Szélességi bejárás (szélességi keresés) – gyakorlat

óravázlat

Milyen feladatot old meg az algoritmus, mik az eredmények

Az algoritmus a gráfokra használt egyik nevezetes bejáró algoritmus: egy tetszőleges kezdő csúcsot megadunk, és az abból úttal elérhető csúcsokat felderíti. A felderített csúcsokhoz egy minimális hosszúságú utat állít elő a kezdőcsúcsból. Ha több ilyen út lenne, az egyiket.

Eredmények:

- d: minimális út hossza (élszáma), ∞, ha a csúcshoz nem vezet út a kiválasztott kezdőcsúcsból.
- π: szülő csúcsot adja meg a kezdőcsúcs és az adott csúcs közötti legrövidebb úton (honnan érkezünk a csúcsba), 0 a kezdőcsúcsnál, valamint azoknál a csúcsoknál, amelyek a kezdőcsúcsból nem érhetők el.
- color: a bejárások használják, egy csúcsnak három állapota lehet, ezt ábrázolja.
    - white - ’fehér’ – még felderítetlen a csúcs, nem találkozott vele a bejárás,
    - grey - ’szürke’ – már találkozott a bejárás a csúccsal, de még nem dolgozta fel,
    - black - ’fekete’ – a csúcs feldolgozása befejeződött, ’kész’ állapotú.

A ’szürke’ csúcsokat egy sor tárolja, abban várakoznak, hogy feldolgozásra kerüljenek.

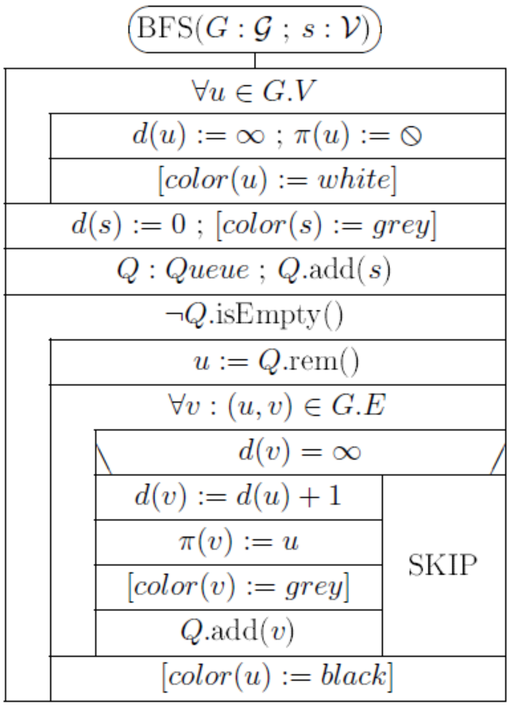
Milyen gráfokon használható

Tetszőleges (nem üres) gráfon használható:

- irányított vagy irányítatlan
- összefüggő, nem összefüggő

Az algoritmus áttekintése

Struktogram áttekintése, műveletigény áttekintése, mi hagyható el belőle (d, pi, color)
Műveletigénynél: G=(V,E), és |V|=n és |E|=m



Műveletigény MT(n,m)	Műveletigény mT(n,m)
$\Theta(n)$	$\Theta(n)$
$\Theta(1)$	$\Theta(1)$
$\Theta(1)$	$\Theta(1)$
$\Theta(n)$	$\Theta(1)$ *csak a kezdőcsúcsot dolgozza fel a ciklus*
$\Theta(m)$	*nullaszor lefutó ciklus is lehet, ha nincs éle a gráfnak az s csúcsból*
összesítve: $\Theta(n+m)$	összesítve: $\Theta(n)$

Mit jelent a fejlécben szereplő „írott G” típus és „írott V” típus:

$\mathrm{BFS}(G : \mathcal{G} \; ; \; s : \mathcal{V})$

Jegyzet 2. fejezetében definiált típusok[1]:

- A gráfok absztrakt algoritmusainak leírásához bevezetjük a $\mathcal{V}$ (vertex, azaz csúcs) absztrakt típust. A $\mathcal{V}$ lesz a gráfok csúcsainak absztrakt típusa. Ez egy olyan elemi típus, amelyben mindegyik csúcshoz tetszőlegesen sok, névvel jelölt címke társítható, és mindegyik címkéhez tartozik valamilyen érték. A $\mathcal{V}$ halmazt az algoritmusok implementációiban legtöbbször az N halmaz reprezentálja, egy n csúcsú gráf csúcsait pedig egyszerűen az 1..n vagy a 0..(n-1) halmaz, attól függően, hogy a tömböket egytől vagy nullától kezdve indexeljük. A csúcsokhoz tartozó címkéket gyakran tömbök reprezentálják.
  d(u), color(u), π(u) például ilyen címkék a szélességi bejárásban.
- Élek halmaza ($\mathcal{E}$)

$\mathcal{E}$
$+\, u, v : \mathcal{V}$

- Gráf típus ($\mathcal{G}$)

$\mathcal{G}$
$+\, V : \mathcal{V}\{\}$
$+\, E : \mathcal{E}\{\} \; // \; E \subseteq V \times V \setminus \{(u,u) : u \in V\} \; // \; \mathrm{edges}$

FONTOS! Az ábrázolással kapcsolatosan fontoljuk meg az éleket bejáró ciklus műveletigényét:

- A $\forall$v: (u,v) $\in$ G.E ciklusban u szomszédjait dolgozza fel az algoritmus. Ez akár minden csúcsra lefutó ciklus lehet. Miért nem szabad úgy elképzelni ezt a ciklust, hogy az élek teljes halmazát bejárva keressük meg az (u,v) éleket?
  Válasz: Θ(n*m)-re növelné a (maximális) műveletigényt, sűrű gráf esetén ez már $\Theta(n^3)$!
- Ez a ciklus benne van egy O(n) lépésszámú ciklusban, így csak a szomszédok hatékony ábrázolása esetén lehet Θ(m) a maximális lépésszám.
- Ritka gráfon, szomszédossági listás ábrázolás esetén a lista első elemének elérése konstans időben történik, majd csak a szomszédokon megy végig a ciklus, így minden csúcsra végrehajtva a lista bejárását, épp Θ(m) lépésszám teljesül.
- Sűrű gráfok esetén, csúcsmátrixos ábrázolást használva, a szomszédok bejárása Θ(n), mivel ez benne van egy n-szer lefutó ciklusban, így maximális lépésszáma $\Theta(n^2)$, viszont sűrű gráf esetén m=|G.E| $\in \Theta(n^2)$, így a Θ(m) korlát ilyenkor is teljesül.

A csúcsok mely jellemzője (címkéje) hagyható el az algoritmusból (d,π,color):

- color: általában kihagyható, mivel d(u)=∞ vizsgálat helyettesíti a color(u)=white vizsgálatot. A szürke és fekete szín sokszor helyettesíthető egy színnel (*nem mindig!*), ha csak azt szeretnénk eldönteni, hogy a csúcs már látókörbe került, vagy sem. Azaz arra használjuk, nehogy többször is bekerüljön a sorba ugyanaz a csúcs, amivel végtelen ciklusba kerülne a bejárás.
- π: a szülő pointer elhagyható, ha nem kell az előállított útvonal a feladathoz.
- d: ha csak a bejárás részét használjuk az algoritmusnak, nem célunk a legrövidebb út hosszának ellőállítása, akkor d elhagyható, de ilyenkor a color mindenképpen szükséges, és gyakran mindhárom állapot fontos: white, grey, black.

[1] Nem tudom pont azt a betűtípust használni, így az MS Word „Script MT Bold” karakterkészletét használtam.

LEJÁTSZÁS

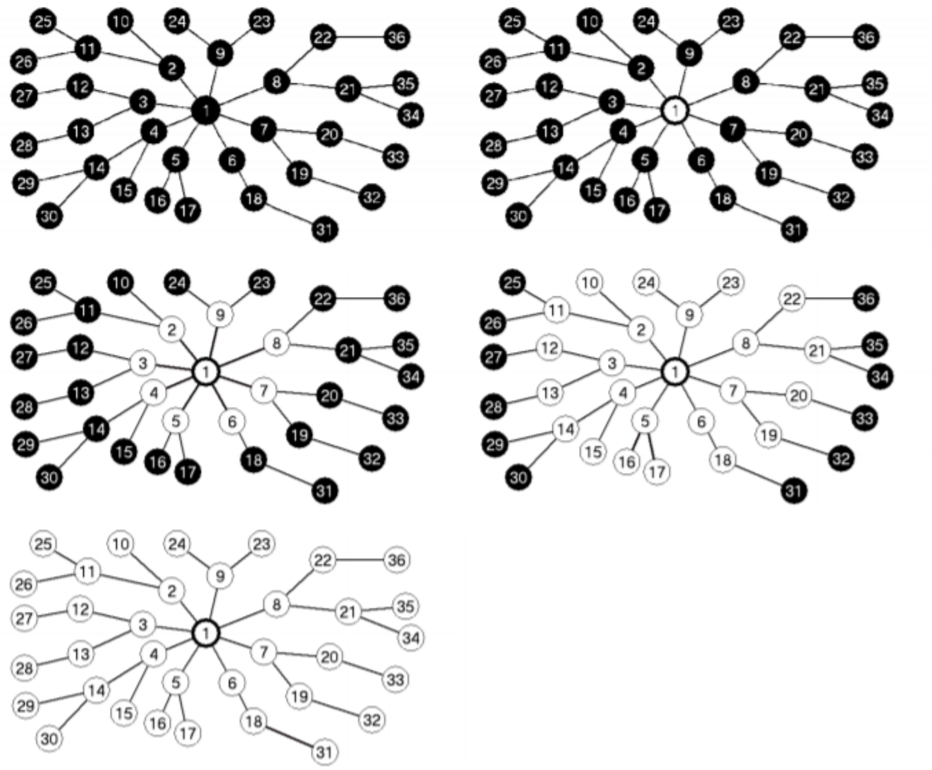
dr Fekete István által készített jegyzetben (https://people.inf.elte.hu/fekete/algoritmusok_jegyzet/) nagyon jó szemléltető ábra van, hogyan terjed egyre nagyobb körökben a kezdő csúcs körül a felderített csúcsok köre. Ebből a jegyzetből másoltam ide egy részletet:

23.1. A szélességi bejárás stratégiája

A bejárási stratégiákról megkapóan szemléletes leírást találhatunk a *Rónyai Lajos*, *Ivanyos Gábor* és *Szabó Réka* szerzőhármas *Algoritmusok* című könyvében. Szabadon és tömören idézzük „az öreg városka girbe-gurba utcáin bolyongó kopott emlékezetű lámpagyújtogató esetét”, illetve, most csak az egyik módszert arra, hogy végül az összes köztéri lámpa világítson.

Az egyik eljárás szerint a lámpagyújtogatót egy este nagyszámú barátja elkíséri a városka főterére, ahol együtt meggyújtják az első lámpát. Utána annyi felé oszlanak, ahány utca onnan kivezet. A különvált kis csapatok meggyújtják az összes szomszédos lámpát, majd tovább osztódnak. A városka lámpáit ilyen módon széltében terjeszkedve érik el.

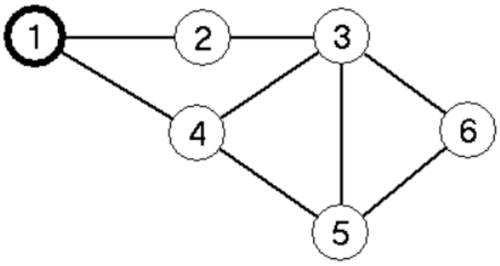
Ha fentről néznénk a várost, ahogy kigyulladnak a lámpák, azt látnánk, hogy a középpontból a város széle felé egyre nagyobb sugarú körben terjed a világosság. Ez a szemléletes alapelve a *szélességi bejárásnak*. A szélességi stratégiát a 23.1. ábrán látható néhány pillanatfelvétel illusztrálja.



23.1. ábra. Szélességi bejárás egy gráfon

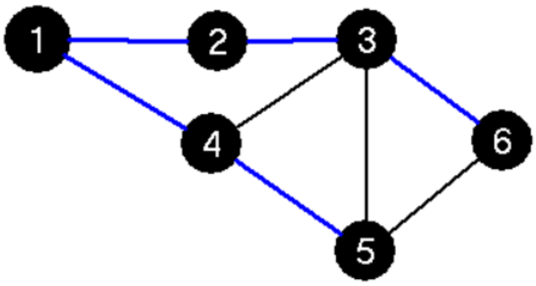
1. Közösen megoldott feladat a bejárás szemléltetésére

A gráf képe (kezdőcsúcs: 1):



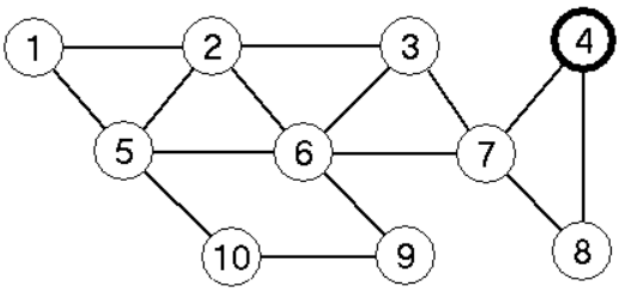
A lejátszáskor kapott táblázat:

Kiterjesztett csúcs	**csúcsok d értékei**						**Sor tartalma**	**csúcsok π értékei**					
	1	*2*	*3*	*4*	*5*	*6*		*1*	*2*	*3*	*4*	*5*	*6*
	0	∞	∞	∞	∞	∞	< **1** >	0	0	0	0	0	0
1, d:0		1		1			< **2**;4 >		1		1		
2, d:1			2				< **4**;3 >			2			
4, d:1					2		< **3**;5 >					4	
3, d:2						3	< **5**;6 >						3
5, d:2							< **6** >						
6, d:3							< >						
	0	1	2	1	2	3		0	1	2	1	4	3



2. Önálló munkára kiadható feladat (bejárás lejátszása önállóan)

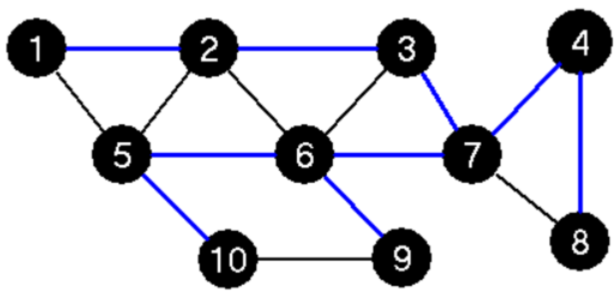
4-es csúcsból indulva játsszuk le a szélességi bejárás algoritmusát.



A táblázat:

Kiterjesztett csúcs	csúcsok d értékei										Sor tartalma	csúcsok π értékei									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*		*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	< 4 >	0	0	0	0	0	0	0	0	0	0
4,d:0							1	1			< 7;8 >							4	4		
7,d:1			2			2					< 8;3;6 >			7			7				
8,d:1											< 3;6 >										
3,d:2		3									< 6;2 >		3								
6,d:2					3				3		< 2;5;9 >					6				6	
2,d:3	4										< 5;9;1 >	2									
5,d:3										4	< 9;1;10 >										5
9,d:3											< 1;10 >										
1,d:4											< 10 >										
10,d:4											< >										
	4	**3**	**2**	**0**	**3**	**2**	**1**	**1**	**3**	**4**		**2**	**3**	**7**	**0**	**6**	**7**	**4**	**4**	**6**	**5**

A szélességi fa:



3. Lefuttattuk a szélességi bejárást egy ismeretlen gráfon, a π értékeket ismerjük, keressünk benne utat, rajzoljuk le a szélességi fát.

Egy ismeretlen gráfon lefuttattuk a szélességi bejárást. A szülő értékek ismertek, az alábbi táblázat szerintiek:

csúcs:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

Adjuk meg, az 5-ös csúcsba vezető utat:

A π(5) értékből visszafelé indulva deríthető ki az út:

csúcs:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

11 → 5

csúcs:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

7 → 11 → 5

csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

$2 \rightarrow 7 \rightarrow 11 \rightarrow 5$

csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

$4 \rightarrow 2 \rightarrow 7 \rightarrow 11 \rightarrow 5$

csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

4-es volt a kezdőcsúcs, tehát az út:

$4 \rightarrow 2 \rightarrow 7 \rightarrow 11 \rightarrow 5$

Következő feladat: rajzoljuk fel a szélességi fát:

csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

4-es a gyökér

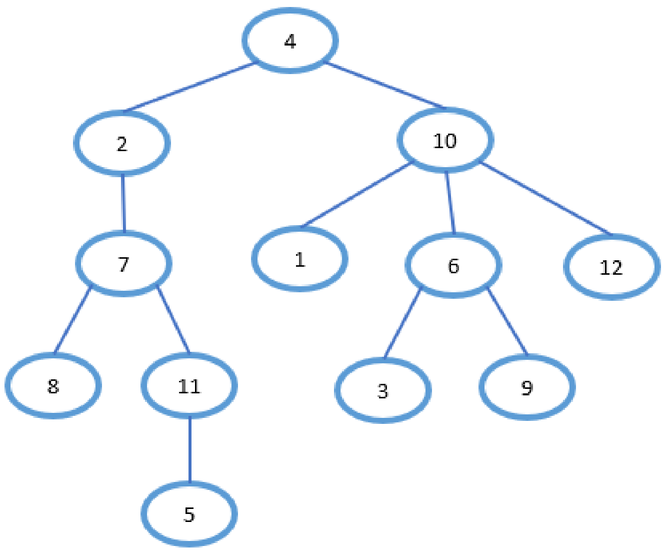
csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

a 4 gyerekei: 2 és 10

csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	**10**	**4**	**6**	**0**	**11**	**10**	**2**	**7**	**6**	**4**	**7**	**10**

2 gyereke:7   10 gyerekei: 1,6 és 12

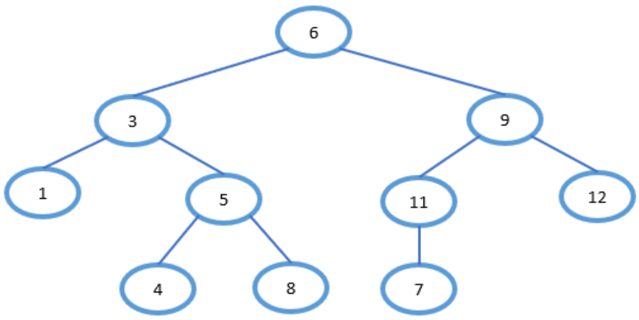
Ezt folytatva a szélességi fa:



Mi lehetett itt a fa (több nulla is van a szülő értékek között):

csúcs:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
szülő:	**3**	**0**	**6**	**5**	**3**	**0**	**11**	**5**	**6**	**0**	**9**	**9**

Megoldás: 2,6 és 10 csúcs szülője nulla. Vegyük észre, hogy közülük csak a 6 szerepel szülőként, ami úgy lehetséges, hogy a 2 és 10 csúcsok nem voltak elérhetőek a 6-os csúcsból. Tehát a fa:



ALGORITMUSOK

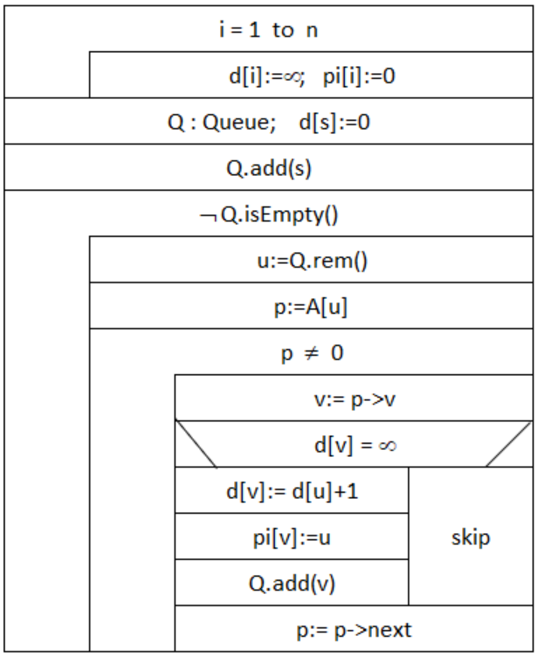
4. Készítsük el a szélességi keresés algoritmusát csúcsmátrixos/szomszédossági listás ábrázolásra, tömböket használva d, pi, color értékekre.

Megállapodások:

- a csúcsok nincsenek ábrázolva, az 1..n természetes számok, azonosítják őket,
- a szomszédossági listás ábrázolás az A/1:Edge*[n] tömbben van,
- a csúcsok d értékei a d/1:N[n] tömbben lesznek,
- a csúcsok π értékei a pi/1:N[n] tömbben lesznek,
- color-t nem használjuk.

Az algoritmus:

BFS(A/1:Edge*[n]; d/1:N[n]; pi/1:N[n]; s:1..n)



d és pi tömbök feltöltése

kezdő csúcs d-je legyen nulla

berakjuk a sorba a kezdő csúcsot

az u csúcs feldolgozása

az u csúcshoz tartozó lista első elemére

állítjuk a p pointert

u szomszédja: p->v csúcs

ha v csúcs még nem került az

algoritmus látóterébe

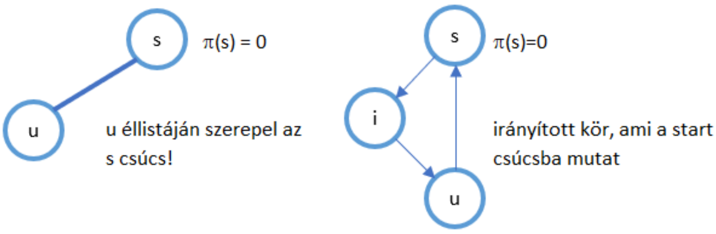
d[v] és pi[v] értéket kap

v csúcs bekerül a sorba

p pointer tovább lép az u csúcs éllistáján

Miért nem lenne helyes a d[v] = ∞ feltétel helyett a pi[v] = 0 feltételt vizsgálni?

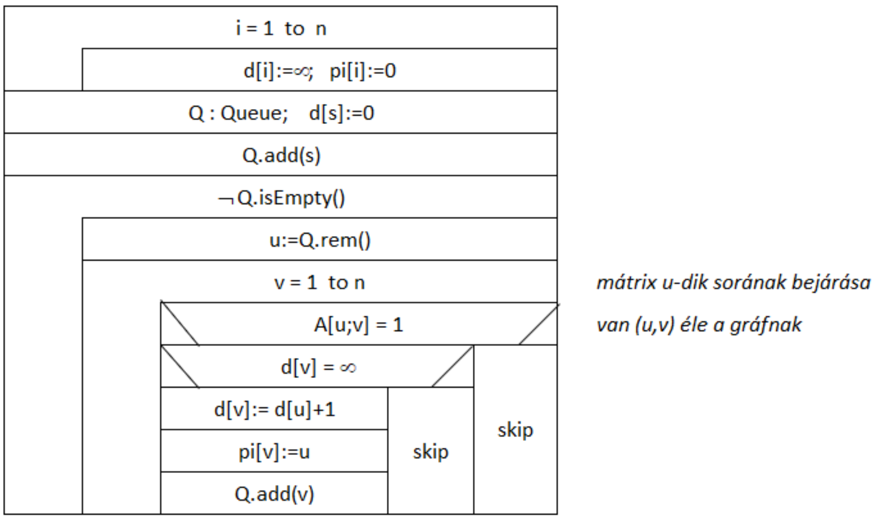
- d[v] = ∞ csak azokra a csúcsokra teljesül, amelyek elsőként kerülnek a látókörbe.
- pi[v] = 0 ezeknél a csúcsoknál nyilván teljesül, de nem csak ezeknél a csúcsoknál! A kezdő csúcsra is igaz! Emiatt az algoritmus hibásan működne: a kezdő csúcsot is úgy kezelné, mintha még nem járt volna ott.
  Ez irányítatlan esetben a kezdőcsúcs bármely szomszédjánál azonnal hibát okozna, irányított gráfnál pedig, ha van olyan irányított köre, ami a kezdő csúcsra mutat.



Mit és hogyan kell változtatni, ha a gráf az A/1:bit[n][n] mátrixban van ábrázolva?

- Csak a szomszédok bejárása fog megváltozni az ábrázolás miatt.
- Mennyi így a műveletigény? A mátrixnak minden sorát bejárjuk, így a belső ciklus (maximális) műveletigénye: $\Theta(n^2)$,
  az A[u,v]=1 feltétel viszont pontosan $\Theta(m)$-szer fog teljesülni.

BFS(A/1:bit[n][n]; d/1:N[n]; pi/1:N[n]; s:1..n)

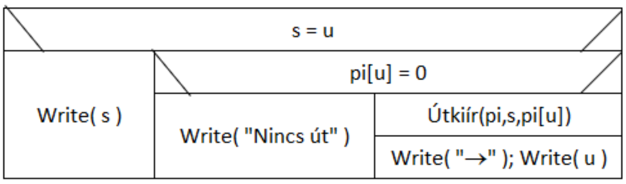


5. Készítsünk rekurzív / iteratív útkiíró algoritmust.

A csúcsokat 1..n azonosítja, szülő értékeket egy pi/1:N[n] tömb, s a kezdőcsúcs, u pedig az a csúcs, amelybe az utat ki akarjuk íratni. Figyeljünk a következőkre: u=s előfordulhat, illetve lehet, hogy a bejárás nem talált utat az u csúcsba, ekkor az algoritmus azt írja ki, hogy „Nincs út.” Az úton a csúcsok közé írjunk egy „→” karaktert.

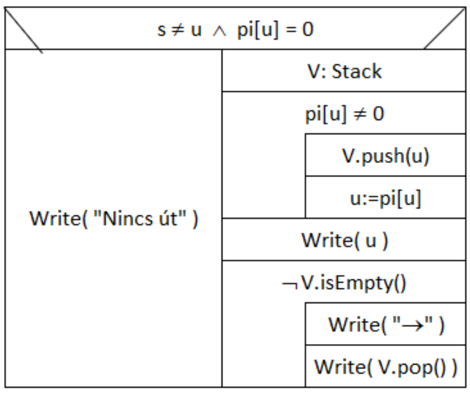
Rekurzív megoldás

Útkiír(pi/1:N[n]; s:1..n; u:1..n)



Iteratív megoldás

Útkiír(pi/1:N[n]; s:1..n; u:1..n)

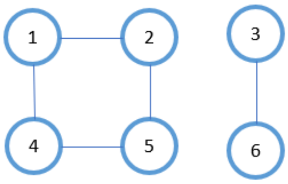


6. Adott egy irányítatlan gráf szomszédossági listás ábrázolásban, döntsük el szélességi bejárás segítségével, hogy a gráf fa-e?

A csúcsokat 1..n azonosítja, a gráf az A/1:Edge*[n] tömbben van ábrázolva. Csak a color-t fogjuk használni. Legyen szín={white,grey,black} típus.

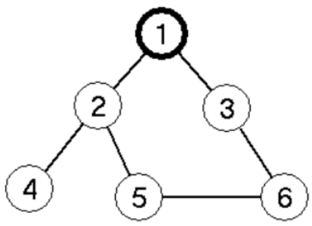
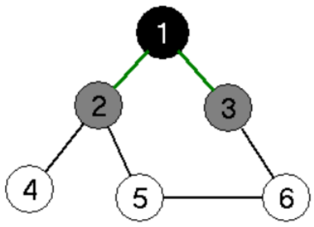
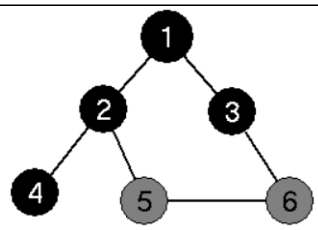
Fának tekintjük azt az irányítatlan gráfot, amely összefüggő és nem tartalmaz kört. Tehát ezt a két tulajdonságot kell ellenőriznünk.

Felmerül a következő egyszerű ötlet: nincs is szükség bejárásra, tudjuk, hogy n csúcsa van a gráfnak, számoljuk hát meg az éleket, n-1 él esetén nem lehet benne kör. Sajnos ez nem teljesen igaz, mert nem tudjuk, hogy a gráf összefüggő-e. Itt van egy egyszerű példa 6 csúcsra, a gráf 5 élt tartalmaz, és nem fa.



Be kell járnunk tehát a gráfot, meg kell vizsgálni, hogy bejárás közben találunk-e kört létrehozó élt, ha nem találkoztunk ilyen éllel, akkor végül még azt kell ellenőrizni, hogy mindegyik csúcsot meglátogattuk-e.

Kör észlelése bejárás közben:

		
Indítsuk el a bejárást az 1-es csúcsból, figyeljük meg a kör feltűnését.	A 2-es és 3-as csúcsok kiterjesztésénél fekete és fehér szomszédokat fogunk látni. Fekete a szülő csúcs, így nem jelent kört.	Amikoraz 5-ös csúcshoz ér a feldolgozás, az 5-ös is találkozik a szülőjével (2-es), aki már fekete, viszont meglátja a 6-os csúcsot, ami szürke. A szürke szín jelzi, hogy kört találtunk a gráfban.

- Tehát egy (u,v) él feldolgozása közben azt kell figyelni majd, hogy szin[v] = grey.
- Ez például egy olyan feladat, ahol két szín nem lenne elég, szükség van a háromféle színre. Megjegyezzük, hogy két szín elég abban az esetben, ha a szülőt nyilvántartó π tömböt még betesszük az algoritmusba, ekkor (u,v) él kört hoz létre, ha szin[v] $\neq$ white és v $\neq$ π[u].
- Azt, hogy minden csúcsot meglátogattunk-e, egyszerű számlálással fogjuk ellenőrizni: bevezetünk egy számlálót. mely a feldolgozott csúcsokat megszámolja. Ha ez végül n, akkor minden csúcsot feldolgoztunk.

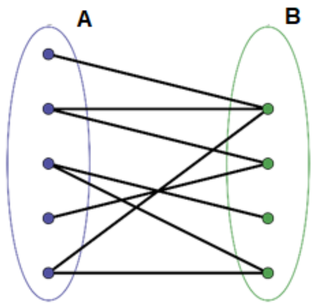
Lássuk tehát az algoritmust:

Fa_e(A/1:Edge*[n]) : Bool

color:szin[1..n] //color, szin típusú tömb létrehozása	*szin={white,grey,black}*
i = 2 to n	*az 1-es csúcsból indítjuk az algoritmust, a*
color[i]:= white	*többit fehérre állítjuk*
Q : Queue color[1]:=grey	*szokásos lépések, fa logikai változó fogja a*
Q.add(1) fa:=igaz c:=0	*ciklusokat leállítani, ha a gráf nem fa*
fa ∧ ¬ Q.isEmpty()	
u:=Q.rem() c:=c+1	*c-ben számoljuk a feldolgozott csúcsokat*
color[u]:=black p:= A[u]	
fa ∧ p ≠ 0	
v:= p->v	*fehér szomszéd esetén folytatódik a bejárás,*
\ color[v]=white: color[v]:=grey; Q.add(v)	*szürke szomszéd kört jelent, a gráf nem fa,*
\ color[v]=grey: fa:=hamis	*fekete szomszéd a csúcs "szülője", nem jelent kört.*
\ color[v]=black: skip	
p:=p->next	
return (fa ∧ c=n)	*ha fa igaz maradt, és minden csúcsot feldolgoztunk, a gráf fa*

7. Adott egy irányítatlan összefüggő gráf, döntsük el a szélességi bejárás segítségével, hogy a gráf páros gráf-e.

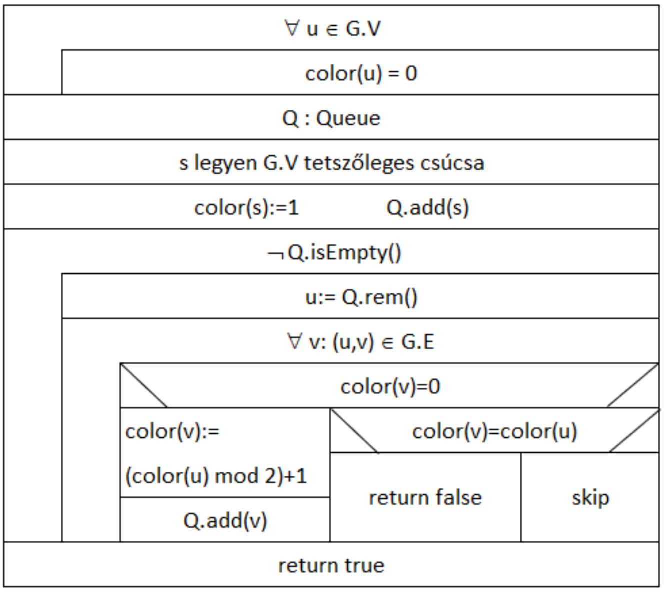
Egy irányítatlan G=(V,E) gráfot akkor nevezünk párosnak, ha a gráf csúcsait két halmazba (V=A∪B) tudjuk osztani úgy, hogy tetszőleges (u,v) él esetén az él végpontjai nem esnek ugyanabba a halmazba, azaz u ∈ A esetén v ∈ B, vagy fordítva.



- Elsőként tegyük fel, hogy a gráf összefüggő (ez nem szükséges feltétele a párosságnak). Szélességi bejárást fogunk használni.
- Három színnel színezzük a gráf csúcsait, kezdetben mindegyik fehér. Amikor egy csúcs látókörbe kerül, piros vagy kék színű lesz, attól függően, hogy milyen színű az a csúcs, amelynek szomszédjaként rátaláltunk.
- A szomszédok vizsgálatánál pedig ellenőrizni kell, hogy nincs-e ugyanolyan színű szomszéd, mint az éppen kiterjesztett csúcs színe, mert ez azt jelenti, hogy a gráf nem páros.
- Az algoritmus páros gráf esetén megad egy lehetséges osztályozást is a csúcsokon: „A” lesz például a kék csúcsok halmaza, „B” pedig a piros csúcsoké.
- A színeket a 0,1,2 egészekkel fogjuk ábrázolni. Csak a color-t használjuk, d, π nem lesz.
- Ábrázolástól függetlenül, az absztrakt gráf típuson oldjuk meg a feladatot.

Az algoritmus:

Páros_e(G: $\mathcal{G}$) : Bool



Minden csúcsot fehérre színezünk.

színek: 0-white; 1-blue; 2-red

Választunk egy tetszőleges csúcsot kezdő csúcsnak, kékre festjük, és berakjuk a sorba.

u lesz a következő kiterjesztett csúcs, szomszédjainak színe szerint:

fehér: ellentett színre állítjuk, és berakjuk a sorba,

ugyanolyan színű, mint u: a gráf nem páros,

ellentett színű: mehet tovább

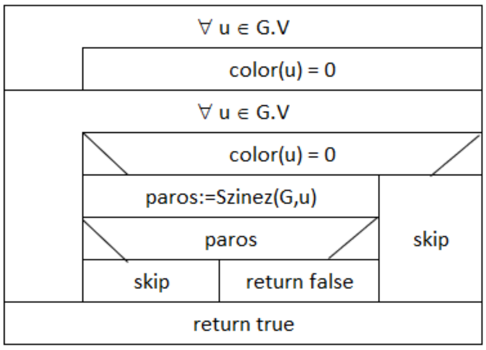
Piros és kék színűek a csúcsok, a gráf páros.

8. Oldjuk meg az előbbi feladatot úgy is, hogy nem biztos, hogy a gráf összefüggő.

- Ha a gráf nem összefüggő, akkor összefüggő komponensekből áll. Ha minden egyes komponenst megvizsgálunk az előbbi módszerrel, és mindegyik páros, a gráf páros.
- Célszerű lesz a szélességi bejárást egy külön algoritmusba kiemelni.
- A megoldás felső szintje gondoskodik a kezdeti fehér szín beállításról minden csúcsra. Majd összefüggő komponensenként megvizsgálja, hogy páros-e a komponens vagy sem, és ezt összegzi. Ezt úgy fogja csinálni, hogy sorba veszi a csúcsokat, és ha fehér színűt talál, az egy még nem feldolgozott komponens egy tetszőleges csúcsa: elindít belőle egy színezést, hogy megvizsgálja páros-e.
- A színezés a szélességi bejárást használja. Az összefüggő komponens bármely csúcsából indítható, megpróbálja piros/kék színekkel kiszínezni a komponens csúcsait. Igazzal vagy hamissal tér vissza aszerint, hogy sikeres volt-e a színezés, vagy nem.
- Visszatérve a felső szintű algoritmusba, ha a komponens páros volt, mehet tovább a csúcsok felsorolása: ha van még fehér csúcs, abból indul egy újabb szélességi színezés. Ha viszont a megvizsgált a komponens nem páros, az algoritmus leáll és nemleges választ ad.

A megoldás felső szintje:

Páros_e(G: $\mathcal{G}$) : Bool



Csúcsokat fehérre állítjuk.

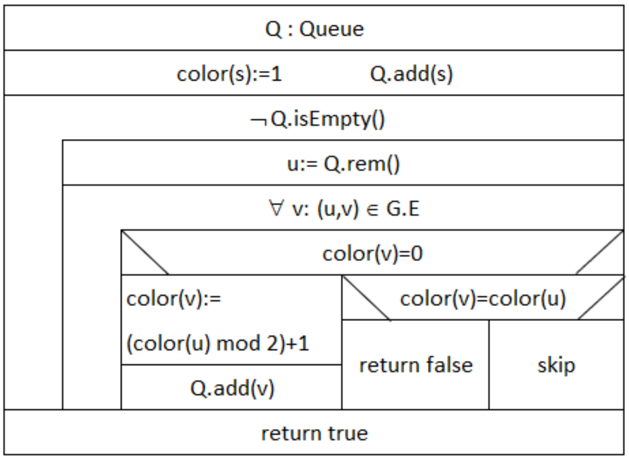
Csúcsonként vizsgálódik,
ha egy fehér csúcsot lát,
elindít egy szinezést.

Ha színezés sikertelen, a gráf
nem páros.

Minden csúcs ki van színezve, a
gráf páros.

A szélességi bejáráson alapuló színezés:

Színez(G : $\mathcal{G}$, s : $\mathcal{V}$) : Bool



A szélességi bejárást használva, megpróbálja
az s-ből elérhető csúcsokat piros és
kék színnel kiszínezni.

Szomszédok vizsgálata az előbb látottak szerint.

Azonos színű szomszéd esetén ez a komponens
nem páros.

Ez az összefüggő komponens páros.