

Szélességi bejárás (szélességi keresés) – gyakorlat

óravázlat

TARTALOM

Tartalom	1
Milyen feladatot old meg az algoritmus, mik az eredmények	2
Milyen gráfokon használható	2
Az algoritmus áttekintése	2
Lejátszás	4
1. Közös megoldott feladat a bejárás szemléltetésére	5
2. Önálló munkára kiadható feladat (bejárás lejátszása önállóan)	5
3. Lefuttattuk a szélességi bejárást egy ismeretlen gráfon, a pi értékeket ismerjük, keressünk benne utat, rajzoljuk le a szélességi fát.	6
Algoritmusok	9
4. Nézzük meg az algoritmust csúcsmátrixos/szomszédossági listás ábrázolásra, tömböket használva d, pi, color értékekre.	9
5. Készítsünk rekurzív / iteratív útkiíró algoritmust.	10
6. Adott egy irányítatlan gráf szomszédossági listás ábrázolásban, döntsük el szélességi bejárás segítségével, hogy a gráf fa-e?	11
7. Adott egy irányítatlan összefüggő gráf, döntsük el a szélességi bejárás segítségével, hogy a gráf páros gráf-e.	12
8. Oldjuk meg az előbbi feladatot úgy is, hogy nem biztos, hogy a gráf összefüggő.	13
Szorgalmi házi feladat	14
9. Sakk: minimum hány ugrással jut a huszár egy start mezőről a cél mezőre?	14

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 , π , $color$)

Műveletigénynél: $G=(V,E)$, és $|V|=n$ és $|E|=m$

BFS($G : \mathcal{G} ; s : \mathcal{V}$)	Műveletigény MT(n,m)	Műveletigény mT(n,m)
$\forall u \in G.V$		
$d(u) := \infty ; \pi(u) := \emptyset$	$\Theta(n)$	$\Theta(n)$
$[color(u) := white]$		
$d(s) := 0 ; [color(s) := grey]$	$\Theta(1)$	$\Theta(1)$
$Q : Queue ; Q.add(s)$	$\Theta(1)$	$\Theta(1)$
$\neg Q.isEmpty()$		
$u := Q.rem()$	$\Theta(n)$	$\Theta(1)$
$\forall v : (u, v) \in G.E$		<i>csak a kezdőcsúcsot dolgozza fel a ciklus</i>
$d(v) = \infty$	$\Theta(m)$	
$d(v) := d(u) + 1$		<i>nullaszer lefutó ciklus is lehet, ha nincs éle a gráfnak az s csúcsból</i>
$\pi(v) := u$		
$[color(v) := grey]$		
$Q.add(v)$		
$[color(u) := black]$	összesítve: $\Theta(n+m)$	összesítve: $\Theta(n)$

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

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

Jegyzet 2. fejezetében definiált típusok¹:

- 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 csúsú 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 címkeket gyakran tömbök reprezentálják.

$d(u)$, $\text{color}(u)$, $\pi(u)$ például ilyen címkek a szélességi bejárásban.

- Élek halmaza (\mathcal{E})

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

- Gráf típus (\mathcal{G})

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

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: $\Theta(n \cdot m)$ -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 $\Theta(m)$ 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 $\Theta(m)$ lépésszám teljesül.
- Sűrű gráfok esetén, csúcsmátrixos ábrázolást használva, a szomszédok bejárása $\Theta(n)$, 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 $\Theta(m)$ korlát ilyenkor is teljesül.

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

- color : általában kihagyható, mivel $d(u) = \infty$ vizsgálat helyettesíti a $\text{color}(u) = \text{white}$ 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 előállítása, 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

- Legyen egy kisebb példa az órán, amin közösen áttekintjük az algoritmus működését, majd önállóan oldjanak meg a hallgatók egy nagyobb (9-10 csúcsból álló gráfon egy lejártszós feladatot).
- Kövessük a jegyzetben megadott szemléltetési módszert, egy táblázattal mutatjuk be az algoritmus működését.

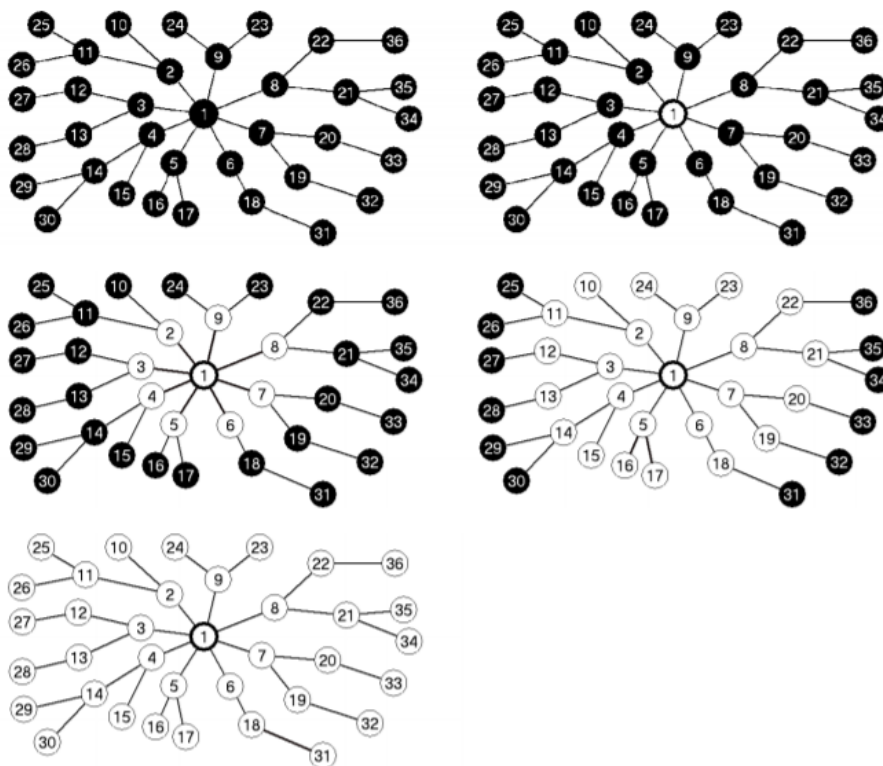
dr Fekete István által készített jegyzetben (https://people.inf.elte.hu/fekete/algorithmusok_jegyzet/) nagyon jó szemléltető ábra van, hogyan terjed egyre nagyobb körökben a kezdő csúcs körül a felderített csúcsok köre. Ebből a jegyzetből másoltam ide egy részletet:

23.1. A szélességi bejárás stratégiája

A bejárás 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 emlékeztető 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 szétében 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.

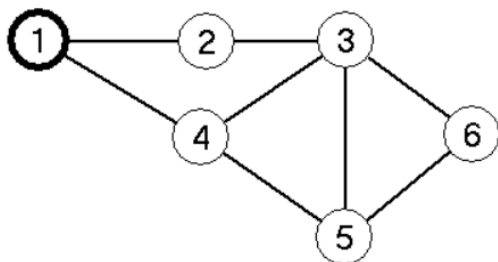


23.1. ábra. Szélességi bejárás egy gráfon

1. Közös megoldott feladat a bejárás szemléltetésére

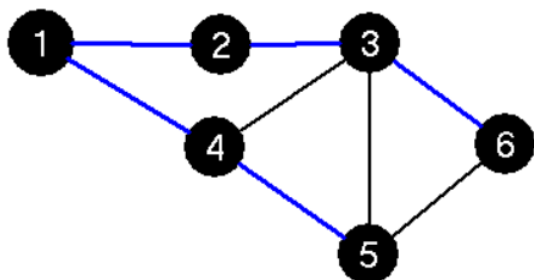
Készíttem egy PPT bemutatót is az itt szereplő példára, azt érdemes levetíteni a gyakorlaton.

A gráf képe (kezdőcsúcs: 1):



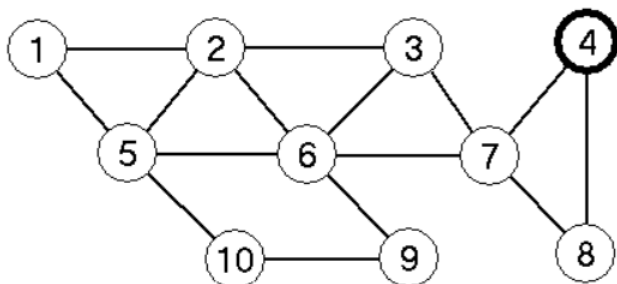
A lejátszáskor kapott táblázat (megtalálható a mellékelt Excel fájlban):

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)

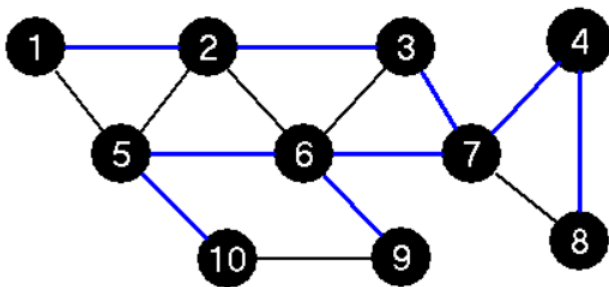
4-es csúcsból indulva játsszuk le a szélességi bejárás algoritmusát.



A táblázat (mellékelve az Excel fájlban):

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 $\pi(5)$ 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 \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

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

2 → 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

4 → 2 → 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

4-es volt a kezdőcsúcs, tehát az út:

4 → 2 → 7 → 11 → 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

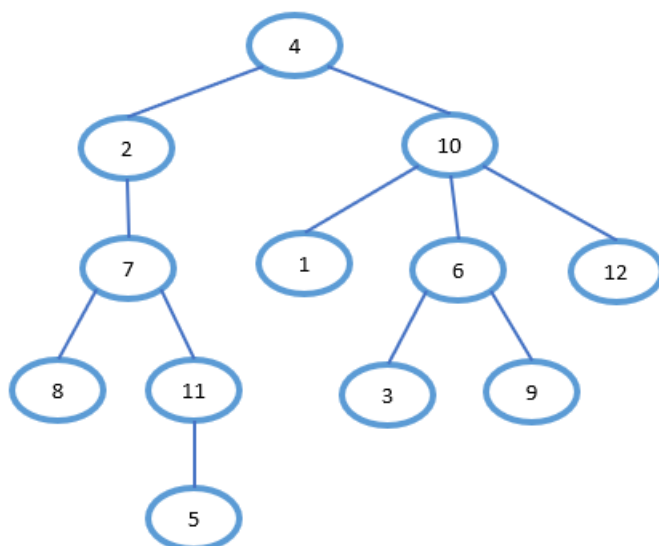
csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	10	4	6	0	11	10	2	7	6	4	7	10

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

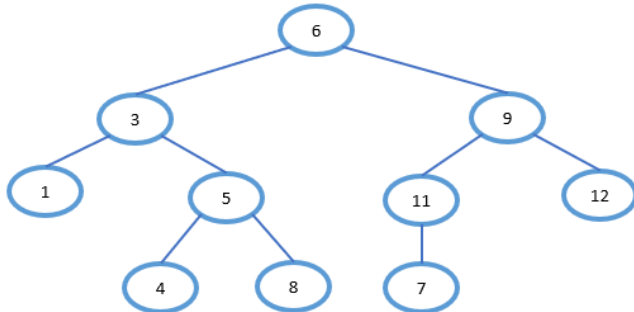
Ezt folytatva a szélességi fa:



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 , π , color értékekre.

Tipp: a szomszédossági listás ábrázolás algoritmusát beszéljük meg közösen, majd a hallgatókat megkérdezzük, hogy hol és miként változna az algoritmus, ha csúcsmátrixszal van a gráf ábrázolva, változna-e a műveletigény? Az elkészítés legyen gyakorló feladat, később megkaphatják a megoldást.

Megállapodások:

- a csúcsok nincsenek ábrázolva, az $1..n$ természetes számok, azonosítják őket,
- a szomszédossági listás ábrázolás az $A[1:\text{Edge}*[n]]$ 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:

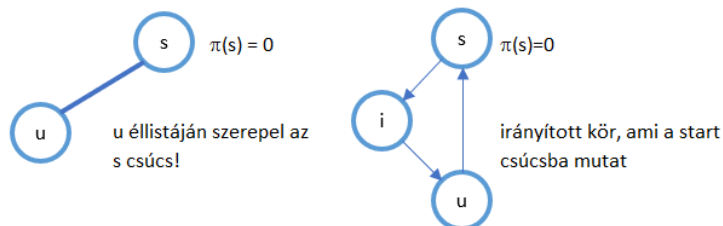
BFS($A[1:\text{Edge}*[n]$; $d[1:N[n]]$; $\pi[1:N[n]]$; $s:1..n$)

$i = 1$ to n	d és π tömbök feltöltése
$d[i] := \infty$; $\pi[i] := 0$	
$Q : \text{Queue}$; $d[s] := 0$	kezdő csúcs d -je legyen nulla
$Q.\text{add}(s)$	berakjuk a sorba a kezdő csúcsot
$\neg Q.\text{isEmpty}()$	
$u := Q.\text{rem}()$	az u csúcs feldolgozása
$p := A[u]$	az u csúcsához tartozó lista első elemére
$p \neq 0$	állítjuk a p pointert
$v := p \rightarrow v$	u szomszédja: $p \rightarrow v$ csúcs
$d[v] = \infty$	ha v csúcs még nem került az
$d[v] := d[u] + 1$	algoritmus látóterébe
$\pi[v] := u$	$d[v]$ és $\pi[v]$ értéket kap
$Q.\text{add}(v)$	v csúcs bekerül a sorba
$p := p \rightarrow \text{next}$	p pointer tovább lép az u csúcs éllistáján

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

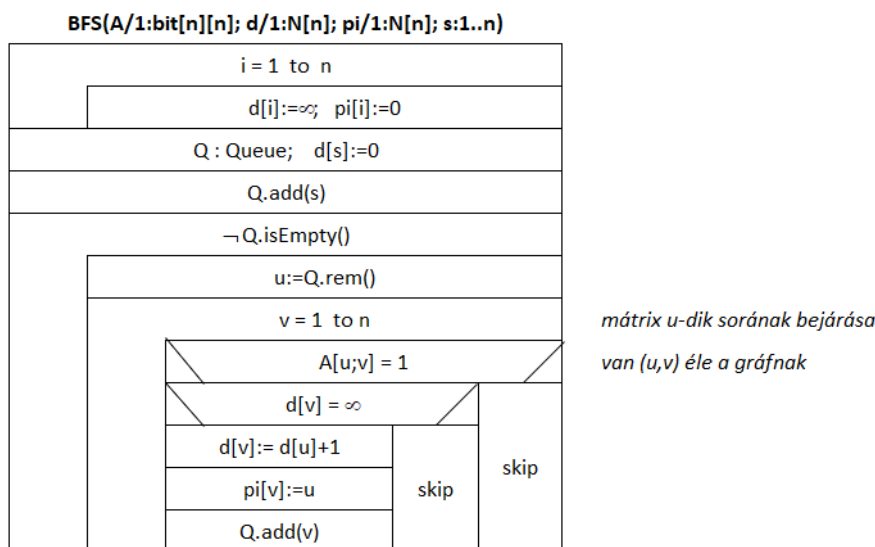
- $d[v] = \infty$ csak azokra a csúcsokra teljesül, amelyek elsőként kerülnek a látókörbe.
- $\pi[v] = 0$ ezeknél a csúcsoknál nyilván teljesül, de nem csak ezeknél a csúcsoknál! A kezdő csúcsra is igaz! Emiatt az algoritmus hibásan működne: a kezdő csúcsot is úgy kezelné, mintha még nem járt volna ott.

Ez irányítatlan esetben a kezdőcsúcs bármely szomszédjánál azonnal hibát okozna, irányított gráfnál pedig, ha van olyan irányított kör, ami a kezdő csúcsra mutat.



Mit és hogyan kell változtatni, ha a gráf az $A[1:\text{bit}[n][n]]$ 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.

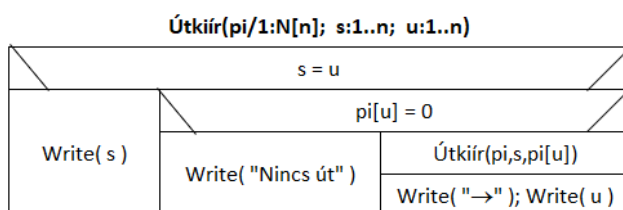


5. Készítsünk rekurzív / iteratív útkiíró algoritmust.

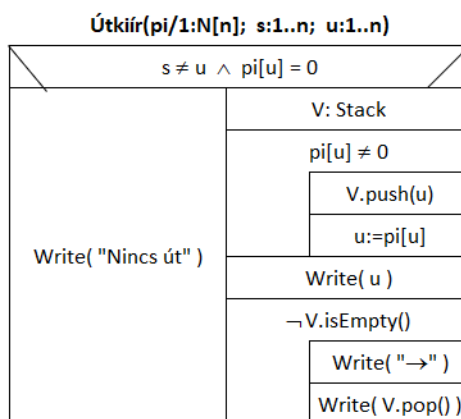
A csúcsokat $1..n$ azonosítja, szülő értékeket egy $pi[1:N[n]]$ tömb, s a kezdőcsúcs, u pedig az a csúcs, amelybe az utat ki akarjuk íratni. Figyeljünk a következőkre: $u=s$ előfordulhat, illetve lehet, hogy a bejárás nem talált utat az u csúcsba, ekkor az algoritmus azt írja ki, hogy „Nincs út.” Az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

Tipp: egyiket oldjuk meg az órán, a másik legyen gyakorló feladat.

Rekurzív megoldás



Iteratív megoldás

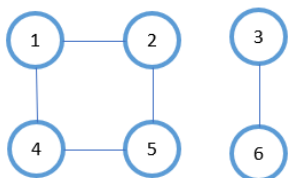


6. Adott egy irányítatlan gráf szomszédossági listás ábrázolásban, döntsük el szélességi bejárás segítségével, hogy a gráf fa-e?

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:

<p>Indítsuk el a bejárást az 1-es csúcsból, figyeljük meg a kör feltűnését.</p>	<p>A 2-es és 3-as csúcsok kiterjesztésénél fekete és fehér szomszédokat fogunk látni. Fekete a szülő csúcs, így nem jelent kört.</p>	<p>Amikor az 5-ös csúcsra 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.</p>

- 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 $v \neq \pi[u]$.
- 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
i = 2 to n	
color[i]:= white	
Q : Queue	color[1]:=grey
Q.add(1)	fa:=igaz c:=0
fa \wedge \neg Q.isEmpty()	
u:=Q.rem()	c:=c+1
color[u]:=black p:= A[u]	
fa \wedge p \neq 0	
v:= p->v	
color[v]=white	color[v]=grey
color[v]=grey	fa:=hamis
Q.add(v)	skip
p:=p->next	
return (fa \wedge c=n)	

szin={white,grey,black}

az 1-es csúcsból indítjuk az algoritmust, a többi fehérre állítjuk

szokásos lépések, fa logikai változó fogja a ciklusokat leállítani, ha a gráf nem fa

c-ben számoljuk a feldolgozott csúcsokat

fehér szomszéd esetén folytatódik a bejárás,

szürke szomszéd kört jelent, a gráf nem fa,

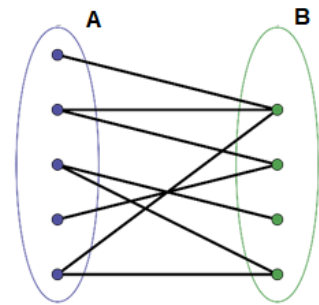
fekete szomszéd a csúcs "szülője", nem jelent kört.

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 ($V=A \cup B$) tudjuk osztani úgy, hogy tetszőleges (u,v) él esetén az él végpontjai nem esnek ugyanabba a halmazba, azaz $u \in A$ esetén $v \in B$, 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:

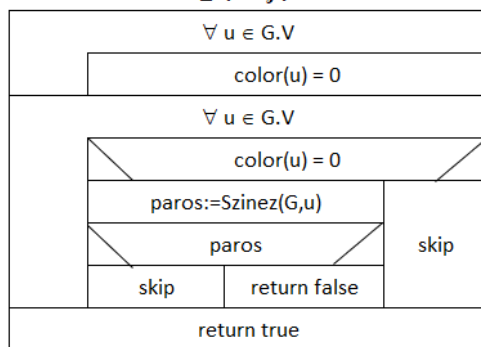
Páros_e(G: G) : Bool	
$\forall u \in G.V$	Minden csúcsot fehérre színezzünk.
color(u) = 0	színek: 0-white; 1-blue; 2-red
Q : Queue	Választunk egy tetszőleges csúcsot kezdő
s legyen G.V tetszőleges csúcsa	csúcsnak, kékre festjük, és berakjuk a
color(s):=1 Q.add(s)	sorba.
$\neg Q.isEmpty()$	
u:= Q.rem()	u lesz a következő kiterjesztett csúcs,
$\forall v: (u,v) \in G.E$	szomszédjainak színe szerint:
color(v)=0	fehér: ellentett színre állítjuk, és berakjuk
color(v):= (color(u) mod 2)+1	a sorba,
Q.add(v)	ugyanolyan színű, mint u: a gráf nem páros,
return false	ellentett színű: mehet tovább
skip	Piros és kék színűek a csúcsok, a gráf páros.
return true	

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 beállításáról 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 színezést.

Ha színezés sikertelen, a gráf

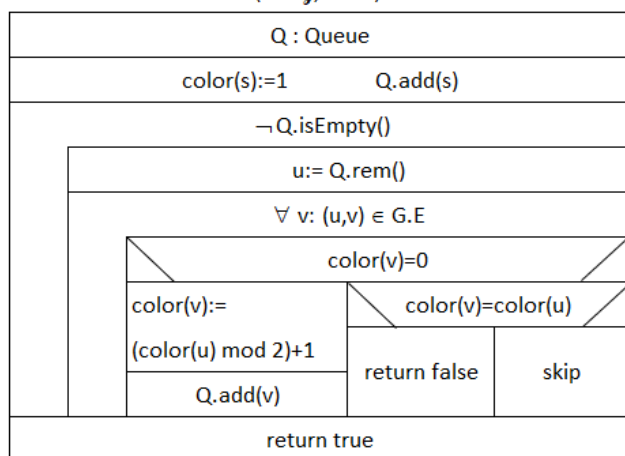
nem páros.

Minden csúcs ki van színezve, a

gráf páros.

A szélességi bejárás alapuló színezés:

Szinez($G: \mathcal{G}, s: \mathcal{V}$) : 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.

SZORGALMI HÁZI FELADAT

9. Sakk: minimum hány ugrással jut a huszár egy start mezőről a cél mezőre?

Sokszor meglepő feladatoknál válik be a szélességi keresés. Első hallásra a feladatnak semmi köze a gráfokhoz. Egy ilyen érdekes feladat a következő:

Adott egy tetszőleges méretű sakktábla, tehát most nem a 8 x 8

-as táblára kell gondolni, hanem egy $n \times m$ -es $n > 0$ és $m > 0$

méretű táblára. Ennek egy adott (i_1, j_1) kockáján áll egy huszár.

El kell juttatni egy (i_2, j_2) kockára, szabályos lépésekkel.

Minimum hány lépésre van szüksége a huszárnak, hogy eljusson a start kockáról a cél kockára?

- Érdemes kicsit közösen gondolkodni, hogy mi köze van a feladatnak a gráfokhoz, és a szélességi bejáráshoz.
- Rá lehet jönni, hogy nem minden esetben lesz megoldása a feladatnak.
- Ha megoldható, akkor általában több út is létezik, de a minimális lépésszám állandó.
- A részletes, és pontos kidolgozásért lehet kicsit több szorgalmi pontot is kitűzni. (pl 5 pont)

A kép illusztráció a feladathoz:

