# Proiectarea Algoritmilor

Curs 6 – Componente Tare Conexe


## Bibliografie

- Giumale – Introducere in Analiza Algoritmilor cap. 5.2
- Cormen – Introducere în Algoritmi cap. 23.5
- http://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm

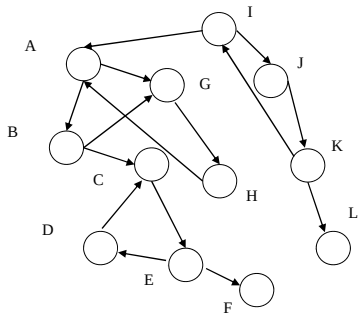
## Componente Tare Conexe (CTC)

- Definiție: Fie G = (V,E) un graf orientat. G este tare-conex ⇔ ∀ u,v∈V ∃ o cale u..v si o cale v..u (u∈R(v) ∧ v∈R(u)).
- Definiție: G = (V,E) graf orientat. G' = (V',E'), V'⊆V, E'⊆E. G' este o CTC a lui G ⇔ G' e tare-conex (∀ u,v∈V', u∈R(v) ∧ v∈R(u)) si G' este maximal.
- Lema 5.6: G = (V,E) graf orientat, G' CTC, ∀ u,v∈V' => ∀ u..v din G are noduri exclusiv in V'
  - Dem: ∀ z a.i. u..z..v => z∈R(u) si v∈R(z). Dar u∈R(v) => z∈R(v) => v si z sunt in aceeași CTC.

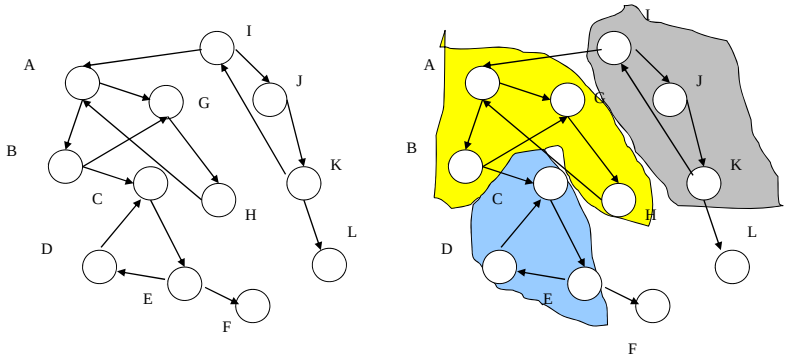




## Exemplu (I) – determinare CTC



## Exemplu (II) – determinare CTC(2)



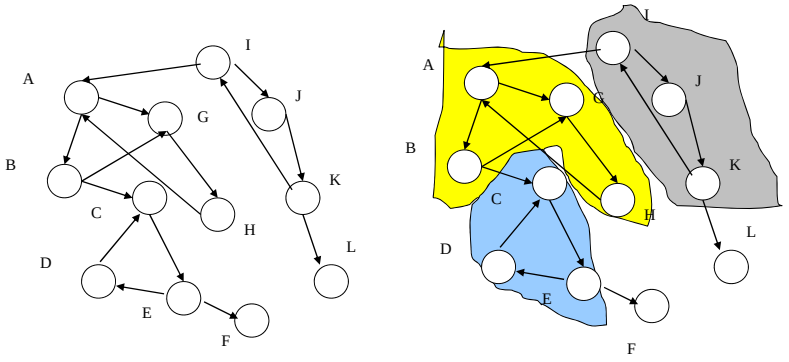



## Componente Tare Conexe (CTC)

- Teorema 5.10: G = (V,E) orientat, G' = (V',E') o CTC a lui G. Toate nodurile $v \in V'$ sunt grupate in același Arb(u) construit de DFS(G), unde u este primul nod descoperit al componentei.
  - Dem: $\forall\ v \in V'$, $v \neq u$, $\exists$ u..v drum cu noduri albe la momentul descoperirii d(u); toate nodurile drumului sunt in V' (conf. Lema 5.6) => (din Teorema drumurilor albe) v este descendent al lui u in Arb(u)

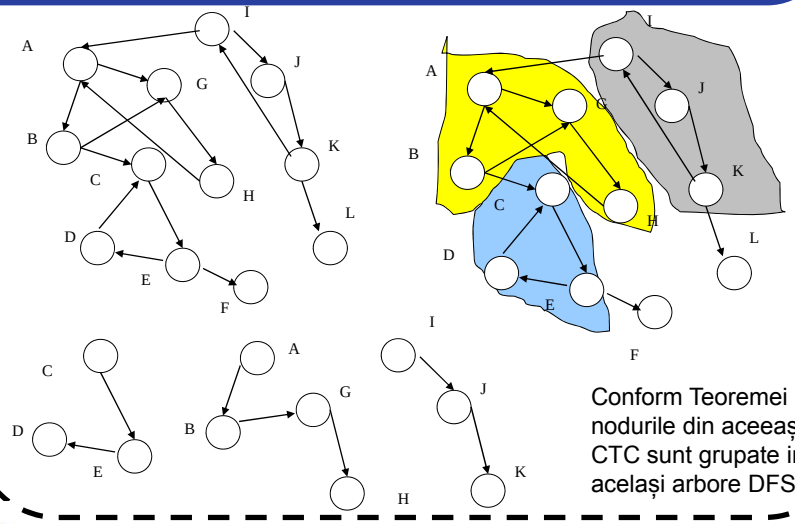
# Exemplu (III) – DFS

- Aplicare DFS pornind din primul nod al fiecarei CTC






# Exemplu (IV) - DFS(2)



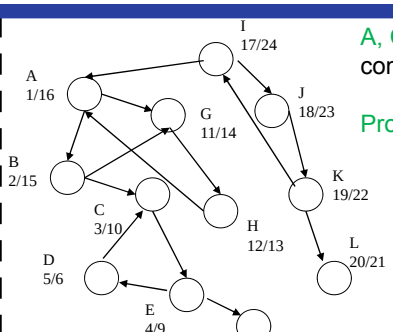

Conform Teoremei nodurile din aceeași CTC sunt grupate in același arbore DFS!

# Componente Tare Conexe (CTC)

- Definiție: G = (V,E) orientat, u∈V. Φ(u) = strămoș DFS al lui u determinat in cursul DFS(G) dacă:
  - Φ(u)∈R(u)
  - f(Φ(u)) = max{f(v) | v∈R(u)}
- Ce e Φ(u)? Φ(u) este primul nod din CTC descoperit de DFS(G)
- Teorema 5.11: Φ(u) satisface următoarele proprietăți:
  1. f(u) ≤ f(Φ(u)) când e egalitate? u este primul nod din CTC
  2. ∀v∈R(u), f(Φ(v)) ≤ f(Φ(u)) ce înseamnă ca e egalitate? u si v sunt in aceeași CTC
  3. Φ(Φ(u)) = Φ(u) Dem: Φ(Φ(u))∈R(Φ(u)) $\overset{2}{\rightarrow}$ f(Φ(Φ(u))) ≤ f(Φ(u)) si $\overset{1}{\rightarrow}$ f(Φ(Φ(u))) ≥ f(Φ(u)) → f(Φ(Φ(u))) = f(Φ(u)) → Φ(Φ(u)) = Φ(u)



# Exemplu (V) – stramosi




A, C, I sunt strămoși DFS ai nodurilor din componenta conexă din care fac parte.

Proprietățile strămoșilor:

- f(u) ≤ f(Φ(u))

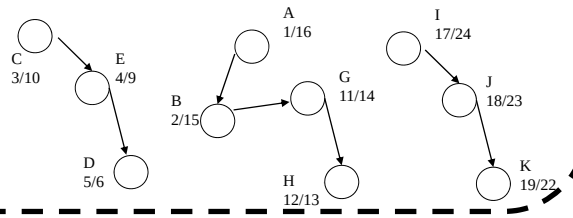
Ex: f(B) ≤ f(Φ(B)) = f(A)

- ∀ v∈R(u), f(Φ(v))≤f(Φ(u))

Ex: E∈R(B), f(C) = f(Φ(E)) ≤ f(Φ(B)) = f(A)

- Φ(Φ(u)) = Φ(u)

Ex: Φ (Φ(E)) = Φ(C) = C = Φ(E)

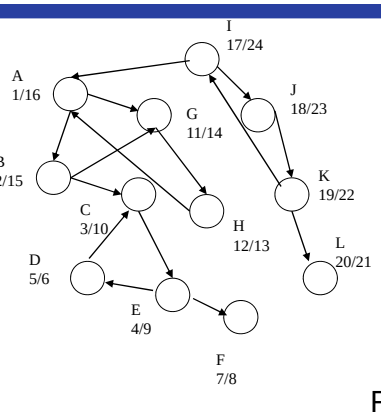


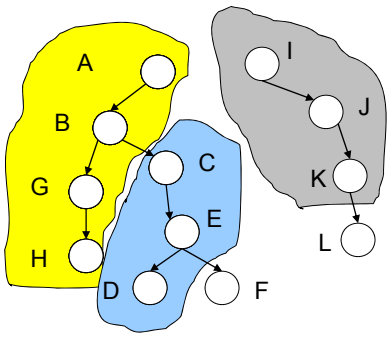

# Componente Tare Conexe (CTC)

- Din Definiție: $\Phi(u) \in R(u)$ && $f(\Phi(u)) = \max\{f(v) \mid v \in R(u)\}$.
- Teorema 5.12. G = (V,E) orientat, $\forall$ u∈V, u este descendent al lui $\Phi(u)$ in Arb($\Phi(u)$) construit de DFS.
  - Dem: prin considerarea tuturor culorilor posibile ale lui $\Phi(u)$ la momentul d(u).
- Teorema 5.13. G = (V,E) orientat, $\forall$ u,v∈V; u si v aparțin aceleiași CTC $\Leftrightarrow$ $\Phi(u) = \Phi(v)$.
  - Dem folosind proprietățile strămoșilor :
    - $\forall$ u,v∈ aceleiași CTC => $\Phi(u) = \Phi(v)$: $v \in R(u)$, $f(\Phi(v)) \leq f(\Phi(u))$ si $u \in R(v)$, $f(\Phi(u)) \leq f(\Phi(v))$ => $f(\Phi(u)) = f(\Phi(v))$ → $\Phi(u) = \Phi(v)$
    - $\Phi(u) = \Phi(v)$ => $\Phi(u) \in R(u)$ => u si $\Phi(u)$ ∈ aceleiași CTC si $\Phi(u) \in R(v)$ => v si $\Phi(u)$ ∈ aceleiași CTC
      => u si v ∈ aceleiași CTC



# Exemplu (VI)



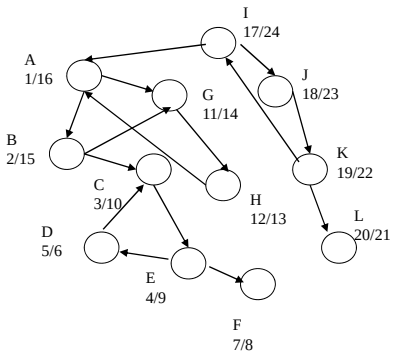

Primul nod dintr-o CTC descoperit la DFS va avea copii in arborele generat de DFS toate elementele componentei conexe!

## Componente Tare Conexe (CTC)

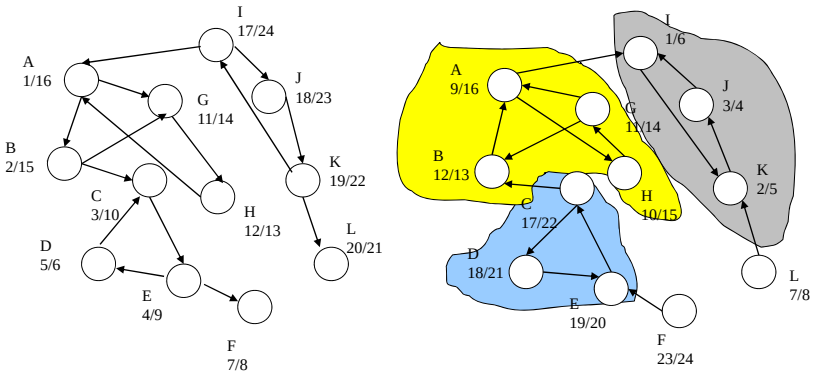
- Probleme:
  - trebuie să eliminăm nodurile care nu sunt in componenta conexă.
  - vrem ca fiecare arbore construit să conțină o CTC.
- Idee – eliminăm nodurile ce nu aparțin CTC! Cum?
  - Dacă aparțin Arb(u) si nu CTC => ∃ u..v si ∄ v..u.
  - → DFS pe graful transpus!



## Exemplu (VII) – DFS ($G^T$)



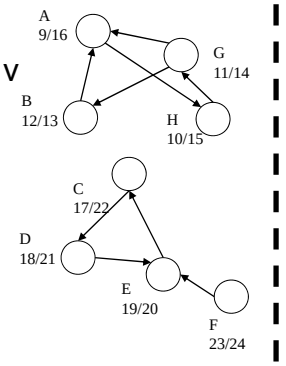
# Exemplu (VIII) – DFS ($G^T$) (2)





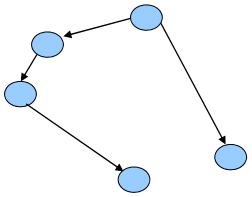

# Componente Tare Conexe (CTC)

- Cazuri in DFS($G^T$):
  - v este in CTC descoperita din u → v poate fi descoperit din u si in DFS($G^T$).
  - v∉CTC dar v∈Arb(u) in DFS(G)→ nu va fi atins in DFS($G^T$) din u.
  - v∉CTC dar ∃ v..u in G → f(v) > f(u) → v va fi deja colorat in negru când se explorează u.



# Observatii

- Înlocuind componentele tare conexe cu noduri obținem un graf aciclic. De ce?
- Pentru că altfel am avea o singură CTC!



- Prima parcurgere DFS este o sortare topologică. De ce?
- Pentru că sortează nodurile in ordinea inversă a timpilor!



# Pseudocod algoritm CTC

- Algoritmul lui Kosaraju:
- CTC(G)
  - DFS(G)
  - $G^T$ = transpune(G)
  - DFS($G^T$) (in bucla principală se tratează nodurile in ordinea descrescătoare a timpilor de finalizare de la primul DFS)
- Componentele conexe sunt reprezentate de pădurea de arbori generați de DFS($G^T$)

Complexitate?

## Algoritmul lui Kosaraju

**Complexitate**

**O(n+m)**

n = numar noduri

m = numar muchii



## Corectitudine algoritm CTC (1)

- Teoremă: Algoritmul CTC calculează corect componentele tare conexe ale unui graf G = (V,E).
- Dem. prin inducție după nr. de arbori de adâncime găsiți de PAD a $G^T$ că vârfurile din fiecare arbore formează o CTC:
  - Fiecare pas demonstrează ca arborele format in acel pas e o CTC, presupunând că toți arborii produși deja sunt CTC.
  - $P_1$: trivial pt. că ∄ arbori anteriori.
  - $P_n \rightarrow P_{n+1}$: Fie arborele T obținut in pasul curent având rădăcina r. Notăm $C_r = \{v \in V \mid \Phi(v) = r\}$.

## Corectitudine algoritm CTC (2)

- Demonstrăm că $u \in T \Leftrightarrow u \in C_r$:
- $\Longrightarrow u \in C_r \rightarrow \exists$ r..u → toate nodurile din $C_r$ ajung in același arbore de PAD. Dar $r \in C_r$ si r e rădăcina lui T → $\forall u \in C_r$ => $u \in T$
- $\Longleftarrow$ demonstrăm că $\forall$ w a.i. $f(\Phi(w)) > f(r)$ sau $f(\Phi(w)) < f(r)$, $w \notin T$
  - Dacă $f(\Phi(w)) > f(r)$ → la d(r), w e deja pus in CTC cu rădăcina $\Phi(w)$ pt. că nodurile sunt considerate in ordinea inversa a timpilor de finalizare → $w \notin T$
  - Dacă $f(\Phi(w)) < f(r)$ → $w \notin T$ pt. că altfel ($w \in T$) → $\exists$ r..w in $G^T$ → $\exists$ w..r in G → $r \in R(w)$, → $f(\Phi(w)) \geq f(\Phi(r)) = f(r)$ (F)
  - → T conține doar nodurile pt. care $\Phi(w) = r$ → $T = C_r$



## Algoritmul lui Tarjan (I)

- Bazat tot pe DFS
- Folosește o singura parcurgere in adâncime.
- Determină din parcurgere care sunt "rădăcinile" CTC.

# Algoritmul lui Tarjan (II)

- index = 0 // nivelul pe care este nodul in arborele DFS
- S = empty // se folosește o stiva care se inițializează cu Ø
- forall v in V do
  - if (v.index is undefined) // se pornește DFS din fiecare nod pe care nu l-am vizitat încă
    - tarjan(v)

- procedure tarjan(v)
  - v.index = index // se setează nivelul nodului v
  - v.lowlink = index // retine strămoșul nodului v
  - index = index + 1 // incrementez nivelul
  - S.push(v) // introduc v in stiva
  - forall (v, v') in E do // se prelucrează succesorii lui v
    - if (v'.index is undefined or v' is in S) // CTC deja identificate sunt ignorate
      - if (v'.index is undefined) tarjan(v') // daca nu a fost vizitat v' intru in recursivitate
      - v.lowlink = min(v.lowlink, v'.lowlink) //actualizez strămoșul
  - if (v.lowlink == v.index) // printez CTC începând de la rădăcina
    - print "CTC:"
    - repeat
      - v' = S.pop // extrag nodul din stiva si il printez
      - print v'
    - until (v' == v) // până când extrag rădăcina




# Algoritmul lui Tarjan (III)

Complexitate

O(n+m)

n = numar noduri

m = numar muchii

# Întrebări?