Proiectarea Algoritmilor

Curs 6 – Introducere în grafuri

Bibliografie

- Giumale – Introducere în Analiza Algoritmilor cap 5 și 5.1
- Cormen – Introducere în Algoritmi cap Algoritmi elementari de grafuri (23) – Reprezentări + Căutări
- http://www.h3c.com/portal/res/200706/01/20070601_108959_image001_201240_57_0.gif
- http://ashitani.jp/gv/
- http://www.graphviz.org
- http://en.wikipedia.org/wiki/PageRank

Plan curs

- Introducere
- Modalități de reprezentare
- Exemple de probleme practice
- Algoritmi de parcurgere
  - BFS
  - DFS

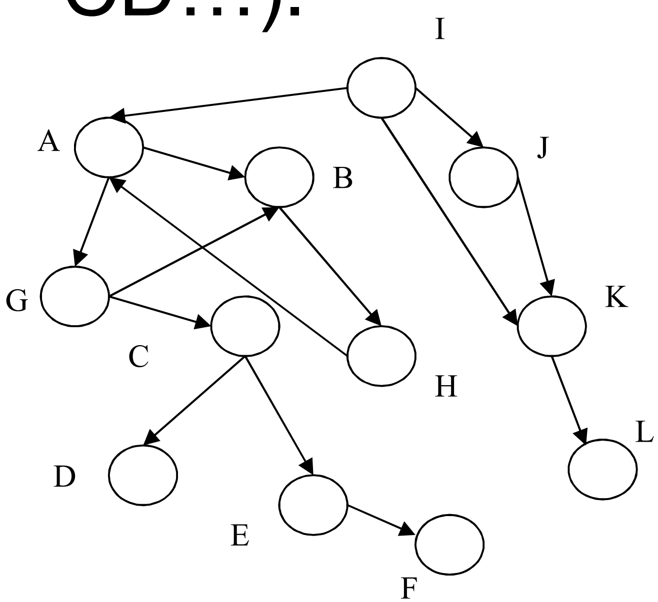
Introducere

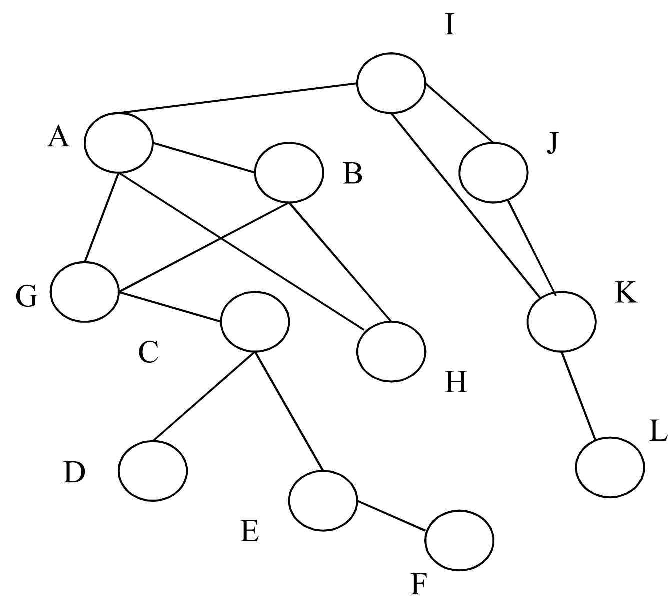
- Circa 6 cursuri în care sunt prezentați algoritmii cei mai importanți pentru prelucrarea grafurilor:
  - Parcurgere
  - Sortare topologică
  - Componente tare conexe
  - Puncte de articulație
  - Punți
  - Arbori minimi de acoperire
  - Drumuri de cost minim
  - Fluxuri maxime

- Încercăm să legăm algoritmii de aplicații cât mai practice.

Tipuri de grafuri

- Orientate: noduri (A-L) + arce (AB, BC, CD…).

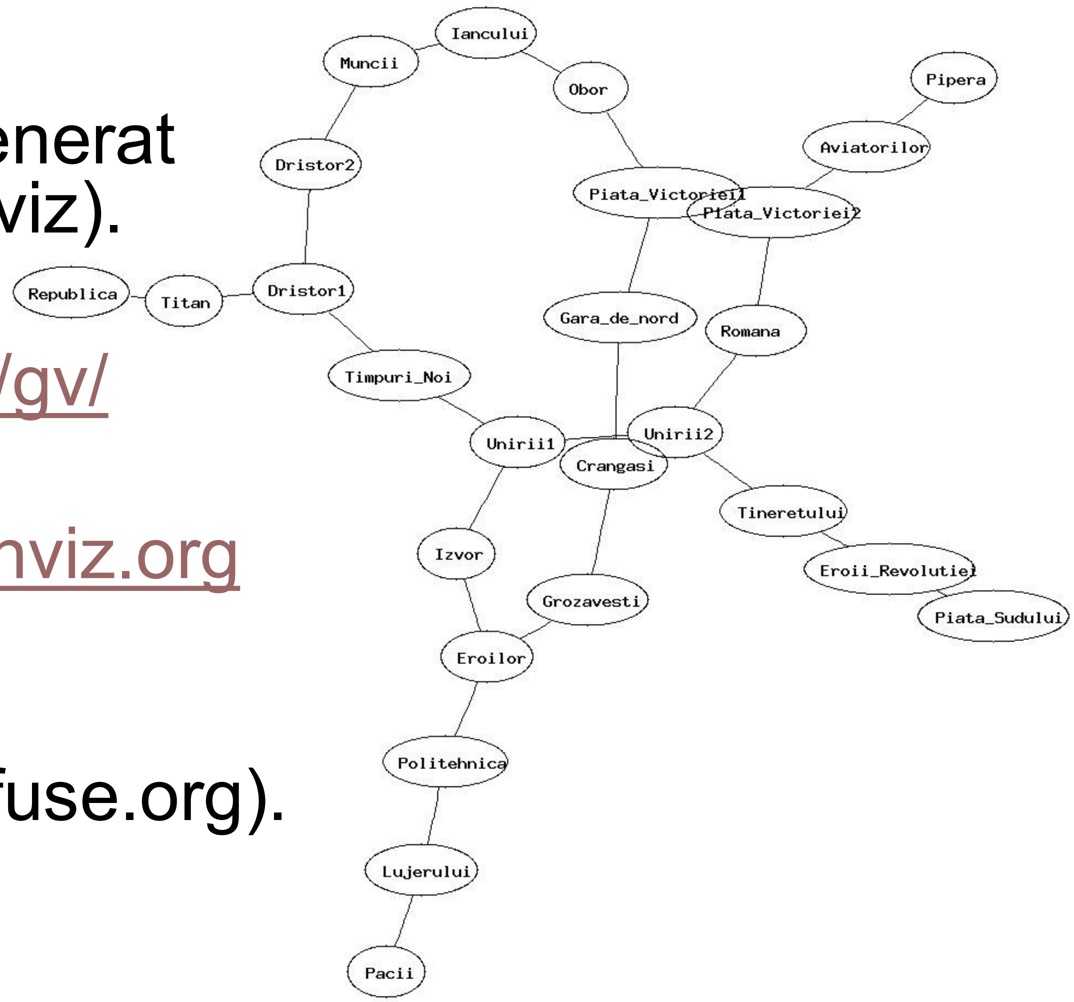




- Neorientate: noduri (A-L) + muchii (AB, BC, CD…).

Exemplu graf neorientat

- Exemplu graf generat cu neato (graphviz).
- http://ashitani.jp/gv/
- http://www.graphviz.org
- Biblioteci pentru vizualizare (Prefuse.org).



Modalități de descriere ale grafurilor

- Reprezentare în memorie:
  - Liste de adiacență;
  - Matrice de adiacență.

- Reprezentarea datelor de intrare:
  - Tupluri (sursă, destinație);
    - Întâlnite mai ales în descrierile folosind baze de date.
  - Limbaje specializate (ex: dot, GraphML, rdf).

Formate de reprezentare

- Listă adiacență :
  - Eroilor: Politehnica, Grozăvești, Izvor
  - Muncii: Dristor2, Iancului…
- Matrice adiacență:

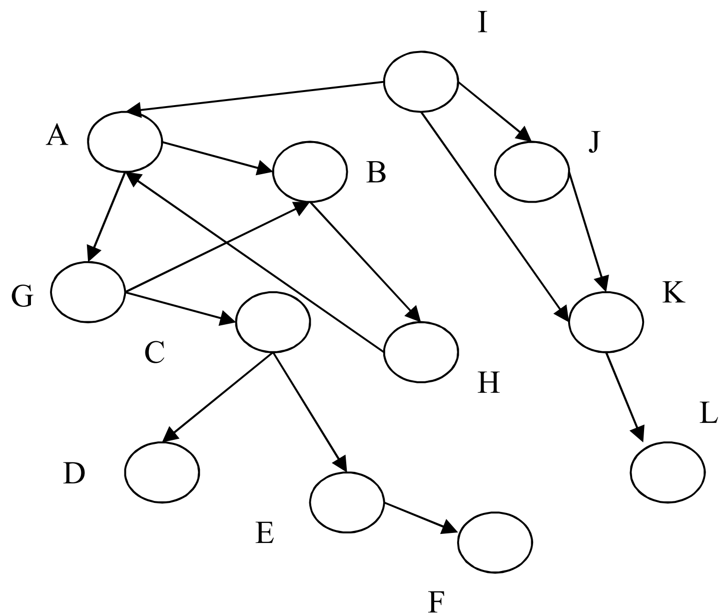
	Unirii2	Tineretului	Romană	…
Unirii2	-	1	1	
Tineretului	1	-		
Romană	1		-	
…				

- Tupluri:
  - (Dristor1; Dristor2)
  - (Eroilor; Grozăvești)
  - …
- Dot:
  - graph G {node;
  - Dristor2--Muncii--Iancului—Obor;
  - Piata_Victoriei1--Gara_de_nord--Crangasi--Grozavesti--Eroilor;
  - Pacii--Lujerului--Politehnica--Eroilor;
  - Republica--Titan--Dristor1--Timpuri_Noi--Unirii1--Izvor--Eroilor;
  - Dristor1--Dristor2;
  - Unirii1--Unirii2;
  - Piata_Victoriei1--Piata_Victoriei2;
  - Piata_Sudului--Eroii_Revolutiei--Tineretului--Unirii2--Romana--Piata_Victoriei2--Aviatorilor--Pipera;
  - }

Formate de reprezentare - GraphML

- GraphML
  - <graphml xmlns="http://graphml.graphdrawing.org/xmlns">
  - <graph edgedefault="undirected">
  - <!-- data schema -->
  - <key id="name" for="node" attr.name="name" attr.type="string"/>
  - <key id="gender" for="node" attr.name="gender" attr.type="string"/>
  - <!-- nodes -->
  - <node id="1">
  - <data key="name">Jeff</data>
  - <data key="gender">M</data>
  - </node>
  - <node id="2">
  - <data key="name">Ed</data>
  - <data key="gender">M</data>
  - </node>
  - <edge source="1" target="2"></edge>
  - </graph>
  - </graphml>

Matrice de adiacență

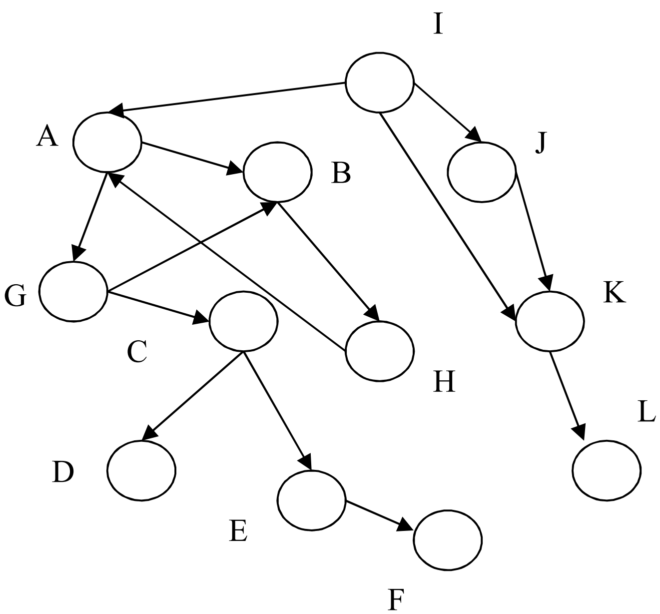


Cum se determină matricea de adiacență?

Matrice de adiacență

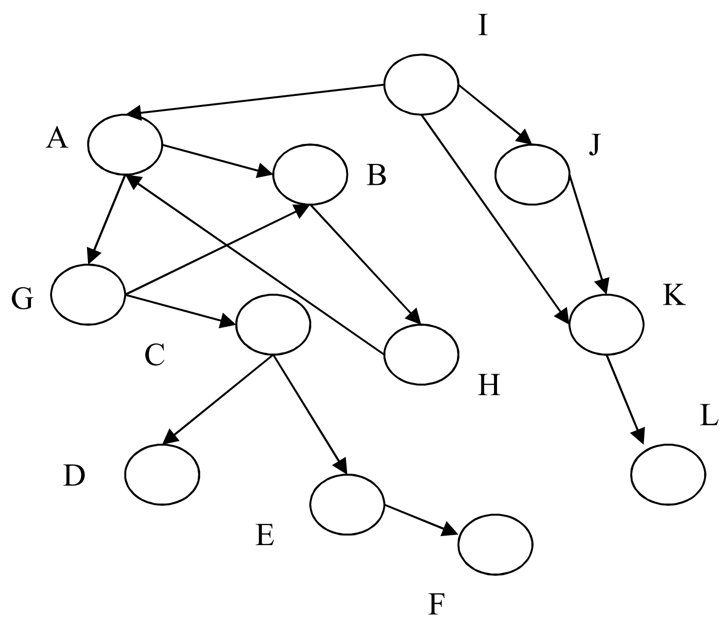
Matricea este rară?

- G – rar
- G – dens



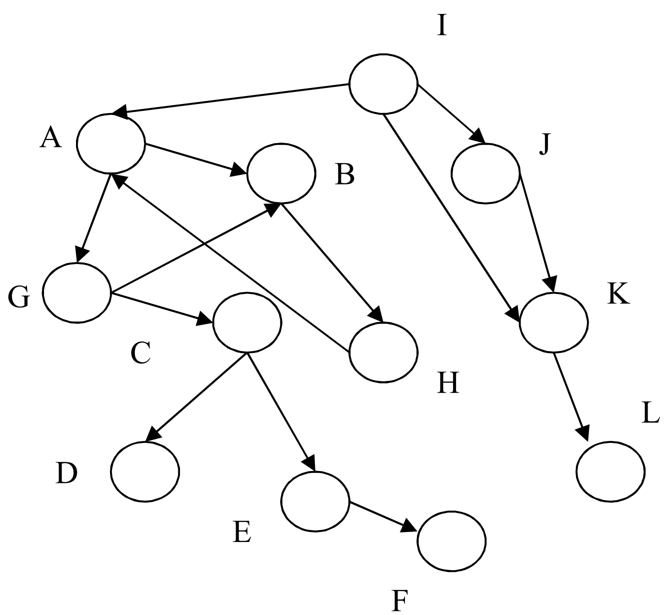
	A	B	C	D	E	F	G	H	I	J	K	L
A		1					1					
B								1				
C				1	1							
D												
E						1						
F												
G		1	1									
H	1											
I	1									1	1	
J											1	
K												1
L												

Vector de adiacență



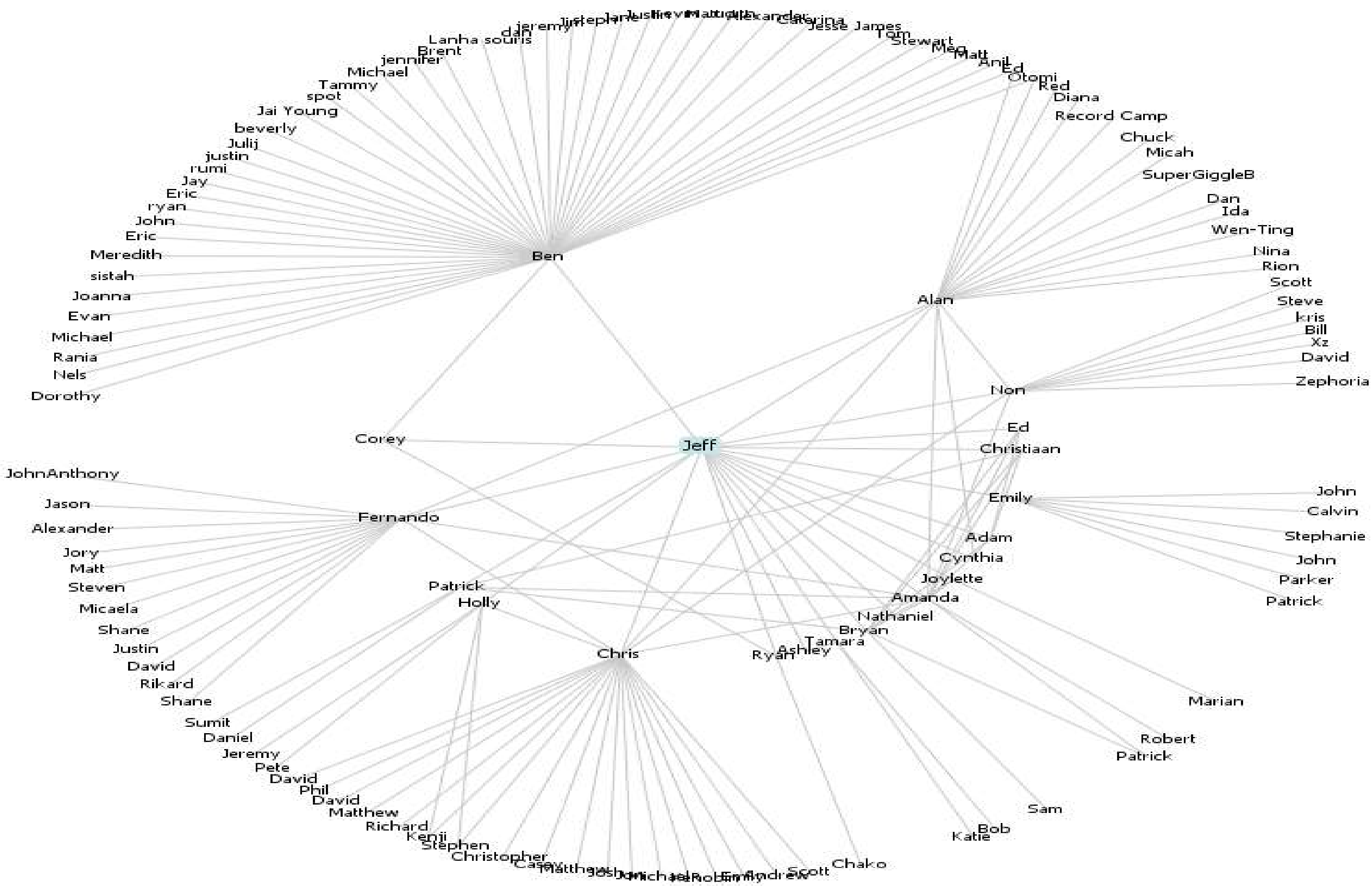
Cum se determină vectorul de adiacență?

Vector de adiacență

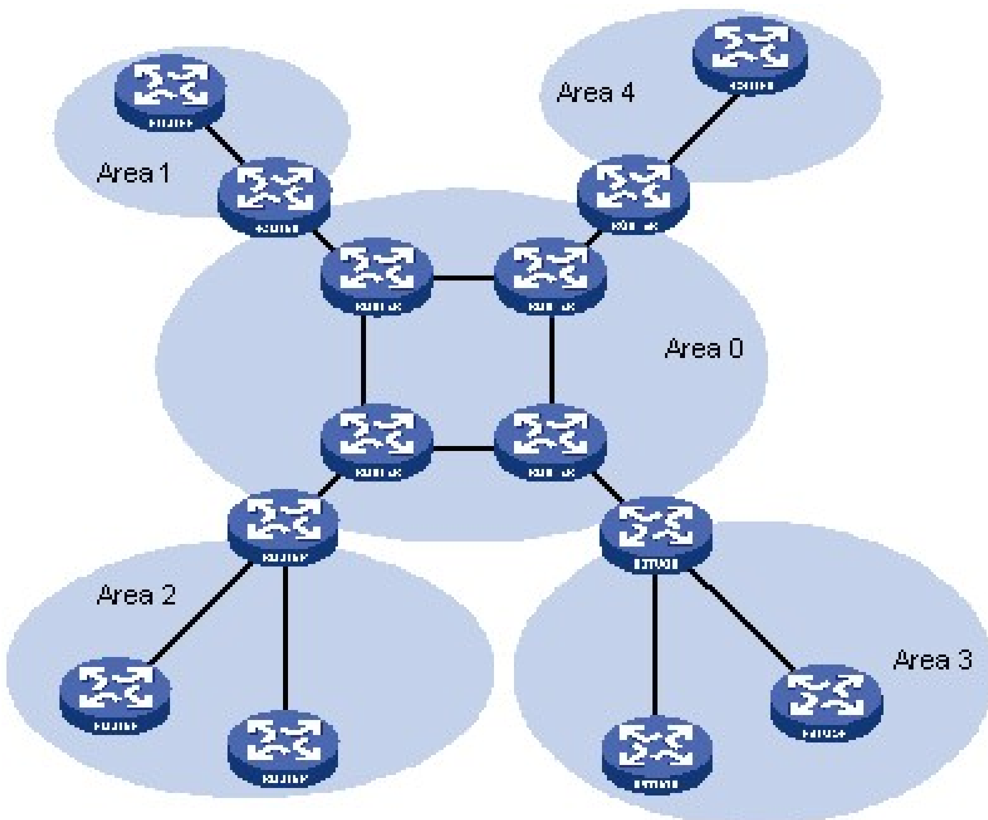


A	**B**	**G**	
B	**H**		
C	**D**	**E**	
D			
E	**F**		
F			
G	**B**	**C**	
H	**A**		
I	**A**	**J**	**K**
J	**K**		
K	**L**		
L			

Utilizări practice - Rețele sociale

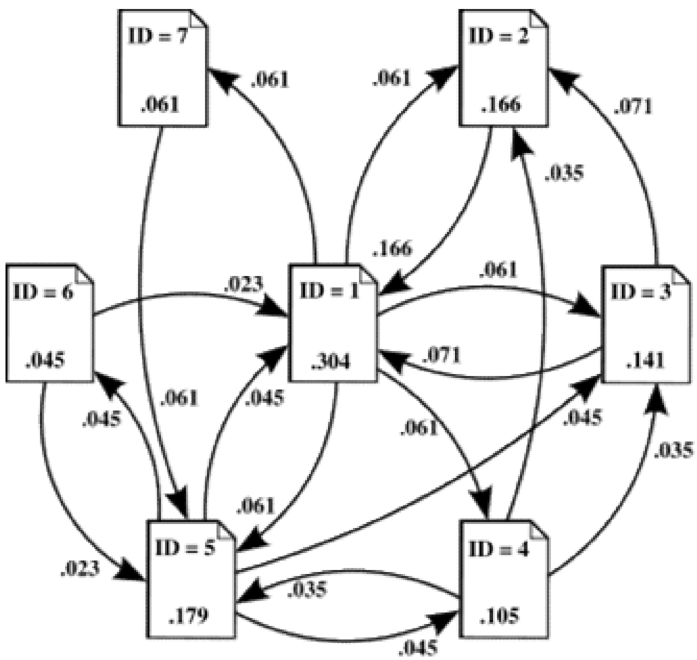


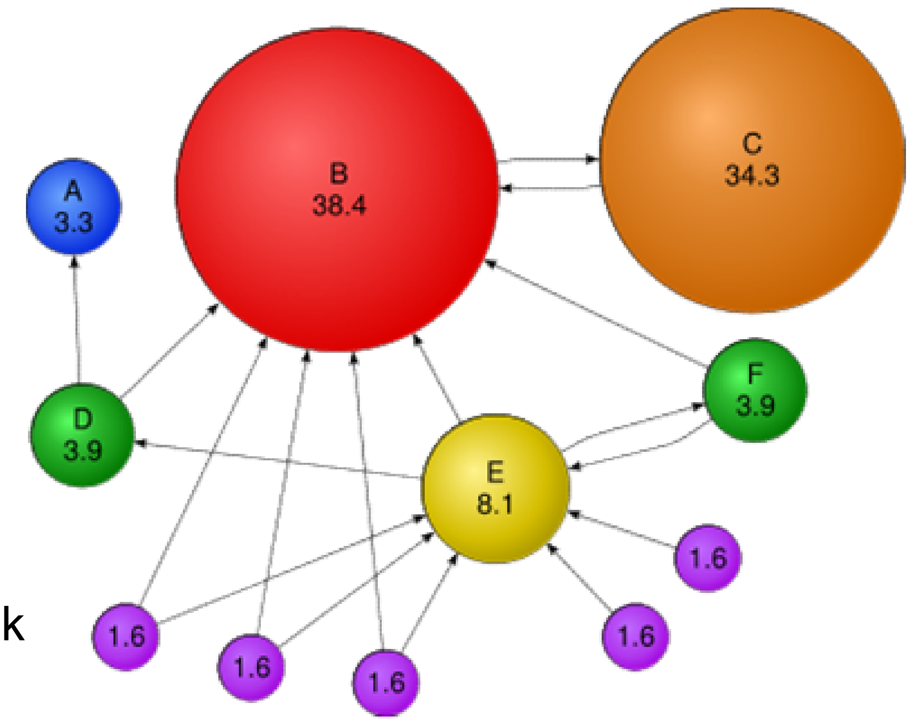
Utilizări practice – Rețele de calculatoare



http://www.h3c.com/portal/res/200706/01/20070601_108959_image001_201240_57_0.gif

Utilizări practice - Web





http://en.wikipedia.org/wiki/PageRank

Utilizări practice

- Hărți, rețele (calculatoare, instalații, etc.), rețele sociale, analiza fluxurilor (semaforizare, proiectarea dimensiunii țevilor de apă).
- Exemple simple:
  - Cel mai scurt drum între punctele A și B pe o hartă.
  - Radialitate – în rețele sociale: gradul în care rețeaua socială a unui individ se întinde în rețeaua globală pentru a schimba date și influență.
  - Page Rank (Google).

Algoritmi de parcurgere – Notații (1)

- G = (V,E);
- V – mulțimea de noduri;
- E – mulțimea de muchii / arce;
- (u,v) – arcul / muchia u,v;
- u..v – drum de la u la v; dacă există mai multe variante notăm u..x..v, u..y..v;
- R(u) - reachable(u) = mulțimea nodurilor ce pot fi atinse pe căi ce pleacă din u;

Algoritmi de parcurgere – Notații (2)

- succs(u) – mulțimea succesorilor lui u (graf orientat) sau mulțimea nodurilor adiacente lui u (graf neorientat);
- c(u) – culoarea nodului – specifică starea nodului la un anumit moment al parcurgerii:
  - Alb – nedescoperit;
  - Gri – descoperit, în curs de prelucrare;
  - Negru – descoperit și terminat (cu semnificații diferite pentru BFS si DFS).
- p(u) (π(u)) – “părintele lui u” – identificator al nodului din care s-a ajuns în nodul u prima oară.

Parcurgere în lățime (BFS)

- Nod de start (sursă): s.
- Determină numărul minim de arce / muchii între s și $\forall$ u $\in$ V = numărul de pași între sursă și orice alt nod din graf (acesta este cel mai scurt drum în condițiile în care nu există o funcție de cost asociată grafului).
- $\delta(s,u)$ – costul optim al s..u; $\delta(s,u) = \infty \Leftrightarrow u \notin R(s)$.
- Dist(s,u) – costul drumului descoperit s..u.
- Ex: Politehnica → restul stațiilor de autobuz (de câte bilete am nevoie?)

BFS – Structura de date

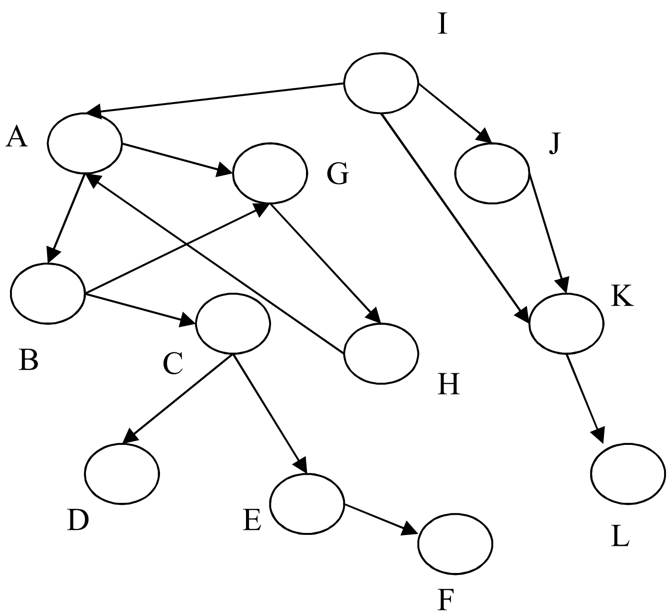
- Folosește o coadă (FIFO) pentru a reține nodurile ce trebuie prelucrate.
- Pentru fiecare nod se rețin:
  - Părintele – π(u) (p(u));
  - Dist(s,u) – distanța până la nodul sursă;
  - Culoarea nodului.

BFS – Algoritm

- BFS(s,G)
  - **Pentru fiecare** nod u (u ∈ V)
    - p(u) = null; dist(s,u) = inf; c(u) = alb; // inițializări
  - Q = (); // se folosește o coadă în care reținem nodurile de prelucrat
  - dist(s,s) = 0; // actualizări: distanța de la sursă până la sursă este 0
  - Q ← Q + s; // adăugăm sursa în coadă → începem prelucrarea lui s
  - c(s) = gri; // și atunci culoarea lui devine gri
  - **Cât timp** (!empty(Q)) // cât timp mai am noduri de prelucrat
    - u = top(Q); // se determină nodul din vârful cozii
    - **Pentru fiecare** nod v (v ∈ succs(u)) // pentru toți vecinii
      - **Dacă** c(v) este alb // nodul nu a mai fost găsit, nu e în coadă
        - **Atunci** { dist(s,v) = dist(s,u) + 1; p(v) = u; c(v) = gri; Q ← Q + v;} // actualizăm structura date
    - c(u) = negru; // am terminat de prelucrat nodul curent
    - Q ← Q - u; // nodul este eliminat din coadă

BFS – Exemplu

Sursa = A

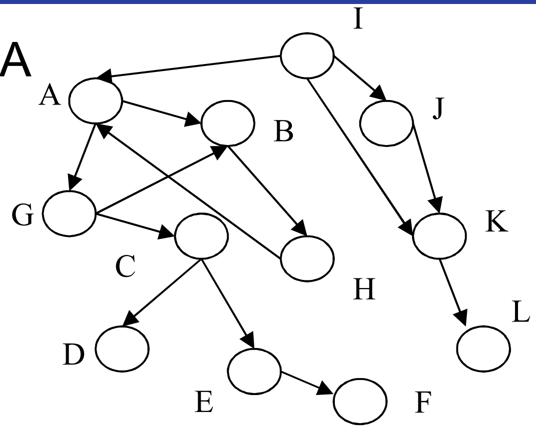


- BFS(s,G)
  - **Pentru fiecare** nod u (u ∈ V)
    - p(u) = null; dist(s,u) = inf; c(u) = alb; // inițializări
  - Q = (); // se folosește o coadă pentru nodurile de prelucrat
  - dist(s,s) = 0; // actualizări: distanța de la s la s e 0
  - Q ← Q + s; // adăugăm sursa în coadă → începem cu s
  - c(s) = gri; // și atunci culoarea lui devine gri
  - **Cât timp** (!empty(Q)) // cât timp am noduri de prelucrat
    - u = top(Q); // se determină nodul din vârful cozii
    - **Pentru fiecare** nod v (v ∈ succs(u)) // pentru toți vecinii
      - **Dacă** c(v) este alb // nodul nu a mai fost găsit, nu e în Q

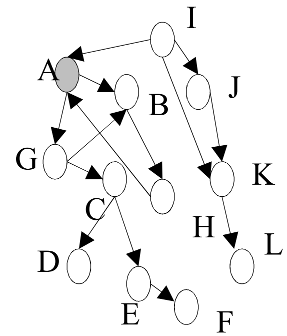
        Atunci { dist(s,v) = dist(s,u) + 1; p(v) = u; c(v) = gri;
        Q ← Q + v;} // actualizăm structura date
    - c(u) = negru; // am terminat de prelucrat nodul curent
    - Q ← Q - u; // nodul este eliminat din coadă

BFS – Evoluția explorării

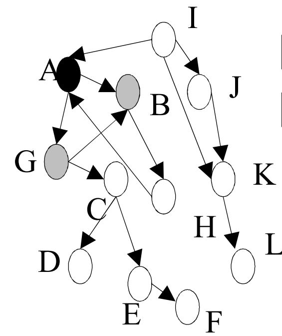
Sursa = A



Q = A; d(A) = 0
p(A) = null

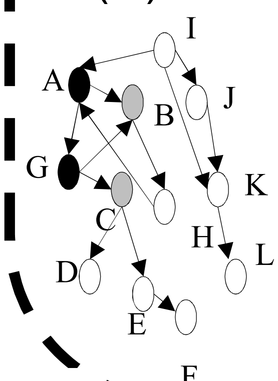


Q = G, B
d(B) = d(G) = 1

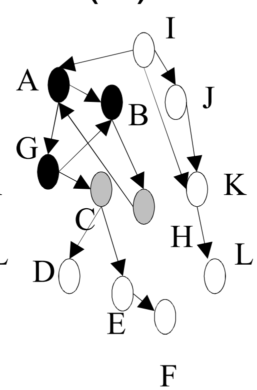


p(B) = A
p(G) = A

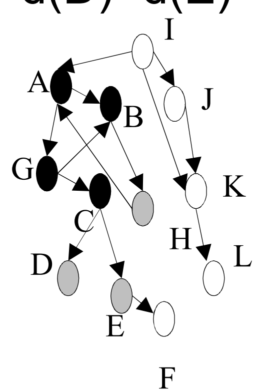
Q = B, C
d(C) = 2



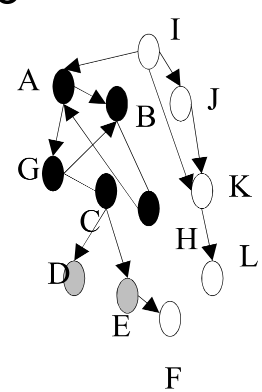
Q = C, H
d(H) = 2



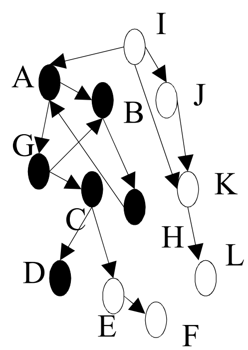
Q = H, D, E
d(D)=d(E)=3



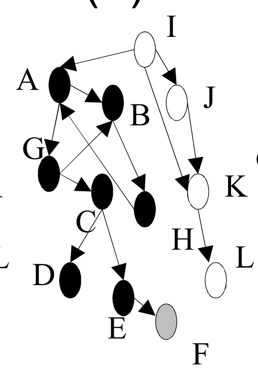
Q = D, E



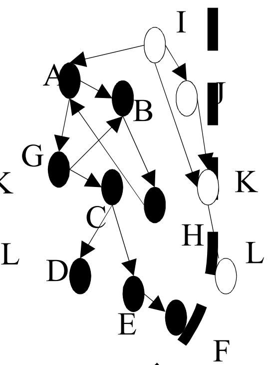
Q = E



Q = F
d(F)=4

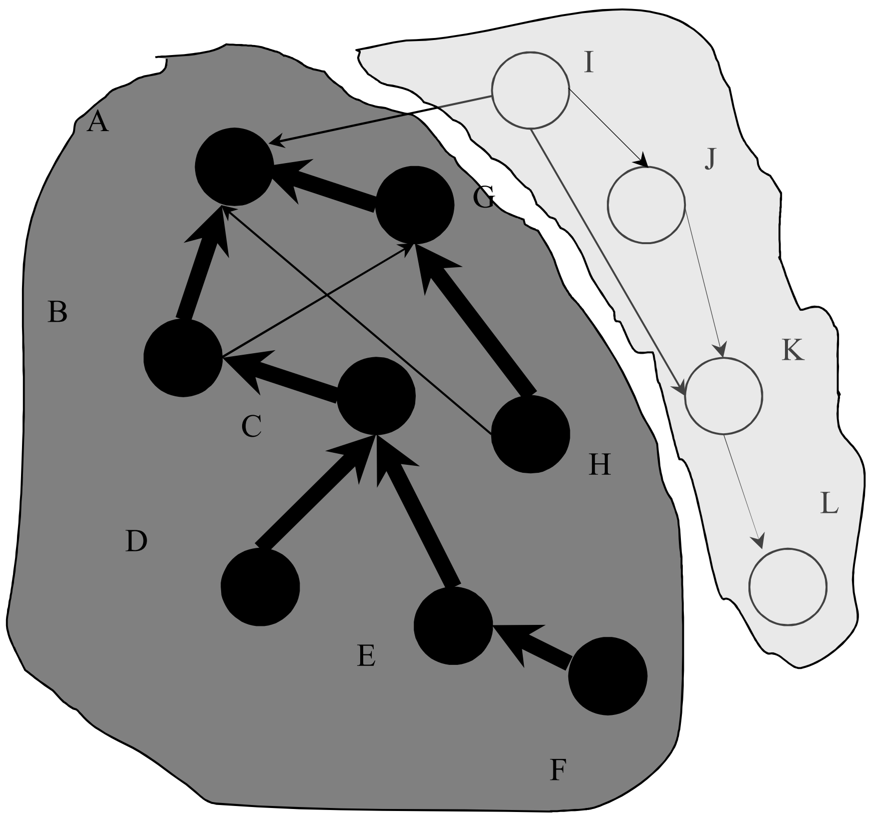


Q = Ø



p(C) = G p(H) = B p(D) = p(E) = C p(F) = E

BFS – Zona de explorare



BFS – Proprietăți (I)

- Lema 5.1. În cursul execuției BFS(s,G) $v \in Q \Leftrightarrow v \in R(s)$.
  - $\rightarrow$ $v \in Q \rightarrow v \in R(s)$: Q conține exclusiv noduri din R(s) (BFS parcurge toate nodurile ce pot fi atinse din s);
    - Caz de bază: $s \in R(s)$ Adevărat
    - Pas inducție: Q' conține doar noduri din R(s) $\rightarrow$ Q $\leftarrow$ Q' + v conține doar noduri din R(s) ($v \in R(s)$)
      - Dacă u = top(Q') $\rightarrow u \in R(s) \rightarrow v \in succs(u) \rightarrow (u,v) \in E \rightarrow v \in R(s)$
  - $\leftarrow$ toate nodurile ce pot fi atinse din s vor fi introduse cândva în coadă. (drum $s = v_0v_1\ldots v_p = v$)
  - Dem prin inducție! $P(i) = \forall\, i \in 0..p$, se execută $Q \leftarrow Q + v_i$
    - Caz de bază: i = 0, se execută $Q \leftarrow Q + s$ Adevărat
    - Pas inducție: P(i) Adevărat $\rightarrow$ P(i+1) Adevărat
      - Conform ipotezei inductive, la un moment dat avem $Q \leftarrow Q + v_i$. $\rightarrow$ cândva, v_i = top(Q) $\rightarrow$ dacă $c(v_{i+1})$ = alb, atunci este introdus in Q ($Q \leftarrow Q + v_{i+1}$). Altfel, $c(v_{i+1}) \neq$ alb $\rightarrow$ v_{i+1} este/a fost deja în coadă pt. că un nod își schimbă culoarea numai la inserție.

BFS – Proprietăți (II)

Lema 5.2. $\forall$ (u,v) $\in$ E, $\delta(s,v) \leq \delta(s,u) + 1$

- $\delta(s,v) \leq \delta(s,u) + 1$ este egalitate când v este descoperit din u; < când v deja a fost descoperit înainte să se ajungă în u.
- Dem prin reducere prin absurd! Pp. $\delta(s,v) > \delta(s,u) + 1$ $\rightarrow$ $\delta(s,u) + 1$ este costul unui drum optim.

- Lema 5.3. La terminarea BFS(s,G) există proprietatea $dist(s,u) \geq \delta(s,u)$.
  - Dem prin inducție folosind Lema 5.1 și Lema 5.2!
    - Caz de bază: $dist(s,s) = 0 = \delta(s,s)$ Adevărat
    - Pas inducție: pt. $\forall$ nod din Q': $dist(s,u) \geq \delta(s,u)$.
      - Cum $dist(s,v) = dist(s,u) + 1$ si $dist(s,u) \geq \delta(s,u)$ $\rightarrow$ $dist(s,v) \geq \delta(s,u) +1 \geq \delta(s,v)$
    - Conf. Lema 5.1, v $\in$ Q <=> v $\in$ R(s) și $dist(s,u) \geq \delta(s,u)$.
    - Dacă v $\notin$ R(s), $dist(s,v) = \delta(s,v) = \infty$

BFS – Proprietăți (III)

- Lema 5.4. După orice execuție a ciclului principal al BFS, Q conține $v_1, v_2, \ldots, v_p$ ai:
  - $Prop(Q) = dist(s,v_1) \leq dist(s,v_2) \leq \ldots \leq dist(s,v_p) \leq dist(s,v_1) + 1$
  - => la un moment dat in coadă sunt elemente de pe același nivel din arborele generat de BFS (sau maxim 1 nivel diferență).
  - Dem prin inducție după numărul de elemente din Q! (demonstrăm invarianța Prop(Q) la inserare și eliminare de elemente în/din Q.)
  - Fie $u = v_1 = top(Q)$
  - Inserție: fie v succ lui u și cul(v) = alb → coada devine: $v_1, v_2, \ldots, v_p, v_{p+1} = v$. $dist(s,v) = dist(s,v_1) + 1 \rightarrow dist(s,v_1) \leq dist(s,v_2) \leq \ldots \leq dist(s,v_p) \leq dist(s,v_{p+1}) \leq dist(s,v_1) + 1$
  - Eliminare: $dist(s,v_1) \leq dist(s,v_2) \rightarrow dist(s,v_1) + 1 \leq dist(s,v_2) + 1 \rightarrow$ coada devine: $v_2, \ldots, v_p$, $\rightarrow dist(s,v_2) \leq \ldots \leq dist(s,v_p) \leq dist(s,v_1) + 1 \leq dist(s,v_2) + 1$
  - Cum relația este respectată, înseamnă că Prop(Q) este adevărată.

BFS – Proprietăți (IV)

- Corolar
  - d(u) = momentul în care nodul u este inserat în coada Q. Atunci:

    $d(u) < d(v) \Rightarrow dist(s,u) \leq dist(s,v)$.

- Teorema 5.1. BFS este corect și după terminare $\delta(s,u) = dist(s,u)$, $\forall$ u din V.
  - Utilizăm notația $V_k = \{ u \in V \mid \delta(s,u) = k \}$
  - Dem prin inducție $P(k) = \{ u \in V_k \mid \delta(s,u) = dist(s,u)$ && $(k > 0 \Rightarrow \pi(u) \in V_{k-1})$ && $(k = 0 \Rightarrow \pi(u) = null) \}$
  - Caz de bază: $k = 0$, $V_0 = \{s\}$, $\delta(s,s) = dist(s,s) = 0$ și $\pi(s) = null$ Adevărat

BFS – Proprietăți (V)

Pas de inducție: $k > 0$, $P(k)$ adevărat → $P(k+1)$ adevărat

- Alegem la întâmplare un nod $v \in V_{k+1}$ → $\delta(s,v) = k+1$ și fie $u \in V_k$ predecesorul lui v pe drumul cel mai scurt s..v.
- Caz 1: u este descoperit după v
  - $dist(s,v) \leq dist(s,u)$ (Corolar 5.1) iar $dist(s,u) = \delta(s,u)$ → $dist(s,v) \leq k$
  - $dist(s,v) \geq \delta(s,v) = k+1$ (Lema 5.3)
- Caz 2: u este descoperit înainte de v și v e descoperit pe arcul (u,v)
  - $dist(s,v) = dist(s,u) + 1$ și $\pi(v) = u$, $u \in V_k$ → $P(k+1)$ este adevărat
- Caz 3: u este descoperit înainte de v și v NU e descoperit pe arcul (u,v). Fie $z \neq u$, a.î. v e descoperit pe arcul (z,v)
  - a) $d(u) < d(z) < d(v)$ → $cul(v) = alb$ la $d(u)$ → v e descoperit pe arcul (u, v)
  - b) $d(z) < d(u)$ → $dist(s,z) \leq dist(s,u) = \delta(s,u)$ → $dist(s,z) \leq k$
    $dist(s,v) = dist(s,z) + 1$ și $k + 1 = \delta(s,v) \leq dist(s,v)$ → $dist(s,z) \geq k$
    → $dist(s,z) = k$ → $dist(s,v) = k + 1 = \delta(s,v)$ → drumul s..z,v e optim
    → $z \in V_k$ → $\pi(v) = z \in V_k$ → $P(k+1)$ este adevărat

Complexitate? Optimalitate? Completitudine?

BFS – Complexitate și Optimalitate

Complexitate:

$O(n+m)$

n = număr noduri

m = număr muchii

Optimalitate: DA

Parcurge tot graful? NU

Parcurgere în adâncime (DFS)

- Nu mai avem nod de start, nodurile fiind parcurse în ordine.

- d(u) = momentul descoperirii nodului (se trece prima oară prin u și e totodată și momentul începerii explorării zonei din graf ce poate fi atinsă din u).
- f(u) = timpul de finalizare al nodului (momentul în care prelucrarea nodului u a luat sfârșit)
  - Tot subarborele de adâncime dominat de u a fost explorat.
  - Alternativ: tot subgraful accesibil din u a fost descoperit și finalizat deja.

DFS – Structura de date

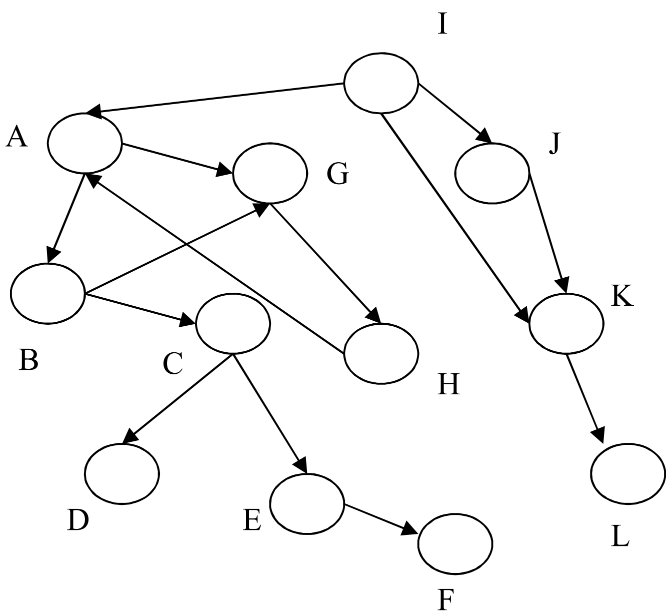
- Folosește o stiva (LIFO) pentru a reține nodurile ce trebuie prelucrate
  - În implementările uzuale, stiva este rareori folosită explicit;
  - Se apelează la recursivitate pentru a simula stiva.
- Folosește o variabilă globală *timp* pe baza căreia se calculează timpii de descoperire și de finalizare ai fiecărui nod.
- Pentru fiecare nod se rețin:
  - Părintele – $\pi(u)$ (p(u));
  - Timpul de descoperire – d(u);
  - Timpul de finalizare – f(u);
  - Culoarea nodului.

DFS – Algoritm

- DFS(G)
  - V = noduri(G)
  - **Pentru fiecare** nod u (u ∈ V)
    - c(u) = alb; p(u) = null; // inițializare structură date
  - timp = 0; // reține distanța de la rădăcina arborelui DFS pană la nodul curent
  - **Pentru fiecare** nod u (u ∈ V)
    - **Dacă** c(u) este alb
      - **Atunci** explorare(u); // explorez nodul

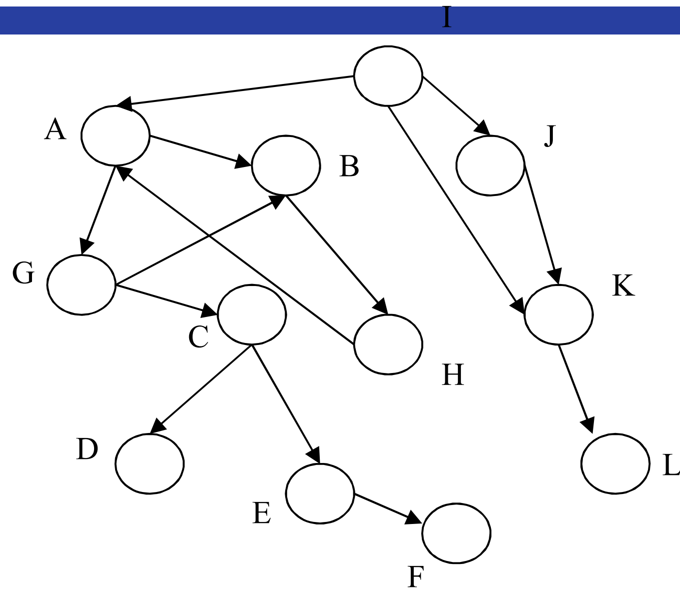
- explorare(u)
  - d(u) = ++ timp; // timpul de descoperire al nodului u
  - c(u) = gri; // nod în curs de explorare
  - **Pentru fiecare** nod v (v ∈ succs(u)) // încerc sa prelucrez vecinii
    - **Dacă** c(v) este alb
      - **Atunci** {p(v) = u; explorare(v);} // dacă nu au fost prelucrați deja
  - c(u) = negru; // am terminat de explorat nodul u
  - f(u) = ++ timp; // timpul de finalizare al nodului u

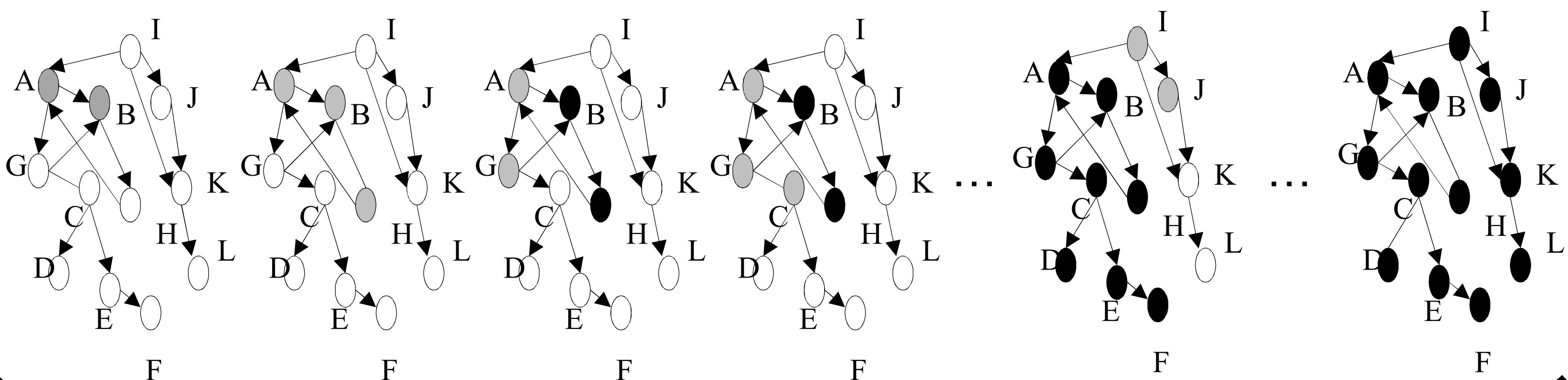
DFS – Exemplu



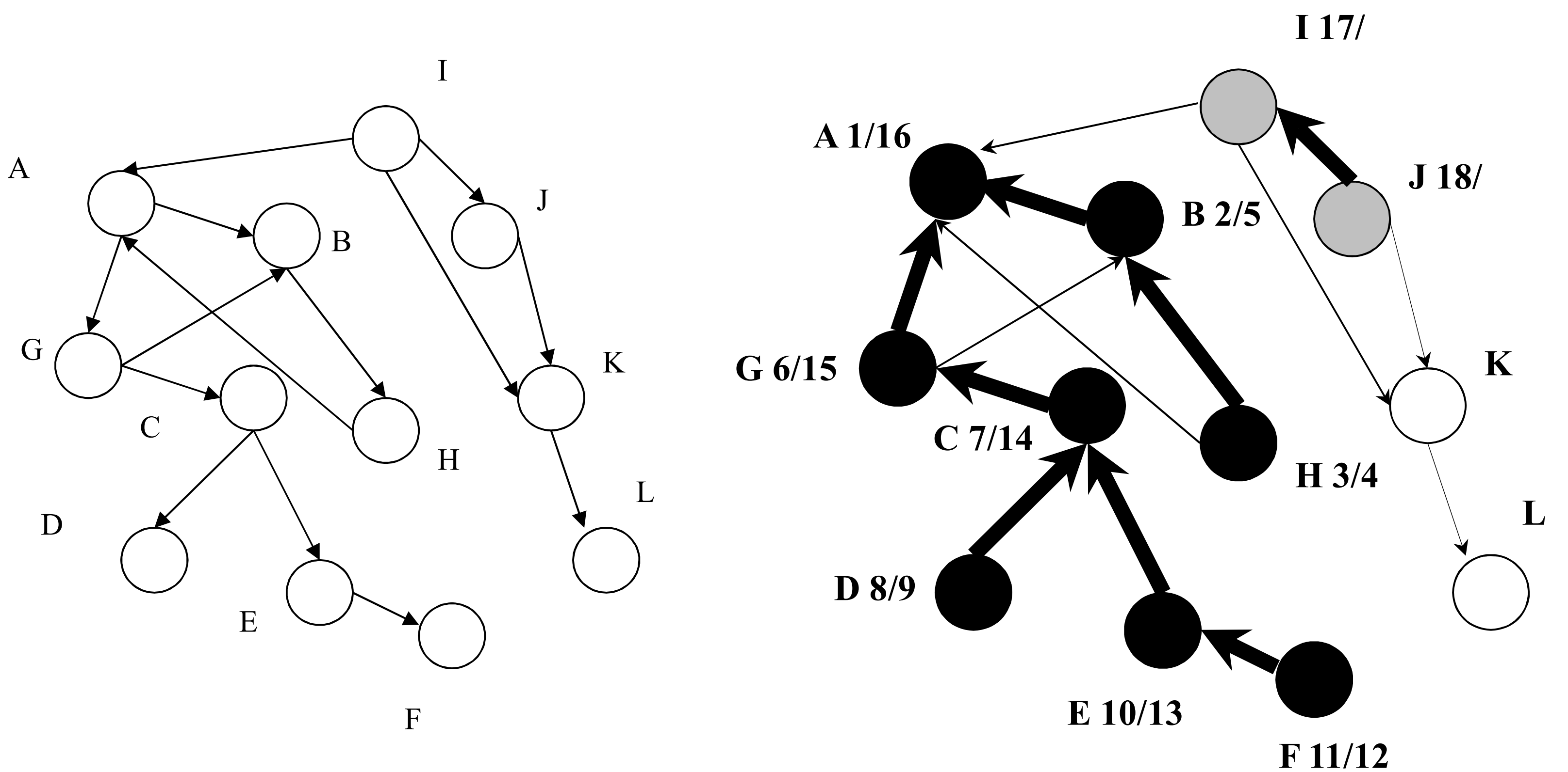
- DFS(G)
  - V = noduri(G)
  - **Pentru fiecare** nod u (u ∈ V)
    - c(u) = alb; p(u) = null; // inițializare structură date
  - timp = 0; // reține distanța de la rădăcina arborelui
    // DFS pană la nodul curent
  - **Pentru fiecare** nod u (u ∈ V)
    - **Dacă** c(u) este alb
      - **Atunci** explorare(u); // explorez nodul
- explorare(u)
  - d(u) = ++ timp; // timpul de descoperire al nodului u
  - c(u) = gri; // nod in curs de explorare
  - **Pentru fiecare** nod v (v ∈ succs(u)) // încerc sa
    // prelucrez vecinii
    - **Dacă** c(v) este alb
      - **Atunci** {p(v) = u; explorare(v);} // dacă nu au
        // fost prelucrați deja
  - c(u) = negru; // am terminat de explorat nodul u
  - f(u) = ++ timp; // timpul de finalizare al nodului u

DFS – Evoluția explorării

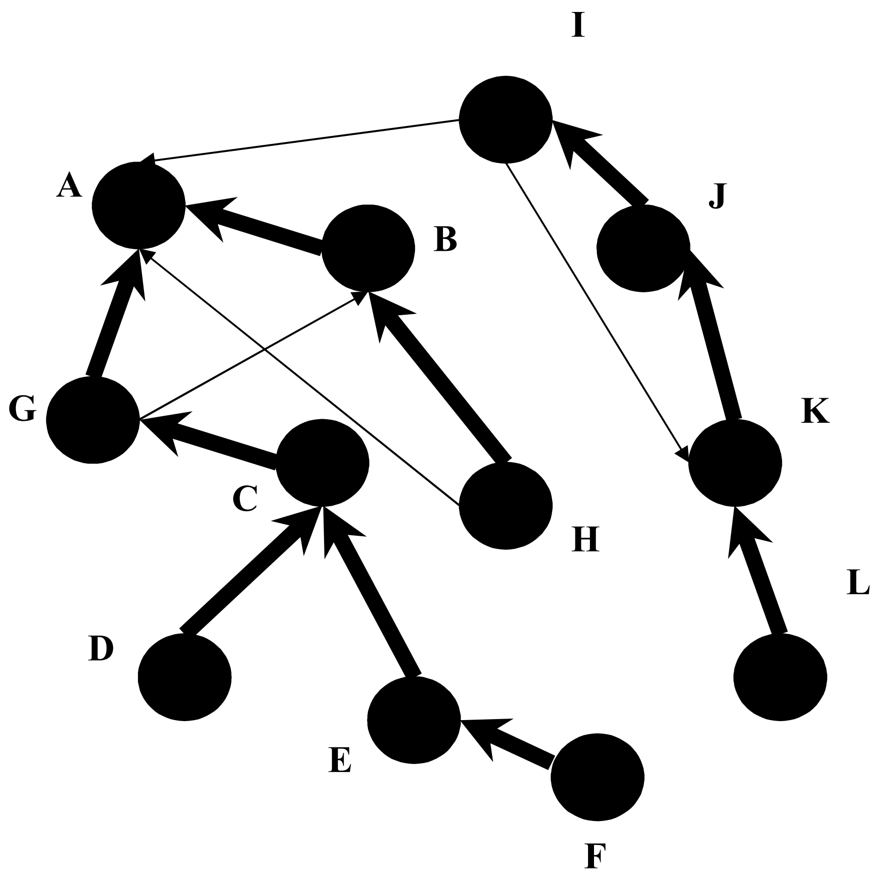




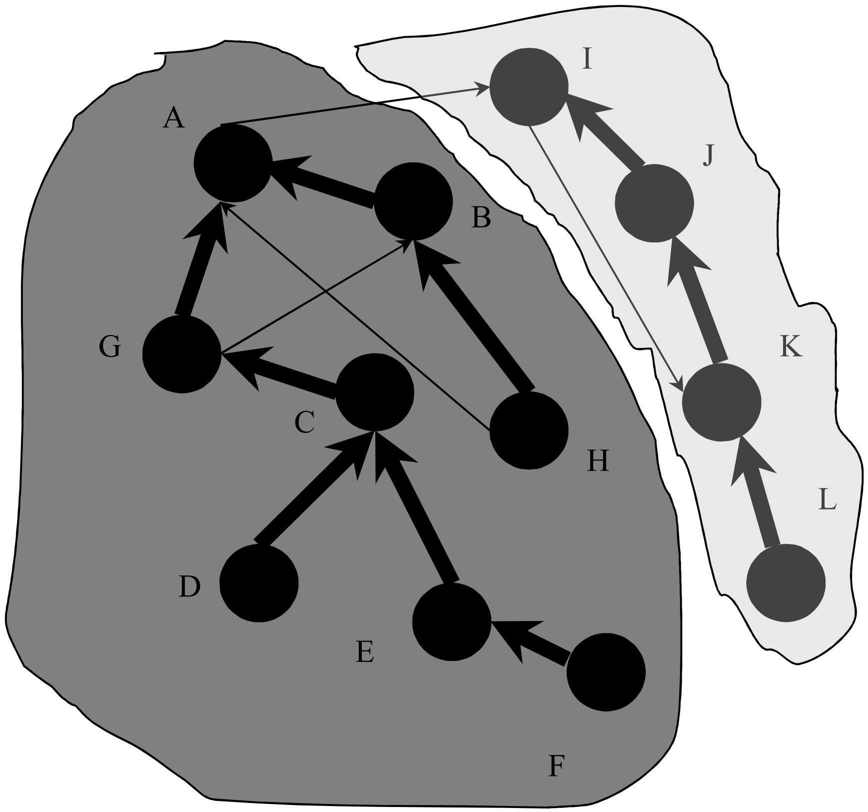
Calcului timpilor



Arborele de parcurgere în adâncime



DFS – Zone de explorare



DFS – Proprietăți (I)

- I(u) = intervalul de prelucrare al nodului (d(u),f(u)).
- Lema 5.5. G = (V,E); u $\in$ V; pentru fiecare v descoperit de DFS pornind din u este construită o cale v, p(v), p(p(v)),…, u.
  - Fie calea u = $v_0 v_1 \ldots v_n$ = v. Dem prin inducție ca $\pi(v_i) = v_{i-1}$!
- Teorema 5.2. G = (V,E); DFS(G) sparge graful G într-o pădure de arbori Arb(G) = { Arb(u); p(u) = null } unde Arb(u) = (V(u),E(u));
  - V(u) = { v | d(u) < d(v) < f(u) } + {u};
  - E(u) = { (v, z) | v, z $\in$ V(u) && p(z) = v }.
  - Dem: Conform algoritmului, se pot identifica noduri în ciclul principal sau din funcția de explorare. Dacă u e descoperit în ciclul principal, atunci $\exists$ o cale către toți succesorii dată de părinți → V(u) = { v | d(u) < d(v) < f(u) } iar arcele sunt chiar cele ce desemnează părinții

DFS – Proprietăți (II)

- Teorema 5.3. Dacă DFS(G) generează 1 singur arbore => G este conex. (Reciproca este adevărată?)

- Teorema 5.4. Teorema parantezelor:
  - $\forall$ u, v avem $I(u) \cap I(v) = \varnothing$ sau $I(u) \subset I(v)$ sau $I(v) \subset I(u)$.
  - Dem prin considerarea tuturor combinațiilor posibile!
  - a) $I(u) < I(v)$: $d(u) < f(u) < d(v) < f(v)$: $v \notin R(u)$ → v rămâne alb pe durata prelucrării lui u → $f(u) < d(v)$
  - b) $I(v) \subset I(u)$: $d(u) < d(v) < f(v) < f(u)$: $v \in R(u)$ → v este descoperit din u și devine negru înaintea terminării prelucrării lui u → $f(v) < f(u)$
  - c) $I(v) < I(u)$: $d(v) < f(v) < d(u) < f(u)$ Analog a)
  - d) $I(u) \subset I(v)$: $d(v) < f(u) < f(u) < f(v)$ Analog b)

DFS – Proprietăți (III)

- Teorema 5.5. $\forall$ u, v $\in$ V, atunci v $\in$ V(u) <=> I(v) $\subset$ I(u).

- Teorema 5.6. Teorema drumurilor albe:
  - G = (V,E); Arb(u); v este descendent al lui u in Arb(u) <=> la momentul d(u) există o cale numai cu noduri albe u..v.
  - Demonstrație prin inducție!
  - v $\in$ V(u) → la momentul d(u) există o cale numai cu noduri albe u..v
    - Dacă v $\in$ V(u) → $\exists$ o cale unică v .. α .. u de pointeri π. Fie un nod oarecare z din calea α. → (Teorema 5.5) d(u) < d(z) < f(z) < f(u) → la d(u), c(z) = alb și cum z a fost ales la întâmplare → toate nodurile de pe calea α sunt albe la d(u).
  - la momentul d(u) există o cale numai cu noduri albe u..v → v $\in$ V(u)
    - Fie u = $v_0 v_1 \ldots v_p$ = v o cale din G, a.î. La d(u) avem c(v_i) = alb, i = 0,p. Dem prin inducție după i că v_i este descendent al lui u în Arb(u).
    - Caz de bază: d(u) < d(v_1) < f(u) → (Teorema 5.5) v_1 descendent al lui u Adevărat
    - Pas inducție: v_i descendent al lui u → v_{i+1} descendent al lui u
      - v_i descendent al lui u → d(u) < d(v_i) < f(v_i) < f(u) . Cum v_{i+1} este alb la d(u) și este succesorul lui v_i → v_{i+1} este descoperit după d(u), dar înainte de f(v_i). → d(u) < d(v_{i+1}) < f(v_i) < f(u) → v_{i+1} descendent al lui u

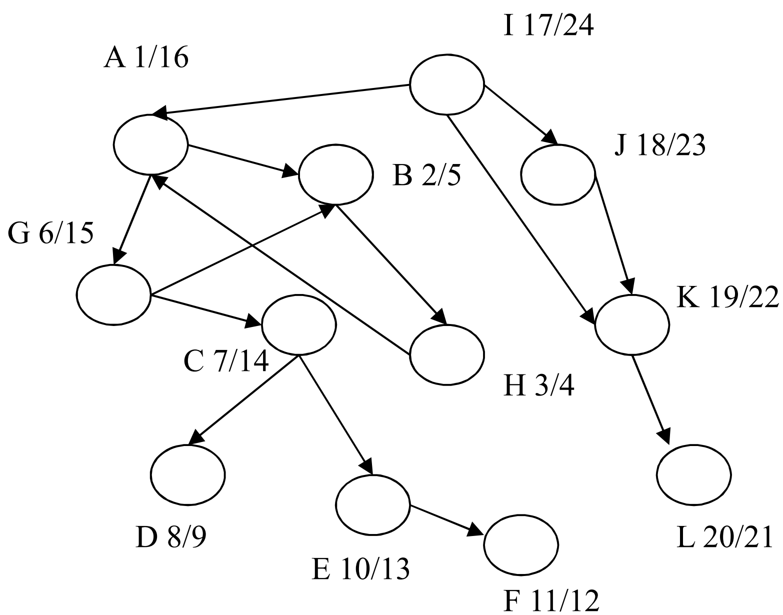
Clasificări ale arcelor grafului (I)

- Arc direct (de arbore)
  - Ce fel de noduri?
- Arc invers (de ciclu)
  - Ce fel de noduri?
- Arc înainte
  - Ce fel de noduri?
- Arc transversal
  - Ce fel de noduri?

Clasificări ale arcelor grafului (II)

- Arc direct (de arbore)
  - între nod gri și nod alb;
- Arc invers (de ciclu)
  - între nod gri și nod gri;
- Arc înainte
  - nod gri și nod negru și $d(u) < d(v)$;
- Arc transversal
  - nod gri și nod negru și $d(u) > d(v)$.

Clasificări ale arcelor grafului (III)



Arc direct (de arbore)

între nod gri și nod alb;

Arc invers (de ciclu)

între nod gri și nod gri;

Arc înainte

nod gri și nod negru și d(u) < d(v);

Arc transversal

nod gri și nod negru și d(u) > d(v).

Arc direct (de arbore):

AB, BH, AG, GC, CD, CE, EF, IJ, JK, KL

Arc invers (de ciclu):

HA

Arc înainte:

IK

Arc transversal:

GB, IA

DFS – Proprietăți (IV)

- Teorema 5.7. Într-un graf neorientat, DFS poate descoperi doar muchii directe și inverse.
  - Dem prin considerarea cazurilor posibile!
  - Fie muchia (u,v) ∈ E și pp. d(u) < d(v). Muchia poate fi străbătută din u sau din v:
    - Caz (u,v): c(u) = gri, c(v) = alb → muchie directă
    - Caz (v,u): la momentul d(u) ∃ o cale cu noduri albe u..v → (Teorema drumurilor albe) v este descendent al lui u în Arb(u) → (Teorema 5.5) d(u) < d(v) < f(v) < f(u) → în intervalul (d(v), f(v)) când se investighează (v,u) c(u) = c(v) = gri → muchie inversă

DFS – Proprietăți (V)

- Teorema 5.8. G = graf orientat; G ciclic <=> în timpul execuției DFS găsim arce inverse.
  - Dem prin exploatarea proprietăților de ciclu și de arc invers!
  - G ciclic → DFS descoperă arce inverse
    - Fie un ciclu și u primul nod din ciclu descoperit de DFS. Atunci ∃ u..v si (v,u) arcul care închide ciclul. d(u) < d(v) (ipoteză) iar u..v conține doar noduri albe → (Teorema drumurilor albe & Teorema 5.5) d(u) < d(v) < f(v) < f(u) → în intervalul (d(v), f(v)) când se explorează (v,u) c(u) = c(v) = gri → arc invers
  - DFS descoperă arce inverse → G ciclic
    - Fie (v,u) arc invers → d(u) < d(v) < f(v) < f(u) → (Teorema 5.5) → v este descendent al lui u în Arb(u) → ∃ calea u..v care inchide ciclul

DFS – Complexitate și Optimalitate

Complexitate:

$O(n+m)$

n = număr noduri

m = număr muchii

Optimalitate: NU

Parcurge tot graful? DA

ÎNTREBĂRI?