Mélységi bejárás

Csima Judit
BME SZIT
csima@cs.bme.hu

2019. november 13.

A mélységi bejárás (Depth-First-Search, DFS) elve

Hasonlóan a szélességi bejáráshoz, a mélységi bejárás célja is az, hogy az (irányított vagy irányítatlan) gráf csúcsait egy kezdőcsúcsból kiindulva bejárja, de ehhez egészen más stratégiát követ, mint a szélességi bejárás. A mélységi bejárás olyan, mint egy tank, megy előre, amíg csak tud, kicsit pontosabban:

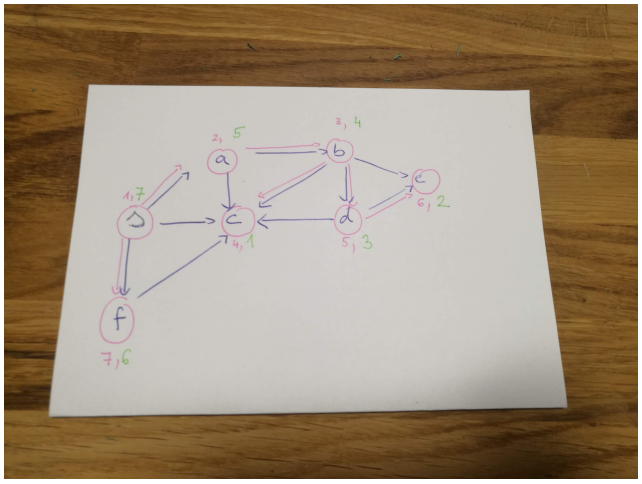
- elindul a kezdőcsúcsból egy irányba (ha a gráf irányított, akkor természetesen az élek irányítását is figyelembe véve) és megy előre addig, amíg tud, azaz amíg az aktuális helyzetéből, a csúcsból, ahol éppen van, vezet ki olyan él, ami még nem bejárt csúcsba vezet,

- ha pedig elakad, mert vagy nincs kimenő él vagy minden kimenő él már bejárt csúcsba vezet, akkor egyet visszalép, azon az élen, ahonnan a mostani helyzetébe érkezett, majd próbál új irányt találni.

A mélységi bejárás annyiban hasonlít a szélességi bejárásra, hogy az a bejárás sem megy soha oda, ahol már volt, de a mélységi bejárás olyan mélyen megy be a gráfba, ahogy csak tud, miközben lehet, hogy a kezdőcsúcs szomszédai közül még sokat nem látogatott meg.

Ennél a bejárásnál nem lesznek szintek, mint a szélességi bejárásnál voltak, a mélységi bejárás nem hullámként terjed, azt azonban célszerű lesz nyilvántartani, hogy a csúcsokat milyen sorrendben látogatjuk meg (ez lesz a bejárási sorrend) és még inkább hasznos lesz (ezt a jövő órán látjuk majd) nyilvántartani, hogy milyen sorrendben fordulunk vissza a csúcsokból: ez lesz a befejezési szám. Az a csúcs, aminek a befejezési száma 1, az az a csúcs, ahonnan először fordultunk vissza.

Az alábbi (irányított) gráfon az s csúcsból futtatva a mélységi bejárást a rózsaszín élekkel fedezem fel az új csúcsokat, a rózsaszín számok mutatják a bejárás, a zöld számok pedig a befejezés sorrendjét, a befejezési számokat.



Megjegyzések, állítások a mélységi bejárásról

1. A fenti példán irányított gráfon futott az eljárás. A mélységi bejárás alkalmazható irányítatlan esetben is például úgy, hogy úgy tekintjük, hogy egy u és v csúcs közti irányítatlan él azt jelenti, hogy mind u-ból v-be, mind v-ből u-ba van irányított él (azaz az élen mindkét irányban lehet haladni).

2. Ha a G gráf irányítatlan, akkor a felfedező élek fát alkotnak, mert

   - a felfedező élek mindig új csúcsba mennek, sose keletkezik kör
   - az új felfedező él mindig kapcsolódik a korábbi felfedező élek egyikéhez

3. Ha G irányítatlan és összefüggő, akkor a felfedező élek feszítőfát adnak G-ben (ez a bejáráshoz tartozó DFS-fa), mert ha egy csúcs elérhető s-ből, akkor a bejárás előbb-utóbb el fog ide érni.

4. A mélységi bejárás segítségével eldönthető egy irányítatlan gráfról, hogy összefüggő-e: ha a bejárást lefuttatva vannak nem bejárt csúcsok, akkor a gráf nem volt összefüggő, különben meg igen.

5. Meg lehet találni azokat a csúcsokat, akik a kezdőcsúcsból elérhetők úton (irányított esetben irányított úton, irányítatlan esetben irányítatlan úton): azok a csúcsok elérhetők s-ből akik az eljárás végén be vannak járva.

6. Ha az összes csúcs be van járva, akkor a felfedező élek minden csúcshoz adnak egy egyértelmű utat s-ből.

7. Ha s-ből minden csúcs elérhető úton, akkor a mélységi bejárás a gráf minden csúcsát pontosan egyszer járja be.

8. Ha vannak olyan csúcsok, amik nem elérhetők el s-ből úton, de mégis be akarjuk járni az összes csúcsot, akkor az s-ből indított bejárás lefutása után indítunk még egy mélységi bejárást egy még bejáratlan csúcsból és ezt ismételgetjük addig, amíg minden csúcsot meg nem látogatunk.

A mélységi bejárás pszeudokódja

Az algoritmus futása során a következőket kell nyilvántartanunk:

- Mely csúcsokat jártam már be? (Ezt azért kell tudni, hogy egy él vizsgálatánál lássam, hogy új csúcsba vezet vagy sem.) Ehhez, a szélességi bejárásnál látott módon, egy n méretű, 1-től n-ig indexelt *bejárva* nevű Boole-értékű tömböt használunk (n a csúcsok száma, a csúcsokról feltesszük, hogy $1, 2, 3 \ldots, n$ címkéjűek), ahol $bejárva[v] = 1$, ha v-t már bejártam, különben $bejárva[v] = 0$.

- Melyik bejárt csúcsot honnan értem el, melyik felfedező élen át? Itt is ugyanazt tesszük, mint a szélességi bejárásnál, a felfedező éleket egy n méretű, 1-től n-ig indexelt *honnan* nevű tömb segítségével tároljuk, amiben $honnan[v] = u$, ha v-t már bejártam az u-ból v-be vezető felfedező éllel, ha pedig v még nincs bejárva akkor $honnan[v] = \star$.

A mélységi bejárás pszeudokódjában lesz egy fő eljárás, ami beállítja a *honnan* és *bejárva* tömböket az elején (annak megfelelően, hogy ekkor csak s van bejárva, saját jogán) és lesz egy függvény, amit többször meghívunk majd (de minden csúcsra maximum egyszer), ez felel meg annak, hogy egy adott csúcsból az összes lehetséges kimenő irányt megvizsgáljuk és ha lehet, akkor továbblépünk arra.
A fő eljárás neve *mélységi_bejárás(G,s)*, a meghívott függvény pedig $DFS(G, v)$, ahol G a gráf, amiben az eljárás fut, a gráf az A-val jelölt szomszédossági mátrix segítségével adott, s a kezdőcsúcs, ahonnan a bejárás indul, v pedig G egy tetszőleges csúcsa.

```
mélységi_bejárás(G,s)
---------------------

bejárva[v] = 1, ha v =s, egyébként bejárva[v] = 0
honnan[v] = v, ha v =s, különben honnan[v] = *
DFS(G,s)



DFS(G,v)
---------

ciklus w = 1-tól n-ig   // végignézem v összes potenciális szomszédját
        ha A[v,w] == 1 és bejárva[w] ==0:    // ha w-be van él és w-t még nem láttuk
            bejárva[w] := 1
            honnan[w] := v
            DFS(G,w)    // megyek tovább w-ból
ciklus vége
```

Nézzük meg a fenti példán, hogy hogyan fut le ez a pszeudokód.
Az elején a fő eljárás beállítja a *bejárva* és *honnan* tömböket és elindítja a $DFS(G, s)$ eljárást.

bejárva:

s	a	b	c	d	e	f
1	0	0	0	0	0	0

honnan:

s	a	b	c	d	e	f
s	*	*	*	*	*	*

A $DFS(G, s)$ eljárásban elindul egy ciklus, ami végigmegy (majd) az A szomszédossági mátrix s-hez tartozó során és azt ellenőrzi minden $A[s, w]$ bejegyzésnél, hogy az 1-e (azaz van-e él s-ből az oszlophoz tartozó w csúcsba) és hogy a w csúcs bejáratlan-e. Ez az ellenőrzés először $w = s$ -re fut le (tegyük fel, hogy a csúcsok s, a, b, c, d, e, f sorrendben vannak a mátrixban), de $A[s, s] = 0$, tehát nincs tennivaló. Ezután azonban $A[s, a] = 1$ és *bejárva[a] = 0*, vagyis ekkor *bejárva[a] = 1* és *honnan[a] =s* lesz és elindul $DFS(G, a)$. Ekkor még $DFS(G, s)$ nem fejeződött be, csak a $DFS(G, s)$ futáson belülről elindult egy másik DFS futás, a $DFS(G, a)$. Majd amikor ez a $DFS(G, a)$ be fog fejeződni, akkor fut tovább a $DFS(G, s)$-en belüli ciklus. Ebben pillanatban a két tömb így néz ki:

bejárva:

s	a	b	c	d	e	f
1	1	0	0	0	0	0

honnan:

s	a	b	c	d	e	f
s	s	*	*	*	*	*

Most a $DFS(G, a)$ eljárásban is elindul egy ciklus, ami végigmegy (majd) az A szomszédossági mátrix a-hoz tartozó során és azt ellenőrzi minden $A[a, w]$ bejegyzésnél, hogy az 1-e (azaz van-e él a-ból az oszlophoz tartozó w csúcsba) és hogy a w csúcs bejáratlan-e. Ez az ellenőrzés először $w = s$ és $w = a$-ra fut le, de $A[a, s] = A[a, a] = 0$, nincs tennivaló. Ezután azonban $A[a, b] = 1$ és *bejárva[b] = 0*, vagyis ekkor *bejárva[b] = 1* és *honnan[b] =a* lesz és elindul $DFS(G, b)$. Ekkor még sem a $DFS(G, s)$ nem fejeződött be, sem a $DFS(G, a)$, csak a $DFS(G, s)$ futáson belülről elindított $DFS(G, a)$-ból elindított $DFS(G, b)$ futás kezdődik el. Ebben pillanatban a két tömb így néz ki:

bejárva:

s	a	b	c	d	e	f
1	1	1	0	0	0	0

honnan:

s	a	b	c	d	e	f
s	s	a	*	*	*	*

Most a $DFS(G, b)$ eljárásban is elindul egy ciklus, ami végigmegy (majd) az A szomszédossági mátrix b-hez tartozó során és azt ellenőrzi minden $A[b, w]$ bejegyzésnél, hogy az 1-e (azaz van-e él b-ből az oszlophoz tartozó w csúcsba) és hogy a w csúcs bejáratlan-e. Ez az ellenőrzés először $w = s, a, b$ értékekre fut le, de itt nincs tennivaló, mert ide nem vezet él b-ből. Ezután azonban $A[b, c] = 1$ és *bejárva[c] = 0*, vagyis ekkor *bejárva[c] = 1* és *honnan[c] =b* lesz és elindul $DFS(G, c)$. Ekkor még sem a $DFS(G, s)$ nem fejeződött be, sem az ebből indított $DFS(G, a)$, sem az ebből indított $DFS(G, b)$, csak elindul $DFS(G, c)$. Ebben pillanatban a két tömb így néz ki:

bejárva:

s	a	b	c	d	e	f
1	1	1	1	0	0	0

honnan:

s	a	b	c	d	e	f
s	s	a	b	*	*	*

Most a $DFS(G, c)$ eljárásban is elindul egy ciklus, ami végigmegy az A szomszédossági mátrix c-hez tartozó során, de mivel itt minden bejegyzés 0, ezért itt nincs tennivaló, a $DFS(G, c)$ úgy fejeződik be, hogy közben sem a *bejárva*, sem a *honnan* tömb nem változott. Ekkor $DFS(G, c)$ ugyan befejeződött, de még fut $DFS(G, s)$, a belőle indított $DFS(G, a)$ és az ebből indított $DFS(G, b)$. Ez utóbbi (miután $DFS(G, c)$ befejeződik) megy tovább a szomszédossági mátrix b-hez tartozó során és mivel $A[b, d] = 1$ és *bejárva[d] = 0* ezért ekkor *bejárva[d] = 1* és *honnan[d] =b* lesz és elindul $DFS(G, d)$. Ebben pillanatban a két tömb így néz ki:

bejárva:

s	a	b	c	d	e	f
1	1	1	1	1	0	0

honnan:

s	a	b	c	d	e	f
s	s	a	b	b	*	*

Most a $DFS(G, d)$ eljárásban is elindul egy ciklus, ami végigmegy (majd) az A szomszédossági mátrix d-hez tartozó során, először $w = s, a, b$ értékeket nézzük, de itt nincs tennivaló, mert ide nem vezet él d-ből, utána $A[d, c] = 1$ ugyan igaz, de *bejárva[c] =1*, vagyis itt sincs tennivaló. Ezután $w = d$-be nem vezet él, de $A[d, e] = 1$ és *bejárva[e] = 0*, vagyis ekkor *bejárva[e] = 1* és *honnan[e] =d* lesz és elindul $DFS(G, e)$.

Ekkor még fut $DFS(G, s)$, a belőle indított $DFS(G, a)$, az ebből indított $DFS(G, b)$, az ebből indtott $DFS(G, d)$ és most indul $DFS(G, e)$. Ebben pillanatban a két tömb így néz ki:

bejárva:

s	a	b	c	d	e	f
1	1	1	1	1	1	0

honnan:

s	a	b	c	d	e	f
s	s	a	b	b	d	*

Most a $DFS(G, e)$ eljárásban is elindul egy ciklus, ami végigmegy az A szomszédossági mátrix e-hez tartozó során, de mivel itt minden bejegyzés 0, ezért itt nincs tennivaló, a $DFS(G, e)$ úgy fejeződik be, hogy közben sem a *bejárva*, sem a *honnan* tömb nem változott. Miután $DFS(G, e)$ befejeződött, az őt elindító $DFS(G, d)$ még fut tovább a $w = f$ potenciális szomszéd vizsgálatával, de $A[d, f] = 0$, vagyis nincs tennivaló, a $DFS(G, d)$ futás is befejeződött, a két tömböt nem változtatjuk. Miután a $DFS(G, d)$ befejeződött, az őt indító $DFS(G, b)$ fut tovább, de b-nek nincs már bejáratlan szomszédja, ezért $DFS(G, b)$ is befejeződik, de az őt indító $DFS(G, a)$ fut tovább. Azonban a-nak sincs már nem bejárt szomszédja, így $DFS(G, a)$ is végetér és már csak a kezdő $DFS(G, s)$ fut. Mindeközben a két tömb nem változott semmit. Ekkor a $DFS(G, s)$-en belül futó ciklus tovább vizsgálja a szomszédossági mátrix s-hez tartozó sorában az a utáni elemeket (ez volt az utolsó, amit már megnézett), de $w = b, c, d, e$ esetén a bejegyzés vagy 0 vagy ha 1, akkor az oszlophoz tartozó csúcs már be van járva. Amikor azonban $w = f$-hez érünk, akkor $A[s, f] = 1$ és *bejárva[f] =0*, így *bejárva[f] = 1* és *honnan[f] =s* lesz és elindul $DFS(G, f)$.
Ebben pillanatban a két tömb így néz ki:

bejárva:

s	a	b	c	d	e	f
1	1	1	1	1	1	1

honnan:

s	a	b	c	d	e	f
s	s	a	b	b	d	s

Most a $DFS(G, f)$ eljárásban is elindul egy ciklus, ami végigmegy az A szomszédossági mátrix f-hez tartozó során, de itt egy kivétellel minden bejegyzés 0 és noha $A[f, c] = 1$, de *bejárva[c] =1*, vagyis nincs tennivaló, a $DFS(G, f)$ úgy fejeződik be, hogy közben sem a *bejárva*, sem a *honnan* tömb nem változott. Mivel $DFS(G, f)$ befejezésével $DFS(G, s)$ is befejeződött, így az egész eljárásnak vége van. Az összes csúcsot bejártuk, a *honnan* tömb segítségével találtunk utakat az összes csúcsba:
a-ba: s,a
b-be: s,a,b
c-be: s, a, b, c
d-be: s, a, b, d
e-be: s, a, b, d, e
f-be: s, f

A mélységi bejárás lépésszáma

A lépésszámot két részletben számoljuk ki, külön a *mélységi_bejárás(G,s)* kódra és külön az egyes $DFS(G, v)$ kódokra. A fő *mélységi_bejárás(G,s)* kód lépésszáma $O(n)$, mert mindkét tömböt $O(n)$ lépés inicializálni és 1 lépés a $DFS(G, s)$ függvényt meghívni (a $DFS(G, s)$-en belüli lépéseket külön számoljuk meg). Egy $DFS(G, v)$-ben a ciklus n-szer fut le, egy lefutás konstans lépés, azaz egy $DFS(G, v)$ lépésszáma $O(n)$. Mivel minden v csúcsra legfeljebb egyszer indul el $DFS(G, v)$, ezért az összes $DFS(G, v)$ futás összes lépésszéma $O(n^2)$, vagyis a teljes kód $O(n) + O(n^2)$, azaz $O(n^2)$.

Mire nem jó a mélységi bejárás?

Láttuk, hogy sok mindenre jó a mélységi bejárás, most beszéljünk egy kicsit arról, hogy mire nem alkalmas. Ellentétben a szélességi bejárással nem lehet vele legrövidebb utakat megtalálni a kezdőcsúcsból a többi csúcsba, hiszen nem hullámokban, szintenként haladunk, előfordulhat, hogy közeli csúcsokat csak sokára, hosszú út bejárása után látogatunk meg.
Nem jó a mélységi bejárás a leghosszabb út megkeresésére sem, mert előfordulhat, hogy egy csúcsba vezet rövid és hosszú út is s-ből a gráfban és lehetséges, hogy a mélységi bejárás a rövid utat találja meg és nem a hosszút.

A pszeudokód mégegyszer, befejezési számokkal

Jó lesz viszont a mélységi bejárás valami másra (ezt majd a jövő órán nézzük), amihez a csúcsok befejezési számozására lesz szükségünk. Nézzük meg most, hogy hogyan lehet a befejezési számok meghatározását beilleszteni a már ismert kódba.

Ehhez egyrészt szükségünk lesz egy *befejezési_számláló* nevű változóra, ebben tároljuk, hogy mi a következő befejezési szám, amit ki fogunk adni (az elején az 1-es), másrészt szükségünk lesz egy *bsz* nevű tömbre, amiben a már kiadott befejezési számokat tároljuk (ez egy n hosszú, a csúcsokkal indexelt tömb lesz, a még nem befejezett csúcsokra $bsz[v] = *$).

Azt kell még kitalálnunk, hogy hol van az apont, amikor a befejezési számot kiadjuk. Ez akkor történik meg, amikor egy v csúcsról kiderül, hogy már nincs más szomszédja, ahova el tudnánk menni, vagyis a $DFS(G, v)$-beli ciklus után.

A fentiek alapján a kód így néz ki:

```
mélységi_bejárás(G,s)
---------------------

bejárva[v] = 1, ha v =s, egyébként bejárva[v] = 0
honnan[v] = v, ha v =s, különben honnan[v] = *
befejezési_számláló: = 1
bsz[v] = * minden v-re
DFS(G,s)
```

```
DFS(G,v)
---------

ciklus w = 1-tól n-ig   // végignézem v összes potenciális szomszédját
        ha A[v,w] == 1 és bejárva[w] ==0:    // ha w-be van él és w-t még nem láttuk
            bejárva[w] := 1
            honnan[w] := v
            DFS(G,w)    // megyek tovább w-ból
ciklus vége
bsz[v] := befejezési_számláló
befejezési_számláló += 1
```

Vegyük észre, hogy a betoldott extra sorok nem befolyásolják a lépésszámot, ennek a verziónak, ami kiszámolja a befejezési számokat is $O(n^2)$ a lépésszáma.