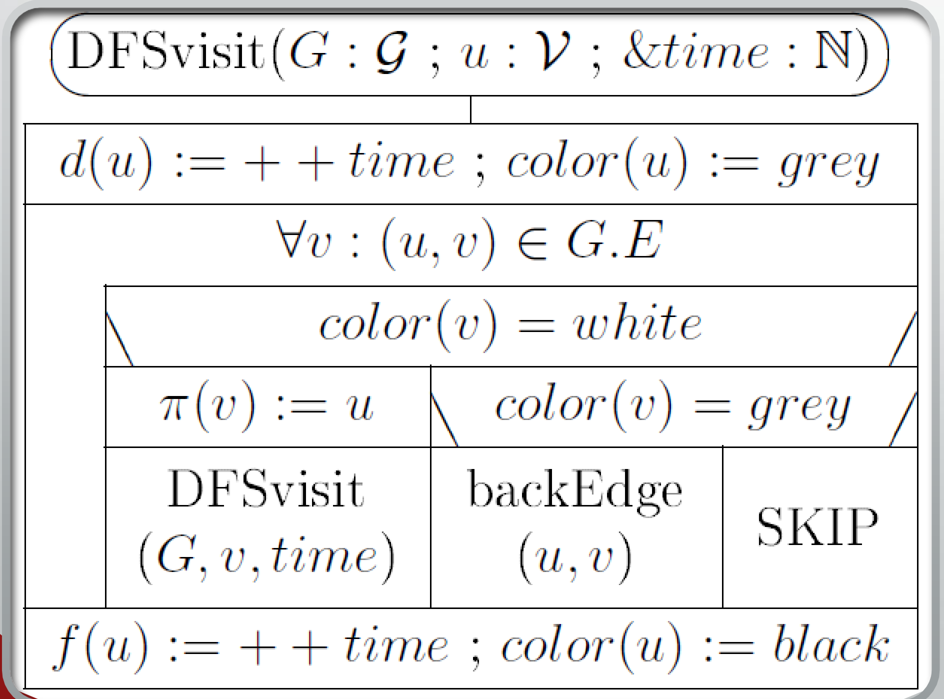
Algoritmusok és adatszerkezetek I.
5. Előadás

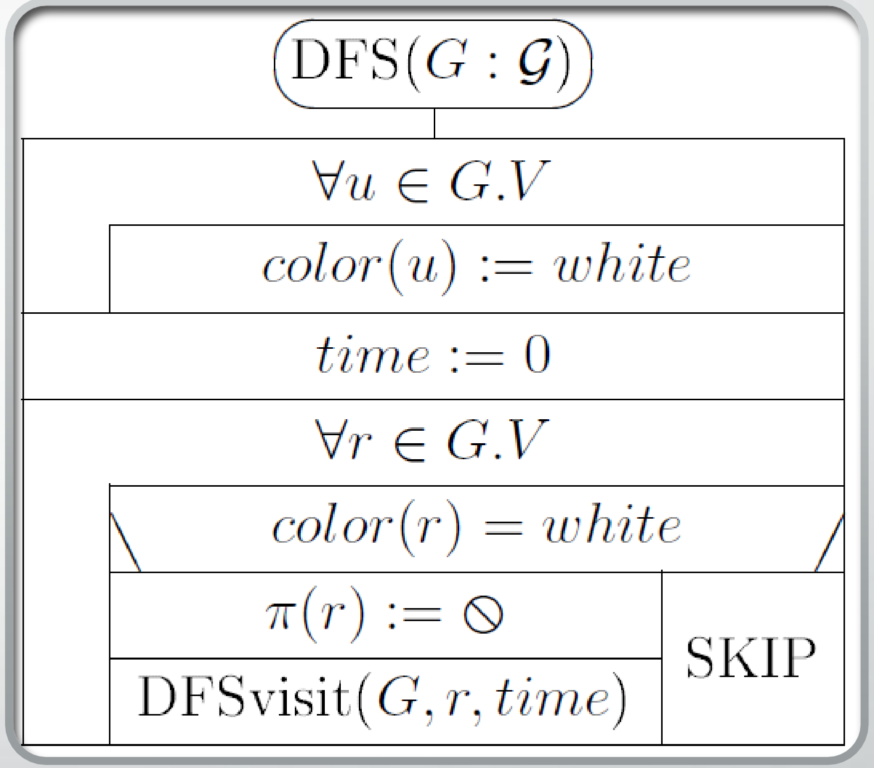
Mélységi gráfkeresés, élek osztályozása, DAG, topológikus rendezés, irányított kör keresése.

Tartalom

Mélységi gráfkeresés (DFS. Depth-first Search)

- Csak egyszerű irányított gráfokra értelmezzük
- Csúcsok színei:
  - Fehér csúcs: érintetlen
  - Szürke csúcs: belőle elérhető csúcsokat járunk be éppen
  - Fekete csúcs: befejeztük





- d(u): elérési idő (discovery time)
- f(u): befejezési idő (finishing time)

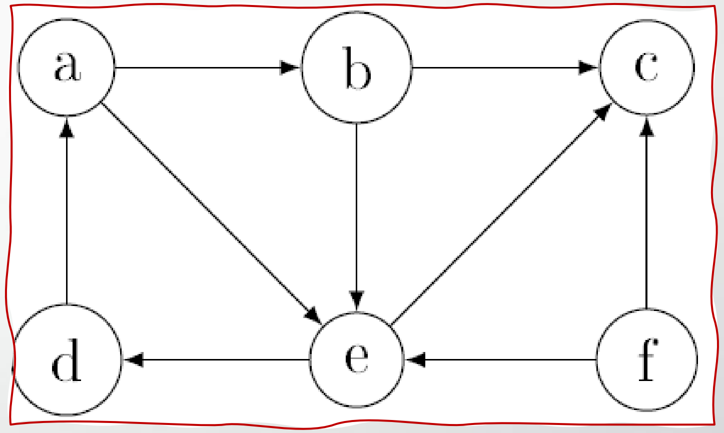
Mélységi feszítő erdő (Depth-first forest)

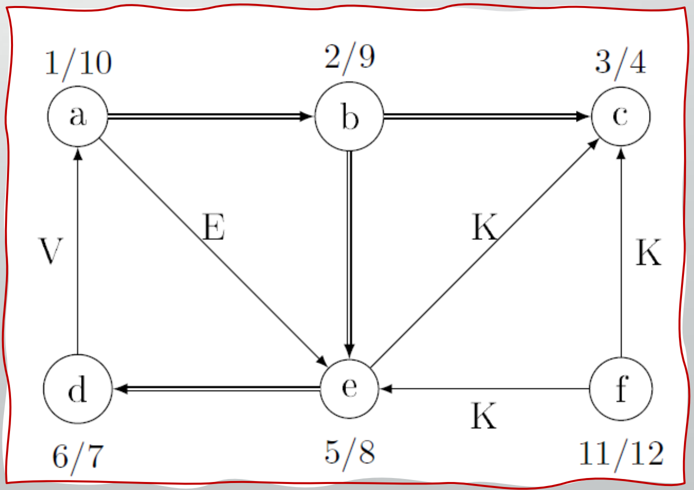
- Mindegyik, a DFS eljárásból indított DFSvisit egy-egy *mélységi fát* számol ki.
- Ezek együtt adják a *mélységi feszítő erdőt*.
- $r \in G.V$ egy mélységi fa gyökere $\Leftrightarrow \pi(r) = \emptyset$
- $(u, v) \in G.E$ egy mélységi fa éle $\Leftrightarrow u = \pi(v)$

A mélységi bejárás szemléltetése

a → b ; e.
b → c ; e.
d → a.
e → c ; d.
f → c ; e.

- A bejárás indeterminisztikus abban, hogy az egyes ciklusokban a csúcsokat milyen sorrendben veszi sorra
  - Szemléltetésben: csúcsokat ábécé rendben, illetve indexek szerint monoton növekvően dolgozzuk fel
- *d*/*f* alakú címkék:
  a csúcs *elérési/befejezési* idő
  (discovery/finishing time)
- Élek osztályozása:
  - fa-élek: dupla szárú nyíl
  - vissza- él: V
  - előre-él: E
  - kereszt-él: K





Az élek osztályozása (Classification of edges)

1. Tétel. A gráf éleinek felismerése:

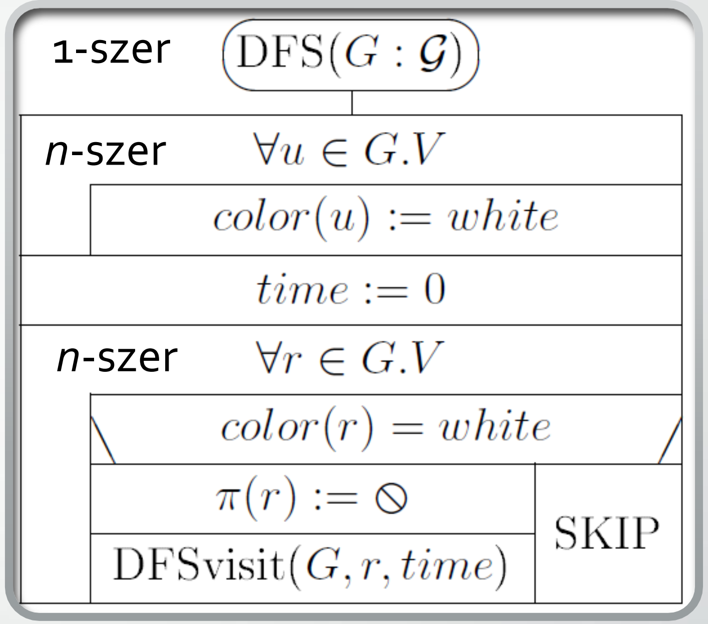
- A mélységi bejárás során tetszőleges (u, v) él feldolgozásakor az él a következő kritériumok alapján osztályozható:
  - (u, v) fa-él (tree edge) $\Leftrightarrow$ a v csúcs még fehér.
  - (u, v) vissza-él (back edge) $\Leftrightarrow$ a v csúcs éppen szürke.
  - (u, v) előre-él (forward edge) $\Leftrightarrow$ a v csúcs már fekete $\wedge$ $d(u) < d(v)$.
  - (u, v) kereszt-él (cross edge) $\Leftrightarrow$ a v csúcs már fekete $\wedge$ $d(u) > d(v)$.

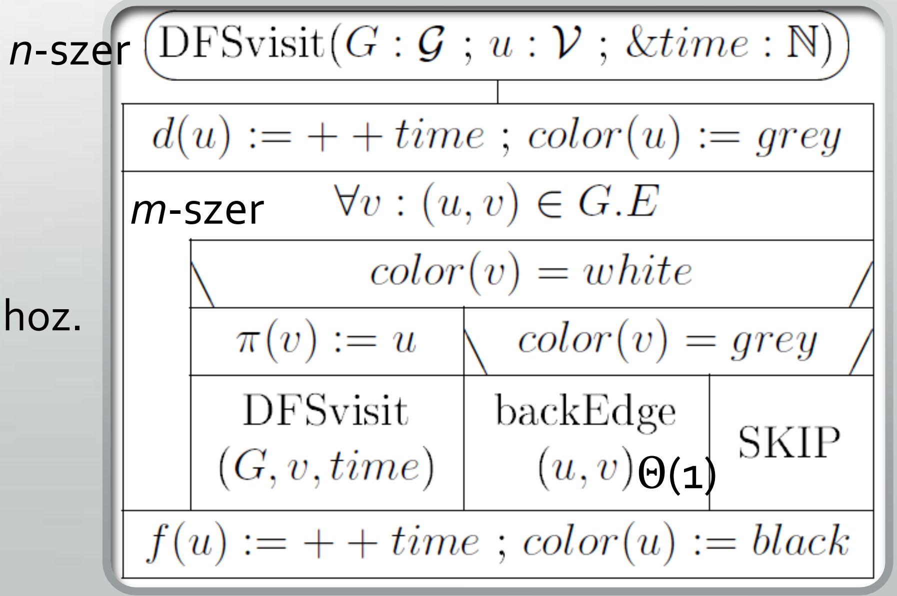
A mélységi gráfkeresés futási ideje

- $n = |G.V|$ és $m = |G.E|$
- DFS mindkét ciklusa *n*-szer fut le.
- DFSvisit rekurzív eljárás:
  - Mindegyik csúcsra, amikor először érjük el (fehér)egyszer
  - **összesen *n*-szer** hívódik meg
- A DFSvisit ciklusa
  - mindegyik csúcsra annyit iterál, amennyi a kimeneti fokszáma.
  - Ez a ciklus tehát a gráf éleit dolgozza fel, mindegyiket egyszer.
  - Összesen ***m*** iterációt végez.
- A DFS csak egyszer hívódik meg.
- Tfh! a backEdge eljárás: csak megjelöl egy visszaélet
  - ➢ $\Theta(1)$ műveletigényű
  - ➢ és műveletigénye hozzávehető az őt meghívó ciklusiterációéhoz.

➢ Pontosan: $(1+2n)+(n+m) = 3n + m + 1$ lépés

➢ $MT(n); mT(n) \in \Theta(n + m)$





A DAG tulajdonság eldöntése

3. Definíció. A G irányított gráf akkor DAG (Directed Acyclic Graph = körmentes irányított gráf), ha nem tartalmaz irányított kört.

- A DAG-ok a gráfok fontos osztályát képezik: sok hasznos algorimus DAGot vár a bemenetén
  - az input ellenőrzése is szükséges lehet
- Ha a mélységi bejárás egy (u, v) vissza-élet talál -> irányított kört is talált a gráfban
  - ekkor a vissza-él a definíciója szerint egy mélységi fában az u csúcs egyik őse a v csúcs
  - és a v-ből u-ba vezető, fa-élekből álló út az (u, v) éllel együtt irányított kört alkot

A DAG tulajdonság eldöntése

- Bebizonyítható, hogy ha a *G* irányított gráf nem DAG (irányított kört tartalmaz) ->a DFS fog visszaélet, és ezzel irányított kört találni
  - Az viszont nem garantált, hogy az összes irányított kört megtalálja

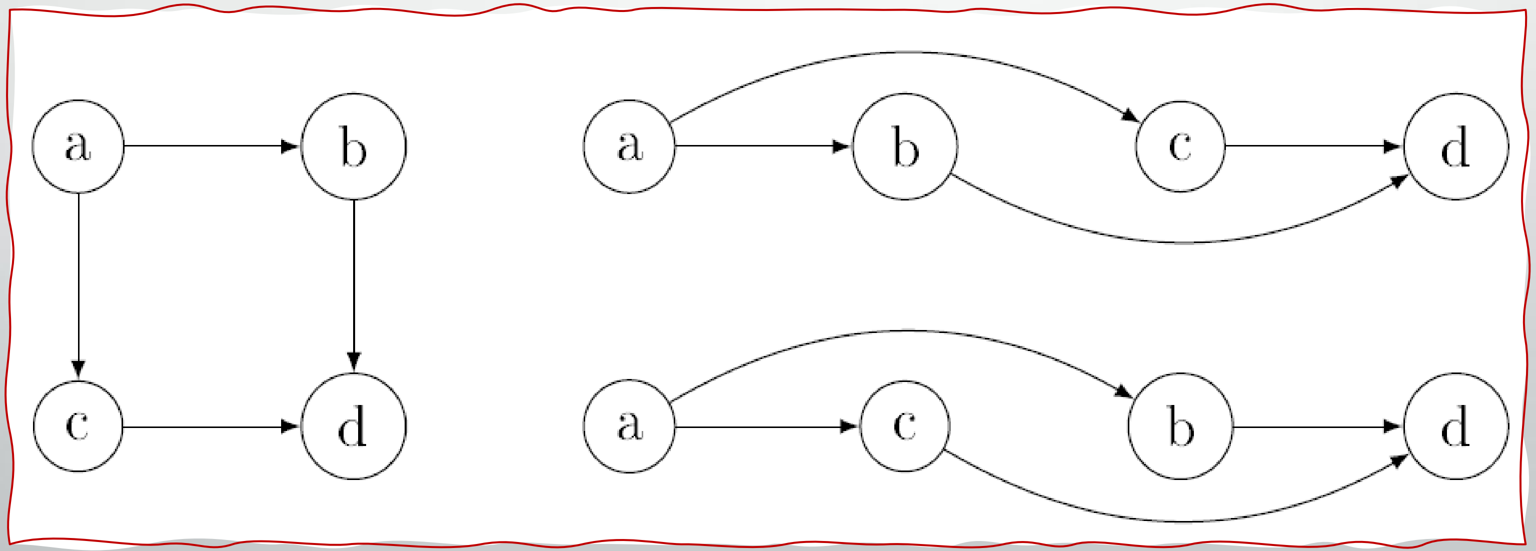
4. Tétel. A *G* irányított gráf DAG $\Leftrightarrow$ a mélységi bejárás nem talál G-ben vissza-élet.
Ha a DFS talál egy (u, v) vissza-élet, akkor az $\langle u, \pi(u), \pi(\pi(u)), \ldots v; u \rangle$ csúcssorozat visszafelé olvasva egyszerű irányított kört ad.

- A tétel alapján a backEdge eljárás akár ki is nyomtathatja a megtalált irányított kört.
  - Ebben az esetben a maximális futási ideje nyilván $\Theta(n)$ lesz

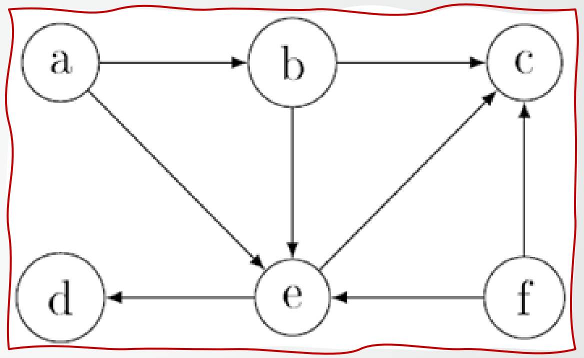
Topologikus rendezés

5. Definíció. Irányított gráf topologikus rendezése alatt a gráf csúcsainak olyan sorba rendezését értjük, amelyben minden él egy-egy később jövő csúcsba (szemléletesen: balról jobbra) mutat.

- Ugyanannak a gráfnak lehet egynél több topologikus rendezése



Tetszőleges DAG-ra a topologikus rendezés algoritmusa



- A DAG topologikus rendezésésében a csúcsok a befejezési idők szerint szigorúan monoton csökkenően jelennek meg.
- Ha az algoritmus indeterminizmusát más konvenció mentén oldanánk fel, gyakran az előbbitől különböző topologikus rendezést kapnánk!
- A topologikus rendezés egy kézenfekvő alkalmazása: az ún. egygépes ütemezési probléma megoldása:
    - A csúcsok: (munka)folyamatok
    - Az élek: a köztük lévő rákövetkezési kényszerek
    - A folyamatokat ezeknek megfelelően kell sorba rakni.

Topologikus rendezés

5. Tétel. Tetszőleges irányított gráfnak pontosan akkor van topologikus rendezése, ha nincs irányított kör a gráfban, azaz a gráf DAG.

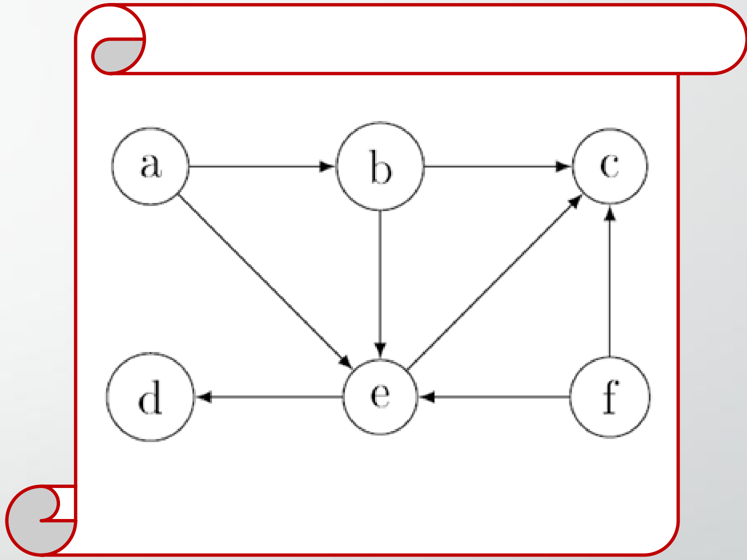
- **Bizonyítás.**
  - ⇒
    - Ha van irányított kör a gráfban, jelölje <u_1, u_2,…, u_k, u_1 >!
    - Ekkor egy tetszőleges topologikus rendezésben u_1 után jön valahol u_2,az után valahol u_3, és így tovább, végül is u_1 után jön u_k,
    - és u_k után u_1 => ellentmondás
    - Ekkor nincs topologikus rendezés (a gráf csúcsain).

Topologikus rendezés

- **Bizonyítás.**
  - $\Leftarrow$
    - Ha nincs irányított kör a gráfban → van olyan csúcs, aminek nincs megelőzője
    - Ha veszünk egy megelőzővel nem rendelkező csúcsot, és töröljük a gráfból →a maradék gráfban nem keletkezik irányított kör
    - ➢ lesz megint legalább egy olyan, amelyiknek nincs megelőzője
    - Sorban a törölt csúcsok adják a topologikus rendezést

Tetszőleges DAG-ra a topologikus rendezés algoritmusának lépései

1. Létrehozunk egy üres vermet
2. Végrehajtuk gráf mélységi bejárását
   - valahányszor befejezünk egy csúcsot, a verem tetejére tesszük
3. A verem tartalmát kiolvasva megkapjuk a gráf csúcsainak topologikus rendezését



Tetszőleges DAG-ra a topologikus rendezés algoritmusa

- Az algoritmus képes ellenőrizni a saját előfeltételét
  - Ha a DFS vissza-élt talál -> a gráf irányított kört tartalmaz ->az algoritmus eredménye: nincs topologikus rendezés
  - (Ilyenkor a verem tartalma nem használható fel.)
- *Feladat:* Írja meg a topológikus rendezés stuktogramját! (szomszédossági listás és szomszédossági mátrixos gráfábrázolások esetén!)
    - Mit tud mondani az algoritmusok hatékonyságáról?

Erősen összefüggő komponensek meghatározása = A redukált gráf előállításának algoritmusa* Lépések:

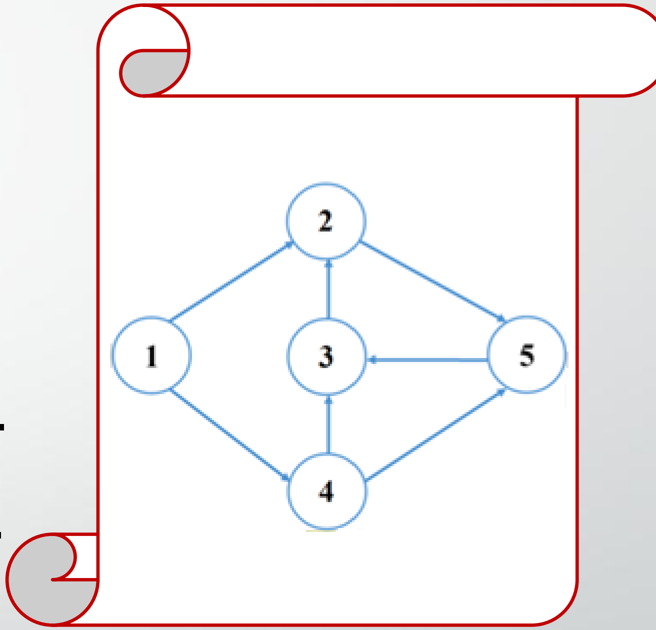
1. A gráfot bejárjuk mélységi bejárással, a csúcsokat a befejezési számok sorrendjében kiírjuk egy verembe.
2. Transzponáljuk a gráfot, azaz megfordítjuk az élek irányítását.
3. Bejárjuk a transzponáltat mélységi bejárással, de nem az alapvető sorrend szerint vesszük a csúcsokat kiinduló csúcsnak, hanem az 1. lépésben készített veremből történő kivétel sorrendjében.

Erősen összefüggő komponensek meghatározása = A redukált gráf előállításának algoritmusa

- A lépések végrehajtásával egy mélységi erdőt kapunk:
  - A fák a gráf erős komponensei lesznek
- Redukált gráf előállítása:
  - EÖK fáinak a csúcsait összevonjuk egy csúccsá
  - A csúcsok között az éleket az eredeti gráf éleinek megfelelően megadjuk

Erősen összefüggő komponensek*

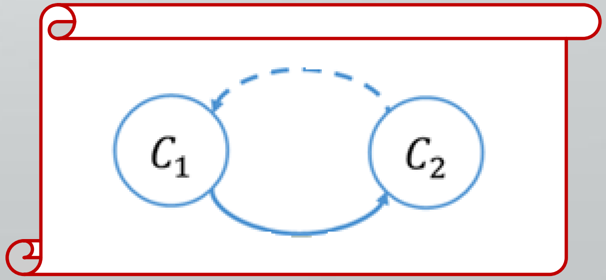
- **Definíció.**
  - Legyen *G=(V,E)* irányított, véges gráf. *G összefüggő*, ha *G* irányítás nélkül összefüggő.
  - *G erősen összefüggő*, ha $\forall\, u, v \in E$ esetén $\exists\, u \rightsquigarrow v$ út.
- **Definíció.** Legyen *G=(V,E)* irányított, véges gráf, $u, v \in V$.
  - Ekkor legyen $u \approx v$, ha léteznek $u \rightsquigarrow v$, illetve $v \rightsquigarrow u$ utak a gráfban.
- **Állítás.** A ≈ reláció ekvivlenciareláció
  - t ≈ osztályozza a *V* csúcshalmazt.



Összefüggő, de nem erősen összefüggő gráf

Erősen összefüggő komponensek*

- **Definíció.** A $\approx$ reláció ekvivalencia osztályait nevezzük a gráf *erős komponense*inek.
- **Állítás.** Egy összefüggő gráf két erős komponense között az élek csak egy irányba mehetnek.
- **Bizonyítás.** Indirekt tegyük fel, hogy két erős komponens között oda-vissza is mehetnek élek.
  - Legyen a két komponens: C_1 és C_2
  - ➢ Ekkor $u \in C_1$ és $v \in C_2$ csúcsok között léteznek $u \rightsquigarrow v$ és $v \rightsquigarrow u$ utak, ugyanis:
    - az erős komponenseken belül: definíció szerint
    - C_1 és C_2 között: az indirekt feltevés szerint létezik út
  - ➢ $u \approx v$, ami ellentmondás



Erősen összefüggő komponensek*

- **Műveletigény:** $T(n) \in O(n + m)$
  - Mélységi bejárás: $O(n + m)$
  - Gráf megfordítása: $O(m)$
  - Veremműveletek
- **Megjegyzés:** Ha a gráf DAG, akkor az algoritmus 3. lépésében említett bejárási sorrend a gráf egy topologikus rendezése.
- **Állítás:** Bármely mélységi bejárás során, egy erős komponens összes csúcsa ugyanabba a mélységi fába kerül.
- **Állítás:** Futtassuk le a fenti algoritmust a $G=(V,E)$ gráfon.
  Legyenek $u,v \in V$ a gráf csúcsai, és tekintsük az algoritmus 3. lépésében kapott mélységi fákat.

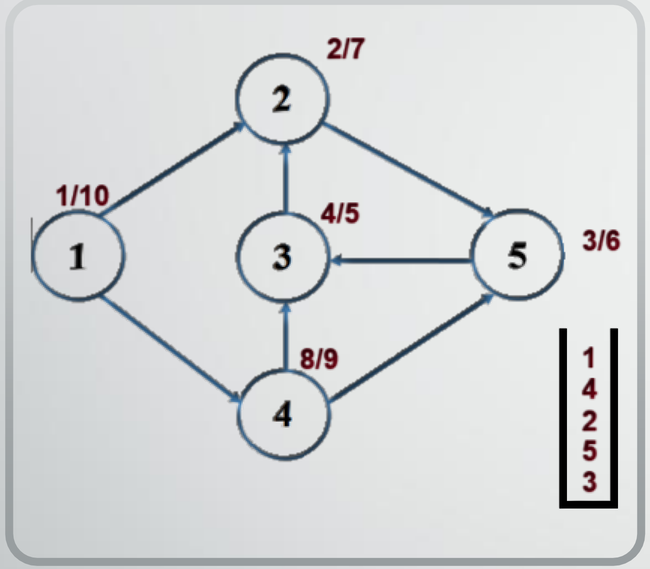
  Ekkor $u \approx v \Longleftrightarrow$ ha u és v ugyanabban a mélységi fában vannak

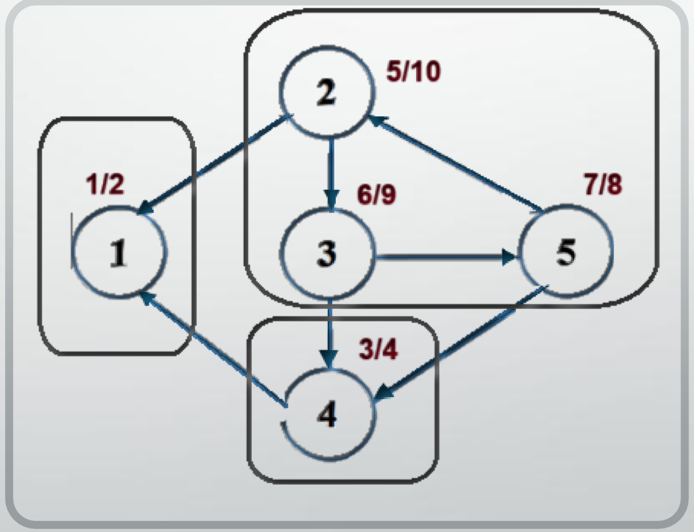
Erősen összefüggő komponensek*

- **Definíció.** Legyen *G=(V,E)* irányított, véges gráf. *G redukált gráf*ja olyan irányított gráf,
  - amelynek csúcsai *G* erős komponensei,
  - és a redukált gráf két csúcsa között, akkor halad él, ha csúcsoknak megfelelő *G* erős komponensei között halad él,
  - továbbá az él irányítása megegyezik az erős komponensek között haladó él(ek) irányításával.
- **Állítás.** A redukált gráf DAG, azaz körmentes irányított gráf.
- **Bizonyítás.** Indirekt tegyük fel, hogy a redukált gráf nem DAG, azaz létezik benne irányított kör.
  - Legyen egy ilyen kör $C_1 \longrightarrow C_{1-} \longrightarrow C_2 \longrightarrow \ldots \longrightarrow C_k \longrightarrow C_1$
  - Ekkor a kör mentén lévő komponensek kölcsönösen elérhetők,
  - vagyis $C_1 \approx C_2 \approx \ldots \approx C_k$, ami ellentmondás.

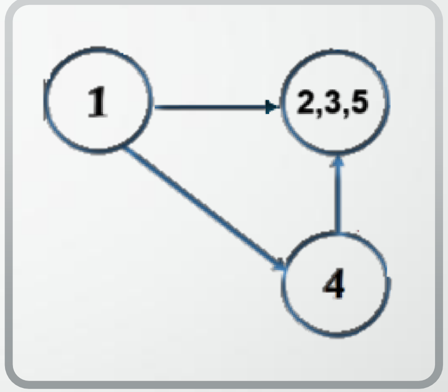
Példa

Erősen összefüggő komponensek





Redukált gráf



Ellenőrző kérdések: Mélységi gráfkeresés

1. Rajzolja le a Mélységi gráfkeresés absztrakt struktogramját!
   - Mit tud a műveletigényéről? (Indokolja is az állítást!)
2. Adja meg az éltípusok definícióját és mondja ki az osztályozásukkal kapcsolatos tételt!
3. Szemléltesse a Mélységi keresést az alábbi irányított gráfon!
   - nemdeterminisztikus esetekben mindig a kisebb indexű csúcsot részesítse előnyben!
   - Jelölje a bejárás során a különböző éltípusokat is!
   - a -> b ; d. b -> c ; d. c-> e. d.-> e. e-> b. f-> c ; e.

Ellenőrző kérdések: topologikus rendezés

1. Mit értünk egy irányított gráf csúcsainak topologikus rendezése alatt?
   - Mondja ki és bizonyítsa be az ezzel kapcsolatos tételt!
2. Mutassa be az alábbi gráf 2 csúcsai topologikus rendezésének a gráf mélységi bejárására épülő algoritmusát!
   - a ! b; d. b ! c; d. c ! e. d ! e. f ! c; e.
   - Nemdeterminisztikus esetekben előny: az alfabetikusan kisebb indexű csúcs
   - Módosítsa egyetlen él behúzásával úgy a gráfot, hogy ne legyen topologikus rendezése!
     - A módosított gráfnak miért nincs topologikus rendezése?
     - Mikor derül ez ki a fent szemléltetett algoritmus végrehajtása során?
3. Mit tud a mélységi bejárásra épülő topologikus rendezés műveletigényéről?

Köszönöm a figyelmet!

Pusztai Kinga

A bemutató Ásványi Tibor: Algoritmusok és adatszerkezetek II. eladásjegyzet: Elemi gráfalgoritmusok és Fekete István: Algoritmusok és adatszerkezetek jegyzete alapján készült.