

1 Vocabularul teoriei grafurilor

- Variații în definiția unui graf
- Grade
- Subgrafuri
- Operații cu grafuri
- Clase de grafuri
- Drumuri și circuite

2 Exerciții pentru seminarul 3 (săptămâna 13-17 octombrie)

Variații în definiția unui graf

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Multigraf: $G = (V, E)$, unde V este o mulțime nevidă (de noduri), și E este un **multiset** (de muchii) pe V , i. e., există o funcție $m : \binom{V}{2} \rightarrow \mathbb{N}$.

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

$e \in \binom{V}{2}$, cu $m(e) > 0$ este o muchie a multigrafului G ; dacă $m(e) = 1$, atunci e este o **muchie simplă**, altfel este o **muchie multiplă** cu **multiplă** $m(e)$.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.

Graful suport al unui graf, G , este graful obținut din G prin înlocuirea fiecărei muchii multiple printr-una simplă.

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Variații în definiția unui graf

Pseudograf (graf general): $G = (V, E)$, unde V este o mulțime (de noduri), și E este un **multiset** (de muchii) peste $V \cup \binom{V}{2}$, i. e., există o funcție $m : V \cup \binom{V}{2} \rightarrow \mathbb{N}$.

$e \in E \cap V$ (i. e., $|e| = 1$) este numită **buclă**.

Graful suport al unui pseudograf G este graful obținut din G prin înlocuirea fiecărei muchii multiple printr-una simplă și prin ștergerea buclilor.

Variații în definiția unui graf

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Digraf: $D = (V(D), E(D))$, unde $V(D)$ este o mulțime (de noduri), și $E(D) \subseteq V(D) \times V(D)$ este o mulțime de **arce** (sau **muchii orientate**).

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Dacă $e \in E$ atunci $e = (u, v)$ (sau simplu $e = uv$) este un arc orientat de la u către v și spunem că:

- u este **extremitatea inițială** of e , v este **extremitatea finală** ale lui e ;
- u și v sunt **adiacente**;
- e este **incident din u și către v** ;
- v este a **sucesor** al lui u și u este a **predecesor** al lui v etc.

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Variații în definiția unui graf

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

- **Pereche simetrică de arce:** (uv, vu) . uv este numit inversul lui vu .
- **Inversul** unui digraf D : se înlocuiește fiecare arc din D cu inversul său.
- **Graful suport** al unui digraf D fără bucle, notat cu $M(D)$, se obține înlocuind fiecare arc din D cu mulțimea corespunzătoare de două noduri (o muchie). $M(D)$ este un multigraf.
- Dacă $M(D)$ este un graf (simplu), atunci D este numit **graf orientat**.
- **Digraf complet simetric:** orice două noduri (distincte) sunt unite printr-o pereche simetrică de arce.
- **Turneu:** un graf orientat complet (orice două noduri (distincte) sunt unite exact printr-un arc).

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

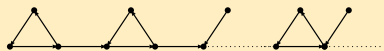
Variații în definiția unui graf

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

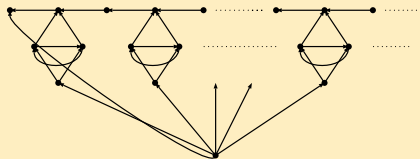
(Di)grafuri infinite: mulțimea nodurilor și/sau mulțimea muchiilor (arcelor) este numărabil infinită.

Un graf infinit este **local finit** dacă $N(v)$ este o mulțime finită, pentru orice nod v .

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -



The Danaids barrel



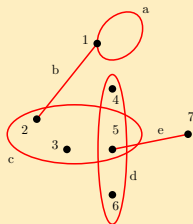
The sisyphus graph

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

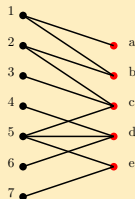
Hipergrafuri (Sisteme de mulțimi finite)

- Muchiile, numite acum **hipermuchi**, nu mai sunt restricționate să fie submulțimi cu două elemente ale mulțimii de noduri. O hipermuchie este submulțime a mulțimii de noduri.
- **Hipergrafuri k -uniforme**: fiecare muchie are cardinalul k .

Fiecare hipergraf poate fi reprezentat ca un graf bipartit:



Hypergraph H



Bipartite graph associated with H

Fie $G = (V, E)$ un graf și $v \in V$.

- **Gradul** unui nod v : $d_G(v) =$ numărul de muchii incidente cu v .
- v este un **nod izolat** dacă $d_G(v) = 0$ și **pendant** (sau **frunză**) dacă $d_G(v) = 1$.

$$\sum_{v \in V} d_G(v) = 2|E|.$$

- **Gradul maxim** $\Delta(G)$ și **gradul minim** $\delta(G)$:

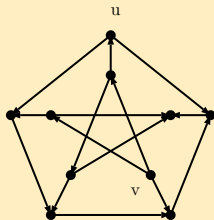
$$\Delta(G) = \max_{v \in V} d_G(v), \quad \delta(G) = \min_{v \in V} d_G(v).$$

- Dacă $\Delta(G) = \delta(G) = k$, atunci G este **k -regulat**.
- **Graf nul**: un graf 0-regulat .

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
 * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
 Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
 Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
 - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.

Exemplu

$$d_G^+(u) = 2, d_G^-(u) = 1; d_G^+(v) = 3, d_G^-(v) = 0$$



C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
 * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
 Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
 Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
 - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

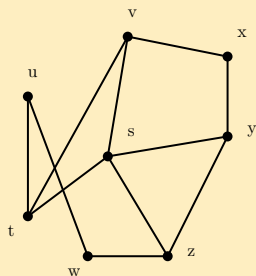
Fie $G = (V(G), E(G))$ un graf.

- **Subgraf** al lui G : un graf $H = (V(H), E(H))$ așa încât $V(H) \subseteq V(G)$ și $E(H) \subseteq E(G)$.
- **Graf parțial** al lui G : un subgraf H al lui G astfel ca $V(H) = V(G)$.
- **Subgraf generat de** $B \subseteq E(G)$ **în** G : un subgraf al lui G , $H = (V(H), E(H))$, astfel că $E(H) = B$ și $V(H) = V(G)$. Se notează prin $\langle B \rangle_G$.
- **Subgraf indus**: un subgraf H al lui G așa încât $E(H) = \binom{V(H)}{2} \cap E(G)$. Dacă $A \subseteq V(G)$, **subgraful indus de** A **în** G este notat cu $[A]_G$ sau $G[A]$.

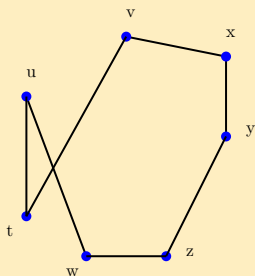
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Exemplu

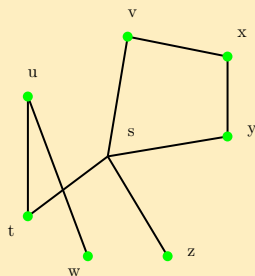
Un graf G , $G - s$, și $G - \{vt, wz, zy\}$.



G



$G - s$



$G - \{vt, wz, zy\}$

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Operații cu grafuri

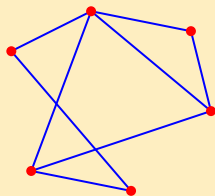
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Operație unară: $G = (V(G), E(G))$

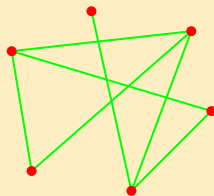
- **Complementul** unui graf G : un graf \overline{G} , cu $V(\overline{G}) = V(G)$ și

$$E(\overline{G}) = \binom{V(G)}{2} \setminus E(G).$$

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.
Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *



G



\overline{G}

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Operații cu grafuri

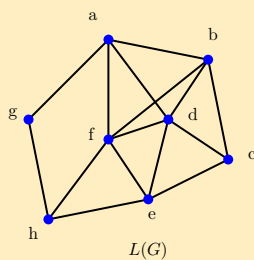
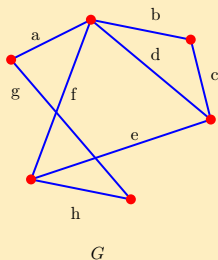
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Operație unară: $G = (V(G), E(G))$

- **Graful reprezentativ al muchiilor (line-graful)** lui G : un graf $L(G)$, cu $V(L(G)) = E(G)$ și

$$E(L(G)) = \{ef : e, f \in E(G), e \text{ și } f \text{ sunt adiacente în } G\}.$$

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.



- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Operații cu grafuri

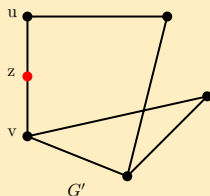
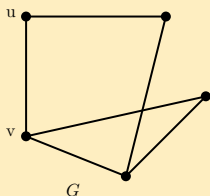
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Operație unară: $G = (V(G), E(G))$

- **Graful obținut din G prin inserarea unui nod nou (z) pe o muchie ($e = uv$):** graful G' , cu $V(G') = V(G) \cup \{z\}$ și

$$E(G') = E(G) \setminus \{uv\} \cup \{uz, zv\}.$$

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *



Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Operații cu grafuri

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

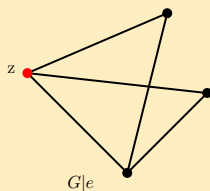
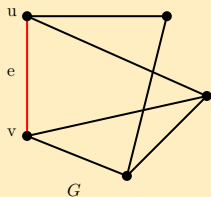
Operație unară: $G = (V(G), E(G))$

- Graful obținut din G prin contractia unei muchii $e = uv \in E(G)$: graful $G|e$ cu

$$V(G|e) = V(G) \setminus \{u, v\} \cup \{z\},$$

$$E(G|e) = E([V(G) \setminus \{u, v\}]_G) \cup \{yz : yu \text{ sau } yv \in E(G)\}.$$

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph



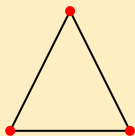
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Operație binară: G, G' cu $V(G) \cap V(G') = \emptyset$

- **Produsul cartezian** al grafurilor G și G' : graful $G \times G'$ cu

$$V(G \times G') = V(G) \times V(G').$$

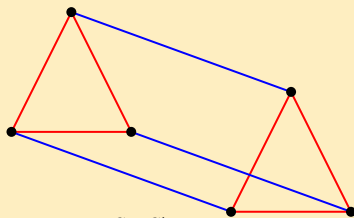
$$E(G \times G') = \{(u, u')(v, v') : u, v \in V(G), u', v' \in V(G'), \\ u = v \text{ și } u'v' \in E(G') \text{ sau } u' = v' \text{ și } uv \in E(G)\}.$$



G



G'



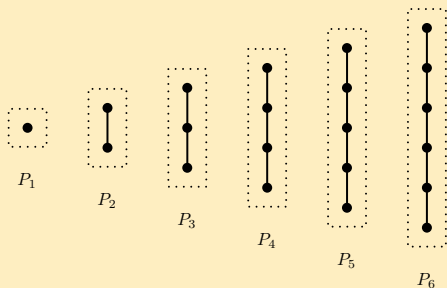
$G \times G'$

Clase de grafuri - Drumurile P_n

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Drumul de ordin n , P_n : $V(P_n) = \{1, 2, \dots, n\}$ și $E(P_n) = \{12, 23, \dots, n-1n\}$.

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph



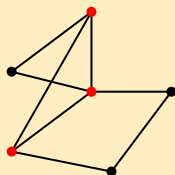
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

O submulțime de k noduri a graf G care induce un graf complet este numită o k -clică.

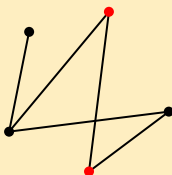
$$\text{numărul de clică al lui } G : \omega(G) = \max_{Q \text{ clică în } G} |Q|.$$

Remarcăm că $\omega(G) = \alpha(\overline{G})$.

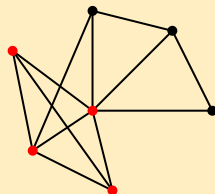
Exemplu



$$\omega(G) = 3$$



$$\omega(G) = 2$$



$$\omega(G) = 4$$

Clase de grafuri - Grafurile bipartite

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

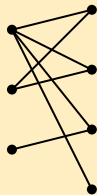
Graf bipartit: un graf G cu proprietatea că $V(G)$ poate fi partiționat în două clase care sunt mulțimi stabile.

Dacă $V(G) = S \cup T$, $S \cap T = \emptyset$, $S, T \neq \emptyset$, cu S și T mulțimi stabile în G , atunci G este notat $G = (S, T; E(G))$.

Graf bipartit complet: $G = (S, T; E(G))$, cu $uv \in E(G)$, $\forall u \in S$ și $\forall v \in T$; se notează cu $K_{s,t}$, unde $s = |S|$, $t = |T|$.

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exemplu



A bipartite graph



$K_{1,1}$



$K_{1,2}$



$K_{2,3}$



$K_{3,3}$

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Graf planar: un graf care poate fi reprezentat într-un plan astfel ca fiecărui nod să îi corespundă un punct al aceluiași plan și fiecărei muchii să îi corespundă o curbă simplă care unește punctele corespunzătoare extremităților și **aceste curbe se intersectează doar în extremitățile lor**. Un graf care nu este planar este numit **graf ne-planar**.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.
Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Grafuri planare: **Problemă de decizie**

PLAN Instanță: G graf.

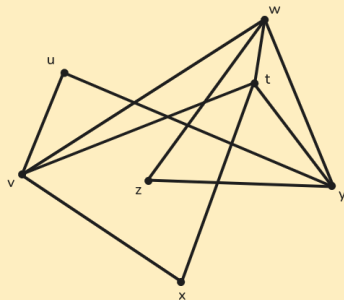
întrebare: Este G planar?

PLAN aparține clase de complexitate P (Hopcroft, Tarjan, 1972, $\mathcal{O}(n + m)$).

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exemplu

Mersuri închise, parcursuri închise, circuite.

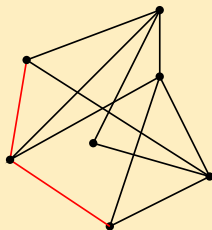


- Un mers închis: $u, uy, y, yt, t, tv, v, vw, w, wv, v, vu, u$.
- Un parcurs închis: $v, vw, w, wz, z, zy, y, yw, w, wt, t, tx, x, xv, v$.
- Un circuit: $u, uv, v, vx, x, xt, t, tw, w, wy, y, yu, u$ (în acest circuit vt este o coardă).

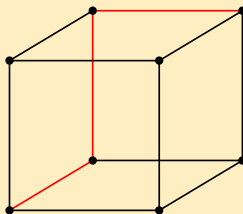
Fie $G = (V, E)$ un graf.

- Distanța în G de la u la v , $d_G(u, v)$ lungimea celui mai scurt drum în G de la u la v (dacă există un astfel de drum).
- Diametrul unui graf G , $d(G)$:

$$d(G) = \max_{u, v \in V} d_G(u, v).$$



$d(G) = 2$



$d(G) = 3$

Fie $G = (V, E)$ un graf.

- **Graf conex**: există câte un drum între orice două noduri ale grafului. Altfel graful este **neconex**.
- **Componentă conexă** a unui graf G : un subgraf maximal conex, H , of G (i. e., nu există vreun subgraf conex H' of G , $H' \neq H$, iar H este subgraf al lui H').
- Orice graf poate fi scris ca o reuniune disjunctă a componentelor sale conexe.
- Următoarea relație binară este o relație de echivalență: $\rho \subseteq V \times V$, dată prin $u\rho v$ (i. e., $(u, v) \in \rho$) dacă există un drum în G între u și v .
- Componentele conexe ale lui G sunt subgrafurile induse de clasele de echivalență ale relației ρ .

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Fie $G = (V, E)$ un graf conex.

- **Punct de articulație (cut-vertex)**: un nod $v \in V$ astfel că $G - v$ este neconex.
- **Mulțime de articulație (vertex cutset)**: o mulțime de noduri $S \subseteq V$ așa încât $G - S$ este neconex.
- Un **arbore** este un graf conex fără circuite.
- Un graf ale cărui componente conexe sunt arbori este o **pădure**.

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

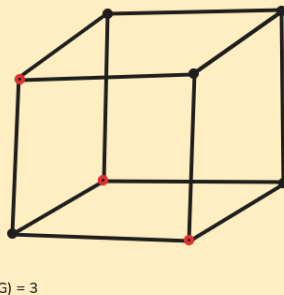
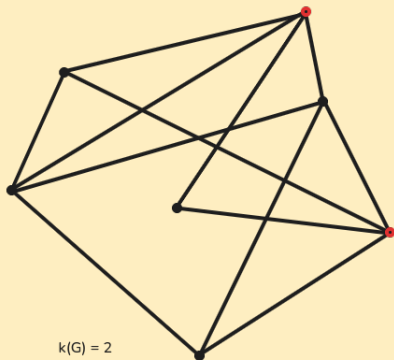
Mai general: într-un graf G care nu este neapărat conex $v \in V(G)$ este un **punct de articulație** dacă $G - v$ are mai multe componente conexe decât G .

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Drumuri și circuite - Conexiune

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Exemplu



Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Fie $G = (V, E)$ un graf conex.

- **Punte (bridge)**: o muchie $e \in E$ astfel că $G - e$ nu este conex.
- **Mulțime de muchii de articulație (edge-cutset)**: O submulțime de muchii $S \subseteq E$ așa încât $G - S$ este neconex.
- **Tăietura generată de o bipartiție** (V_1, V_2) a lui V este

$$E(V_1, V_2) = \{xy \in E : x \in V_1, y \in V_2\}.$$

- Pentru $p \in \mathbb{N}^*$, G este **graf p -muchie-conex** dacă G nu are o mulțime de muchii de articulație de cardinal $< p$.
- **Numărul de conexiune pe muchii, $\lambda(G)$** , al unui graf G este

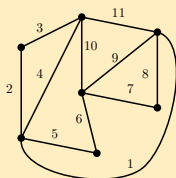
$$\lambda(G) = \max \{p \in \mathbb{N}^* : G \text{ este } p\text{-muchie-conex}\}.$$

Mai general: într-un graf G care nu este neapărat conex $e \in E(G)$ este o **punte** dacă $G - e$ are mai multe componente conexe decât G .

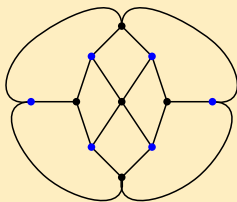
Fie G un (di)graf.

- G este **eulerian** dacă există un parcurs închis în G care trece prin fiecare muchie a lui G . Un astfel de **parcurs** este numit **eulerian**.
- G este **hamiltonian** dacă există un circuit în G care trece prin fiecare nod al lui G . Un astfel de **circuit** este numit **hamiltonian**.

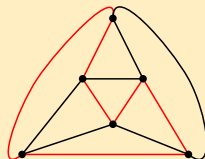
Recunoașterea (di)grafurilor euleriene se face în timp polinomial (**Euler, 1736**).



an Eulerian graph



a non Hamiltonian graph
(bipartite of odd order)



a Hamiltonian graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

Probleme hamiltoniene

HAM Instanță: G un graf.

Întrebare: Este G hamiltonian?

NP-completă (Karp, 1972).

NH Instanță: G un graf.

Întrebare: Este G ne-hamiltonian?

NH \in NP?

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exerciții pentru seminarul 3

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Exercițiul 5. Fie G un graf cu $n \geq 2$ noduri. Arătați că:

- (a) Dacă G este conex, atunci conține cel puțin un nod care nu este punct de articulație.
- (b) Dacă $n \geq 3$ și G este conex atunci conține două noduri care nu sunt puncte de articulație.
- (c) Adevărat sau fals: dacă G este conex iar $x \in V(G)$ nu este un punct de articulație al său, atunci x este o frunză într-un arbore parțial al lui G ?

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.

Exercițiul 6. Fie G un graf conex care nu conține două noduri pendante (frunze) cu un vecin în comun. Arătați că există două noduri adiacente prin ștergerea cărora G nu se deconectează.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exerciții pentru seminarul 3

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Exercițiul 7. Fie G un graf și H graful său reprezentativ al muchiilor ($H = L(G)$). Arătați că H este $K_{1,3}$ -free.

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Exercițiul 8. Fie G un graf. Arătați că:

- Dacă G are exact două noduri de grad impar, atunci aceste două noduri sunt unite printr-un drum în G .
- Dacă G este conex cu toate nodurile de grad par, atunci G are o muchie care nu este punte (ștergerea ei nu deconectează graful).
- Dacă G este conex cu toate nodurile de grad par, atunci nicio muchie a lui G nu este punte.
- Dacă G este conex cu toate nodurile de grad par și $|G| > 1$, atunci G este eulerian.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exerciții pentru seminarul 3

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Exercițiul 9. Fie G un graf. Arătați că

- (a) Numărul de noduri de grad impar este par.
- (b) Dacă G este conex și are k noduri de grad impar, atunci G este o reuniune $\lfloor k/2 \rfloor$ parcursuri disjuncte pe muchii.

Exercițiul 10. Fie G un graf astfel ca $N_G(u) \cup N_G(v) = V(G)$, $\forall u, v \in V(G)$, $u \neq v$. Arătați că G este graf complet.

Exercițiul 11. Fie G un graf cu proprietatea că $d_G(u) + d_G(v) \geq |G| - 1$, $\forall u, v \in V(G)$, $u \neq v$. Arătați că diametrul lui $d(G) \leq 2$.

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

