Feszítőfa
Szélességi bejárás

Csima Judit
BME SZIT
csima@cs.bme.hu

2019. november 6.

Feszítőfa

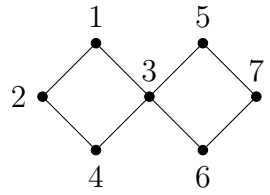
Először néhány további definíciót vezetünk be, melyekre a tanult algoritmusok során lesz szükségünk.

Séta, út, kör

Egy G irányítatlan gráfban **sétának** nevezzük csúcsok egy $v_1v_2 \dots v_s$ sorrendjét, ha az egymást követő csúcsok (v_i és v_{i+1}) között van él a gráfban, vagyis ha végig lehet sétálni a gráf éleit használva a $v_1, v_2, \dots, v_s$ csúcsokon ebben a sorrendben. A séta nem biztos, hogy a gráf minden csúcsát tartalmazza és egy csúcsot többször is érinthetünk.

Az alábbi gráfban például séta a $2, 1, 3, 6, 7, 5$ sorrend, de séták a $2, 1, 3, 6, 7, 5, 3, 4$ és $2, 1, 3, 6, 7, 5, 3$ sorrendek is.

G_1:



Egy G irányítatlan gráfban **útnak** nevezzük azt a sétát, amiben nincs ismétlődő csúcs, például az előbbi gráfban út a $2, 1, 3, 6, 7, 5$ séta, de nem út a $2, 1, 3, 6, 7, 5, 3, 4$ és a $2, 1, 3, 6, 7, 5, 3$ séta.

Egy G irányítatlan gráfban **körnek** nevezzük azt a sétát, amiben nincs ismétlődő csúcs (tehát egy út), ahol még az is teljesül, hogy az utolsó és az első csúcs között is van él. Például az előbbi gráfban kör a $2, 1, 3, 4$ séta, de nem kör a $2, 1, 3, 6, 7, 5, 3, 4$ és a $2, 1, 3, 5$.

Hasonló fogalmak vannak irányított gráfokra is:

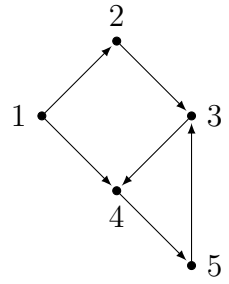
Egy G irányított gráfban **irányított sétának** nevezzük csúcsok egy $v_1v_2 \dots v_s$ sorrendjét, ha az egymást követő csúcsokra igaz, hogy van él v_i-ből és v_{i+1}-be, vagyis ha az élek irányítását is figyelembe véve végig lehet sétálni a gráf éleit használva a $v_1, v_2, \dots, v_s$ csúcsokon ebben a sorrendben. A séta nem biztos, hogy a gráf minden csúcsát tartalmazza és egy csúcsot többször

is érinthetünk.

Egy G irányított gráfban **irányított útnak** nevezzük azt az irányított sétát, amiben nincs ismétlődő csúcs.

Egy G irányított gráfban **irányított körnek** nevezzük azt az irányított sétát, amiben nincs ismétlődő csúcs (tehát egy irányított út), ahol még az is teljesül, hogy az utolsó csúcsból van él az első csúcsba.

Az alábbi gráfban irányított séták például az $1, 2, 3, 4, 5, 3$ és $1, 4, 5, 3$, de ezek közül csak az $1, 4, 5, 3$ sorrend irányított út. Irányított kör például az $5, 3, 4$ sorrend.

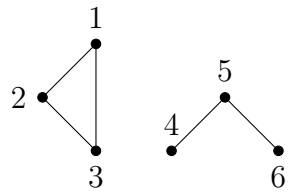


Fák, feszítőfák

Egy G irányítatlan gráfot **összefüggőnek** nevezünk, ha igaz, hogy mindegyik csúcsából mindegyik másik csúcsába vezet út.

Összefüggő gráf például az előbb látott G_1 gráf, de nem összefüggő az alábbi gráf:

G_2:



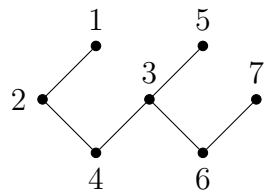
Egy nem összefüggő irányítatlan gráf nem bővíthető összefüggő részeit **komponenseknek** nevezzük. (A nem bővíthető azt jelenti, hogy már nincs olyan csúcsa a gráfnak, amit még az összefüggőség megtartása mellett bele tudnánk venni a komponensbe.)

Például az előbbi G_2 gráfnak két komponense van, az $1, 2, 3$ csúcsokból és köztük menő élekből álló rész és a $4, 5, 6$ csúcsokból és köztük menő két élből álló rész. Nem komponense a gráfnak az $1, 2$ csúcsokból és a köztük menő egy darab élből álló rész, mert ehhez még hozzá lehet venni a 3-as csúcsot is.

Nagyon fontos definíció a következő:

Egy G irányítatlan gráfot akkor nevezünk **fának**, ha összefüggő és körmentes (azaz nincsen benne kör).

A már látott G_1 és G_2 gráfok egyike sem fa (noha G_1 összefüggő, de van benne kör, G_2 pedig még csak nem is összefüggő és még kör is van benne). Fa viszont a következő gráf:



1. állítás
Ha G egy legalább két csúcsból álló fa, akkor G-ben biztosan van legalább kettő elsőfokú csúcs.

Bizonyítás
Vegyük a G-ben található leghosszabb utat (ha több ilyen is van, akkor egy ilyet), legyen ez $v_1v_2 \dots v_s$. Ennek az útnak a két végpontja, v_1 és v_s biztosan elsőfokú lesz, hiszen egy él illeszkedik rájuk (v_1-re a v_1v_2 él, v_s-re pedig a $v_{s-1}v_s$ él) és több él nem vezethet ki a végpontokból, mert

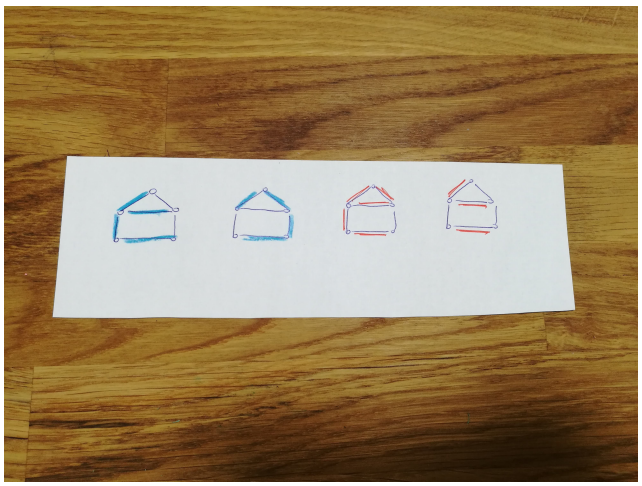
- v_1-ből nem mehet él v_2-n kívül másik útbeli csúcsba, mert akkor kör lenne, ami nem lehet, mert G egy fa (hasonló igaz v_s-re is)
- ha v_1-ből menne él egy úton kívüli pontba, akkor ezzel az éllel kiegészítve az eddigi utat hosszabb utat kapnánk, ami nem lehetséges, mert feltevésünk szerint ez volt a leghosszabb út a gráfban.

2. állítás
Ha G egy $n \geq 2$ csúcsból álló fa, akkor G-ben $n-1$ él van.

Bizonyítás
Teljes indukcióval:
$n = 2$-re az állítás igaz, mert ekkor a fa két csúcsból és egy darab, a két csúcs közt menő élből áll.
Tegyük fel, hogy igaz az állítás n-re és lássuk be $n+1$-re, azaz lássuk be, hogy egy $n+1$ csúcsú F fának n éle van.
Az előző állítás szerint az F fában van elsőfokú csúcs, hagyjuk el ezt a csúcsot és a rá illeszkedő élet az F fából. Amit így kapunk az is fa lesz (mert kör nem keletkezhetett ettől és mivel elsőfokú csúcsot hagytunk el, a gráf összefüggő maradt), aminek n csúcsa van. Az indukciós feltevés szerint az új fának $n-1$ éle van, tehát F-nek n éle volt.

Egy G irányítatlan gráf **feszítőfája** egy olyan F gráf melynek csúcsai megegyeznek G csúcsaival, élei G élei közül kerülnek ki és F fa (azaz körmentes és összefüggő).

Az alábbi ábrán egy öt csúcsú gráf látható, négyszer lerajzolva. A kék élek az első két rajzon feszítőfát alkotnak a gráfban, de a piros élek nem (a 3. rajzon azért nem mert a piros élek gráfja nem körmentes, a 4. rajzon pedig nem összefüggő).



Az F feszítőfa a G gráf "csontváza", csak F éleit használva el lehet jutni G minden csúcsából minden másik csúcsba (mert F összefüggő és G minden csúcsát tartalmazza), ráadásul egyértelműen, azaz csak egy út van F-ben bármely két csúcs között (mert két út együtt már egy kört jelentene, de az F-ben nem lehet, mert F egy fa). F ezen utóbbi jó tulajdonsága miatt kommunikációs hálózatokban használnak feszítőfákat az üzenetek küldésére, mert például ha csak az F feszítőfa éleit használjuk az üzenetek továbbítása során, akkor biztosan nem fognak az üzenetek körbe-körbe járni. A feszítőfáknak további számos alkalmazása van, jó lenne tehát tudni, hogy mely gráfokban van feszítőfa és amikben van ott jó lenne valami algoritmust találni egy feszítőfa megkeresésére.
Az világos, hogy ha G-ben van feszítőfa, akkor G összefüggő (mert már F élein át is van út bárhonnan bárhova), most pedig mutatni fogunk egy olyan algoritmust, a szélességi bejárást, ami összefüggő gráfokban talál is egy feszítőfát.

Szélességi bejárás (BFS, Breadth-First-Search)

A szélességi bejárás számos dologra jó lesz:

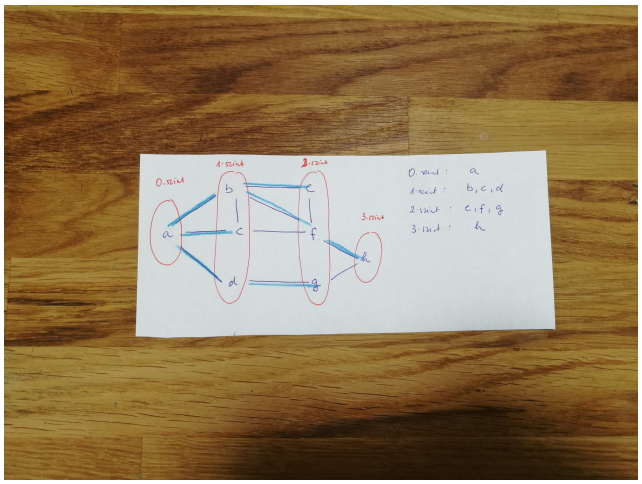
1. a segítségével be tudjuk járni a gráf csúcsait úgy, hogy minden csúcsot csak egyszer érintünk
2. ha a gráf, amiben futtatjuk összefüggő, akkor az algoritmus ad egy feszítőfát a gráfban
3. a segítségével el tudjuk dönteni egy irányítatlan gráfról, hogy összefüggő-e
4. és még valami másra is jó lesz :)

A szélességi bejárás elve a következő:

- A G gráf egy s csúcsából indulunk, ez a 0. szint
- Az első fázisban meglátogatjuk az s összes szomszédját, ezek a csúcsok alkotják az 1. szintet.
- A második fázisban meglátogatjuk az első fázisban felkeresett csúcsok összes olyan szomszédját, amit még nem látogattunk meg. Ezek a csúcsok leszek a 2. szinten, ők az s szomszédainak szomszédai. A 2. szint csúcsait úgy látogatjuk meg, hogy ha egy u_1 csúcsot előbb láttunk az 1. fázisban, mint a szintén az 1. fázisban meglátogatott u_2 csúcsot, akkor u_1 szomszédait előbb nézzük meg, mint u_2 szomszédait.
- A 3. fázisban meglátogatjuk a 2. fázisban felkeresett csúcsok összes olyan szomszédját, amit még nem látogattunk meg. Ezek a csúcsok leszek a 3. szinten, ők az s szomszédainak a szomszédainak a szomszédai. A 3. szint csúcsait úgy látogatjuk meg, hogy ha egy u_1 csúcsot előbb láttunk a 2. fázisban, mint a szintén a 2. fázisban meglátogatott u_2 csúcsot, akkor u_1 szomszédait előbb nézzük meg, mint u_2 szomszédait.
- Általában is a k. fázisban meglátogatjuk az előző fázisban felkeresett csúcsok összes olyan szomszédját, amit még nem látogattunk meg, úgy, hogy ha egy u_1 csúcsot előbb láttunk az előző fázisban, mint a szintén az előző fázisban meglátogatott u_2 csúcsot, akkor u_1 szomszédait előbb nézzük meg, mint u_2 szomszédait.

- Ezt addig csináljuk, amíg az utoljára bejárt szinten levő csúcsoknak vannak olyan szomszédai, ahol még nem jártunk.

Az alábbi ábra azt mutatja, hogy ezzel a stratégiával hogyan járjuk be egy gráf éleit. A kékkel jelölt élek a "felfedező" élek, ezekkel jutunk új csúcsokba, a pirossal jelölt halmazok az egyes szintek.



Ebben a gráfban az $a = s$ kezdőcsúcsból indulunk és az első fázisban meglátogatjuk az a csúcs összes szomszédját b, c, d sorrendben az ab, ac, ad éleken át.
A második fázisban sorban megvizsgálom az 1. fázisban megtalált b, c, d csúcsokból kiinduló éleket: először a b csúcs ba, bc, be, bf éleit nézem meg, ezekből a ba, bc nem vezetnek még nem látott csúcsba, de a be és bf élekkel megtalálom az e és f csúcsokat, ezek bekerülnek a 2. szintre. Nem végeztünk még azonban az 1. szint összes csúcsának szomszédaival, még hátravannak a c és d csúcsok szomszédai. A c csúcsra illeszkedő ca, cb, cf élek egyike sem vezet még nem látott csúcsba, a d csúcsnál a da, df élek szintén nem, de a dg él felfedezi a g csúcsot. Így a 2. szintre az e, f, g csúcsok kerültek, az ezekből kilépő éleket fogjuk a 3. fázisban végignézni, de az összes ilyen él közül csak az fh vezet új csúcsba, a 3. szintre csak a h csúcs került be. A 4. fázisban a h csúcs éleit nézzük meg, de ezek egyike sem vezet új helyre, az algoritmus leáll.

Mielőtt a pszeudokódot is felírnánk vegyük észre a következőket:

1. A felfedező élek fát alkotnak.
   Ez azért igaz, mert az új élek mindig új csúcsba vezetnek (sose lépünk vissza, nem keletkezik kör) és az új felfedező élek mindig egy korábbihoz illeszkednek, a gráf ágról ágra bővül.

2. Ha G összefüggő, akkor minden csúcsa elérhető s-ből ezért az algoritmus is előbb-utább el fog érni minden csúcsot, vagyis a felfedező élek fája minden csúcsát tartalmazza a G gráfnak, vagyis ez egy feszítőfa lesz.

BFS 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.)

- Melyik bejárt csúcsot honnan értem el, melyik felfedező élen át? (Hogy tudjam, hogy mely élek alkotják a feszítőfa éleit a végén.)

- Mik azok a csúcsok, akiket már bejártam, de még nem néztem át az éleiket új szomszédok után kutatva?

Ezekre a nyilvántartandó dolgokra a következő megoldásokat fogjuk használni:

- Lesz egy n méretű, 1-től n-ig indexelt *bejárva* nevű Boole-értékű tömbünk (n a csúcsok száma, a csúcsokról feltesszük, hogy $1, 2, 3 \ldots, n$ címkéjűek), ahol $bejárva[v] = 1$, ha v-t már bejártam, különben $bejárva[v] = 0$.

- Lesz egy n méretű, 1-től n-ig indexelt *honnan* nevű tömb, amiben $honnan[v] = u$, ha v-t már bejártam az u-ból v-be vezető felfedező éllel, ha pedig v még nincs bejárva akkor $honnan[v] = \star$.

- Azokat a csúcsokat, amiket már bejártunk, de amiknek a szomszédait még nem láttuk egy Q listában tároljuk úgy, hogy a listába a végére rakjuk be az új csúcsot, amit éppen bejárunk és az elejéről vesszük azt, aminek a szomszédait meg akarjuk nézni. Ez egy úgy nevezett FIFO (First In First Out) lista, más néven egy sor, ez biztosítja, hogy ha egy csúcsot előbb járunk be egy másiknál, akkor a szomszédait is előbb fogjuk megnézni.

Most már készen állunk a szélességi bejárás pszeudokódjára. Ebben A jelöli a G gráf szomszédossági mátrixát és s jelöli a kezdőcsúcsot, ahonnan a bejárás indul.

```
bejárva[v] = 1, ha v =s, egyébként bejárva[v] = 0
Q = {s}
honnan[v] = v, ha v =s, különben honnan[v] = *

ciklus amíg Q-ban van csúcs:    // még van, akinek nem néztem a szomszédait
    v := Q első csúcsa
    vegyük ki v-t Q-ból
    ciklus w = 1-től n-ig       // végignézem az összes potenciális szomszédot
        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
            w-t Q végére rakom
    ciklus vége
ciklus vége
```

Nézzük végig, hogy hogyan változik a *bejárva* tömb, a *honnan* tömb és a Q sor a szélességi bejárást futtatva a korábbi példán. Tegyük fel, hogy a gráf csúcsai $a = 1, b = 2, c = 3, d = 4, e = 5, f = 6, g = 7, h = 8$ címkékkel címzettek, a szomszédossági mátrix elemeire $A[a, b]$ alakban fogunk hivatkozni ($A[1, 2]$ helyett).

Az algoritmus kezdetekor:
bejárva:

a	b	c	d	e	f	g	h
1	0	0	0	0	0	0	0

$Q = \{a\}$

honnan:

a	b	c	d	e	f	g	h
a	*	*	*	*	*	*	*

Először az a csúcs kerül a kódbeli v csúcs szerepébe. Végigmegyünk a szomszédossági mátrixban az a csúcs során, de $A[a, a] = 0$ miatt az első bejegyzésnél nem kell semmit tennünk. Viszont $A[a, b] = 1$ és $bejárva[b] = 0$ miatt $bejárva[b] = 1$ és $honnan[b] = a$ lesz, továbbá b bekerül a Q sor végére. Hasonló dolog történik $A[a, c] = 1$ és $bejárva[c] = 0$ miatt, illetve $A[a, d] = 1$ és $bejárva[d] = 0$ miatt is, a többi csúcsba pedig nem vezet él a-ból, így azt kapjuk (mire a külső ciklus $v = a$ választással lefut):
bejárva:

a	b	c	d	e	f	g	h
1	1	1	1	0	0	0	0

$Q = \{b, c, d\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	*	*	*	*

Most a b csúcs kerül a kódbeli v csúcs szerepébe. Végigmegyünk a szomszédossági mátrixban az b csúcs során, itt csak $A[b, a], A[b, c], A[b, e], A[b, f]$ 1-es, de ezek közül $bejárva[a] = 1$ és $bejárva[c] = 1$, így itt nincs tennivaló. Viszont $bejárva[e] = 0$ és $bejárva[f] = 0$ miatt e és f bejáródnak, bekerülnek Q-ba és a *honnan* értékük b lesz, vagyis ezt kapjuk:
bejárva:

a	b	c	d	e	f	g	h
1	1	1	1	1	1	0	0

$Q = \{c, d, e, f\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	b	b	*	*

Most a c csúcs kerül a kódbeli v csúcs szerepébe, de nincs olyan bejáratlan csúcs, amibe menne él c-ből, így ezt kapjuk (csak Q változik):

a	b	c	d	e	f	g	h
1	1	1	1	1	1	0	0

bejárva:
$Q = \{d, e, f\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	b	b	*	*

Most a d csúcs kerül a kódbeli v csúcs szerepébe, egyetlen bejártalan szomszédja g:
bejárva:

a	b	c	d	e	f	g	h
1	1	1	1	1	1	1	0

$Q = \{e, f, g\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	b	b	d	*

Most az e csúcs kerül a kódbeli v csúcs szerepébe, de nincs bejáratlan szomszédja (csak Q változik ismét):
bejárva:

a	b	c	d	e	f	g	h
1	1	1	1	1	1	1	0

$Q = \{f, g\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	b	b	d	*

Most az f csúcs kerül a kódbeli v csúcs szerepébe, egyetlen bejáratlan szomszédja h:
bejárva:

a	b	c	d	e	f	g	h
1	1	1	1	1	1	1	1

$Q = \{g, h\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	b	b	d	f

A következő két körben g, majd h kerül a kódbeli v csúcs szerepébe, de egyiknek sincsen egyetlen bejáratlan szomszédja sem, vagyis csak Q változik két lépésben üressé:

bejárva:

a	b	c	d	e	f	g	h
1	1	1	1	1	1	1	1

$Q = \{\}$

honnan:

a	b	c	d	e	f	g	h
a	a	a	a	b	b	d	f

Ekkor az algoritmus leáll.

Mire jó a szélességi bejárás?

Foglaljuk össze, hogy eddig mit láttunk, mire jó a szélességi bejárás:

- Láttuk, hogy a felfedező élek fát alkotnak és hogy ha G összefüggő, akkor ez a fa minden csúcsot elér vagyis feszítőfát kaptunk. Tehát a szélességi bejárás talál egy feszítőfát, ha a gráf összefüggő (ha meg nem összefüggő, akkor nincs is feszítőfa).
- Ha G összefüggő, akkor a szélességi bejárás bejárja a gráf csúcsait, mindegyiket pontosan egyszer, ez hasznos lehet, ha a gráf csúcsaiban tárolunk valami információt, amit össze akarunk gyűjteni.
- Ha G összefüggő, akkor a felfedező élek feszítőfájában egyértelműen létezik út a kezdőcsúcsból minden másik csúcsába a gráfnak, ez üzenetek gazdaságos továbbítása során hasznos lehet. Ezeket az utakat a *honnan* tömb segítségével tudjuk megtalálni.
- El tudjuk dönteni, hogy G összefüggő-e: lefuttatjuk a szélességi bejárást és ha a végén még vannak bejáratlan csúcsok, akkor a gráf nem volt összefüggő, különben meg összefüggő volt és kaptunk is benne egy feszítőfát.
- El tudjuk dönteni, hogy mely csúcsok érhetők el a kezdőcsúcsból úttal: azok, amik be lesznek járva az algoritmus végén.

Van még egy feladat, amit meg tudunk oldani a szélességi bejárás segítségével: meg tudjuk határozni az s csúcsból az összes többi csúcsba vezető legkevesebb élből álló utakat. Ehhez csak azt kell észrevennünk, hogy ha egy v csúcsba i darab élből áll a legkevesebb élből álló út s-ből kiindulva, akkor az s-ből indított szélességi bejárás i-edik szintjére fog a v csúcs kerülni. Ez azt jelenti, hogy csak azt kell a kódunkba belerakni, hogy amikor egy csúcs bejáródik, akkor melyik

szintre kerül, ez azonban nyilvántartható, mert ha egy w csúcs a v csúcsból induló felfedező éllel járódik be, akkor w eggyel magasabb szintre fog kerülni, mint ahol v volt. A szintek nyilvántartására egy *távolság* nevű, 1-től n-ig a csúcsokkal indexelt tömböt fogunk használni, az algoritmus végén ez tartalmazza majd az s-től vett távolságokat (hány élből álló úttal lehet elérni a csúcsokat s-ből), magukat az utakat pedig a *honnan* tömbből tudjuk kiolvasni (hiszen a legrövidebb utak a felfedező élek fájában levő egyértelmű utak lesznek).

A kód a következő lesz, csak két extra sor van benne, egyik a *távolság* tömb inicializálására, a másik sor pedig az éppen bejárt csúcs távolságának beállítására.

```
bejárva[v] = 1, ha v =s, egyébként bejárva[v] = 0
Q = {s}
honnan[v] = v, ha v =s, különben honnan[v] = *
távolság[v] = 0, ha v =s, különben távolság[v] = * //  a kezdocsúcs távolsága 0


ciklus amíg Q-ban van csúcs:    // még van, akinek nem néztem a szomszédait
    v := Q első csúcsa
    vegyük ki v-t Q-ból
    ciklus w = 1-tól n-ig       // végignézem az összes potenciális szomszédot
        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
            távolság[w] := távolság[v] + 1 // w eggyel nagyobb szintre kerül, mint v
            w-t Q végére rakom
    ciklus vége
ciklus vége
```

Szélességi bejárás itányított gráfokban

Eddig irányítatlan gráfokban használtuk a szélességi bejárást, de minden pontosan ugyanígy működik iráyított esetben is, az egyetlen különbség, hogy amikor egy v csúcs szomszédait nézzük, akkor csak azok számítanak szomszédnak, akikbe megy él v-ből. A pszeudokódban semmi változtatásra nincs szükség, hiszen irányított esetben az $A[v, w] == 1$ ellenőrzés éppen ezt nézi meg.

Az irányítatlan gráfokhoz hasonlóan, irányított gráfokban is hasznos a szélességi bejárás:

- El tudjuk dönteni, hogy mely csúcsok érhetők el a kezdőcsúcsból irányított úttal (azok, amik be lesznek járva az algoritmus végén) és a felfedező élek, iletve a *honnan* tömb segítségével utakat is tudunk találni s-ből minden elérhető csúcsba.
- Ha a gráf irányított, akkor nem lehet összefüggőségről beszélni (ez a fogalom csak irányítatlan gráfokban értelmezett), de azt el lehet dönteni, hogy elérhető-e minden csúcs s-ből irányított úton.
- Az s-től vett távolságokat ugyanúgy lehet meghatározni, mint irányítatlan esetben, annak nyilvántartásával, hogy a csúcsok melyik szintre kerülnek a bejárás során (lásd az előző kódot).

Szélességi bejárás lépésszáma

A pszeudokód eleje, ahol a tömbök kezdeti értékének beállítása (az inicializálás) történik $O(n)$ lépés, mert három n mérteű tömböt kell feltöltenünk (mindegyik $O(n)$ lépés), ezen kívül még Q-t kell $\{s\}$-re állítanunk, ami konstans.

A külső ciklus legfeljebb n-szer fut le, mert minden futásnál egy új csúcs kerül a v csúcs szerepébe (és minden csúcs csak egyszer kerül ide, akkor, amikor kikerül Q-ból, ahova pedig csak egyszer kerül be, amikor bejáródik). A külsü ciklus magja két lépésből és egy belső ciklusból áll, a belső ciklus n-szer fut le és konstans sok lépésből áll, vagyis a belső ciklus $O(n)$, így a külső ciklus magja is $O(n)$, vagyis a külső ciklus egésze $O(n^2)$. Az egész kód tehát $O(n) + O(n^2)$, tehát $O(n^2)$ lépés.

Ha nem minden csúcs érhető el s-ből

Ha nem minden csúcs érhető el s-ből (irányítatlan esetben ez azt jelenti, hogy a gráf nem összefüggő), de mégis be akarjuk járni az összes csúcsot, akkor azt tesszük, hogy az algoritmus egy s csúcsból indított futása után újraindítjuk egy még nem bejárt csúcsból és ezt addig ismételjük, amíg lesznek bejáratlan csúcsok. Így irányítatlan esetben a nem összefüggő gráf komponenseiben fogunk feszítőfákat kapni. Be lehet látni, hogy ebben az esetben is igaz lesz, hogy egy n csúcsú gráfban az egész eljárás lépésszáma $O(n^2)$.