# A DFS algoritmus*

Korábban már megismerkedtünk a BFS (Szélességi keresés) algoritmussal, amely bejárja egy adott $G$ gráf adott $s$ csúcsából elérhető csúcsokat és eközben több feladatot is hatékonyan megold: meghatározza a többi csúcs $s$-től való távolságát (amennyiben minden él hosszát 1-nek tekintjük), eldönti, hogy összefüggő-e a gráf, illetve meghatározza az $s$-et tartalmazó komponens egy feszítőfáját. A BFS eljárás által adott BFS-fát szemléletesen a lehető „legszélesebbnek" érezhetjük: $s$-nél a lehető legtöbb irányba ágazik, majd minden ág ismét a lehető legtöbb felé ágazik tovább, stb.

Létezik azonban a gráfok bejárására egy másik, nagyon sok alkalmazással bíró megközelítés is: ez a DFS (*Depth First Search*, magyarul *Mélységi keresés*). Ez bizonyos értelemben épp a BFS-sel ellentétes stratégiát alkalmaz: $s$-ből indulva addig halad „előre", míg el nem akad; ekkor visszalép egyet és ismét elakadásig megy, stb. A DFS által adott feszítőfa tehát nem széles, hanem inkább „mély" lesz. Ösztönösen is ezt a megközelítést választja mindenki, ha egy ismeretlen terepet, például egy épületet akar felderíteni: senkinek nem jutna eszébe a BFS logikája szerint először az $s$ „bejárat" melletti szobákat végiglátogatni (és közben folyton visszarohanni $s$-hez), majd csak ezután merészkedni a bejárattól két szobányi távolságra. Ehelyett természetesnek tűnik mindig új és új szobákba továbblépve addig bolyongani az épületben, amíg további, még meg nem látogatott szobába már nem nyílik ajtó és csak ekkor visszafordulni az eggyel korábbi szobába.

A BFS algoritmust irányítatlan és irányított gráfokra is leírtuk és hasonló a helyzet a DFS-sel is: mindkét fajta gráfra alkalmazható. Ennek ellenére, az alábbiakban irányított gráfokra adjuk meg az algoritmus részletes leírását (mert a gyakorlati alkalmazásokban ez fordul elő többször) és csak röviden ejtünk szót az irányítatlan esetről. Irányított gráfban viszont könnyen előfordulhat, hogy az $s$ kezdőpontból irányított úton elérhető csúcsok halmaza kisebb, mint a gráf csúcshalmaza – még akkor is, ha a gráf egyébként irányítatlan értelemben összefüggő. Ezért a DFS algoritmus nagyon egyszerű stratégiát alkalmaz annak érdekében, hogy a gráf minden csúcsát elérje: ha már bejárta az $s$-ből elérhető csúcsokat és mégsem érte el a gráf összes csúcsát, akkor egy tetszőleges eléretlen pontból újraindítja a bejárást és ezt egészen addig ismétli, amíg minden csúcs bejárttá nem válik.

Az algoritmus a működése során a csúcsokat kétféleképpen is megszámozza: a *mélységi számozás* azt mutatja meg, hogy az algoritmus hányadjára ért el egy csúcsot; a *befejezési számozás* viszont azt írja le, hogy az eljárás hányadjára fejezte be egy csúcsnak és „leszármazottainak" a végigvizsgálását. Az előbbit $d(v)$, az utóbbit $f(v)$ fogja jelölni (az angol *depth*, illetve *finish* szavakból).

Így az algoritmus a működése során az alábbi adatokat tartja nyilván:

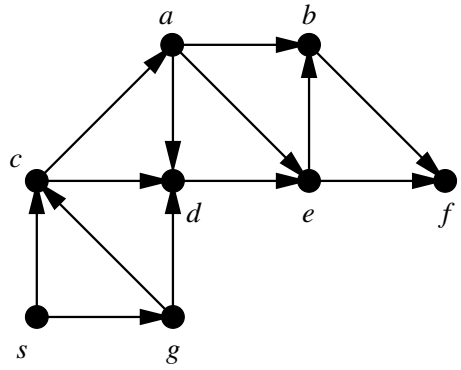
- $d(v)$ $(v \in V)$: a $v$ csúcs mélységi száma
- $f(v)$ $(v \in V)$: a $v$ csúcs befejezési száma
- $m(v)$ $(v \in V)$ : a $v$-t **m**egelőző csúcs – vagyis az, amiből a bejárás $v$-t elérte
- $a$: a jelenleg aktív csúcs
- $D$: az eddigi legnagyobb mélységi szám
- $F$: az eddigi legnagyobb befejezési szám

DFS ALGORITMUS

**Bemenet:** Egy $n$ csúcsú $\overrightarrow{G} = (V,E)$ irányított gráf és egy $s \in V$ csúcs

```
1   d(s) ← 1, minden v ∈ V, v ≠ s-re d(v) ← *
2   minden v ∈ V-re f(v) ← *
3   minden v ∈ V-re m(v) ← *
4   D ← 1; F ← 0; a ← s
5   ciklus
6       ha létezik olyan e = av⃗ él, amelyre d(v) = *, akkor:
7           D ← D + 1
8           d(v) ← D
9           m(v) ← a
10          a ← v
11      különben:
12          F ← F + 1
13          f(a) ← F
14          ha m(a) ≠ *, akkor:
15              a ← m(a)
16          különben:
17              ha van olyan v csúcs, amelyre d(v) = *, akkor:
18                  a legyen egy ilyen v csúcs
19              különben:
20                  stop
21  ciklus vége
```

Az eljárás működését az alábbi gráfon illusztráljuk:




**1. ábra**

Lefuttatva az algoritmust az alábbi adatok keletkeznek:

| $v$ | $s$ | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ |
|---|---|---|---|---|---|---|---|---|
| $d(v)$ | 1 | 8 | 6 | 7 | 3 | 4 | 5 | 2 |
| $f(v)$ | 8 | 5 | 2 | 6 | 4 | 3 | 1 | 7 |
| $m(v)$ | $*$ | $c$ | $e$ | $g$ | $g$ | $d$ | $e$ | $s$ |

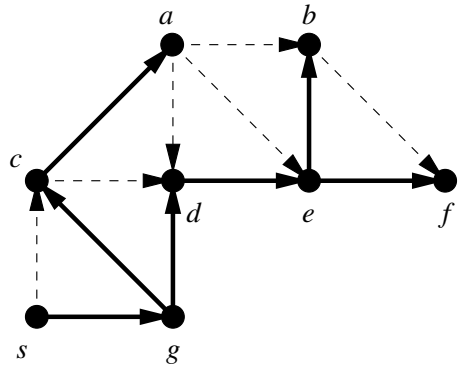
Természetesen ez csak az egyik lehetséges helyes futása az algoritmusnak: az eljárás leírásában nincs arra vonatkozó megkötés, hogy az éppen aktuális csúcsból melyik, még bejáratlan szomszédjába lépjünk tovább.

A fenti példában egyszer sem volt szükség új gyökérpont választására (vagyis a 18. sor végrehajtására): a 14. sor végrehajtásakor $m(a) = *$ először $a = s$-re fordult elő, amikor már minden $v$ csúcsra $d(v) \neq *$ volt, így az eljárás megállt. Említettük azonban, hogy ez nem mindig van így, csak ha $s$-ből $\overrightarrow{G}$ minden csúcsa elérhető irányított úton (mint a fenti példában is). Az általános esetben az algoritmus leállásakor azokra a $v$ csúcsokra lesz $m(v) = *$, amelyek az eljárás során valamikor gyökérpontok voltak.

A DFS algoritmus futásidejéről hasonlókat mondhatunk, mint a BFS-ről: minden élen legföljebb egyszer próbál továbblépni, ami (függetlenül attól, hogy ez sikerül-e) csak konstansnyival járul hozzá a futásidőhöz. Ezért a teljes futásidő felülről becsülhető $c \cdot m$-mel, ahol $m$ a gráf élszáma és $c$ alkalmas konstans. Így a DFS eljárás is rendkívül hatékony: a lépésszáma lineáris (vagyis az input méretének lineáris függvényével felülről becsülhető). Azonban itt is igaz, hogy ezzel a felületes elemzéssel több fontos részletet figyelmen kívül hagytunk: a $c \cdot m$ futásidőhöz feltételeznünk kell, hogy a gráf alkalmasan megválasztott adatstruktúrában lett megadva (hogy minden csúcsra gyorsan kiolvasható módon rendelkezésre álljanak az arra illeszkedő élek) és hogy $\overrightarrow{G}$-nek nincs izolált csúcsa (hogy ne lehessen sokkal több csúcsa, mint éle), illetve az implementációnál is gondosabban kell eljárni, mint ami a fenti pszeudokódból látszik (például azért, hogy amikor egy csúcs újra aktívvá válik, ne próbáljunk másodszor is egy olyan élen továbblépni belőle, amivel korábban már kísérleteztünk).

## A DFS-erdő

Hasonlóan a BFS algoritmushoz, a DFS esetében is külön figyelmet érdemelnek azok az élek, amelyek minden olyan $v$-re, amelyre $m(v) \neq *$ (tehát amelyek nem voltak gyökérpontok) $m(v)$-ből $v$-be mutatnak. A fenti példára ezek az alábbi ábrán láthatók (a többi élt szaggatott vonallal jelöltük).




**2. ábra**

Ebben a példában ezek az élek fát alkotnak (ha eltekintünk az élek irányításától), de ugyanez általában nem igaz (csak akkor, ha $s$-en kívül nem volt több gyökérpont). Az viszont könnyen láthatóan igaz, hogy az $m(v)$-ből $v$-be mutató élek (irányítatlan értelemben)

olyan erdőt (vagyis körmentes részgráfot) alkotnak, amely $\overrightarrow{G}$ minden csúcsát tartalmazza. Ennek az erdőnek a neve *DFS-erdő*.

A DFS-erdő lehetőséget teremt a $\overrightarrow{G}$ éleinek alábbi osztályozására, ami a DFS algoritmus számos alkalmazásában szerephez jut.

**1. Definíció.** *Tegyük fel, hogy az $s$ csúcsból indítva lefuttattuk a DFS algoritmust a $\overrightarrow{G}$ irányított gráfban. Jelölje a futáshoz tartozó DFS-erdőt $F$. Legyen $e = \overrightarrow{uv}$ a $\overrightarrow{G}$-nek egy tetszőleges éle. Ekkor*

1. *$e$-t* faélnek *nevezzük, ha $e \in E(F)$;*
2. *$e$-t* előreélnek *nevezzük, ha nem faél, de $F$-ben van $u$-ból $v$-be irányított út (vagyis $v$ „leszármazottja” $u$-nak);*
3. *$e$-t* visszaélnek *nevezzük, ha $F$-ben van $v$-ből $u$-ba irányított út (vagyis $v$ „őse” $u$-nak);*
4. *$e$-t* keresztélnek *nevezzük, ha $F$-ben sem $u$-ból $v$-be, sem $v$-ből $u$-ba nincs irányított út (vagyis $u$ és $v$ között nincs „egyenes ági leszármazási viszony”).*

Fontos kiemelni, hogy az itt bevezetett fogalmaknak csak a DFS egy konkrét futását feltételezve van értelme: ugyanaz az $e$ él könnyen tartozhat a fenti négy osztály közül egy másikba is, ha a DFS-nek egy másik (egyébként helyes) futását feltételezzük $\overrightarrow{G}$-n. Például a DFS futtatására látott fenti példában az $\overrightarrow{sc}$ él előreél és az összes többi, a DFS-erdőbe nem tartozó él keresztél. Visszaélet tehát ebben a példában nem találunk; később látni fogjuk, hogy ennek a ténynek fontos következményei vannak.

Fentebb már említettük, hogy a DFS eljárás (helyes implementáció esetén) $\overrightarrow{G}$ minden $e$ élével egyszer foglalkozik. Ezért a gyakorlati alkalmazások számára nagyon hasznos, hogy ebben a pillanatban az is megállapítható (és akár eltárolható), hogy $e$ a fenti négy kategória közül melyikbe tartozik – annak ellenére is, hogy ekkor még nem ismert a teljes DFS-erdő. Így tehát a DFS eljárás minimális kiegészítésével az is elérhető, hogy az (a futásidő érdemi növekedése nélkül) a $\overrightarrow{G}$ minden élét besorolja a fenti definíció szerinti osztályok valamelyikébe. Ennek a részleteit mondja ki az alábbi tétel.

**2. Tétel.** *Tegyük fel, hogy a $\overrightarrow{G}$ irányított gráfra a DFS algoritmust futtatva az éppen az $e = \overrightarrow{av}$ élen próbál továbblépni (így az aktív csúcs jelenleg $a$). Ekkor $e$ erre a DFS bejárásra vonatkozóan akkor és csak akkor lesz*

1. *faél, ha $d(v) = *$;*
2. *előreél, ha $d(v) > d(a)$;*
3. *visszaél, ha $d(v) < d(a)$ és $f(v) = *$;*
4. *keresztél, ha $d(v) < d(a)$ és $f(v) \neq *$.*

*Bizonyítás:* Ha $d(v) = *$ az eljárás 6. sorának végrehajtásakor, akkor a 9. sor szerint $m(v) = a$ lesz, így $e$ valóban faél.

A többi állítás belátásához képzeletben egészítsük ki az algoritmus működését egy $T$ változóval, amely az időt méri. (Gyakorlatban erre nincs szükség, csak a bizonyítást segíti.) A ciklus indulása előtt (a 4. sor után) inicializáljuk $T$-t is 0-nak, majd a 8. és a 13. sor után is szúrjunk be egy-egy $T \leftarrow T + 1$ utasítást. Így az algoritmus futása során minden $v$ csúcshoz tartozik egy kezd$(v)$ kezdési idő (a $T$-nek az az értéke, amikor $d(v)$ értéket

kapott) és egy bef($v$) befejezési idő (amikor $f(v)$ értéket kapott), illetve beszélhetünk a $v$-hez tartozó $I(v)$ időintervallumról is (ami kezd($v$)-től bef($v$)-ig tart).

Egyszerű, de hasznos észrevétel, hogy az algoritmus működésének az $I(v)$ intervallumban zajló szakasza a DFS egy szabályos, a $v$-ből indított és a következő gyökérpont választásáig tartó futtatásának felel meg az azon $x$ csúcsokból (és az eredetileg ezek között futó élekből) álló gráfon, amelyeket az eljárás $v$ előtt még nem ért el – vagyis amelyekre a kezd($v$) pillanatban még $d(x) = *$ teljesült. (Itt eltekintettünk attól a részletkérdéstől, hogy a mélységi és befejezési számozás ebben az esetben nem 1-től indul, hanem azoktól az értékektől, amelyeknél kezd($v$)-kor tartottak.) Jelölje az eljárás ezen szakaszához tartozó DFS-erdőt $F_v$. Ekkor $F_v$ egyrészt nyilván egy fa, másrészt azonos a teljes algoritmusból előálló $F$ DFS-erdőnek azzal a részgráfjával, ami a $v$-ből $F$-ben irányított úton elérhető csúcsokból (és a köztük futó $F$-beli élekből) áll.

Tegyük most fel, hogy az algoritmus az $\overrightarrow{av}$ élen a $T$ pillanatban próbál továbblépni. Ekkor $T$-ben az $I(a)$ már elkezdődött, de még nem ért véget, vagyis kezd$(a) \leq T \leq$ bef$(a)$.

Ha $d(v) > d(a)$, akkor a kezd($a$) pillanatban még $d(v) = *$ volt, amiből a fentiek szerint $v \in V(F_a)$. Ezért $v$ valóban elérhető $F_a$-ban (és így persze $F$-ben is) irányított úton $a$-ból, így $e$ valóban előreél.

Ha $d(v) < d(a)$, akkor kezd($v$)-ben még $d(a) = *$ volt. Ha emellett a $T$-ben $f(v) = *$ is igaz, akkor kezd$(v) <$ kezd$(a) \leq T <$ bef$(v)$, vagyis $T$-ben $I(v)$ még tart. Ezért a fentiek szerint $a \in V(F_v)$, vagyis $a$ érhető el irányított úton $v$-ből $F_v$-ben és ezért $F$-ben is. Ezért $e$ valóban visszaél.

Végül ha a $T$ pillanatban $d(v) < d(a)$ és $f(v) \neq *$, akkor a kezd($a$) pillanatban $I(v)$ már véget ért, így $a \notin V(F_v)$ és $v \notin V(F_a)$. Ezért valóban nincs irányított út $F$-ben sem $a$-ból $v$-be, sem $v$-ből $a$-ba, vagyis $e$ keresztél. □

Mielőtt rátérnénk a DFS algoritmus alkalmazásaira, vizsgáljuk meg röviden annak az irányítatlan gráfokra vonatkozó változatát is. Ehhez a fenti pszeudokódban csak a 6. sort kell megváltoztatni:

6 **ha** létezik olyan $e = \{a, v\}$ él, amelyre $d(v) = *$, **akkor:**

Vagyis az $a$-ból induló irányított élek helyett minden, az $a$-ra illeszkedő irányítatlan élen megpróbálunk továbblépni $a$-ból. Ha $G$ irányítatlan és összefüggő, akkor a DFS-erdője nyilván egy feszítőfa lesz, hiszen ekkor $s$-ből minden $G$-beli pont elérhető. (Ezért a BFS-hez hasonlóan a DFS is használható egy gráf összefüggőségének a vizsgálatára, illetve egy feszítőfa meghatározására.)

A $G$ irányítatlan gráfon futtatott DFS elképzelhető úgy is, mintha azon a $\overrightarrow{G}$ irányított gráfon hajtanánk végre, amelyet $G$-ből kapunk úgy, hogy annak minden $e = \{u, v\}$ élét helyettesítjük az $e' = \overrightarrow{uv}$ és $e'' = \overrightarrow{vu}$ irányított élekkel (noha a gyakorlatban persze fölösleges volna az élhalmaz megduplázása). Ebből kiindulva könnyű végiggondolni, hogy hogyan alakul át a $G$ éleinek fent bevezetett osztályozása az irányítatlan esetben. Faélek nyilván továbbra is vannak, azonban a DFS-erdőbe nem tartozó élek sokkal egyszerűbben leírhatók. Egyrészt az előreélek egybeesnek a visszaélekkel (ezeket az irányítatlan esetben visszaélnek szokták hívni), hiszen ha a $G$-ből készített $\overrightarrow{G}$ irányított gráfban $e' = \overrightarrow{uv}$ előreél, akkor $e'' = \overrightarrow{vu}$ visszaél (és fordítva) mert $u$-ból $v$-be pontosan akkor létezik irányított út, ha $v$-ből $u$-ba létezik. Másrészt könnyű látni, hogy $G$-ben nem lehetnek keresztélek: ha $\overrightarrow{G}$-ben $e' = \overrightarrow{uv}$ keresztél volna, akkor $e'' = \overrightarrow{vu}$-nak is annak kellene lennie (mert mindkét

állítás azt fejezi ki, hogy $u$ és $v$ között nincs irányított út a $\overrightarrow{G}$-beli DFS-erdőben), így a fenti tétel szerint $d(u) < d(v)$ és $d(v) < d(u)$ is teljesülne, ami nyilván lehetetlen. Következésképp az irányítatlan gráfban futtatott DFS algoritmus esetében bármely, az erdőbe nem tartozó él végpontjai között kell legyen „egyenes ági leszármazási viszony".

## Aciklikus irányított gráfok, topologikus rendezés

Egy $\overrightarrow{G}$ irányított gráfot akkor nevezünk *aciklikusnak*, ha nem tartalmaz irányított kört. (Elterjedt elnevezés az ilyen gráfokra a *DAG* is, ami az angol *Directed Acyclic Graph* kifejezés rövidítése.) Rengeteg gyakorlati alkalmazásban merülnek fel olyan irányított gráfok, amelyek a feladat természetéből fakadóan aciklikusak. Ennek a ténynek a jelentősége abban rejlik, hogy sok algoritmikus feladat jóval hatékonyabban oldható meg aciklikus irányított gráfokra, mint az általános esetben.

Tekintsük például azt a $\overrightarrow{G}$ irányított gráfot, aminek a csúcsai a Földön élő emberek és $x$-ből akkor vezet él $y$-ba, ha $y$ gyermeke $x$-nek. $\overrightarrow{G}$ nyilván aciklikus, de ezt a tényt különösen látványossá tehetjük a következő módon: képzeletben állítsuk fel egyetlen sorba a Föld összes emberét, mégpedig balról jobbra születési idő szerinti növekvő sorrendben (feltételezve, hogy semelyik két ember nem született hajszálra azonos pillanatban). Ekkor $\overrightarrow{G}$ minden éle balról jobbra mutat (hiszen mindenki idősebb a saját gyerekeinél); így $\overrightarrow{G}$-ben az élek mentén haladva egyre inkább jobbra kerülünk, vagyis valóban lehetetlen irányított kört bezárni. Ezt az egyszerű gondolatot általánosítja az alábbi definíció.

**3. Definíció.** *Legyen $\overrightarrow{G} = (V,E)$ irányított gráf és $(v_1, v_2, \ldots, v_n)$ a $\overrightarrow{G}$ csúcsainak egy felsorolása. A $(v_1, v_2, \ldots, v_n)$ sorozatot* topologikus rendezésnek *nevezzük, ha $\overrightarrow{G}$ minden $e = \overrightarrow{xy}$ élére $x$ előbb van a sorozatban, mint $y$ (vagyis ha $x = v_i$ és $y = v_j$, akkor $i < j$).*

Például az 1. ábra irányított gráfjának topologikus rendezése az $(s,g,c,a,d,e,b,f)$ sorozat. De természetesen nem minden irányított gráfnak van topologikus rendezése: ha a gráfban van irányított kör, akkor a csúcsok egyetlen sorbaállítása sem lehet jó, mert a kör mentén haladva olyan élet is kell találnunk, ami alacsonyabb sorszámú csúcsba visz minket vissza. Így topologikus rendezése csak aciklikus irányított gráfoknak lehet. Ezért érdekes az alábbi, egyszerű tétel.

**4. Tétel.** *A $\overrightarrow{G}$ irányított gráfnak akkor és csak akkor van topologikus rendezése, ha aciklikus.*

*Bizonyítás:* A feltétel szükségessége magától értetődő (és az imént láttuk be). Az elégségesség bizonyításához először azt mutatjuk meg, hogy minden aciklikus irányított gráf tartalmaz *nyelőt* – vagyis olyan csúcsot, amelyből nem indul ki él. Ehhez vegyünk a gráfban egy $P$ leghosszabb irányított utat és legyen $v$ ennek a végpontja. ($P$ valóban létezik, mert $\overrightarrow{G}$-t véges gráfnak feltételeztük). Ekkor $v$ szükségképpen nyelő: $v$-ből nem vezethet irányított él sem $P$ egy korábbi pontjába (mert akkor irányított kör keletkezne), sem egy $P$-n kívüli pontba (mert akkor $P$-nél hosszabb irányított út keletkezne).

Válasszunk tehát $\overrightarrow{G}$-ben egy nyelőt, legyen ez $v_n$. Most hagyjuk el $\overrightarrow{G}$-ből $v_n$-et (és a rá illeszkedő éleket), a kapott gráf legyen $\overrightarrow{G}'$. Nyilván $G'$ is aciklikus, így ebben is

választhatunk egy nyelőt, ez legyen $v_{n-1}$, stb. Az eljárást hasonlóan folytatva kapjuk (jobbról balra) a $(v_1, v_2, \ldots, v_n)$ sorozatot, ami nyilván topologikus rendezése $\overrightarrow{G}$-nek. □

A topologikus rendezés léte a fő oka annak a fent már említett jelenségnek, hogy számos algoritmikus feladat sokkal hatékonyabban oldható meg irányított gráfokra. Tekintsük például a *legrövidebb út* problémát: ismert, hogy tetszőleges élsúlyozott irányított gráfban hatékony algoritmussal meghatározhatjuk egy adott $s$ csúcsból az összes többibe vezető legrövidebb utak hosszát (illetve magukat az utakat is): ha az élsúlyok nemnegatívak, akkor a Dijkstra algoritmus $c \cdot n^2$, általános élsúlyok esetén, negatív összsúlyú irányított kört nem tartalmazó gráfban pedig a Ford algoritmus $c \cdot m \cdot n$ futásidő alatt oldja meg ezt a feladatot (ahol $n$, illetve $m$ a gráf csúcsainak, illetve éleinek a száma, $c$ pedig egy alkalmas konstans). Nagyon nagy méretű gráfok esetében azonban még ezek a lépésszámok is túl nagynak bizonyulhatnak, ezért hasznos, hogy aciklikus irányított gráf esetében jóval hatékonyabban is dolgozhatunk.

LEGRÖVIDEBB ÚT ACIKLIKUS IRÁNYÍTOTT GRÁFBAN

**Bemenet:** Egy $\overrightarrow{G} = (V, E)$ aciklikus irányított gráf, a csúcsainak egy $(s = v_1, v_2, \ldots, v_n)$ topologikus rendezése és egy $w : E(\overrightarrow{G}) \to \mathbb{R}$ (valós értékű) súlyfüggvény

```
1   t(v_1) ← 0, minden i > 1-re t(v_i) ← ∞
2   ciklus: i fut 2-től n-ig
3       ha létezik olyan e = v_j v_i él, amelyre t(v_j) ≠ ∞, akkor:
4           t(v_i) ← min{t(v_j) + w(e) : e = v_j v_i, t(v_j) ≠ ∞}
5   ciklus vége
```

A fenti, pofonegyszerű eljárás helyesen határozza meg az $s = v_1$ csúcsból az összes többi $v$ csúcs $t(v)$ távolságát. Valóban: ha $j < i$, akkor az $s$-ből $v_j$-be vezető legrövidebb (vagy bármilyen) úton $v_i$ nyilván nem lehet rajta. Ezért az $s$-ből $v_i$-be vezető legrövidebb út (ha ilyen egyáltalán létezik) egy $s$-ből $v_j$-be vezető legrövidebb útnak a $\overrightarrow{v_j v_i}$ éllel való megtoldásából áll valamely $j < i$ értékre. Ha tehát feltételezzük, hogy az eljárás már helyesen meghatározta az $s$-ből az $s = v_1, v_2, \ldots, v_{i-1}$ csúcsokba vezető legrövidebb utak hosszát (és kezdetben az $i = 1$ értékre, vagyis az $s$-re nyilván ez a helyzet), akkor a ciklus magjában számolt minimum ezt $v_i$-re is helyesen teszi meg. Ha viszont minden $\overrightarrow{v_j v_i}$ él esetén $t(v_j) = \infty$ (vagy ha ilyen él egyáltalán nincs), akkor nyilván $v_i$-be sem vezet $s$-ből irányított út, így $t(v_i) = \infty$ szintén helyes.

Érdemes azt is megjegyezni, hogy a Dijkstra és Ford algoritmusnál látotthoz hasonlóan ezt az eljárást is könnyen lehetne úgy módosítani, hogy a kimenetéből ne csak a legrövidebb utak hossza, hanem maguk az utak is kiolvashatók legyenek (de a részleteket itt mellőzzük).

Mi a fenti eljárás futásideje? Ez is nagyban függ attól, hogy a $\overrightarrow{G}$ milyen módon (vagyis milyen adatsruktúrával) van megadva. De ha például minden $v$ csúcsra rendelkezésre áll a $v$-be érkező élek listája, akkor minden él konstansnyi idővel járul hozzá a lépésszámhoz, így az algoritmus futásideje $m$ élű gráfra $c \cdot m$ lesz (ahol $c$ alkalmas konstans és feltételeztük, hogy $\overrightarrow{G}$-nek nincs izolált csúcsa). Ez pedig a $c \cdot m \cdot n$ futásidejű Ford algoritmusnál mindenképp sokkal jobb és ha $m$ nagyságrendje $c \cdot n^2$-nél kisebb, akkor a Dijkstra algoritmusnál is (de rosszabb persze ennél sem lehet, mert $m$ nagyságrendje legföljebb $c \cdot n^2$).

Fontos megemlíteni azt is, hogy egy tetszőleges élsúlyozott irányított gráfban az $s$-ből a többi csúcsba vezető *leghosszabb* irányított út meghatározására nem ismert polinomiális algoritmus (és nagyon valószínű, noha nem bizonyított, hogy ilyen nem is létezik – mert ez a feladat úgynevezett *NP-nehéz* probléma). Ha azonban a gráf aciklikus, akkor a feladat hatékonyan megoldhatóvá válik: ehhez a fenti algoritmust csak annyiban kell módosítani, hogy a 4. sorban szereplő minimumot maximumra cseréljük (és a módszer helyességének a bizonyítása is azonos). Az így módosított eljárásnak a legfontosabb alkalmazása az úgynevezett *PERT módszer* (aminek a részletes leírása megtalálható például Katona Gyula, Recski András és Szabó Csaba *A számítástudomány alapjai* című tankönyvében).

## Topologikus rendezés a DFS algoritmussal

A fenti, aciklikus irányított gráfban legrövidebb (vagy leghosszabb) irányított utakat kereső algoritmus feltételezte, hogy $\overrightarrow{G}$-nek eleve adott egy topologikus rendezése. De mi van akkor, ha csak $\overrightarrow{G}$ adott és a topologikus rendezés meghatározása is a feladat megoldásának a része? A 4. Tétel bizonyításában látott gondolat (vagyis nyelők ismételt keresése és elhagyása a gráfból) elvileg alkalmas a topologikus rendezés algoritmikus meghatározására is, de $c \cdot n^2$ lépésszámú eljárást eredményezne. Ez azért kellemetlen, mert ezzel az imént látott legrövidebb (vagy leghosszabb) utakat kereső algoritmus lépéssszáma is $c \cdot n^2$-re romlana, ha azt a topologikus rendezés meghatározásával kell kezdenünk. Ezért nagyon hasznos, hogy $\overrightarrow{G}$ aciklikussága eldönthető, illetve aciklikus $\overrightarrow{G}$-re a topologikus rendezés meghatározható a DFS algoritmussal már $c \cdot m$ futásidőben is.

**5. Tétel.** *Futtassuk le a DFS algoritmust a $\overrightarrow{G}$ irányított gráfban az $s$ csúcsból indítva. $\overrightarrow{G}$ akkor és csak akkor aciklikus, ha az eljárás során nem keletkezik visszaél és ebben az esetben a befejezési számozás szerinti fordított sorrendben felsorolva $\overrightarrow{G}$ csúcsait topologikus rendezést kapunk.*

*Bizonyítás:* Ha keletkezik visszaél és $e = \overrightarrow{av}$ ilyen, akkor $\overrightarrow{G}$ nyilván tartalmaz irányított kört: az $F$ DFS-erdő tartalmaz $v$-ből $a$-ba irányított utat (mert $e$ visszaél), amit $e$-vel kiegészítve irányított körré zárhatunk. Tegyük fel ezért, hogy nem keletkezik visszaél. Ha megmutatjuk a tétel második állítását, vagyis hogy a csúcsoknak a befejezési számozás szerinti fordított sorrendje topologikus rendezés, akkor ebből nyilván $\overrightarrow{G}$ aciklikussága is következni fog, így a tétel bizonyítása teljes lesz.

Azt kell tehát megmutatnunk, hogy $\overrightarrow{G}$ minden $e = \overrightarrow{av}$ élére $f(v) < f(a)$. Mivel feltettük, hogy visszaél nem keletkezett, ezért $e$ lehet faél, előreél vagy keresztél. A 2. Tétel bizonyításában használt elnevezéseket használva tegyük fel, hogy a DFS eljárás az $e = \overrightarrow{av}$ élen a $T$ időpillanatban próbált továbblépni. Ekkor tehát $T$ az $I(a)$ intervallumban van.

Ha $e$ faél vagy előreél, akkor kezd$(v)$ is $I(a)$-ba tartozik. Mivel a 2. Tétel bizonyításában írtak szerint $I(v)$-ben lényegében egy $v$-ből indított DFS algoritmus zajlik azokon az $x$ csúcsokon, amelyekre kezd$(v)$-ben $d(x) = *$ teljesül, ezért az $e$ él vizsgálata biztosan nem $I(v)$ alatt történt (hiszen $d(a) < d(v)$ miatt kezd$(v)$-ben már $d(a) \neq *$). Vagyis $T$-ben $I(v)$ már véget ért, ezért bef$(v) < T \leq$ bef$(a)$. Ebből tehát $f(v) < f(a)$ is következik.

Ha viszont $e$ keresztél, akkor a 2. Tétel szerint a $T$ pillanatban $f(v)$ már kapott egy ($*$-tól különböző) értéket. Viszont $T$-ben $a$ az éppen aktív csúcs volt, ezért ekkor még

$f(a) = *$ teljesült. Mivel az eljárás mindig egyre nagyobb befejezési számokat ad a csúcsoknak, az algoritmus leállásakor $f(v) < f(a)$ valóban igaz. □

A fenti tétel szerint tehát a DFS algoritmus futásának eredményéből könnyen eldönthető, hogy $\overrightarrow{G}$ aciklikus-e és ha igen, megkapható $\overrightarrow{G}$ egy topologikus rendezése: $\overrightarrow{G}$ akkor és csak akkor aciklikus, ha minden $e = \overrightarrow{uv}$ élre $f(u) > f(v)$ és ha ez igaz, akkor a csúcsok $f(v)$ értékek szerinti csökkenő sorrendje topologikus rendezést ad. (Ezt a rendezést pedig nem kell külön elvégezni, az eljárás futása közben ez is eltárolható.) Ebből pedig következik, hogy aciklikus irányított gráfban akkor is meghatározható az $s$-ből a többi csúcsba vezető legrövidebb (vagy leghosszabb) irányított út $c \cdot m$ futásidőben, ha $\overrightarrow{G}$ egy topologikus rendezése nem része a bemenetnek. A DFS algoritmusnak emellett számos más alkalmazása is ismert, de ezekre ebben a rövid bevezetőben nem térünk ki.