

Algoritmusok és adatszerkezetek II.

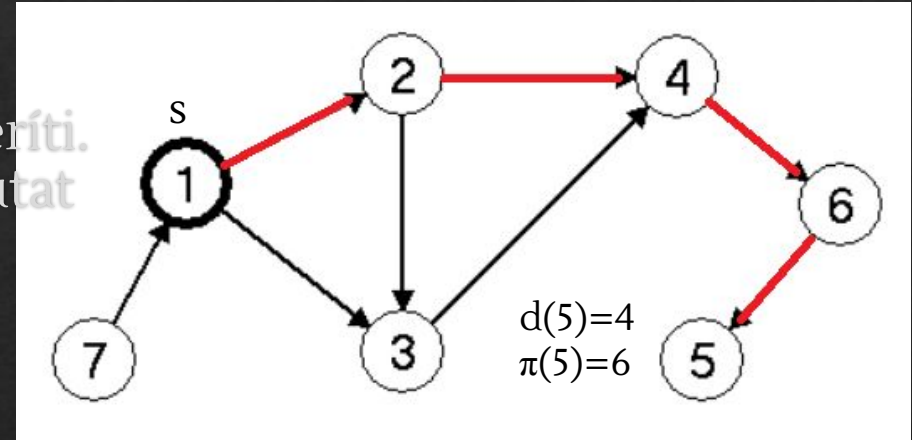
6. gyakorlat

Tartalom:
Szélességi keresés

- ◊ Szorgalmi házi feladatok megoldása
- ◊ Szélességi keresés
- ◊ Lejátszás, példa
- ◊ Lejátszás, gyakorló feladatok
- ◊ Szélességi bejárás szomszédossági listával ábrázolt gráfon
- ◊ Út kiírása
- ◊ Írányítatlan gráf fa-e?
- ◊ Írányítatlan gráf páros-e?
- ◊ Szorgalmi házi feladat

Szélességi keresés (szélességi bejárás)

- ♦ Milyen feladatot old meg?
- ♦ Az algoritmus a gráfokra használt egyik nevezetes bejáró algoritmus: egy tetszőleges kezdő csúcsot megadunk, és az abból úttal elérhető csúcsokat felderítjük. A felderített csúcsokhoz egy minimális hosszúságú utat állít elő a kezdőcsúcsból. Ha több ilyen út lenne, az egyiket.
- ♦ Eredmények:
 - $d(u)$: minimális út hossza (élszáma).
 - ♦ Értéke a kezdő csúcsra 0,
 - ♦ ∞ , 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).
 - ♦ Értéke 0 a kezdőcsúcsnál, valamint azoknál a csúcsoknál, amelyek a kezdőcsúcsból nem érhetők el.



Milyen gráfokon használható?

- ◆ Tetszőleges (nem üres) gráfon használható:
 - irányított vagy irányítatlan
 - összefüggő, nem összefüggő

Csúcsok színezése

- ◆ A bejáró algoritmusok, hogy ne „körözzenek” gyakran színezést használnak.
- ◆ Színezés jelentése: 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.

Már felderített, de még nem kész csúcsokat tároló adatszerkezet egy SOR

- A szürke csúcsok vannak a sorban, onnan kerülnek ki feldolgozásra, ez okozza a jellegzetes „szélességi köröket” a kezdő csúcs körül a bejárásnál.

Az algoritmus, és a műveletigénye

<div style="text-align: center;"> $\text{BFS}(G : \mathcal{G} ; s : \mathcal{V})$ </div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> $\forall u \in G.V$ <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $d(u) := \infty ; \pi(u) := \emptyset$ </div> <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $[color(u) := white]$ </div> $d(s) := 0 ; [color(s) := grey]$ $Q : Queue ; Q.add(s)$ $\neg Q.isEmpty()$ <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $u := Q.rem()$ <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $\forall v : (u, v) \in G.E$ <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $d(v) = \infty$ </div> <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $d(v) := d(u) + 1$ </div> <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $\pi(v) := u$ </div> <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $[color(v) := grey]$ </div> <div style="border: 1px solid black; padding: 2px; margin: 2px;"> $Q.add(v)$ </div> <div style="border: 1px solid black; padding: 2px; margin: 2px; text-align: center;"> SKIP </div> </div> $[color(u) := black]$ </div> </div>	<div>Műveletigény</div> <div>MT(n,m)</div> <div>$\Theta(n)$</div> <div>$\Theta(1)$</div> <div>$\Theta(1)$</div> <div>$\Theta(n)$</div> <div>$\Theta(m)$</div> <div>összesítve: $\Theta(n+m)$</div>	<div>Műveletigény</div> <div>mT(n,m)</div> <div>$\Theta(n)$</div> <div>$\Theta(1)$</div> <div>$\Theta(1)$</div> <div>$\Theta(1)$</div> <div>csak a kezdőcsúcsot dolgozza fel a ciklus</div> <div>nullaszer lefutó ciklus is lehet, ha nincs éle a gráfnak az s csúcsból</div> <div>összesítve: $\Theta(n)$</div>
---	--	---

$G=(V,E)$
 $|V|=n$
 $|E|=m$

Az absztrakt gráf típus

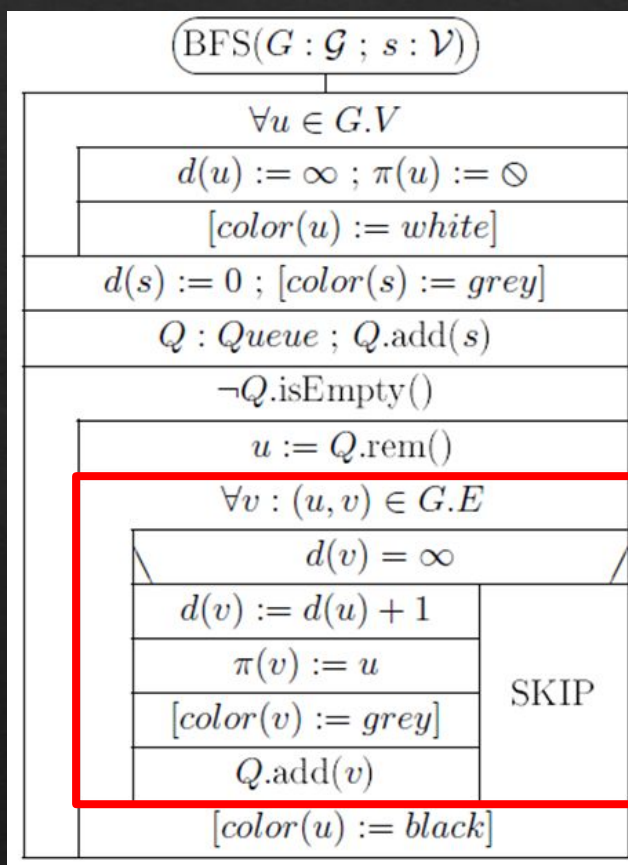
- ♦ A fejlécben található típusok jelentése $\text{BFS}(G : \mathcal{G} ; s : \mathcal{V})$
- ♦ A gráfok absztrakt algoritmusainak leírásához bevezetjük a V (vertex, azaz csúcs) absztrakt típust. A V lesz a gráfok csúcsainak absztrakt típusa. Ez egy olyan elemi típus, amelyben mindegyik csúcshoz tetszőlegesen sok, névvel jelölt címke társítható, és mindegyik címkéhez tartozik valamilyen érték.
- ♦ A 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 címkeket gyakran tömbök reprezentálják.
- ♦ $d(u)$, $\text{color}(u)$, $\pi(u)$ például ilyen címkek a szélességi bejárásban.
- ♦ Élek halmaza (E)

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

- ♦ Gráf típus (G)

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

Hatékonyság és gráf ábrázolás összefüggése



- \diamond A $\forall (u,v) \in G.E$ 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?

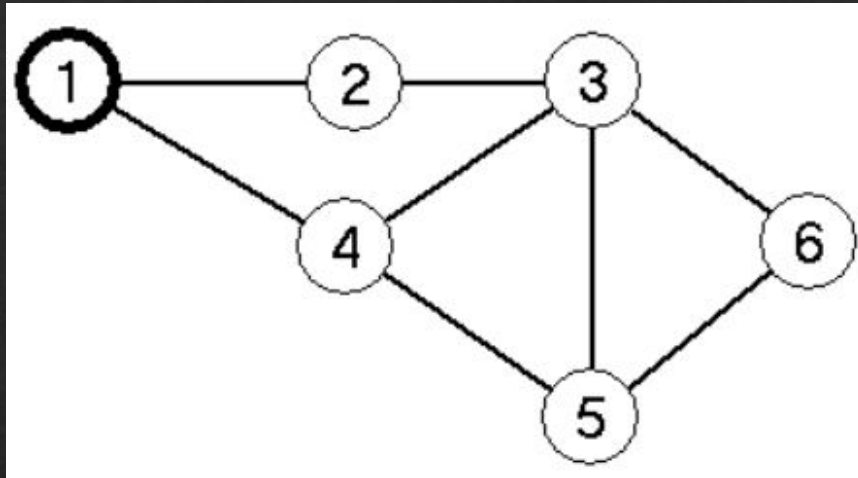
- Mert akkor egy menet $\Theta(m)$ lenne.
- Ha minden csúcsra végrehajtódik: $\Theta(n*m)$ lenne a szélességi keresés műveletigénye, azaz $MT(n,m) = \Theta(n*m)$
- Ami sűrű gráf esetén már $\Theta(n^3)$ lépésszámot jelent!

- \diamond **Ritka gráf, éllista ábrázolás:**
 $A[u]$ pointerű egyszerű listát kell bejárnia, ha minden csúcsra lefut, akkor is a műveletigénye csak: $\Theta(m)$
- \diamond **Sűrű gráf, csúcsmátrixos ábrázolás:**
 u szomszédainak feldolgozásához a mátrix u -dik sorát járja be, ha minden csúcsra lefut, akkor a műveletigény: $\Theta(n^2)$, de mivel sűrű gráfok esetén az élek száma: $m \in \Theta(n^2)$, így ez szintén $\Theta(m)$

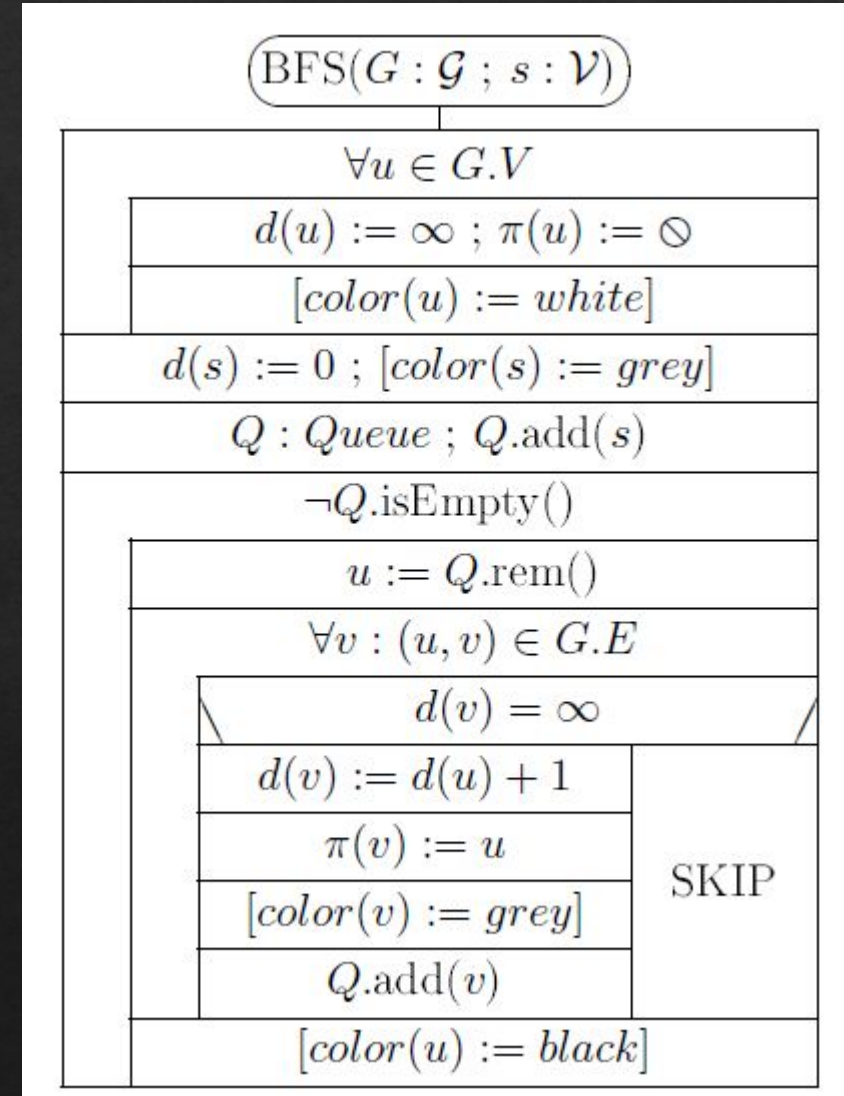
A csúcsok mely jellemzője (címkéje) hagyható el az algoritmusból (d , π , color):

- ♦ 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.

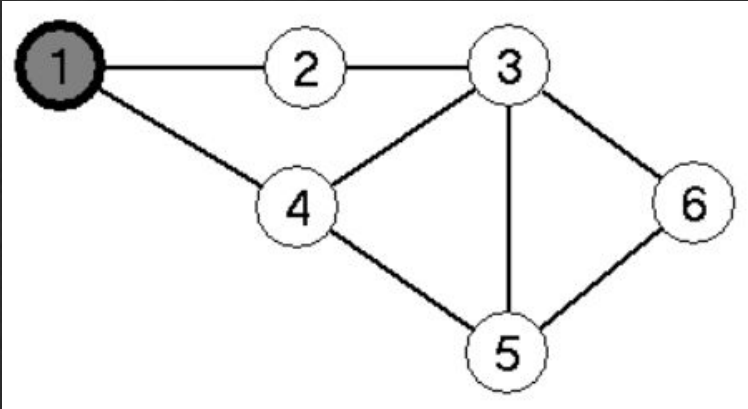
Mutassuk be a szélességi bejárás működését a megadott gráfon



- ◊ Kezdőcsúcs legyen az 1-es csúcs
- ◊ **FONTOS:** lejátszásnál a csúcs szomszédjait mindig nagyság szerint növekvő sorrendben fogjuk feldolgozni!



- Elindul az algoritmus, feltölti a kezdőértékekkel a gráf csúcsait (a csúcsokhoz tartozó címkéket).
- A sorba betesszük az 1-es csúcsot, színe:

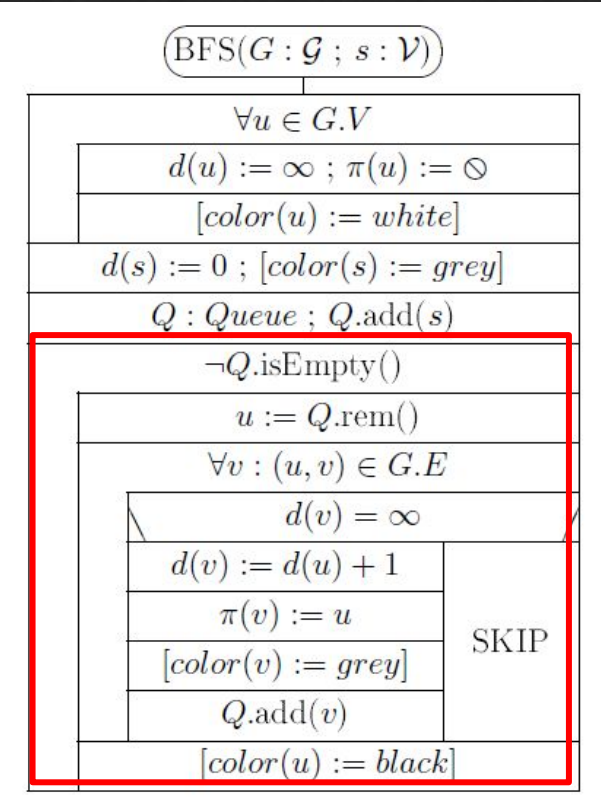
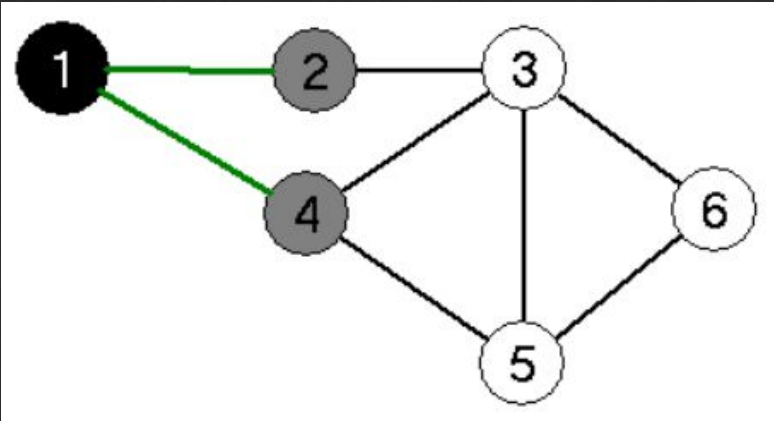


S

$\text{BFS}(G : \mathcal{G} ; s : \mathcal{V})$	
$\forall u \in G.V$	
$d(u) := \infty ; \pi(u) := \emptyset$	
$[color(u) := white]$	
$d(s) := 0 ; [color(s) := grey]$	
$Q : Queue ; Q.add(s)$	
$\neg Q.isEmpty()$	
$u := Q.rem()$	
$\forall v : (u, v) \in G.E$	
<div style="display: flex; justify-content: space-between; align-items: center;"> $d(v) = \infty$ <div style="border-left: 1px solid black; border-right: 1px solid black; width: 10px;"></div> </div>	
$d(v) := d(u) + 1$	<div style="font-size: 2em; font-weight: bold; text-align: center;">SKIP</div>
$\pi(v) := u$	
$[color(v) := grey]$	
$Q.add(v)$	
$[color(u) := black]$	

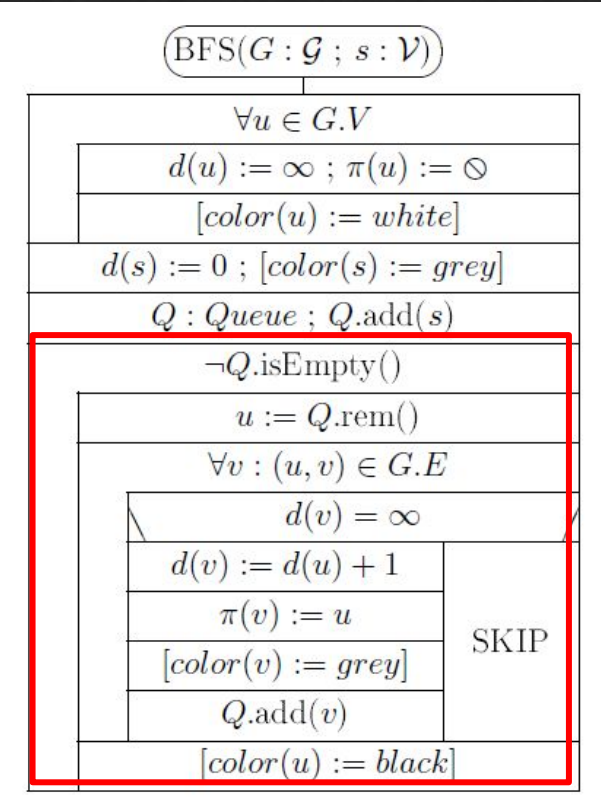
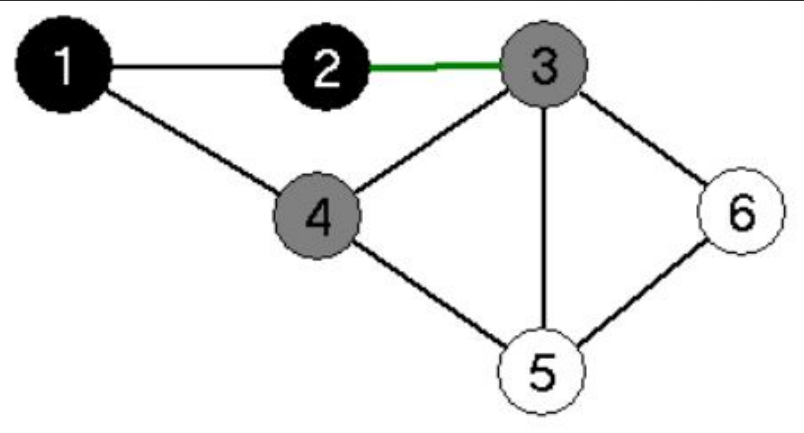
Kiterjesztett csúcs	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0

- ◊ Kiveszi a sorból az 1-es csúcsot.
- ◊ Szomszédok: 2 és 4
- ◊ Mindkettő fehér, szürkére színezi őket, d értékük $d(1)+1=1$, lesz, szülőjük 1.
- ◊ 2 és 4 bekerülnek a sorba.
- ◊ 1 színe fekete lesz



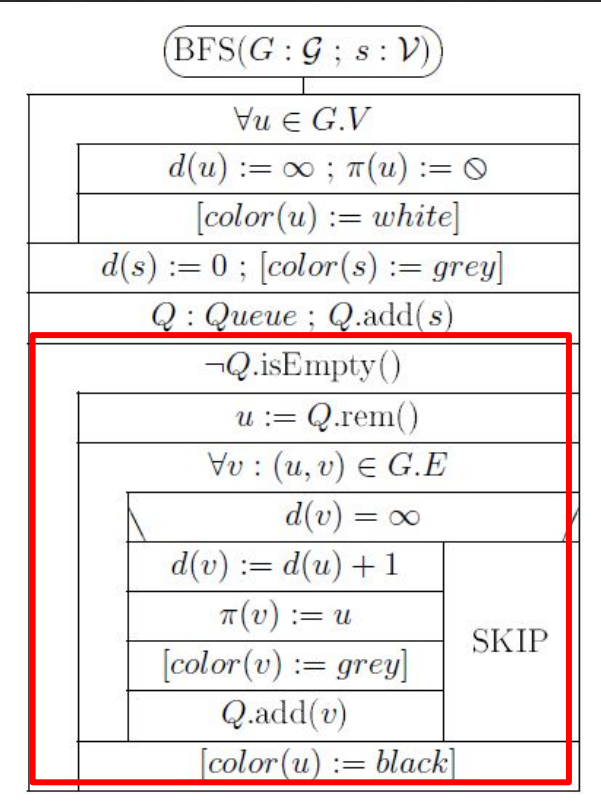
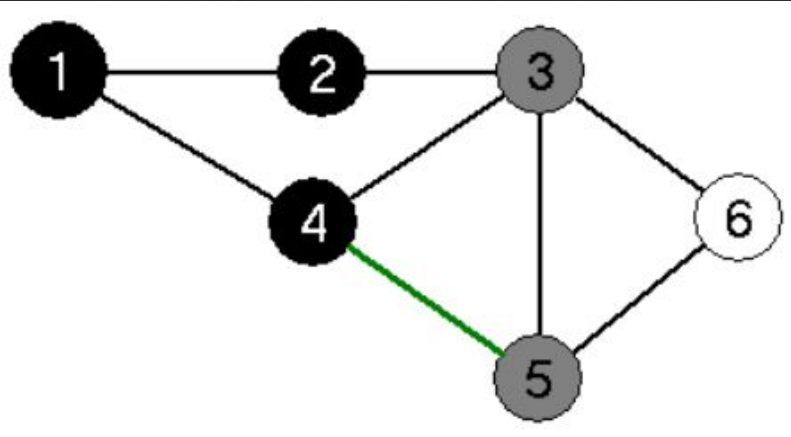
Kiterjesztett csúcs	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0
1, d:0		1		1			< 2;4 >		1		1		

- ◊ Kiveszi a sorból az 2-es csúcsot.
- ◊ Szomszédok: 1 és 3
- ◊ 1 fekete, kész csúcs
- ◊ 3 fehér: szürkére színezi, d értéke $d(2)+1=2$, lesz, szülője 2.
- ◊ 3 bekerül a sorba, 2 színe fekete lesz.



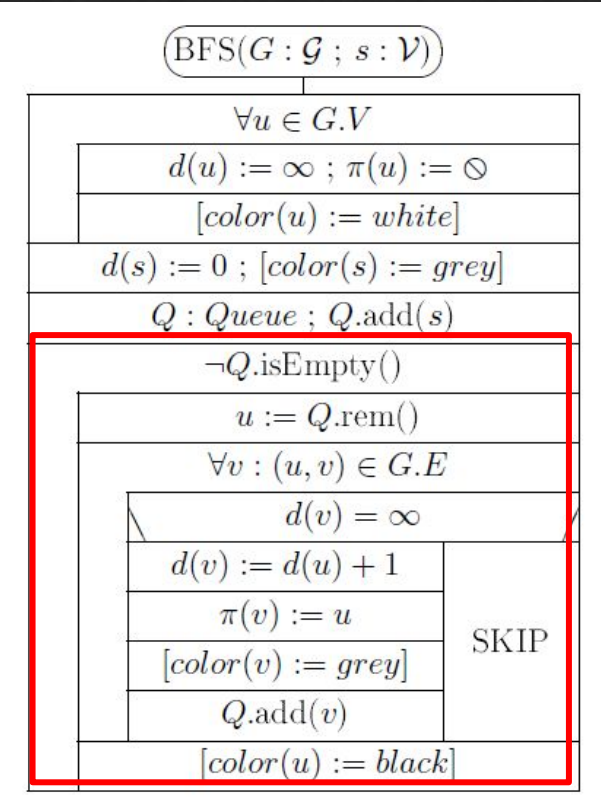
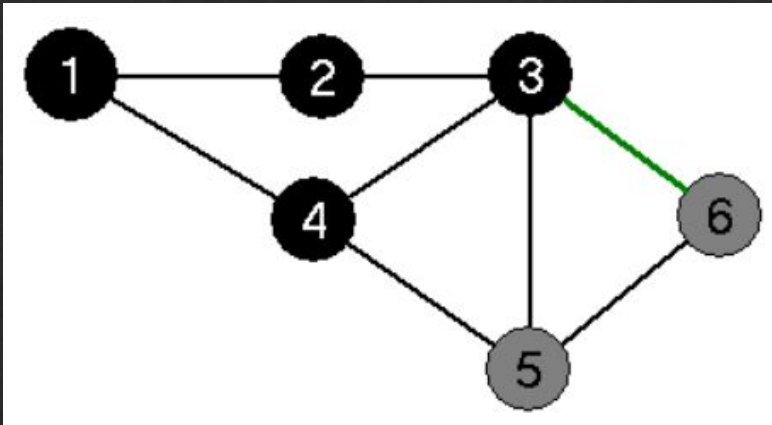
Kiterjesztett	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
csúcs	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0
1, d:0		1		1			< 2;4 >		1		1		
2, d:1			2				< 4;3 >			2			

- ◊ Kiveszi a sorból az 4-es csúcsot.
- ◊ Szomszédok: 1 és 3 és 5
- ◊ 1 fekete, 3 szürke, skip ágon fut
- ◊ 5 fehér: szürkére színezi, d értéke $d(4)+1=2$, lesz, szülője 4.
- ◊ 5 bekerül a sorba, 4 színe fekete lesz.



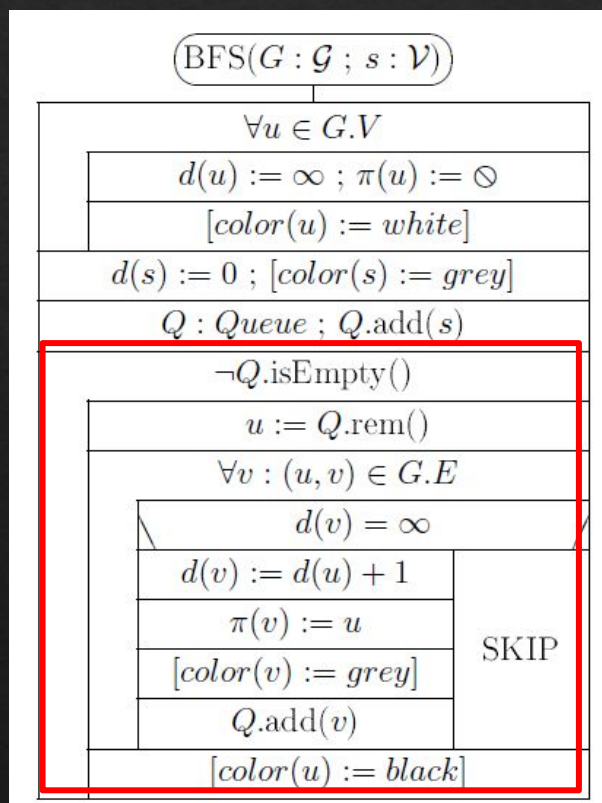
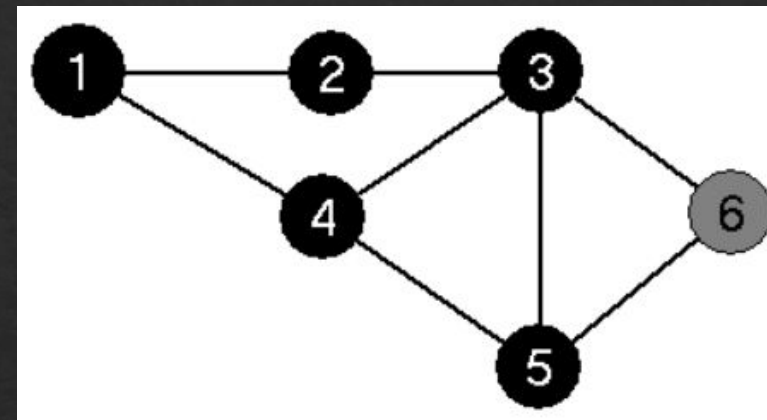
Kiterjesztett csúcs	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0
1, d:0		1		1			< 2;4 >		1		1		
2, d:1			2				< 4;3 >			2			
4, d:1					2		< 3;5 >					4	

- ◊ Kiveszi a sorból a 3-as csúcsot.
- ◊ Szomszédok: 2, 4, 5 és 6
- ◊ 2 és 4 fekete, 5 szürke: skip ágon fut
- ◊ 6 fehér: szürkére színezi, d értéke $d(3)+1=3$, lesz, szülője 3.
- ◊ 6 bekerül a sorba, 3 színe fekete lesz.



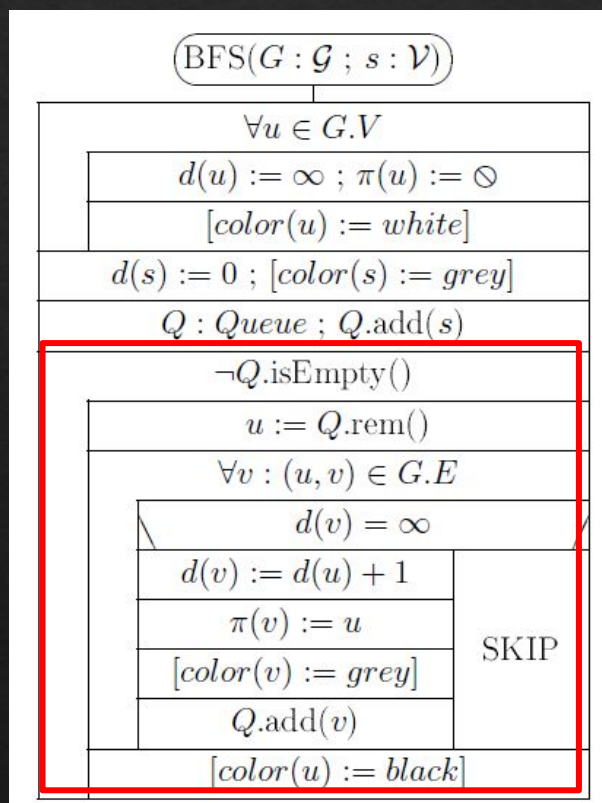
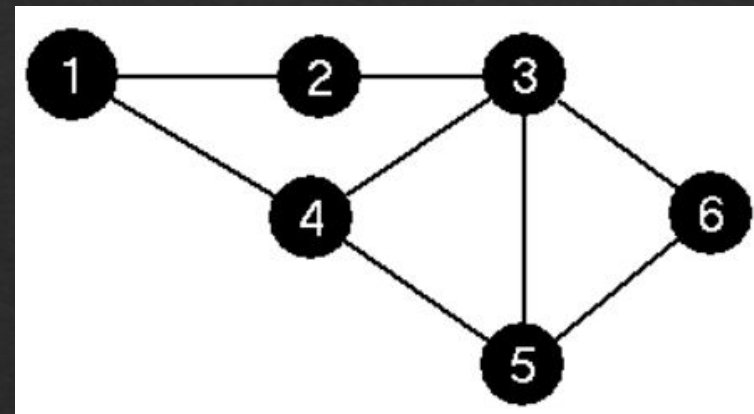
Kiterjesztett csúcs	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0
1, d:0		1		1			< 2;4 >		1		1		
2, d:1			2				< 4;3 >			2			
4, d:1					2		< 3;5 >					4	
3, d:2						3	< 5;6 >						3

- ◊ Kiveszi a sorból a 5-ös csúcsot.
- ◊ Szomszédok: 3, 4 és 6
- ◊ 3 és 4 fekete, 6 szürke: skip ágon fut
- ◊ d és π értékek nem változnak
- ◊ 5 színe fekete lesz, a sorba nem kerül új csúcs



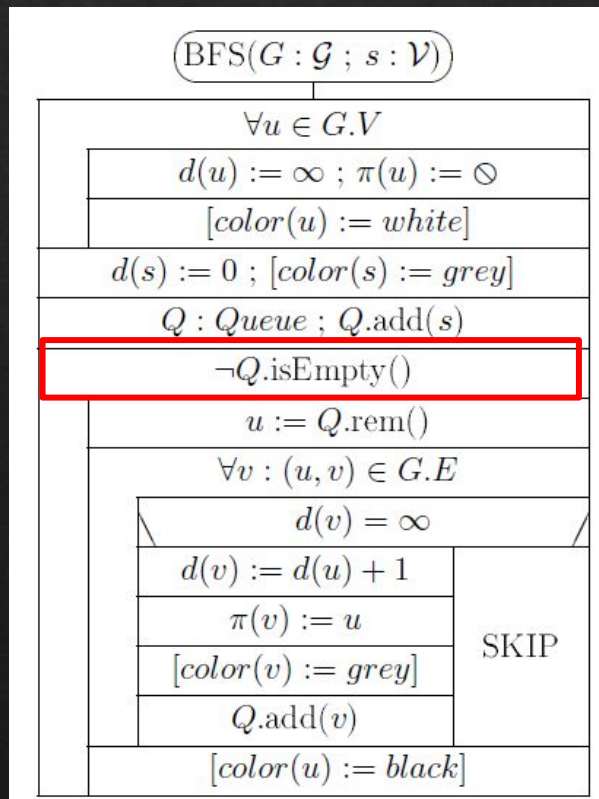
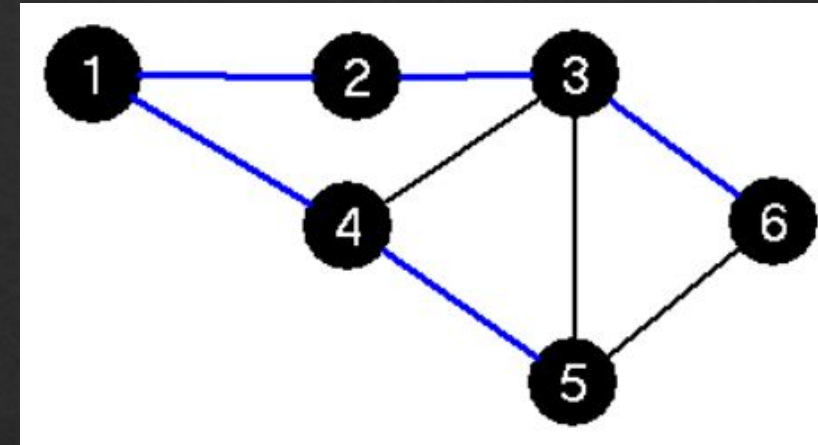
Kiterjesztett csúcs	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0
1, d:0		1		1			< 2;4 >		1		1		
2, d:1			2				< 4;3 >			2			
4, d:1					2		< 3;5 >					4	
3, d:2						3	< 5;6 >						3
5, d:2							< 6 >						

- ◊ Kiveszi a sorból a 6-os csúcsot.
- ◊ Szomszédok: 3 és 5
- ◊ Mindkettő fekete: skip ágon fut
- ◊ d és π értékek nem változnak
- ◊ 6 színe fekete lesz, a sorba nem kerül új csúcs



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							< >						

- ◊ A sor üres, véget ért a főciklus.
- ◊ A csúcsok d és π értékeiből kiolvashatók az eredmények.
- ◊ Vizsgáljuk meg a kapott szélességi fát.

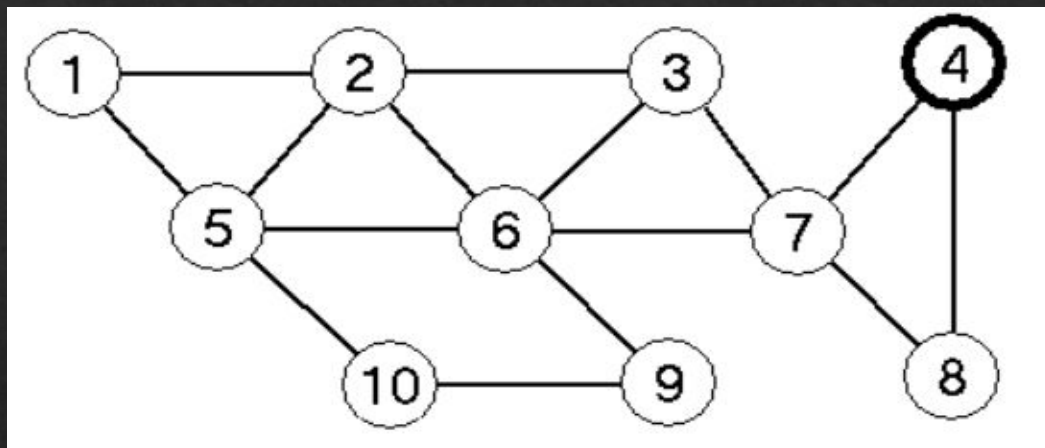


Kiterjesztett csúcs	csúcsok d értékei						Sor tartalma	csúcsok π értékei					
	1	2	3	4	5	6		1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	< 1 >	0	0	0	0	0	0
1, d:0		1		1			< 2;4 >		1		1		
2, d:1			2				< 4;3 >			2			
4, d:1					2		< 3;5 >					4	
3, d:2						3	< 5;6 >						3
5, d:2							< 6 >						
6, d:3							< >					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

Vége

1. gyakorló feladat

- ◊ Az előző példához hasonlóan mutassuk be a szélességi bejárás algoritmusát az alábbi gráfon:



Kezdő csúcs legyen a 4-es.

- ◊ Segítség: `szelessegi_gyakorlo.xlsx`
- ◊ Rajzoljuk be a kapott szélességi fát a gráfba.

2. gyakorló feladat

- ♦ Egy ismeretlen gráfon lefuttattuk a szélességi bejárást. A szülő értékek ismertek, az alábbi táblázat szerinti

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 → 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 → 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

3. gyakorló feladat

- ♦ 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

Rajzoljuk le 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

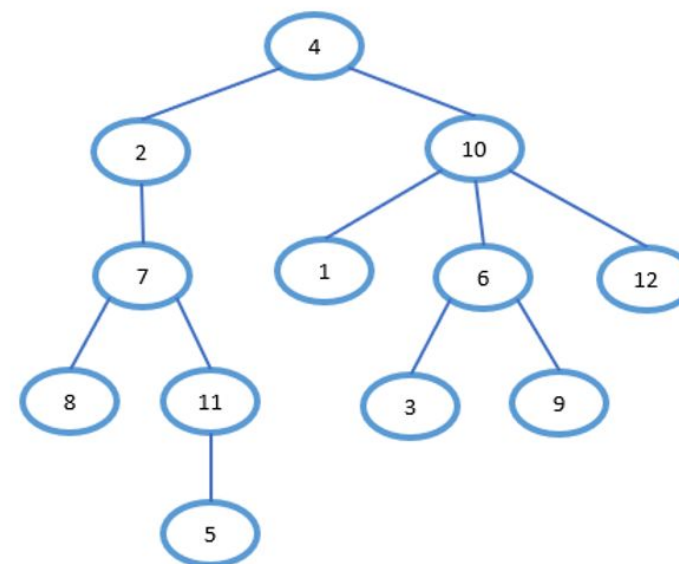
csúcs:	1	2	3	4	5	6	7	8	9	10	11	12
szülő:	10	4	6	0	11	10	2	7	6	4	7	10

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

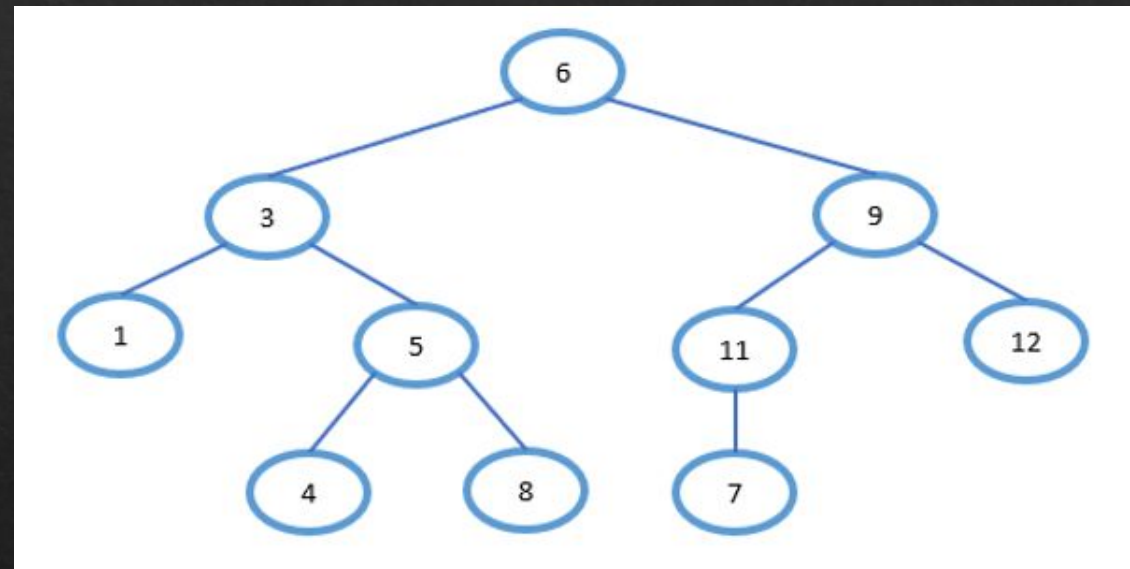
2 gyereke: 7 10 gyerekei: 1, 6 és 12

Ezt folytatva a szélességi fa:



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

- ◊ Mi lehetett a fa?
- ◊ Több nulla is van a szülő értékek között!
- ◊ Csak 6 lehetett a gyökér, mert 2 és 10 nem fordul elő szülőként.
- ◊ Azaz nem vezet út a 6-os csúcsból 2 és 10 csúcsokba.



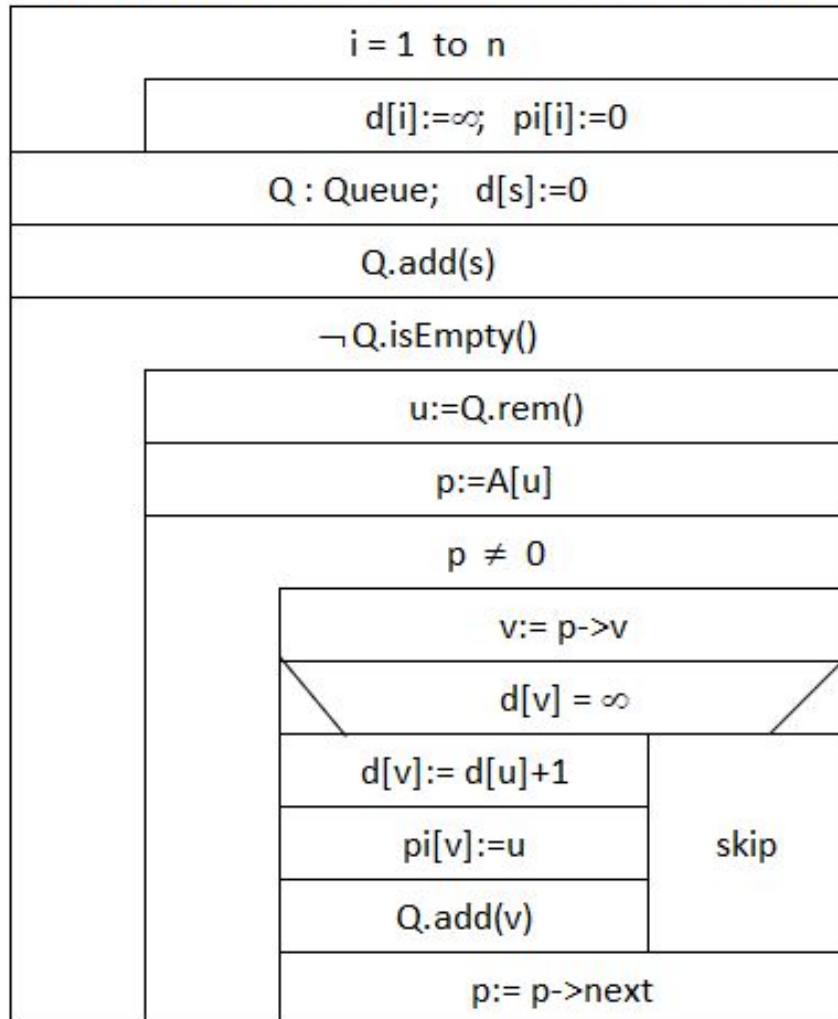
Készítsük el a szélességi keresés algoritmusát csúcsmátrixos/szomszédossági listás ábrázolásra, tömböket használva d , π , $color$ értékekre.

◆ **Szomszédossági listás ábrázolással:**

- 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.

Szélességi keresés szomszédossági listával ábrázolt gráfon

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 csúcsához tartozó lista első elemére

állítjuk a p pointert

u szomszédja: p->v csúcs

ha v csúcs még nem került az

algorithmus látóterébe

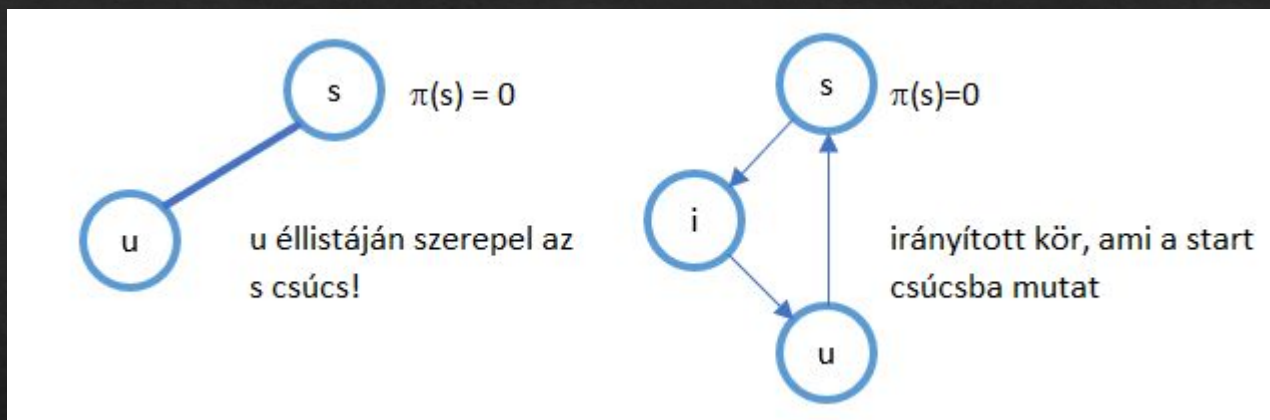
d[v] és pi[v] értéket kap

v csúcs bekerül a sorba

p pointer tovább lép az u csúcs éllistáján

Használhatnánk-e a
d[v] = ∞ feltétel
helyett a pi[v] = 0
feltételt?

- ◊ Válasz: nem
- ◊ A feltétel azt ellenőrzi, hogy az adott csúccsal találkoztunk-e már.
- ◊ Ha a csúcs $d(v) = \infty$, az akkor és csak akkor fordul elő, ha a csúcsot eddig még nem fedeztük fel.
- ◊ Viszont $\pi(v)=0$ nem jelenti azt, hogy a csúccsal még nem találkoztunk!

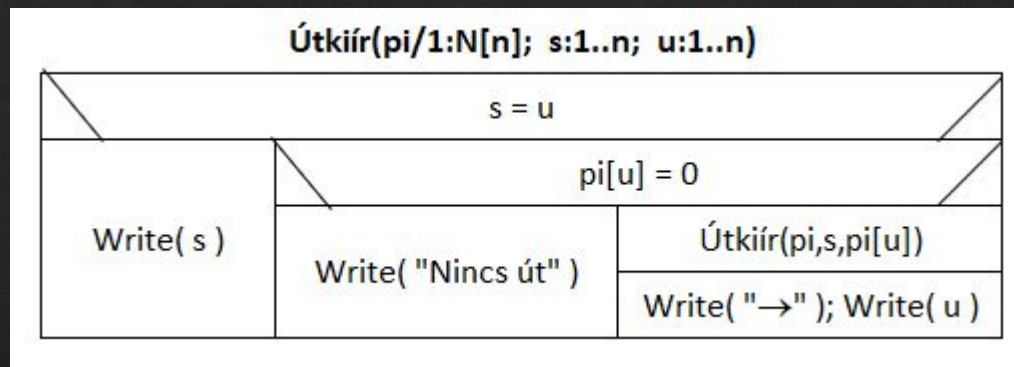


4. gyakorló feladat

- ◆ Készítsük el a szálességi bejárás algoritmusát csúcsmátrixszal ábrázolt gráfra.
- ◆ Vizsgáljuk meg a kapott algoritmus műveletigényét!

Út kiíró algoritmus

- 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 írunk egy „→” karaktert.
- Rekurzív változat:



- Gyakorló feladatként készítsük el az iteratív változatot (verem segítségével)!

Szélességi kereséssel megoldható feladatok

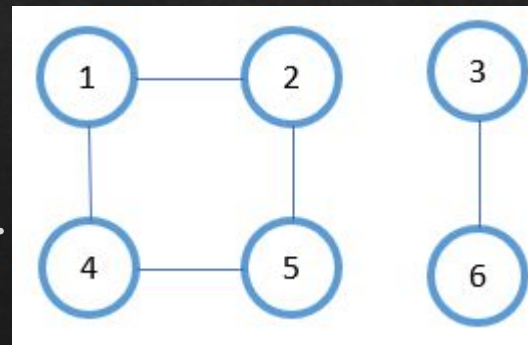
- ◆ Számos gráfokkal kapcsolatos feladat megoldásának alapja lehet a szélességi bejárás.

Nézzünk meg példaként néhány ilyen feladatot:

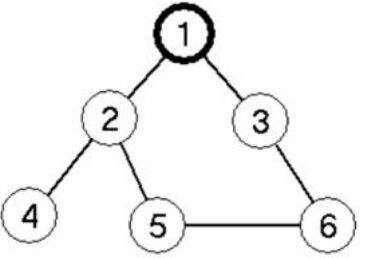
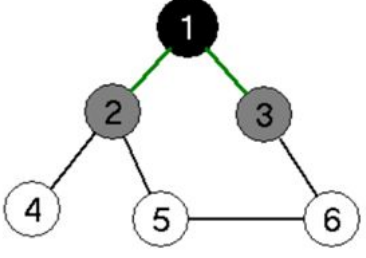
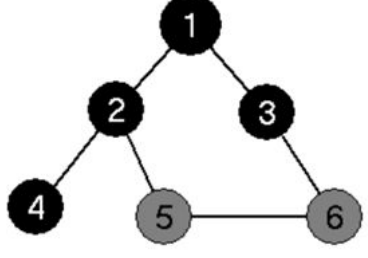
- Adott egy tetszőleges irányítatlan gráf, döntsük el, hogy fa-e!
- Adott egy tetszőleges irányítatlan, összefüggő gráf, döntsük el, hogy páros-e!
- Adott egy tetszőleges irányítatlan gráf, döntsük el, hogy páros-e! A gráf nem biztos, hogy összefüggő!

Írányítatlan 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.
- ◊ Jó ez?
- ◊ 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 talákoztunk 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 $\text{szin}[v] = \text{grey}$.
- 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 $\text{szin}[v] \neq \text{white}$ és $v \neq \pi[u]$.
- Azt, hogy minden csúcsot meglátogattunk-e, egyszerű számlálással fogjuk ellenőrizni: bevezetünk egy számlálót, mely a feldolgozott csúcsokat megszámlálja. Ha ez végül n , akkor minden csúcsot feldolgoztunk.

A megoldás

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

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

szin={white,grey,black}

az 1-es csúcsból indítjuk az algoritmust, a

többet 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,

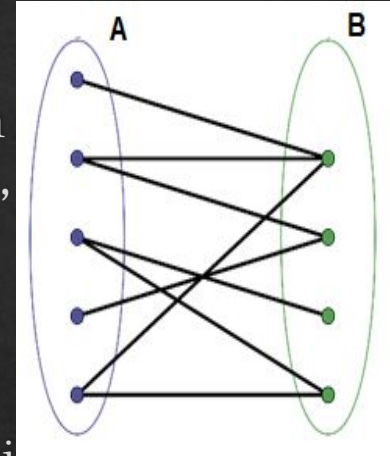
szürke szomszéd kört jelent, a gráf nem fa,

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

ha fa igaz maradt, és minden csúcsot feldolgoztunk, a gráf fa

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 jelentené, 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 egészszekkel fogjuk ábrázolni. Csak a color-t használjuk, d, π nem lesz.
- ◊ Ábrázolástól függetlenül, az absztrakt gráf típuson oldjuk meg a feladatot.

A megoldás

Páros_e($G: \mathcal{G}$) : Bool

$\forall u \in G.V$		
color(u) = 0		
Q : Queue		
s legyen G.V tetszőleges csúcsa		
color(s):=1		Q.add(s)
$\neg Q.isEmpty()$		
u:= Q.rem()		
$\forall v: (u,v) \in G.E$		
color(v)=0		
color(v):=	color(v)=color(u)	
(color(u) mod 2)+1	return false	skip
Q.add(v)		
return true		

Minden csúcsot fehérre színezzük.

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,

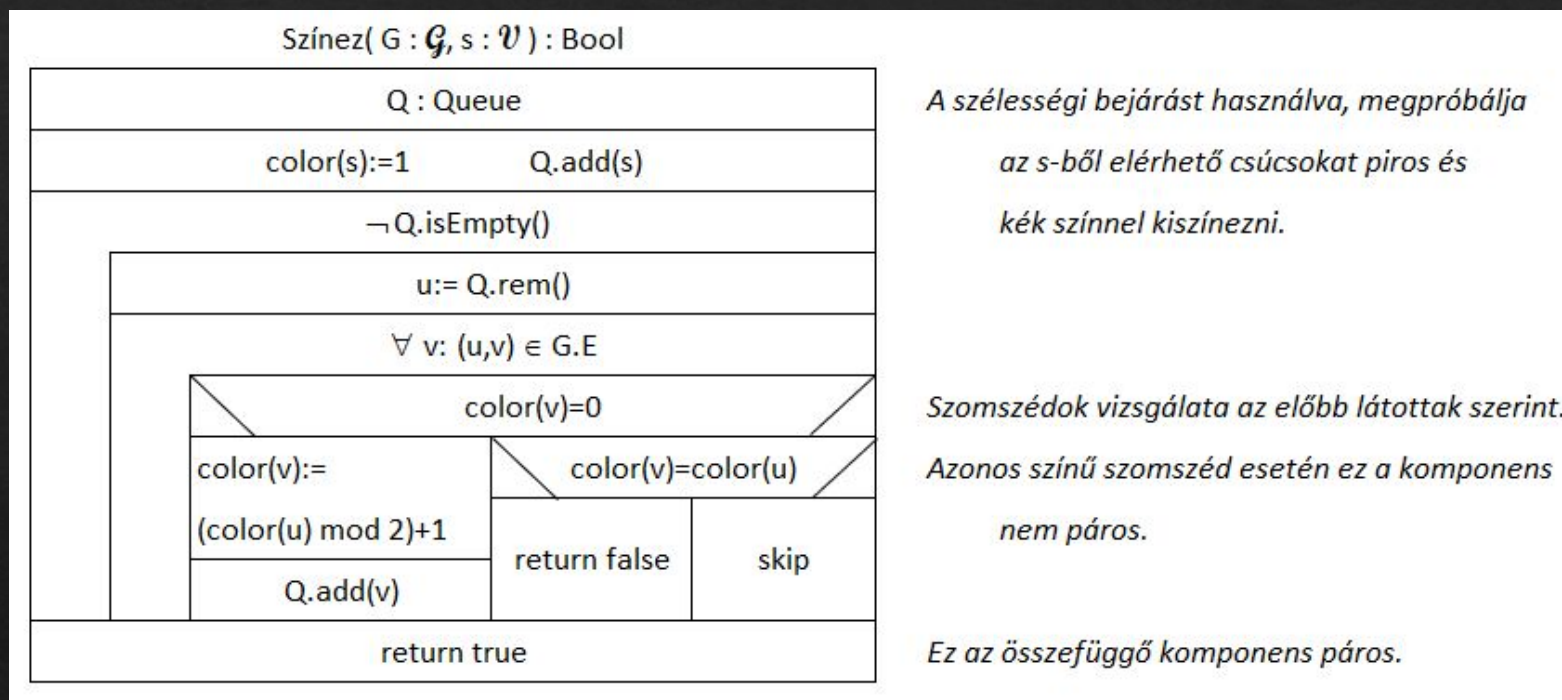
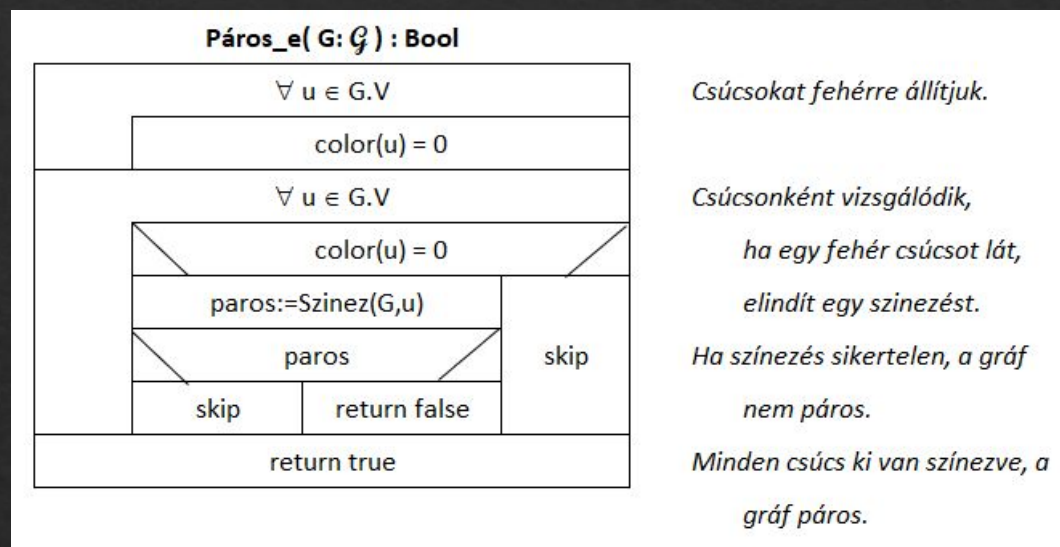
ugyanolyan színű, mint u: a gráf nem páros, ellentett színű: mehet tovább

Piros és kék színűek a csúcsok, a gráf páros.

Vizsgáljuk meg úgy a párosságot, hogy nem tudjuk a gráfról, hogy összefüggő-e.

- ◊ Ha a gráf nem összefüggő, akkor összefüggő komponensekből áll. Ha minden egyes komponenst megvizsgálunk az előbbi módszerrel, és ha mindegyik páros, a gráf páros.
- ◊ Két szintre bontjuk az algoritmust, a szélességi bejárást egy külön algoritmusba kiemeljük.
- ◊ 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, akkor 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:



Szorgalmi házi feladat

- ◆ 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?
 - Mi köze van a feladatnak a szélességi bejáráshoz?
 - Mindig megoldható?
 - Több út is lehetséges, melyik az igazi?

