A BFS algoritmus*

A BFS (*Breadth First Search*, magyarul *Szélességi keresés*) algoritmus egy adott G gráf adott s pontjából meghatározza az összes v csúcs $t(v)$ távolságát – vagyis az s-ből v-be vezető legrövidebb út hosszát (ha ilyen út egyáltalán van). Itt az út *hossza* alatt egyszerűen az utat alkotó élek számát értjük. Ezzel együtt azt is eldönti, hogy a gráf összefüggő-e, illetve ha nem az, akkor meghatározza az s-et tartalmazó komponens csúcsait. Az eljárást számos összetettebb gráfelméleti algoritmus is használja szubrutinként.

A BFS algoritmus működésének alapötlete rendkívül egyszerű: ha már megtaláltuk az s-től $0,1,\ldots,t-1$ távolságra lévő csúcsokat, akkor a t távolságra lévők éppen azok, amelyek ezek között nem szerepelnek, de szomszédosak egy $t-1$ távolságra lévővel. Az s-től 0 távolságra nyilván csak maga s van. Így az eljárás először megkeresi az 1 távolságra lévő, vagyis s-sel szomszédos pontokat. Majd az ezek közül valamelyikkel, de s-sel nem szomszédos csúcsokat 2 távolságra lévőnek nyilvánítja. Ezek után a 2 távolságra lévők valamelyikével szomszédos, de eddig nem érintett csúcsokat 3 távolságra lévőnek nyilvánítja, stb.

Persze az algoritmus az s-től azonos távolságra lévő csúcsokat nem tudja egyszerre bejárni, így ezek között is kialakít egy (véletlenszerű) sorrendet. Így végül is az összes, s-sel egy komponensbe tartozó csúcsot felsorolja valamilyen sorrendben.

Az alábbi pszeudokóddal a fenti, szemléletes képnél kicsit pontosabban is leírjuk a BFS algoritmust. Az eljárás a működése során az alábbi adatokat tartja nyilván:

- $b(i)$ $(i=1,2,\ldots)$: az i-edikként **b**ejárt csúcs
- $t(v)$ $(v\in V)$: v **t**ávolsága s-től
- $m(v)$ $(v\in V, v\neq s)$: a v-t **m**egelőző csúcs az algoritmus által megtalált, s-ből v-be vezető legrövidebb úton
- j: az eddig bejárt csúcsok száma
- k: a jelenleg aktív csúcs sorszáma a $b(1),b(2),\ldots$ sorozatban

BFS ALGORITMUS

Bemenet: Egy $G=(V,E)$ gráf és egy $s\in V$ csúcs

0. lépés. $j\leftarrow 1$, $k\leftarrow 1$, $b(1)\leftarrow s$, $t(s)\leftarrow 0$, minden $v\in V$, $v\neq s$-re $t(v)\leftarrow *$

1. lépés.

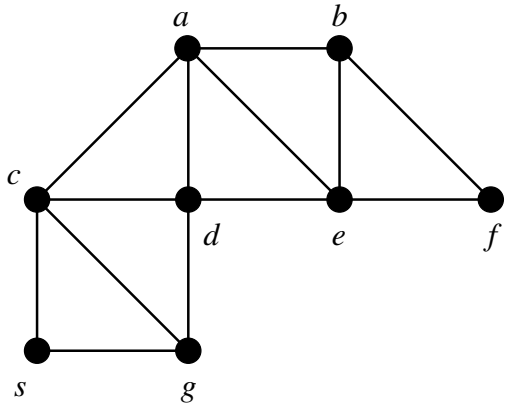
- Ha a $b(k)$ csúcsnak van olyan v szomszédja, amelyre $t(v)=*$, akkor:
  - $j\leftarrow j+1$
  - $b(j)\leftarrow v$
  - $t(v)\leftarrow t\big(b(k)\big)+1$
  - $m(v)\leftarrow b(k)$
  - Folytassuk az **1. lépésnél**.

2. lépés.

- Ha $k=j$, akkor STOP.
- $k\leftarrow k+1$
- Folytassuk az **1. lépésnél**.

* Összeállította: Szeszlér Dávid

Az eljárás működését az alábbi gráfon illusztráljuk:



Lefuttatva az algoritmust az alábbi adatok keletkeznek:

i	1	2	3	4	5	6	7	8
$b(i)$	s	c	g	a	d	b	e	f

v	s	a	b	c	d	e	f	g
$t(v)$	0	2	3	1	2	3	4	1

v	s	a	b	c	d	e	f	g
$m(v)$		c	a	s	c	a	b	s

Természetesen ez csak az egyik lehetséges helyes futása az algoritmusnak: attól függően, hogy az éppen aktuális csúcs még bejáratlan szomszédait milyen sorrendben veszi sorra, a BFS-nek számos más helyes futása is elképzelhető.

Az eljárásban az $m(v)$ változók szerepe az, hogy az s-ből az egyes csúcsokba vezető legrövidebb utaknak ne csak a hossza, hanem az ilyen utak egyike is kiolvasható legyen a kimenetből. Valóban: egy tetszőleges v csúcsból az $m(v), m\big(m(v)\big), \ldots$ sorozaton haladva mindig eggyel csökken a s-től vett távolság, így végül $t(v)$ lépésben érünk el s-be. Például a fenti gráf esetében az f csúcsból az $m(f) = b$, $m(b) = a$, $m(a) = c$, $m(c) = s$ lépésekkel valóban $t(f) = 4$ hosszú úton jutunk el s-be.

Az algoritmus helyességének belátásához könnyű végiggondolni, hogy az a működése során végig fenntartja a következő (a **0. lépés** végrehajtása után rögtön teljesülő) tulajdonságokat:

- ha $t(v) \neq *$, akkor s-ből v-be vezet olyan $t(v)$ hosszúságú út, amelynek ($v \neq s$ esetén) az utolsó előtti csúcsa $m(v)$;
- ha $t(v) \neq *$, akkor s-ből v-be nem vezet $t(v)$-nél rövidebb út.

Emellett az algoritmus leállása után azoknak a v csúcsoknak a C halmaza, amelyekre $t(v) \neq *$ éppen az s-ből úton elérhető pontok halmaza – vagyis az s-et tartalmazó komponens csúcshalmaza. Valóban: minden $v \in C$ csúcs egyrészt nyilván elérhető s-ből, másrészt mivel v az eljárás egy pontján az éppen aktív csúcs volt, így nem lehet C-be nem tartozó szomszédja. Így a G gráf pontosan akkor összefüggő, ha $C = V(G)$.

Minden algoritmus, így a BFS esetében is fontos meggondolni annak a lépésszámát. Mivel a BFS eljárás szempontjából a párhuzamos és hurokélek érdektelenek, ezért feltehetjük, hogy a bemenetként kapott G gráf egyszerű. Tegyük fel továbbá, hogy G úgy van megadva, hogy minden v csúcshoz tartozik egy lista, amely felsorolja a v szomszédait. A gráfoknak ezt az igen egyszerű megadási módját *szomszédsági listának* (*adjacency list*) szokták nevezni. (Itt most eltekintünk az implementációnak a további technikai részleteitől – mint például hogy a csúcsok szomszédait soroló listákat hogyan praktikus tárolni.) Ezt használva minden él kettővel növeli a listák összhosszát, az él mindkét végpontjánál eggyel-eggyel. Így egy m élű gráfra a listák összhossza $2m$. (A bemenet mérete valójában ennél több, mert egyrészt minden csúcshoz el kell tárolni a hozzá tartozó lista kezdetének helyét, másrészt minden egyes listaelem is elfoglal valamennyi helyet a memóriában – de ezektől a technikai részletektől ismét eltekintünk.)

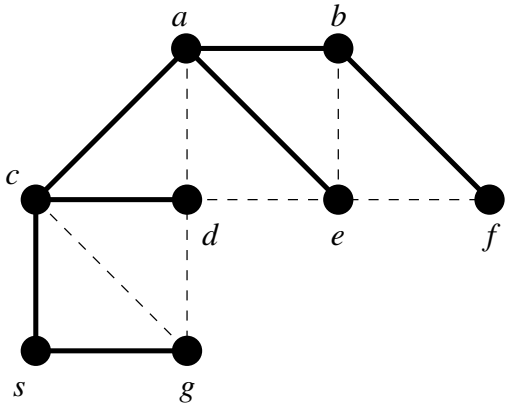
A BFS eljárás a v csúcs aktívvá válásakor (vagyis amikor $b(k) = v$ bekövetkezik) a v-re illeszkedő éleket vizsgálja végig és így $d(v)$-vel, v fokszámával arányos lépést tesz. Mivel az összes pont fokának összege a gráf élszámának kétszerese, ezért elmondható, hogy a BFS egy m élű gráfon legföljebb $c \cdot m$ futásidő után megáll (ahol c valamilyen rögzített konstans). Mivel a bemenetben is $2m$ volt a csúcsokhoz tartozó listák összhossza, ezért elmondhatjuk, hogy a BFS nagyon hatékony, lineáris futásidejű algoritmus.

Fontos azonban rögzíteni, hogy az élszámmal arányos futásidő garantálásához a bemenet megadásának módja egyáltalán nem mellékes körülmény. Tegyük fel például, hogy G *szomszédsági mátrixszal* (*adjacency matrix*) van megadva – vagyis $V(G) = \{v_1, v_2, \ldots, v_n\}$ esetén egy olyan $n \times n$-es $A(G)$ mátrixszal, amelyben minden $1 \leq i, j \leq n$-re az i-edik sor és a j-edik oszlop kereszteződésében álló elem a v_i és v_j között futó élek száma. (Ez tehát 0 vagy 1, ha továbbra is feltételezzük, hogy G egyszerű.) Ez azért lényeges különbség, mert a szomszédsági listához képest a szomszédsági mátrixban egy v_i csúcs szomszédai sokkal kevésbé könnyen elérhető módon állnak rendelkezésre, végig kell értük pásztáznunk az $A(G)$ teljes i-edik sorát (vagy oszlopát). Így a BFS eljárásban már nem elég $d(v_i)$-vel arányos időt fordítanunk az éppen aktív v_i csúcsra, hanem csúcsonként $c \cdot n$, összesen $c \cdot n^2$ lépésszámot kapunk. Más kérdés, hogy mivel $A(G)$ egy $n \times n$-es mátrix, ezért ebben az esetben az input mérete is n^2-tel arányos (hiszen egyszerű gráfra csúcspáronként 1 bitnyi információt kell tárolnunk), így az algoritmus ekkor is lineáris futásidejű – annak ellenére is, hogy a gráfot hatékonyabban tárolva jobb futásidőt is elérhettünk volna.

Tanulságként azt vonhatjuk le, hogy egy algoritmus futásidejének pontos elemzésekor olyan, elsőre talán technikai részletkérdésnek tűnő szempontokat is figyelembe kell vennünk, mint az input megadásához használt adatstruktúra. Azt azonban gyakran már egy felületes, a részleteket figyelmen kívül hagyó elemzéssel is megállapíthatjuk, hogy a vizsgált algoritmus polinomiális futásidejű, vagyis alkalmazásokban hatékonynak tekinthető.

A BFS-fa

A BFS algoritmus fenti példán való futtatásából külön figyelmet érdemel az utolsó táblázat – általában pedig azok az élek, amelyek minden $v \neq s$ esetén v-t $m(v)$-vel kötik össze. A fenti ábra gráfjára ezek az alábbi ábrán láthatók (a többi élt szaggatott vonallal jelöltük).



Megfigyelhető, hogy a szóban forgó élek egy fát, mégpedig az input G gráf s-et tartalmazó C komponensének egy feszítőfáját alkotják. Ez általában is így van: a C csúcshalmazon a minden v csúcsra v-t $m(v)$-vel összekötő élek alkotta F gráf egyrészt összefüggő, hiszen minden C-beli csúcsból F élein eljuthatunk s-be, így bármely két C-beliből is egymásba. Másrészt F körmentes is, mert a BFS eljárás a működése során folyamatosan fenntartja azt a tulajdonságot, hogy az F-be addig bekerült élek körmentes részgráfot alkotnak: valóban, amikor az 1. lépés végrehajtása során $m(v)$ értéket kap, akkor v elsőfokú az addig megtalált F-beli élek gráfjában (hiszen csak $m(v)$-vel szomszédos), így egyetlen kör sem tartalmazhatja az újonnan F-be került v és $m(v)$ közötti élt.

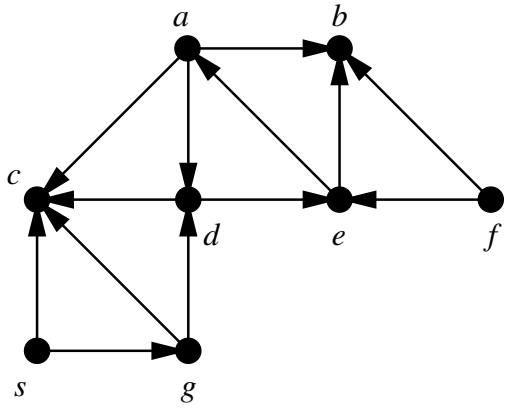
Az így kapott F fát *BFS-fának* nevezzük; ez tehát összefüggő G esetén feszítőfája G-nek és bármely v csúcsra az s-et v-vel összekötő F-beli út a legrövidebbek egyike az s-ből v-be vezető G-beli utak közül.

BFS algoritmus irányított gráfban

A gráf fogalmának bevezetését olyan alkalmazások motiválták, amelyekben egy (véges) halmaz elemei közötti, páronkénti kapcsolatok játszanak szerepet. A gráf csúcsai lehettek például egy társaság tagjai és az élek a köztük lévő ismeretségek; vagy a csúcsok lehettek egy térkép csomópontjai és az élek az őket összekötő útszakaszok; vagy a csúcsok egy számítógép-hálózat csomópontjai és az élek a köztük lévő közvetlen összeköttetések, stb. Azonban nagyon gyakoriak az olyan alkalmazások is, amelyekben az elemek közötti kapcsolatok nem (feltétlen) szimmetrikusak. Így például egy társaság tagjai között lehetnek nem kölcsönös ismeretségek (ha például jelen van egy közismert ember), egy térképen lehetnek egyirányú útszakaszok és egy számítógépes hálózatban is előfordulhat, hogy két csomópont között csak egyirányú kommunikáció lehetséges. Ezeket a helyzeteket *irányított gráfokkal* modellezhetjük.

Egy irányított gráf tehát szintén csúcsokból és élekből áll, azonban minden élnek iránya is van: nem elég információ, hogy az e él az u és a v csúcsok között halad, tudnunk

kell azt is, hogy az u-ból a v-be, vagy a v-ből az u-ba mutat. (Természetesen nincs akadálya annak sem, hogy egy irányított gráf az $u \to v$ és a $v \to u$ éleket is tartalmazza – sőt, bármelyiket akár több példányban is, hasonlóan az irányítatlan gráfokban megengedett párhuzamos élekhez.) Rajzban az irányított éleket egyszerűen nyilakkal szokás jelezni; így például az alábbi ábrán egy irányított gráf látható.



Az adott pontból induló legrövidebb irányított utak megtalálásának feladata alkalmazásokban szintén gyakran felmerülő probléma. (*Irányított út* alatt nyilván olyan utat értünk, amelyen végighaladva sosem megyünk szembe az élek irányításával.) Ez a feladat például akkor, ha egy egyirányú linkeket is tartalmazó számítógép-hálózatban szeretnénk információt továbbítani úgy, hogy az adatcsomagok a lehető legkevesebb közbülső csomóponton haladjanak át. Szerencsére ennek a feladatnak a megoldására nem kell vadonatúj algoritmust kidolgoznunk, a BFS eljárás ezt is megoldja. Ehhez csupán egy minimális, értelemszerű módosításra van szükség a fenti pszeudokódban: az 1. lépésben a „Ha a $b(k)$ csúcsnak van olyan v szomszédja, amelyre $t(v) = *$, akkor…” esetvizsgálat helyett a „Ha a $b(k)$ csúcsból vezet (irányított) él olyan v csúcsba, amelyre $t(v) = *$, akkor…” szükséges. Ettől eltekintve az algoritmus azonos a fentivel.

Például a fenti irányított gráfra lefuttatva az eljárást az alábbi adatok keletkeznek (az egyik lehetséges futást feltételezve):

i	1	2	3	4	5	6	7
$b(i)$	s	c	g	d	e	a	b

v	s	a	b	c	d	e	f	g
$t(v)$	0	4	4	1	2	3	$*$	1

v	s	a	b	c	d	e	f	g
$m(v)$		e	e	s	g	d		s

Látható, hogy itt már az f csúcs nem elérhető s-ből irányított úton. Általában is igaz, hogy a BFS eljárás leállásakor azon v csúcsoknak a C halmaza, amelyekre $t(v) \neq *$, éppen az s-ből irányított úton elérhető pontok halmaza. Érdemes megfigyelni azt is, hogy a

BFS-t ugyancsak a fenti irányított gráfra, de a d csúcsból indítva a d-n kívül csak az a, b, c és e csúcsok volnának elérhetők – vagyis az elérhető csúcsok halmaza még akkor is nagyban függ a kiindulópont választásától, ha a gráf az élek irányítását figyelmen kívül hagyva egyébként összefüggő. Így az s-ből elérhető pontok C halmazát az irányított esetben már nem hívhatjuk komponensnek — ezt a fogalmat ilyenkor sokkal körültekintőbben kell használnunk (de ennek a részleteire itt nem térünk ki).

Az $m(v)$ változók szerepe a BFS eljárás irányított változatában nyilván azonos a korábbival: tetszőleges v csúcsra (legalábbis azokra, amelyekre $t(v) \neq *$ és így $m(v)$ is értéket kapott) ezeket használva lépegethetünk vissza v-től s-ig, hogy egy s-ből v-be vezető legrövidebb irányított utat is megkapjunk. Ismét hasznos külön kiemelni azokat az irányított éleket, amelyek $m(v)$-ből v-be vezetnek valamilyen v-re: ezek (az élek irányításától most eltekintve) megint egy F fát alkotnak, amelynek csúcshalmaza az s-ből elérhető csúcsokból áll és az F élei mentén s-ből eljuthatunk az összes többi csúcsba a lehető legrövidebb úton (legalábbis az ilyen utak egyikén). A BFS algoritmus fenti futásából származó BFS fát az alábbi ábra mutatja: