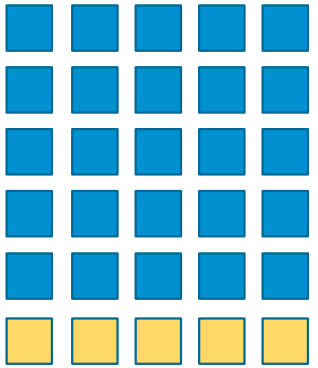
G(V, E)

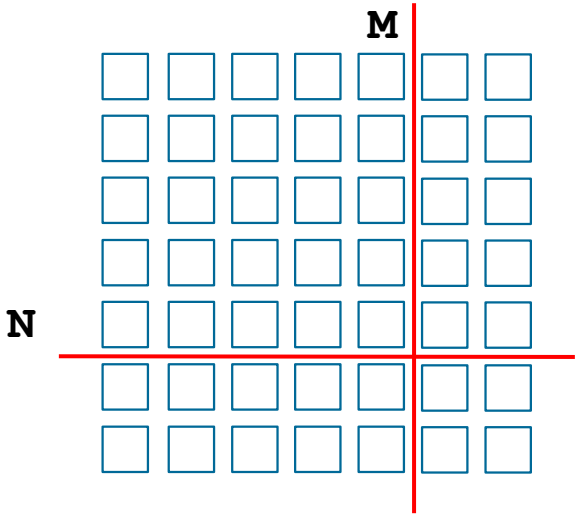
Adăugare vârf



	Matricea de incidență
Cu dublare a structurii de date	$O(\|V\|\|E\|)$
Cu actualizare a structurii curente	$O(1)$



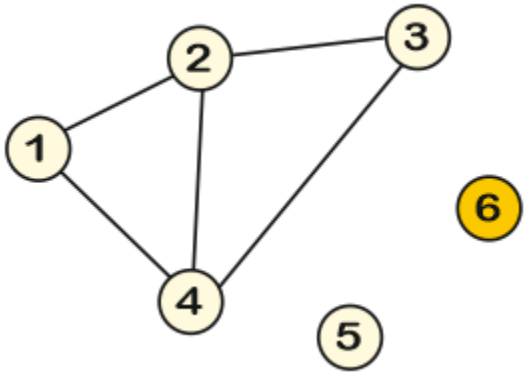




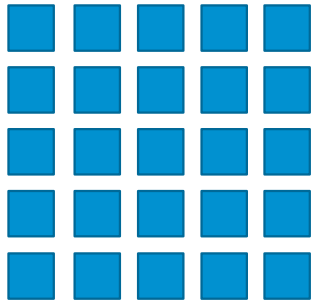

```
N = N + 1
```

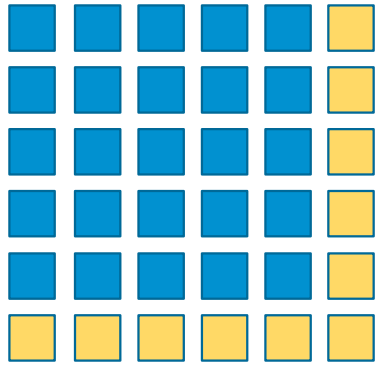
G(V, E)

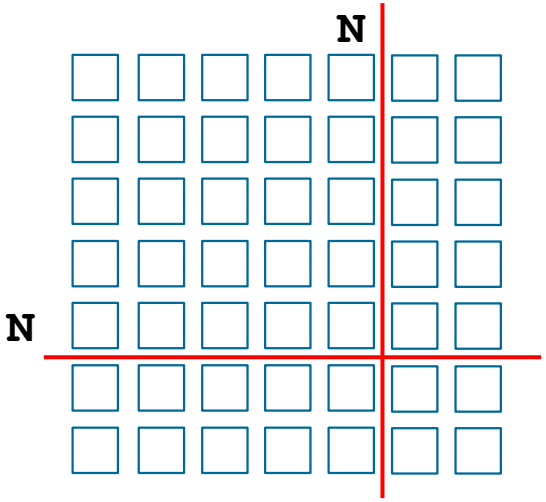
Adăugare vârf



	Matricea de adiacență
Cu dublare a structurii de date	$O(\|V\|^2)$
Cu actualizare a structurii curente	$O(1)$



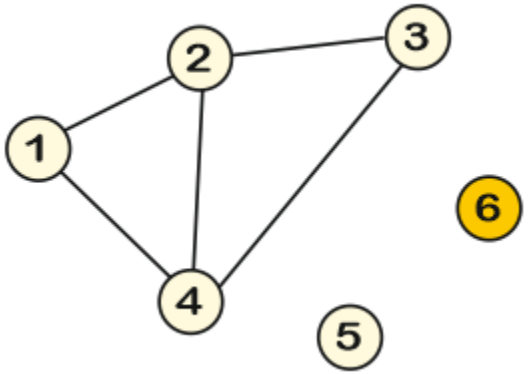




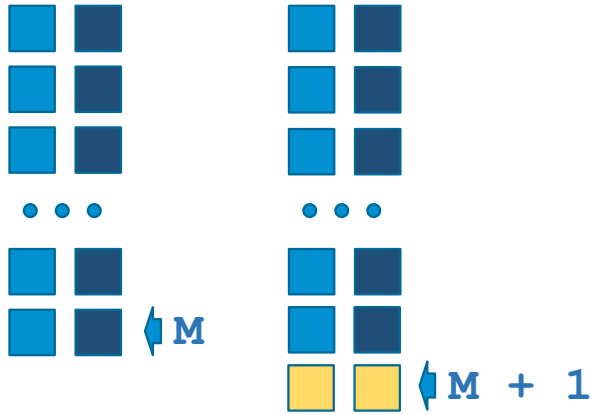

```
N = N + 1
```

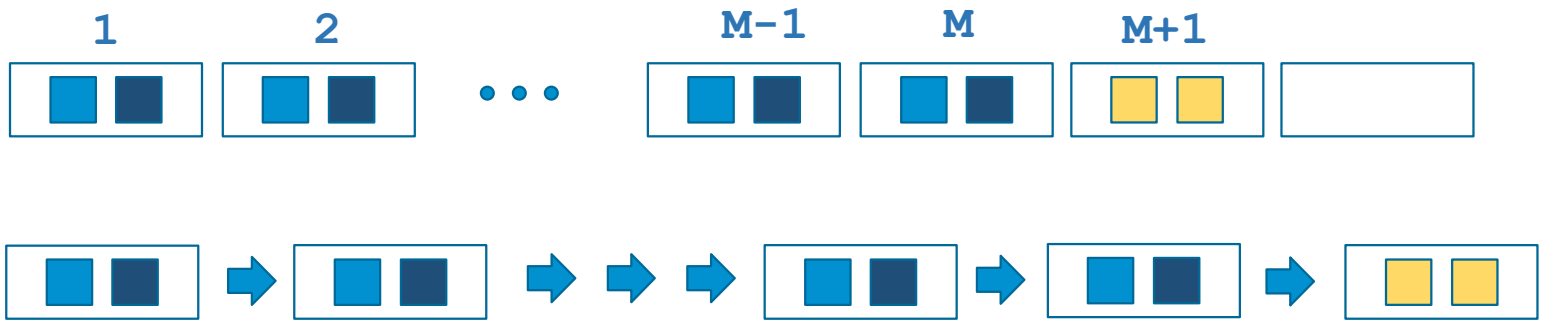
G(V, E)

Adăugare vârf



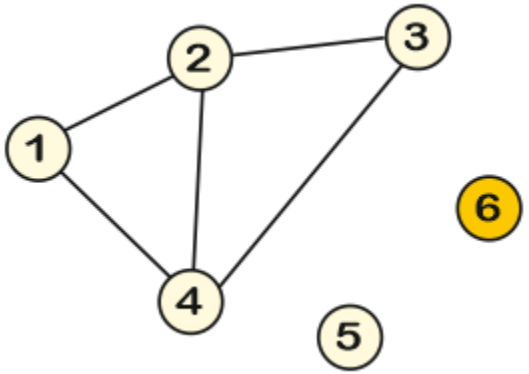
	Lista de muchii
Cu dublare a structurii de date	$O(\|E\|)$
Cu actualizare a structurii curente	$O(1)$ / acces direct $O(\|E\|)$ / acces secvențial



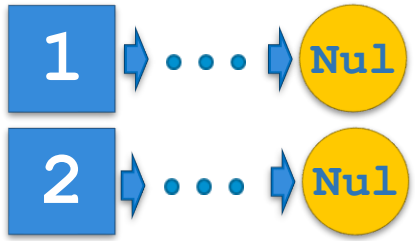


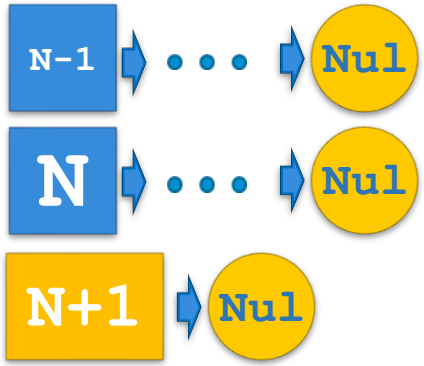
G(V, E)

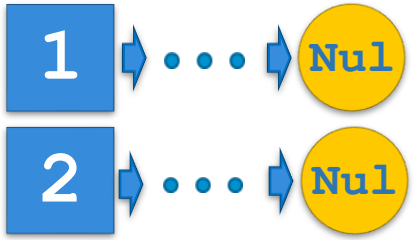
Adăugare vârf

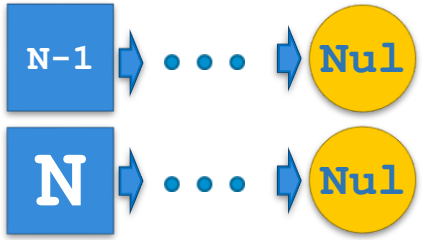


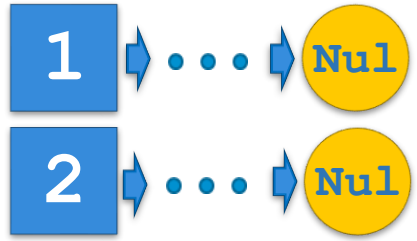
	Lista de vecini
Cu dublare a structurii de date	$O(\|V\|^2)$
Cu actualizare a structurii curente	$O(1)$

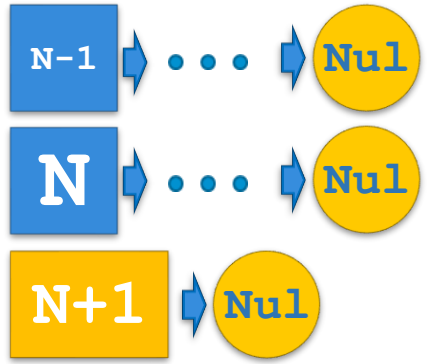






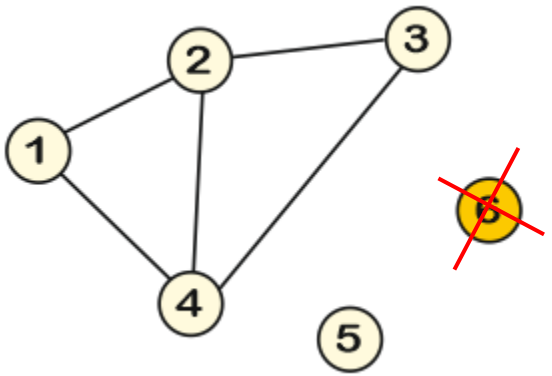






G(V, E)

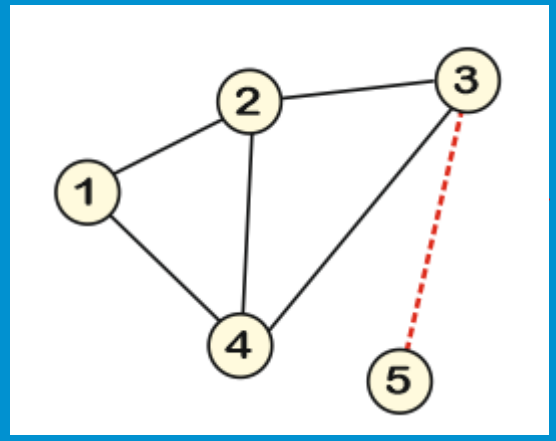
Eliminare vârf



Numărul estimat de operații				
	Matricea de incidență	Matricea de adiacență	Lista de muchii	Lista de vecini
Cu dublare a structurii de date	$O(\|V\|\|E\|)$	$O(\|V\|^2)$	$O(\|E\|)$	$O(\|V\|^2)$
Cu actualizare a structurii curente	$O(\|V\|\|E\|)$	$O(\|V\|^2)$	$O(\|E\|)$	$O(\|V\|^2)$

G(V, E)

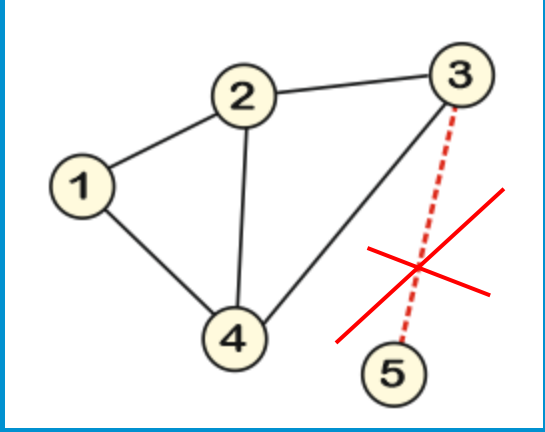
Inserare muchie



Numărul estimat de operații				
	Matricea de incidență	Matricea de adiacență	Lista de muchii	Lista de vecini
Cu dublare a structurii de date	$O(\|V\|\|E\|)$	$O(\|V\|^2)$	$O(\|E\|)$	$O(\|V\|^2)$
Cu actualizare a structurii curente	$O(\|V\|)$ -fără reindexare $O(\|V\|\|E\|)$ - cu reindexare	$O(1)$	$O(1)$	$O(1)$

G(V, E)

Eliminare muchie

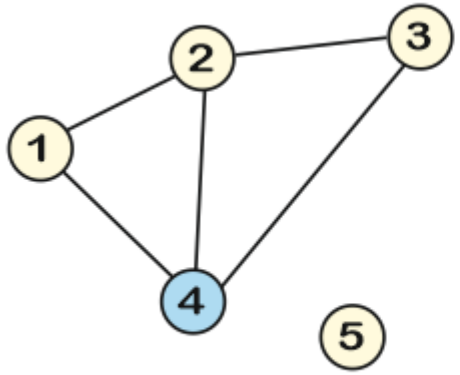


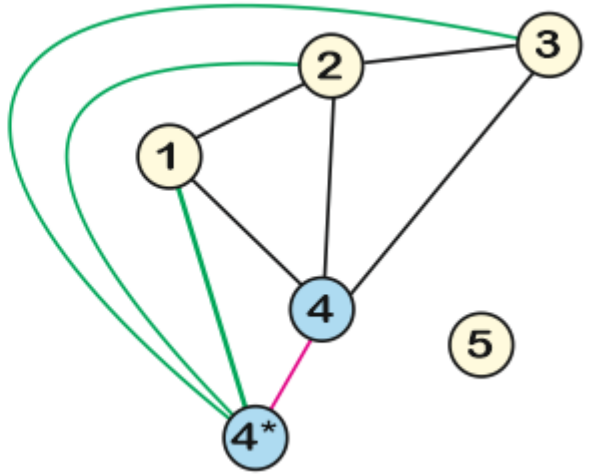
Numărul estimat de operații				
	Matricea de incidență	Matricea de adiacență	Lista de muchii	Lista de vecini
Cu dublare a structurii de date	$O(\|V\|\|E\|)$	$O(\|V\|^2)$	$O(\|E\|)$	$O(\|V\|^2)$
Cu actualizare a structurii curente	$O(\|V\|)$ -fără reindexare $O(\|V\|\|E\|)$ - cu reindexare	$O(1)$	$O(\|E\|)$	$O(\|V\|)$

G(V, E)

Dublare vârf

Fie graful $G=(V,E)$ și $(N=|V|,\ M=|E|)$ și $v_i \in V$. Operația de dublare a vârfului v_i presupune adăugarea unui vârf nou v_j : $\Gamma(v_j) = \Gamma(v_i) \cup \{v_i\}$ și a unei muchii noi (v_i, v_j).





Exercițiul 2.1: Estimați complexitatea operației de dublare a vârfului pentru
a) matricea de adiacență
b) lista de vecini

G(V, E)

Contracție vârfuri

Fie graful $G = (V, E)$, $(N = |V|,\ M = |E|)$ și $v_i,\ v_j \in V,\ (v_i,\ v_j) \notin E$.

Operația de contracție a vârfurilor $v_i,\ v_j$ presupune adăugarea unui vârf nou

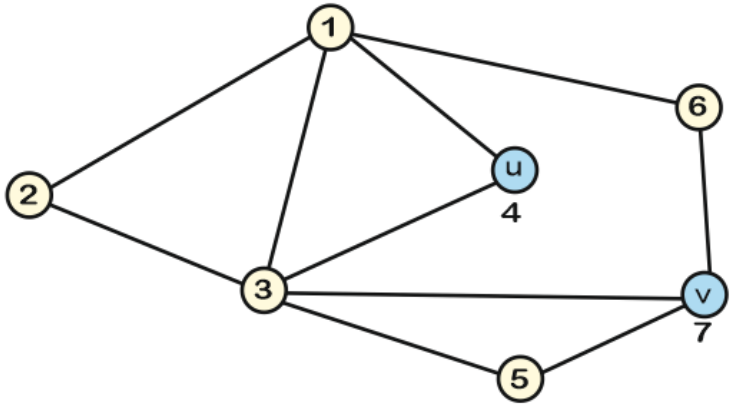
v_z : $\Gamma(v_z) = \Gamma(v_i) \cup \Gamma(v_j) - \{v_i,\ v_j\}$ și eliminarea vârfurilor v_i, v_j:

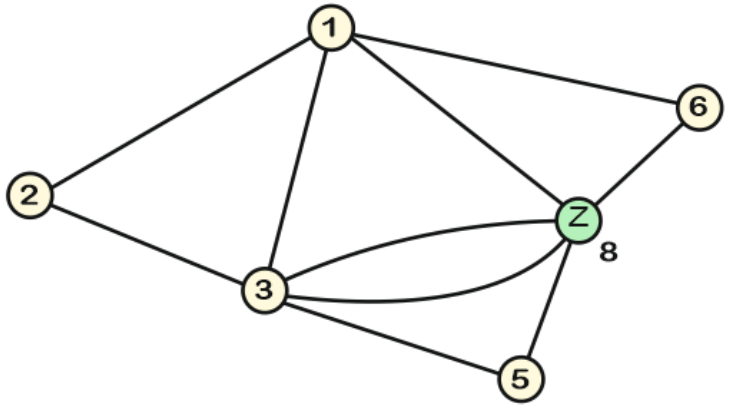
$G^* = (V^*, E^*)$:

$V^* = V - \{v_i,\ v_j\} \cup v_z$

Respectiv, pentru fiecare muchie $(u, v) \in E$,

dacă $u \in \{v_i, v_j\}\ u \leftarrow v_z$, dacă $v \in \{v_i, v_j\}\ v \leftarrow v_z$



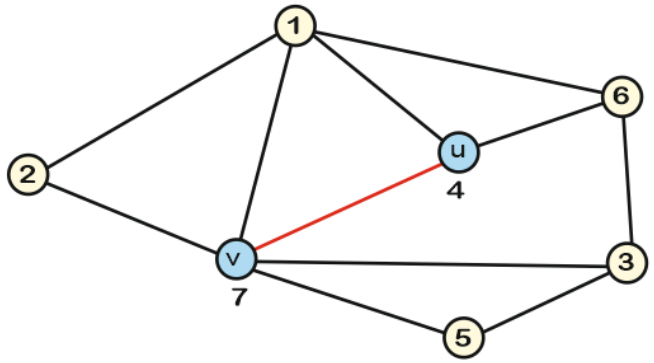


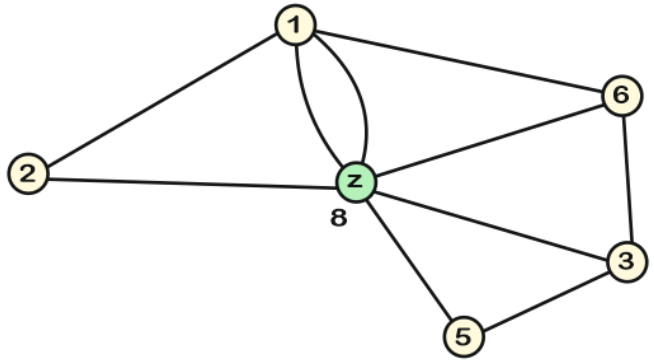
G(V, E)

Contracție muchii

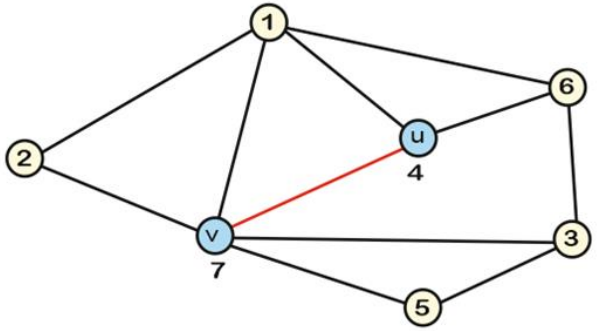
Fie graful $G = (V, E)$, $(N = |V|,\ M = |E|)$ și $v_i,\ v_j \in V,\ (v_i,\ v_j) \in E$.
Operația de contracție a muchiei (v_i, v_j) presupune adăugarea unui vârf nou
v_z : $\Gamma(v_z) = \Gamma(v_i) \cup \Gamma(v_j) - \{v_i,\ v_j\}$ și eliminarea vârfurilor v_i, v_j:
$G^* = (V^*, E^*)$:
$V^* = V - \{v_i,\ v_j\} \cup v_z$
Respectiv, pentru fiecare muchie $(u, v) \in E$:

- dacă $u \in \{v_i, v_j\}\ u \leftarrow v_z$,
- dacă $v \in \{v_i, v_j\}\ v \leftarrow v_z$,
- dacă $v \in \{v_i, v_j\}$ & $u \in \{v_i, v_j\}, E = E - (u, v)$.





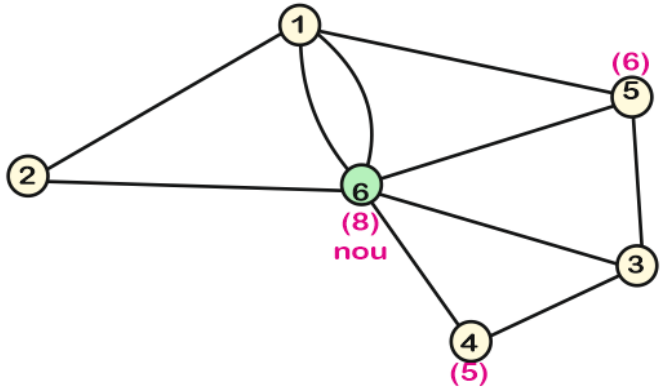
Exemplu implementare



0	1	0	1	0	1	1
1	0	0	0	0	0	1
0	0	0	0	1	1	1
1	0	0	0	0	1	1
0	0	1	0	0	0	1
1	0	1	1	0	0	0
1	1	1	1	1	0	0

0	1	0	0	0	1	0	1
1	0	0	0	0	0	0	1
0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
1	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0
1	1	1	0	1	1	0	0

0	1	0	0	1	1
1	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	0	1
1	0	1	0	0	1
1	1	1	1	1	0



Demonstrație implementare

Contracția vârfurilor

Operații complexe

Operații complexe

Unare – sunt aplicate asupra unui singur graf. Produc un graf nou sau generează un rezultat pe graful existent

Binare – sunt aplicate asupra a două grafuri. Produc în calitate de rezultat un graf nou.

Operații pe grafuri

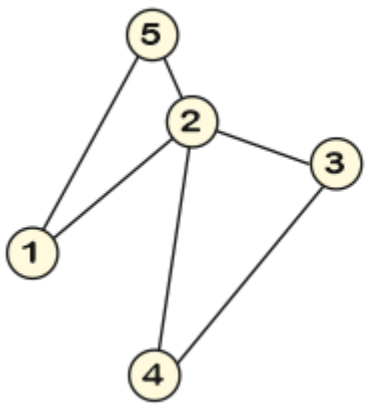
Operații complexe unare:

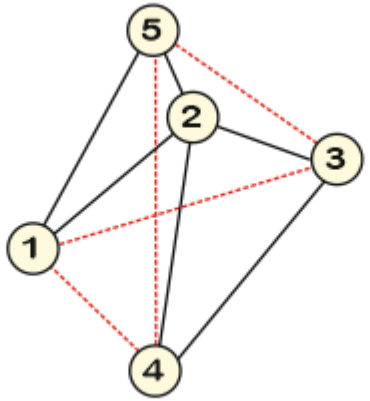
- Graf complementar
- Graf transpus
- Graf asociat
- Parcurgere în adâncime
- Parcurgere în lățime

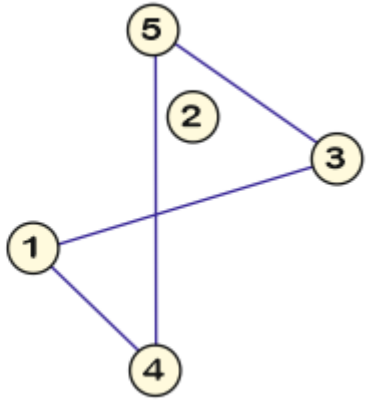
Operații pe grafuri:

Graful complementar

Fie $G=(V,E)$. Prin *graf complementar* se înțelege graful $\tilde{G}=(V,\tilde{E})$ unde $\tilde{E}=\{(u,v): u,v\in V; (u,v)\notin E\}$








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

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

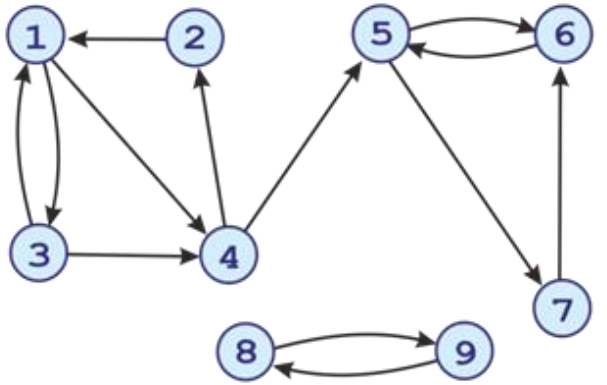
Operații pe grafuri:

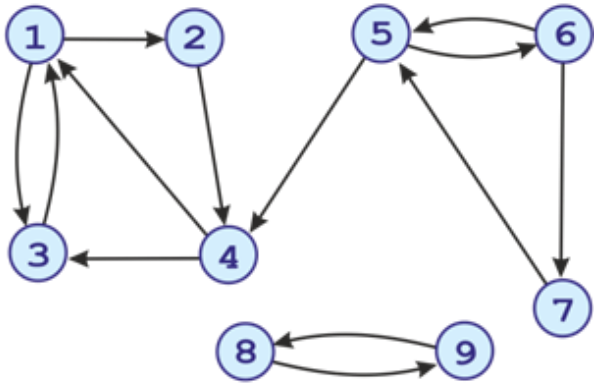
Transpunerea

Pentru un graf orientat $G=(V,E)$, graful transpus este $G^T=(V,E^T)$ unde $E^T=\{(v_i,v_j):(v_j,v_i)\in E\}$. Graful transpus are aceleaşi componente tare conexe ca şi graful inițial.

Fie graful $G=(V,E)$ reprezentat prin matricea de adiacență $A(n\times n)$. Matricea de adiacență a grafului $G^T=(V,E^T)$ se obține conform următoarei formule:

$$A^T_{i,j}=\begin{cases}1 & dacă \quad A_{j,i}=1\\ 0 & în\ caz\ contrar\end{cases} \qquad i,j=1,\ldots,n$$

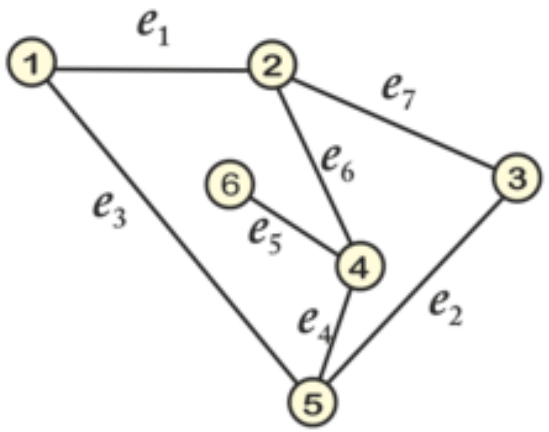


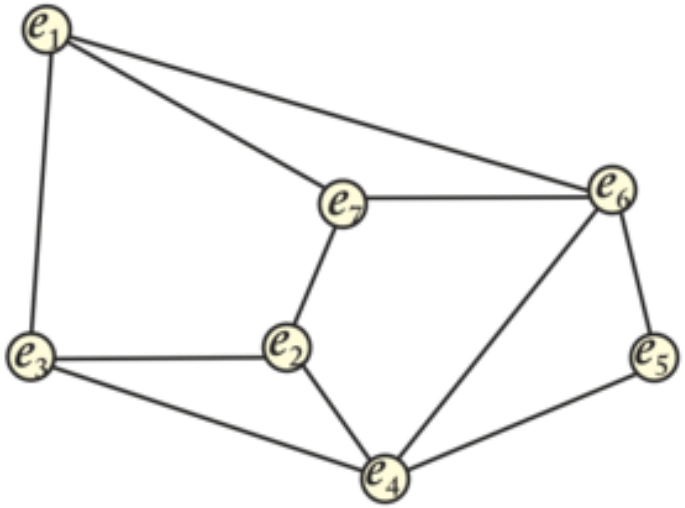


Operații pe grafuri:

Graful asociat

Fie graful neorientat $G=(V,E)$ $E=\{e_1,...,e_M\}$. Graful asociat pentru $G=(V,E)$ se va numi graful $G_A=(V_A,E_A)$, unde $V_A=\{e_1,...,e_M\}$ iar $E_A=\{(e_i,e_j):\exists u\in V, u\in e_i \,\&\, u\in e_j\}$.





Pas 1. Se creează lista muchiilor din $G=(V,E)$. $L=\{e_1,\ldots e_m\}$.

Pas 2. Se creează tabloul bidimensional E' – matricea de adiacență a grafului G_A.

Pas 3. Pentru toți i de la 1 la m-1.

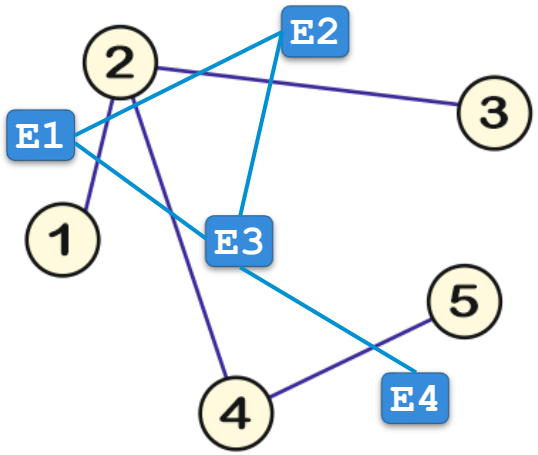
Pentru toți j de la i+1 la m

Dacă în $G=(V,E)$ $e_i \cap e_j \neq \varnothing; (e_i,e_j \in E)$,

atunci $E'_{i,j} \leftarrow E'_{j,i} \leftarrow 1$ altfel $E'_{i,j} \leftarrow E'_{j,i} \leftarrow 0$

Operații pe grafuri:

Graful asociat



E1	E2	E3	E4
1	2	2	4
2	3	4	5

Intrare: graful $G=(V,E)$, matricea de adiacență a căruia este descrisă în tabloul bidimensional **a**.

Ieşire: matricea de adiacență a grafului $G_A=(V_A,E_A)$. Matricea de adiacență este localizată în tabloul bidimensional b.

list[i]

Operații pe grafuri:

Graful asociat

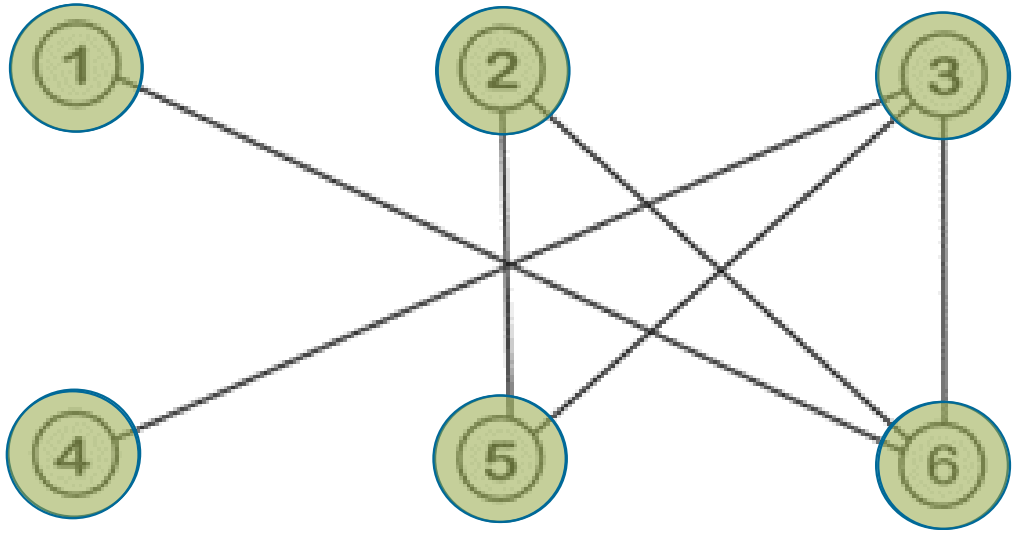
```
void asociat ()
{
    int i, j;
// modelare lista muchii
    for (i = 1; i <= n; i++)
      for (j = 1 + i; j <= n; j++)
        if (a[i][j]!=0 )
        {m++; list[m].v1 = i; list[m].v2 = j;}
// modelare matrice adiacenta
     for (i = 1; i <= m ; i++)
        for (j = 1 + i; j <= m; j++)
            if (list[i].v1 == list[j].v1 || list[i].v1 == list[j].v2 ||
            list[i].v2 == list[j].v1 || list[i].v2 == list[j].v2)
                b[i][j] = b[j][i] = 1;
    return;
}
```

Parcurgeri

DFS

BFS

În caz general problema parcurgerii se formulează în felul următor: Fie dat graful $G=(V,E)$. Pentru un vârf dat $v \in V$ să se determine mulțimea $U \subseteq V$: $\forall u \in U$ există cel puțin o cale între v şi u.

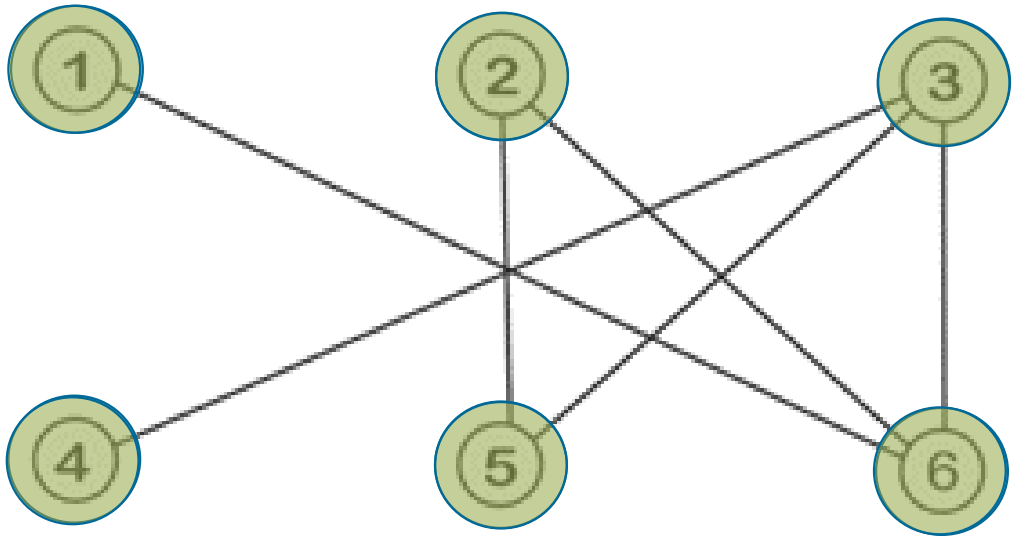


Parcurgeri

DFS

BFS

În caz general problema parcurgerii se formulează în felul următor: Fie dat graful $G=(V,E)$. Pentru un vârf dat $v \in V$ să se determine mulțimea $U \subseteq V$: $\forall u \in U$ există cel puțin o cale între v şi u.



Pentru ambele parcurgeri: descrierea grafului `G(V, E)` este dată de matricea de adiacență `a[][]`, în calitate de structură auxiliară este folosit vectorul de stări `b[]`, nodurile parcurse se afișează în output-ul standard în ordinea parcurgerii.

Parcurgeri

DFS

BFS

DFS

```
int DFS (int s)
{
 int i;
 b[s] = 1;
 for(i = 1; i <= n; i++)
        if(a[s][i] != 0
        && b[i] == 0)
                DFS(i);
   printf("%d ", s);
   return 0;
}
```

BFS

```
int BFS (int s)
{
  int i, st, dr;
  b[s] = 1; c[1] = s; st = dr = 1;
  while (st <= dr)
  {
   for(i = 1; i <= n; i++)
    if(a[c[st]][i] != 0 && b[i] == 0)
        { dr++; c[dr] = i; b[i] = 1;}
    printf("%d ", c[st]);
    st++;
  }
  return 0;
}
```

Operații pe grafuri

Operații complexe binare:

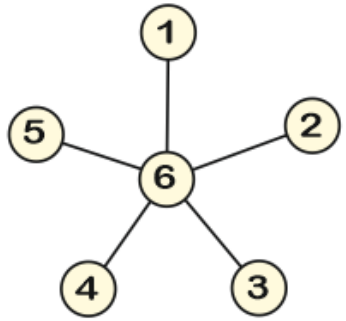
- Reuniunea grafurilor
- Intersecția grafurilor
- (Inter)Conectarea grafurilor
- Diferența grafurilor

Operații pe grafuri

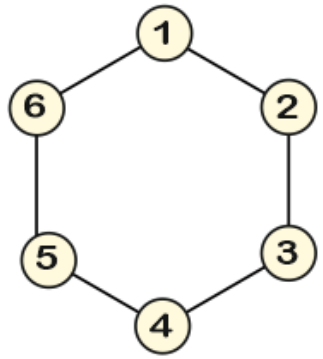
Reuniunea grafurilor

Def. Fie dat graful $G_1 = (V_1, E_1)$ și graful $G_2 = (V_2, E_2)$. Reuniune a grafurilor G_1, G_2 va fi graful $G_r = (V_r, E_r)$, în care

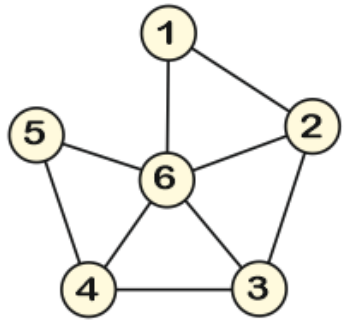
$V_r = V_1 \cup V_2$, $E_r = E_1 \cup E_2$



G_1



G_2



$G_1 \cup G_2$

$V_1 = \{1, 2, 3, 4, 5, 6\}$

$E_1 = \{\{1, 6\}, \{2, 6\}, \{3, 6\}, \{4, 6\} \{5, 6\}\}$

$V_2 = \{1, 2, 3, 4, 5, 6\}$

$E_2 = \{\{1, 6\}, \{2, 1\}, \{3, 2\}, \{4, 3\} \{4, 5\}, \{5, 6\}\}$

$V_r = \{1, 2, 3, 4, 5, 6\}$

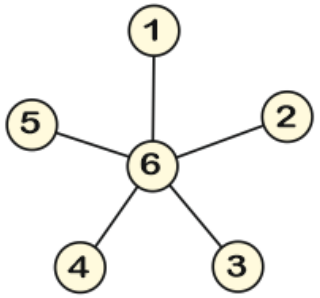
$E_r = \{\{1, 6\}, \{2, 1\}, \{3, 2\}, \{4, 3\} \{5, 6\}, \{2, 6\} \{3, 6\}, \{4, 6\} \{5, 4\}\}$

Operații pe grafuri

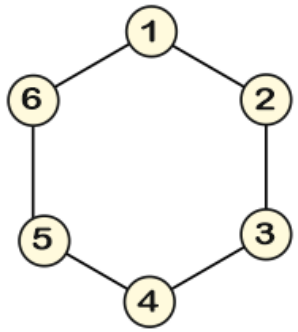
Intersecția grafurilor

Def. Fie dat graful $G_1 = (V_1, E_1)$ și graful $G_2 = (V_2, E_2)$. Intersecție a grafurilor G_1, G_2 va fi graful $G_r = (V_r, E_r)$, în care

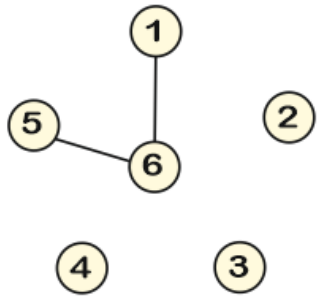
$V_r = V_1 \cup V_2$, $E_r = E_1 \cap E_2$



G_1



G_2



$G_1 \cap G_2$

$V_1 = \{1, 2, 3, 4, 5, 6\}$

$E_1 = \{\{1, 6\}, \{2, 6\}, \{3, 6\}, \{4, 6\} \{5, 6\}\}$

$V_2 = \{1, 2, 3, 4, 5, 6\}$

$E_2 = \{\{1, 6\}, \{2, 1\}, \{3, 2\}, \{4, 3\} \{4, 5\}, \{5, 6\}\}$

$V_r = \{1, 2, 3, 4, 5, 6\}$

$E_r = \{\{1, 6\}, \{5, 6\}\}$

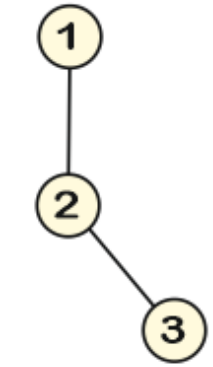
Operații pe grafuri

(Inter) Conectarea grafurilor

Def. Fie dat graful $G_1 = (V_1, E_1)$ și graful $G_2 = (V_2, E_2)$.

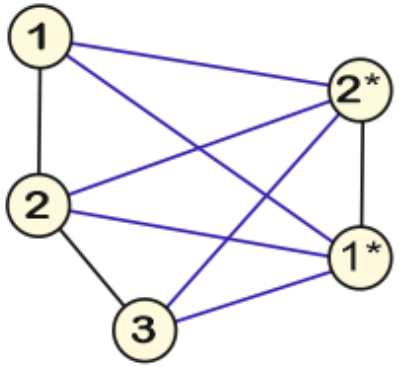
(Inter)Conectare a grafurilor G_1, G_2 va fi graful $G_r = (V_r, E_r)$, în care

$$V_r = V_1 + V_2\,, \quad E_r = E_1 \cup E_2 \cup \{\{u, v\} : u \in V_1, v \in V_2\}$$



G_1

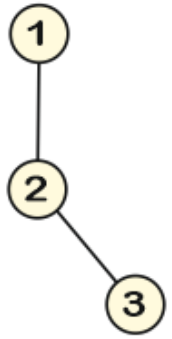
G_2



$G_r = Ic(G_1, G_2)$

Operații pe grafuri

(Inter) Conectarea grafurilor



G_1

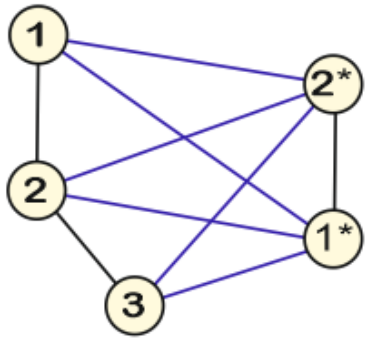
$V_1 = \{1, 2, 3\}$

$E_1 = \{\{1, 2\}, \{2, 3\}\}$

G_2

$V_2 = \{1, 2\}$

$E_2 = \{\{2, 1\}\}$



$G_r = Ic(G_1, G_2)$

$V_r = \{1, 2, 3, 1^*, 2^*\}$

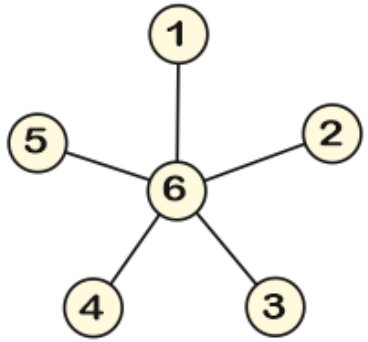
$E_r = \{\{1, 2\}, \{2, 3\}, \{1^*, 2^*\}, \{1^*, 1\} \{1^*, 2\}, \{1^*, 3\}, \{2^*, 2\}, \{2^*, 3\}, \{2^*, 1\}\}$

Operații pe grafuri

(Inter) Conectarea grafurilor

Def. Fie dat graful $G_1 = (V_1, E_1)$ și graful $G_2 = (V_2, E_2)$. Diferență a grafurilor G_1, G_2 va fi graful $G_r = (V_r, E_r)$, în care

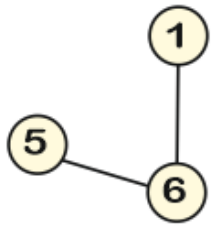
$V_r = V_1 \cup V_2$, $\quad E_r = \{\{u, v\} : \{u, v\} \in E_1 \,\&\&\, \{u, v\} \notin E_2$



G_1

$V_1 = \{1, 2, 3, 4, 5, 6\}$

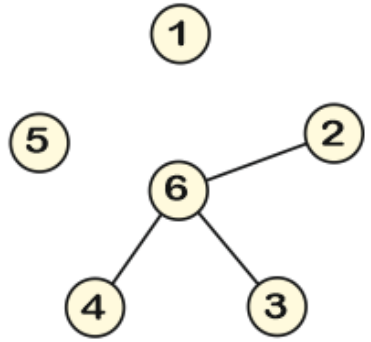
$E_1 = \{\{1, 6\}, \{2, 6\}, \{3, 6\}, \{4, 6\} \{5, 6\}\}$



G_2

$V_2 = \{1, 5, 6\}$

$E_2 = \{\{1, 6\}, \{5, 6\}\}$



$G_1 - G_2$

$V_r = \{1, 2, 3, 4, 5, 6\}$

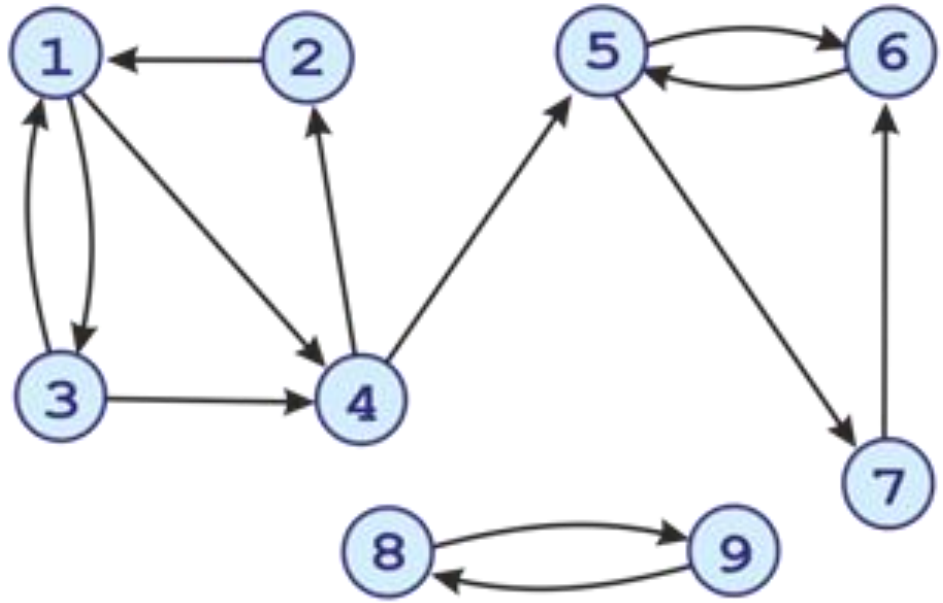
$E_r = \{\{4, 6\}, \{2, 6\}, \{3, 6\}\}$

Aplicații

Determinarea componentelor tare conexe în grafuri orientate

Def. *Componentă tare conexă* a unui graf orientat $G=(V,E)$ se numeşte mulțimea maximală de vârfuri $V' \subseteq V$: $\forall v_i, v_j \in V'$ există cel puțin câte un lanț $v_i \mapsto v_j$ şi $v_j \mapsto v_i$.

Descriere



Pas 1. Se construieşte graful $G^T = (V, E^T)$

Pas 2. Se lansează căutarea în adâncime pornind de la fiecare vârf necercetat din G^T. Pentru fiecare parcurgere se memorează cumulativ ordinea de cercetare a vârfurilor în vectorul f.

Pas 3. Se lansează căutarea în adâncime pe graful inițial G, consecutiv, pornind de la ultimul vârf inclus în f către primul, după vârfurile necercetate.

Pas 4. La fiecare căutare în adâncime realizată în pasul 3, se afişează vârfurile cercetate – acestea formează o componentă tare conexă.

Algoritmul Kosaraju

Lucrul individual :

Elaborați o aplicație de consolă cu interfață text pentru realizarea operațiilor complexe pe grafuri. În calitate de structură pentru datele inițiale folosiți matricea de adiacență

În următoarea sesiune:

- Structuri dinamice de date: structuri liniare.