Algoritmusok és adatszerkezetek I.
8. Előadás

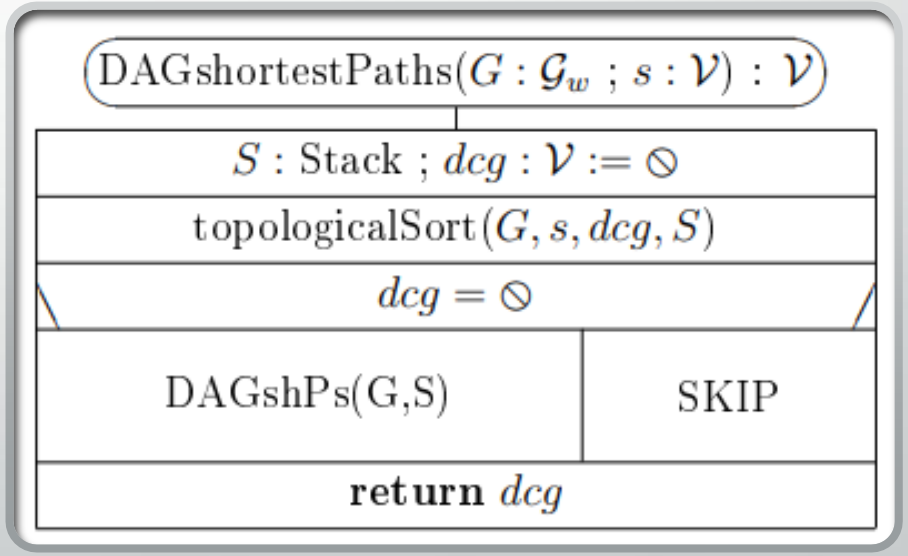
DAG legrövidebb utak egy forrásból. Negatív kör, Sor-alapú Bellman-Ford algoritmus, menetek, negatív kör keresése.

Tartalom

DAG legrövidebb utak egy forrásból (DAGshP)

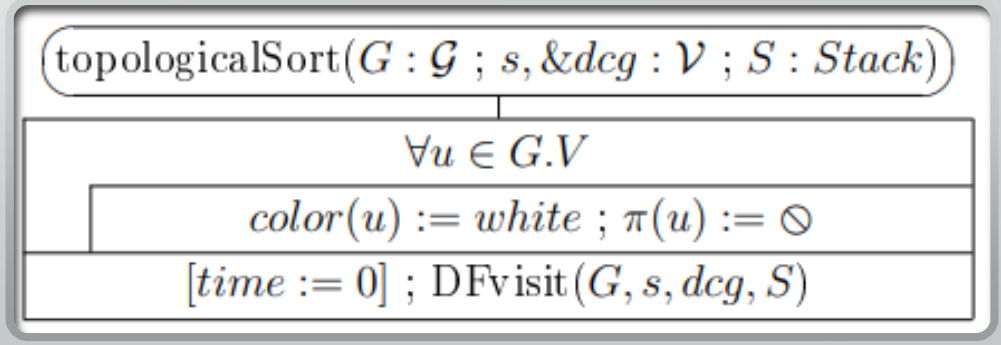
- **Előfeltétel:** A $\mathcal{G} : \mathcal{G}_w$ gráf irányított, és a gráfban nincs *s*-ből elérhető irányított kör
  - Nincs a gráfban *s*-ből elérhető negatív kör -> a legrövidebb utak egy forrásból feladat megoldható.

- Az algoritmus ellenőrzi az előfeltételét
  - Ha teljesül, a DAGshortestPaths() függvény $\oslash$ értékkel tér vissza
  - Különben megtalál egy irányított kört, aminek egyik csúcsával tér vissza
  - Ebből indulva a π címkék mentén a kör fordított irányban bejárható



DAG legrövidebb utak egy forrásból (DAGshP)

- A topologikus rendezés csak az *s*-ből elérhető részgráfot próbálja topologikusan rendezni.
- A *time* nevű változóra és a hozzá kapcsolódó utasításokra csak a topologikus rendezés szemléltetésének megkönnyítése végett van szükségünk.
  - Ezek az implementációkból elhagyhatók
    - a vonatkozó utasítások szögletes zárójelben
  - Time: az egyszerűség kedvéért globális



DAG legrövidebb utak egy forrásból (DAGshP)

DFvisit($G : \mathcal{G}$; $u, \&dcg : \mathcal{V}$; $S : Stack$)

- $color(u) := grey$; $[d(u) := ++time]$
- $\forall v : (u, v) \in G.E$ **while** $dcg = \oslash$
  - $color(v) = white$
    - $\pi(v) := u$
    - DFvisit(G, v, dcg, S)
  - $color(v) = grey$
    - $\pi(v) := u$; $dcg := v$
    - skip
- $[f(u) := ++time]$; $color(u) := black$; S.push(u)

$$MT(n, m) \in \Theta(n + m)$$

$$mT(n, m) \in \Theta(n)$$

DAGshPs($G : \mathcal{G}_w$; S : Stack)

- $\forall v \in G.V$
  - $d(v) := \infty$; $\pi(v) := \oslash$ // distances are still ∞, parents undefined
  - // $\pi(v)$ = parent of v on $s \rightsquigarrow v$ where $d(v) = w(s \rightsquigarrow v)$
- $s := S$.pop()
- $d(s) := 0$ // s is the root of the shortest-path tree
- $u := s$ // going to calculate shortest paths form s to other vertices
- $\neg S$.isEmpty()
  - // check, if $s \rightsquigarrow u \rightarrow v$ is shorter than $s \rightsquigarrow v$ before
  - $\forall v : (u, v) \in G.E \wedge d(v) > d(u) + G.w(u, v)$
    - $\pi(v) := u$; $d(v) := d(u) + G.w(u, v)$
  - $u := S$.pop() // $s \rightsquigarrow u$ is optimal now

DAGshP tételek

1. Tétel. A legrövidebb utak egy forrásból algoritmus az s-ből elérhető csúcsokba optimális utat talál. Az s-ből el nem érhető x csúcsokra $d(x) = \infty$ és $\pi(x) =$ ⦸ lesz.

- **Bizonyítás.**
  - Az s-ből indított részleges topologikus rendezés
    - az s-ből elérhető csúcsokat teszi az S verembe
      - a megfelelő sorrendben

    ➢ pontosan ezeket a csúcsokat terjesztjük ki, a topologikus sorrendnek megfelelően
    - A többi x csúcsra az inicalizálás eredményeként $d(x) = \infty$ és $\pi(x) =$ ⦸ marad
      - ui. ha egy csúcs nem érhető el s-ből, akkor egyetlen G-beli megelőzője sem.

Bizonyítás folytatása

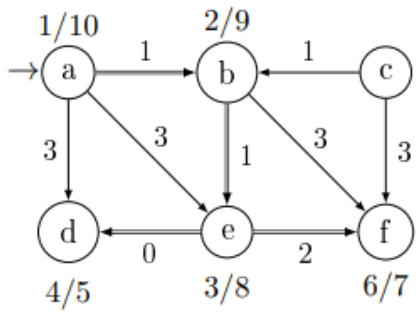
- Elegendő belátni, hogy amikor egy *u* csúcsot kiválasztjuk kiterjesztésre
  - (először az *u* := *s*, később az *u* := *S*.*pop*() utasítással)
  - ➢ *u*-ba már optimális utat találtunk
- Az előbbi állítás *u* = *s* esetben nyilván igaz
- Tegyük fel, hogy adott *v* ≠ *s* csúcs kiválasztása előtt mindegyik kiterjesztésre kiválasztott csúcsra igaz volt
- Mivel a *v* csúcsnak topologikus sorrend szerinti és így a *G*-beli megelőzőit is a *v* kiválasztásának pillanatában már kiterjesztettük a *v* csúcsnak adott $s \overset{opt}{\rightsquigarrow} v$ úton vett közvetlen *u* megelőzőjét is
  - mégpedig úgy, hogy *u*-ba a kiterjesztésekor már optimális utat találtunk
  - így legkésőbb az $u \rightarrow v$ él feldolgozásakor *v*-be is optimális utat találtunk már

A DAG legrövidebb utak egy forrásból algoritmus szemléltetése

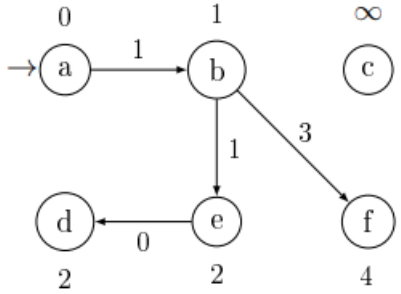
- $s = a$
- Először topologikusan rendezzük az s-ből elérhető részgráfot
  - (A dupla nyilak a mélységi fa fa-éleit jelölik.)
- Ezután a szokásos inicializálásokat követően a csúcsokat a topologikus sorrendjüknek (S) megfelelően terjesztjük ki

ex-pand $d\,v$	changes of d and π labels					
	a	b	c	d	e	f
	$0\,\varnothing$	$\infty\,\varnothing$	$\infty\,\varnothing$	$\infty\,\varnothing$	$\infty\,\varnothing$	$\infty\,\varnothing$
0 a		-2 a		-3 a	0 a	
-2 b					-1 b	1 b
-1 e				-5 e		
1 f						

$a \to b, 1 ; d, 3 ; e, 3.$
$b \to e, 1 ; f, 3.$
$c \to b, 1 ; f, 3.$
$e \to d, 0 ; f, 2.$
$S = \langle a, b, e, f, d \rangle$



- A legrövidebb utak fája $s = a$ esetén
- Az s-ből elérhetetlen csúcsot vagy csúcsokat is feltüntetjük, ∞ d értékkel



Sor-alapú (Queue-based) Bellman-Ford algoritmus (QBF)

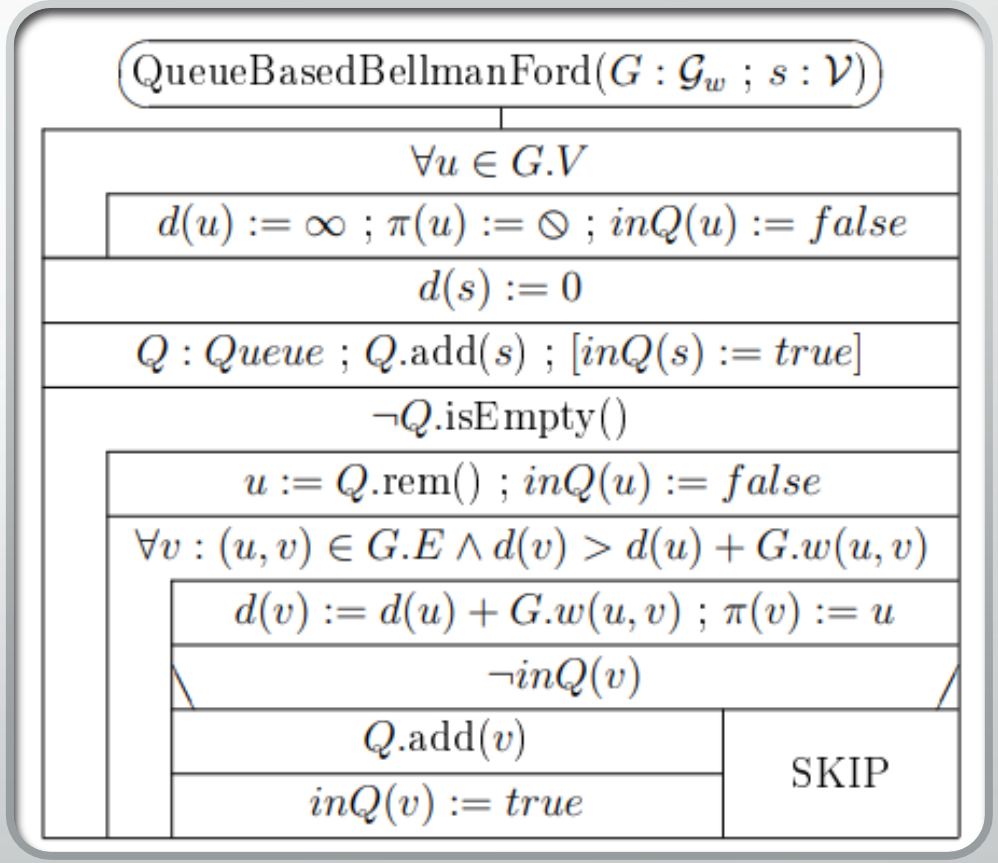
- **Előfeltétel:** Nincs az *s*-ből elérhető negatív kör.

  (Irányított gráf esetén lehetnek negatív élsúlyok.)

- Tarján Róbert Endre amerikai matematikus elemezte és publikálta
  - Breadth-first Scanning néven
- QBF ~ szélességi keresés
  - de az utak hosszát a Dijkstra, a DAGshP és a Bellman-Ford algoritmushoz hasonlóan
  - mint az út menti élsúlyok összegét határozza meg

Sor-alapú (Queue-based) Bellman-Ford algoritmus (QBF) működése

- Kezdetben csak az *s* start (forrás) csúcsot teszi a sorba.
- Inicializálások ~BFS-hez
- Fő ciklus: a sor első elemét kiveszi és kiterjeszti,
- Különbség: mindegyik közvetlen rákövetkezőjére vizsgálat
  - nem talált-e bele rövidebb utat mint eddig
- Ha igen: megfelelően módosítja a *d* és a *π* értékét
- Ha a módosított *d* és *π* értékű csúcs pillanatnyilag nincs benne a sorban, beteszi a sor végére



QBF működése

- A sor (Queue) adattípusra nincs „∈” műveletünk
  - vagy átlátszó sor típust kellene bevezetnünk és megvalósítanunk
  - vagy minden *v* csúcsra értelmezünk egy $inQ(v)$ logikai értékű címkét
    - Igaz ⇔ amikor *v* eleme a sornak
  - Implementációja:
    - feltéve, hogy a csúcsokat 1-től *n*-ig sorszámoztuk
    - $inQ/1 : \mathbb{B}[n]$
- Eredmény:
  - Ha nincs a gráfban *s*-ből elérhető negatív kör:
    - minden *s*-ből elérhető *v* csúcsra meghatározott egy $s \overset{opt}{\rightsquigarrow} v$ utat
    - üres sorral megáll
  - Ha a gráf tartalmaz az *s*-ből elérhető negatív kört
    - a kiterjesztések egy ilyen mentén „körbejárnak”
    - ➢ a sor sosem ürül ki
    - ➢ és az algoritmus végtelen ciklusba kerül

A negatív körök kezelése

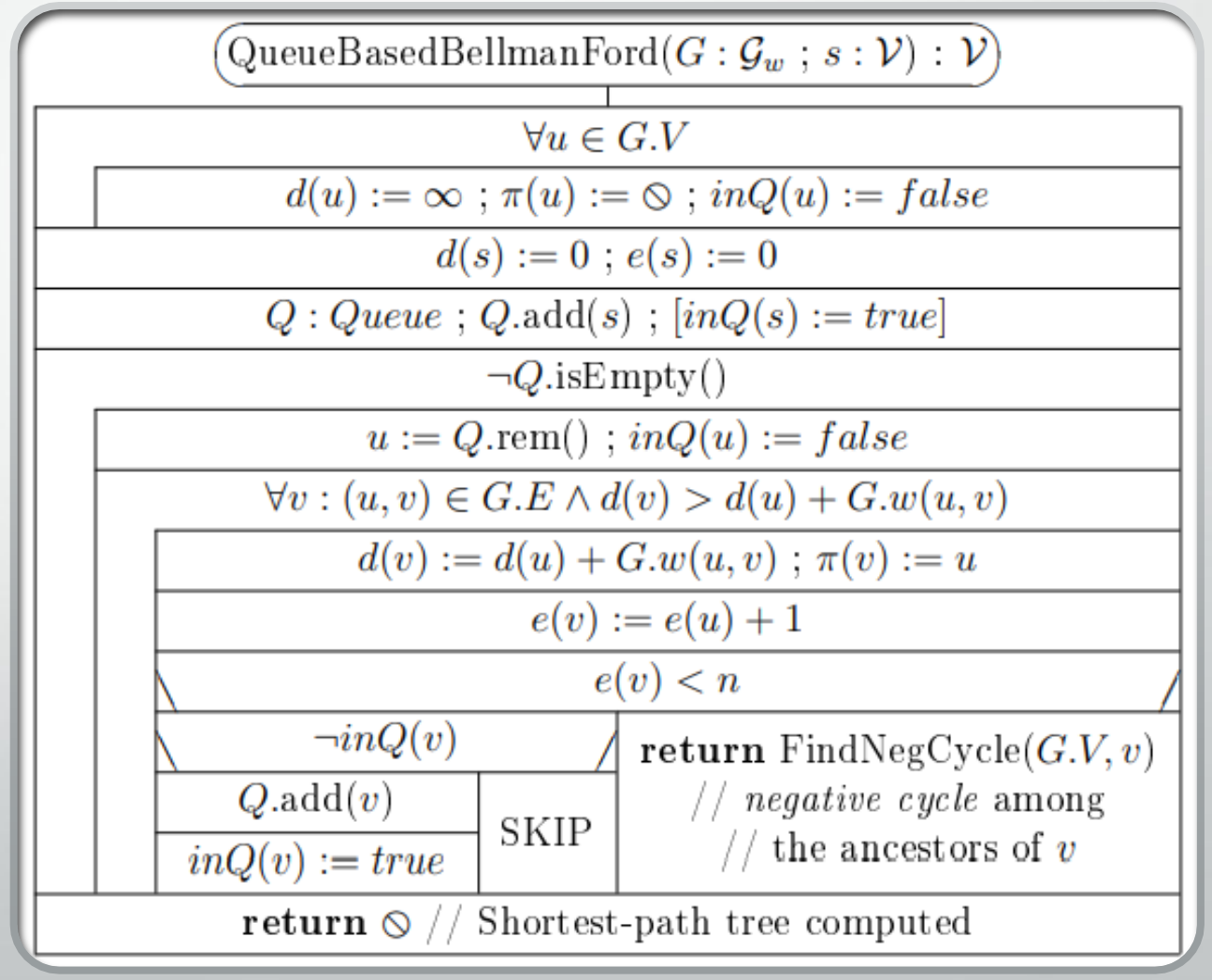
- Bevezetjük a gráf *v* csúcsaira az *e*(*v*) címkét
  - Ennyi élt tartalmaz s ⇝ v
- Ha nincs a gráfban *s*-ből elérhető negatív kör ⇔ QBF futása során minden, a fő ciklusban kezelt *v* csúcsra *e*(*v*) < *n*
  - Fordítva: Ha van a gráfban *s*-ből elérhető negatív kör ⇔ a QBF futása során van olyan, a fő ciklusban kezelt *v* csúcs, amelyre *e*(*v*) ≥ *n*.
- Körök keresése
  - Ez a *v* csúcs vagy része egy negatív körnek, amit a csúcsok *π* címkéi mutatnak meg
  - vagy a *π* címkéken keresztül el lehet belőle jutni egy ilyen negatív körbe, ami gráf csúcsainak *n* száma miatt összesen legfeljebb *n* lépésben azonosítható

Változtatások az algoritmusban a negatív körök kezeléséhez

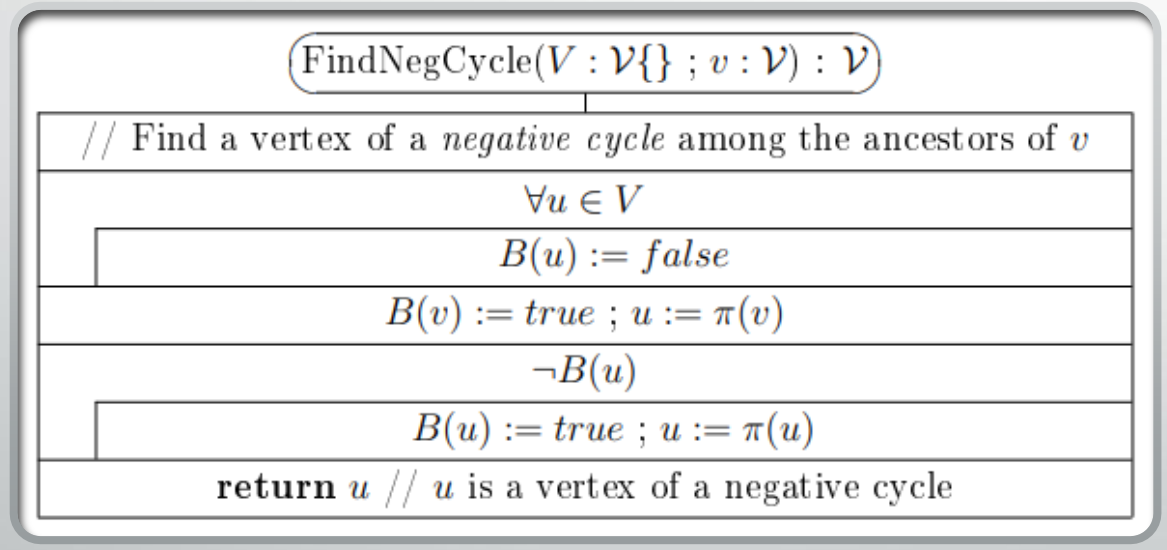
- Az *s*-ből elért *v* csúcsokra a $d(v)$, a $\pi(v)$ és az $inQ(v)$ mellett az $e(v)$ címkét is nyilvántartjuk
- Ha valamely (u, v) él mentén végzett közelítés során $e(v) \geq n$
  - meghatározzuk valamelyik negatív kör egyik csúcsát
  - az algoritmus ezzel tér vissza
- Ha a QBF futása során mindegyik közelítés után $e(v) < n$
  - üres sorral állunk meg
  - Az algoritmus ⦸ extremális hivatkozással tér vissza
- Az így kiegészített QBF $O(n*m)$ idő alatt minden esetben befejeződik

A negatív körök kezelésével kiegészített struktogramok

- Egy tetszőleges *v* csúcs *d*, *e* és *π* címkéjének módosítása után a *v* csúcsot a sor végére tesszük ⇔
  - $e(v) < n$
  - *v* nincs benne a sorban
- Ha biztosan nincs a gráfban *s*-ből elérhető negatív kör
  - ➢ az algoritmus fentebbi változata alkalmazható



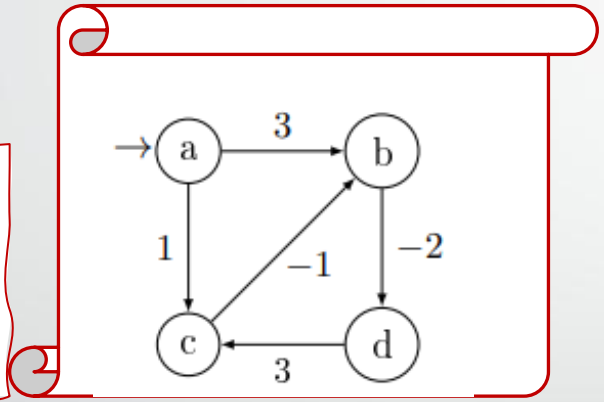
A negatív körök kezelésével kiegészített struktogramok



A legrövidebb utak fája meghatározásának szemléltetése

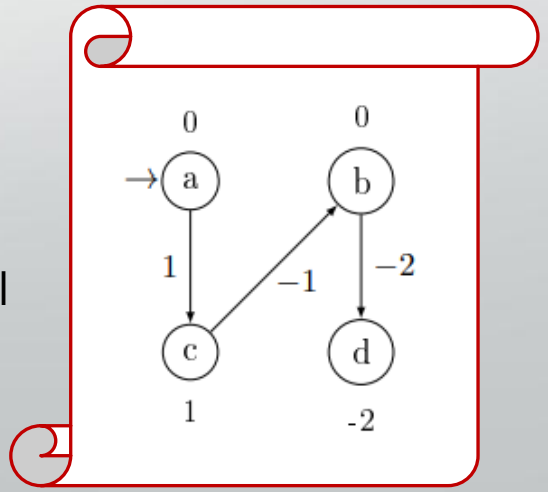
- Mindegyik *s*-ből elérhető *v* csúcsra kiszámítunk egy $s \overset{opt}{\rightsquigarrow} v$ utat

$a \to b,\ 3\ ;\ c,\ 1.$
$b \to d,\ -2.$
$c \to b,\ -1.$
$d \to c,\ 3.$



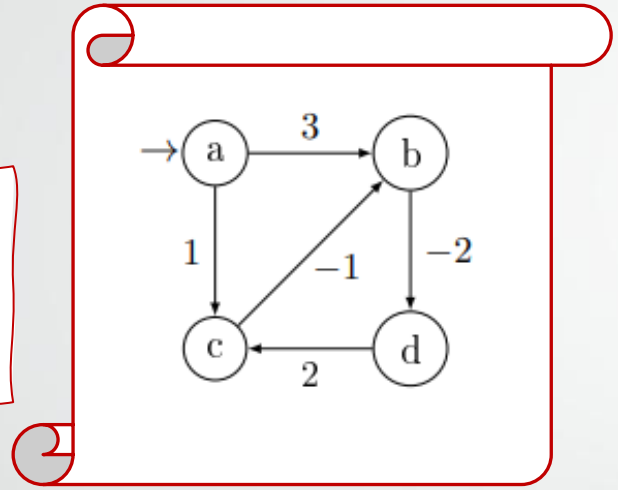
ex-pand	changes of $d\,\pi\,e$				Q :
	a	b	c	d	Queue
$d\,\mathcal{V}\,e$	0 ⊘ 0	∞ ⊘	∞ ⊘	∞ ⊘	⟨ a ⟩
0 a 0		3 a 1	1 a 1		⟨ b, c ⟩
3 b 1				1 b 2	⟨ c, d ⟩
1 c 1		0 c 2			⟨ d, b ⟩
1 d 2					⟨ b ⟩
0 b 2				−2 b 3	⟨ d ⟩
−2 d 3					⟨ ⟩

- A legrövidebb utak fája *s* = *a* esetén
- A csúcsoknál csak a *d* értékeket tüntetjük fel



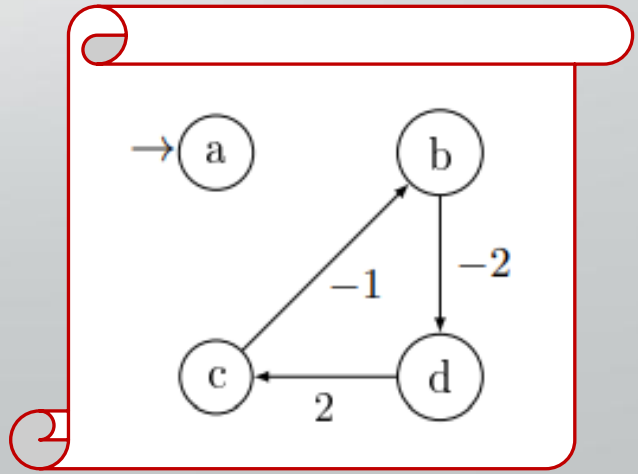
A negatív körök kezelésének szemléltetése

a → b, 3 ; c, 1.
b → d, −2.
c → b, −1.
d → c, 2.



ex-pand $d\,\mathcal{V}\,e$	changes of $d\,\pi\,e$ a	b	c	d	Q : Queue
	0 ⊘ 0	∞ ⊘	∞ ⊘	∞ ⊘	⟨ a ⟩
0 a 0		3 a 1	1 a 1		⟨ b, c ⟩
3 b 1				1 b 2	⟨ c, d ⟩
1 c 1		0 c 2			⟨ d, b ⟩
1 d 2					⟨ b ⟩
0 b 2				−2 b 3	⟨ d ⟩
−2 d 3			0 d ④		⟨ ⟩

- Itt megállunk, mert $e(c) = 4 = n$
  - negatív kör található a *c* csúcs ősei közt
- A π értékek szerint visszafelé haladva megtalálhatjuk a negatív kört



A sor-alapú Bellman-Ford algoritmus (QBF) elemzése

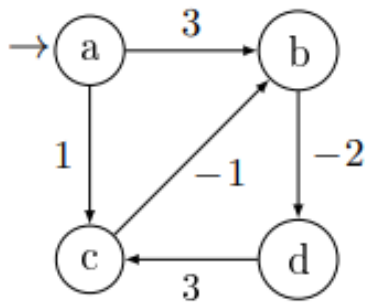
- Az alábbi elemzésben arra az esetre szorítkozunk, amikor nincs a gráfban *s*-ből elérhető negatív kör.
- Az algoritmus helyességének bizonyítása és az algoritmus hatékonyságának vizsgálata szempontjából is alapvető a QBF futásának menetekre bontása

10. Definíció. Menet rekurzív definíciója:

- 0. menet: a start csúcs (s) feldolgozása
- (*i*+1). menet: az *i*. menet végén a sorban levő csúcsok feldolgozása

Az s $\overset{opt}{\rightsquigarrow}$ v utak megkeresésének szemléltetése menetszámlálással

a → b, 3 ; c, 1.
b → d, −2.
c → b, −1.
d → c, 3.

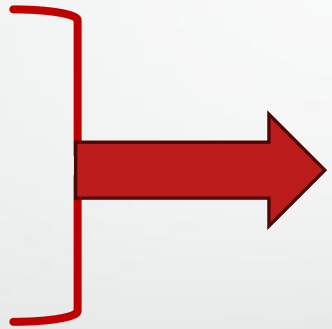


ex-pand $d\,\mathcal{V}\,e$	changes of $d\,\pi\,e$ a	b	c	d	Q : Queue	Pass
	0 ⊘ 0	∞ ⊘	∞ ⊘	∞ ⊘	⟨ a ⟩	
0 a 0		3 a 1	1 a 1		⟨ b, c ⟩	0.
3 b 1				1 b 2	⟨ c, d ⟩	1.
1 c 1		0 c 2			⟨ d, b ⟩	1.
1 d 2					⟨ b ⟩	2.
0 b 2				−2 b 3	⟨ d ⟩	2.
−2 d 3					⟨ ⟩	3.

QBF elemzése

11. Tulajdonság. A gráf tetszőleges *u* csúcsára

- ha *s*-ből nem érhető el negatív kör
- és az *u* csúcsba vezet *k* élből álló $s \overset{opt}{\rightsquigarrow} u$ út



- a *k*. menet elején már $d(u) = w(s \overset{opt}{\rightsquigarrow} u)$
- és $<s, \ldots, \pi^2(u), \pi(u), u>$ egy optimális út

- **Bizonyítás.** *k* szerinti teljes indukcióval

QBF elemzése

12. Lemma. Ha *s*-ből nem érhető el negatív kör

- tetszőleges *u* elérhető csúcsra ∃ $s \overset{opt}{\rightsquigarrow} u$ út, ami legfeljebb $n-1$ élből áll

- **Bizonyítás.**
  - Ebben az esetben az *s*-ből induló körmentes utak hossza ≤ mint a kört is tartalmazók
  - Tetszőleges körmentes útnak viszont legfeljebb $n-1$ éle van
  - Véges sok körmentes út van a gráfban
  - *s*-ből tetszőleges *v* csúcsba is véges sok körmentes út van
  - ezek között létezik egy optimális

QBF elemzése

13. Tétel. (A fenti Lemma és Tulajdonság következménye)
Ha s-ből nem érhető el negatív kör

- Tetszőleges s-ből elérhető *u* csúcsra van olyan $s \overset{opt}{\rightsquigarrow} u$ út, amit az *n*−1. menet elejére már a QBF kiszámolt
- Az *n*−1. menet végére kiürül a sor, az algoritmus pedig $O(n * m)$ időben megáll
  - $MT(n, m) \in O(n * m)$
- Az elméleti műveletigény becslés: rossz hatékonyság

QBF elemzése

- Gyakorlati tesztek: $\Theta(n)$ átlagos műveletigény, ha
  - Pozitív élsúlyú
  - Véletlenszerűen generált
  - Szomszédossági listás gráfábrázolás esetén
  - Nagyméretű
  - *s*-ből összefüggő
  - ritka gráfokra: ($m \in O(n)$)

  = aszimptotikusan az elméleti minimummal

➢ Hálózatokon való útkeresésnél alkalmazzák

- A hálózatok jelentős része nagyméretű, ritka gráffal modellezhető

Ellenőrző kérdések

1. Mit számol ki a Sor-alapú (Queue-based) Bellman-Ford algoritmus?
   - Adja meg a struktogramját!
   - Mit értünk a fenti program futásának menetei alatt? Mi a menetekhez kapcsolódó alapvető tulajdonság?
   - Adjon az algoritmus műveletigényére aszimptotikus felső becslést, és indokolja is állítását!
   - Honnét ismerhető fel, hogy van-e a gráfban a startcsúcsból elérhető negatív kör? Hogyan lokalizálható egy ilyen negatív kör?
   - Szemléltesse az algoritmus működését az alábbi gráfon, a *b* csúcsból indítva!
     - *b*-> *c*, 2 ; *e*, 4. *c*-> *d*, -1 ; *e*, 1. *d*-> *b*, -1 ; *e*, 2 ; *f*, 2. *e* -> *f*, -2.
2. Írja le röviden, szóban vagy struktogrammal, azt a tanult algoritmust, mellyel irányított, súlyozott, körmentes gráfokra a leghatékonyabb módon határozhatjuk meg a start csúcsból a többi csúcsba vezető legrövidebb utak fáját! (Negatív élköltségek is megengedettek.)
   - Mekkora a műveletigénye? Miért?
   - Szemléltesse a működését az alábbi gráfon, ahol a csúcsok topologikus rendezése <*a*, *b*, *c*, *d*, *e*, *f*>, és a *b* a startcsúcs! A legrövidebb utak fáját rajzolja is le!
     - *a*-> *d*, 2; *f*, 3. *b*-> *c*, 2; *e*, 4. *c*-> *d*, -1; *e*, 1. *d*-> *e*, 2; *f*, 2. *e* -> *f*, -2.

Köszönöm a figyelmet!

Pusztai Kinga

A bemutató Ásványi Tibor: Algoritmusok és adatszerkezetek II. eladásjegyzet:Élsúlyozott gráfok és algoritmusaik alapján készült.