Gráfok
3. előadás

Szélességi bejárás alkalmazásai

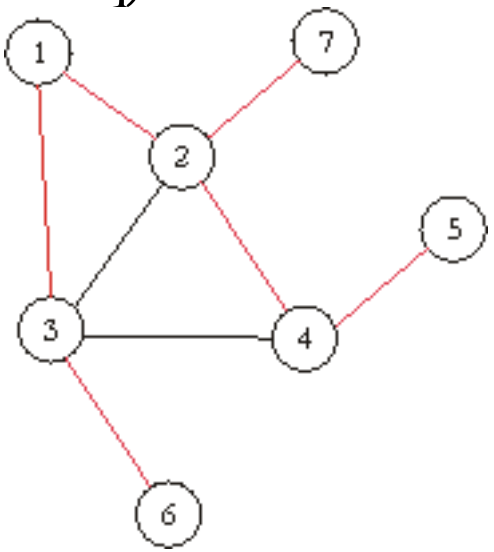
Legrövidebb utak minden pontba (p→q)

- Szélességi keresés a p. csúcsból.
- A Honnan vektor alapján bármely csúcsból visszafelé haladva megkapjuk az oda vezető legrövidebb utat.

Adott ponton áthaladó legrövidebb út (p→x→q)

- Legrövidebb út keresés p-ből x-be.
- Legrövidebb út keresés x-ből q-ba.

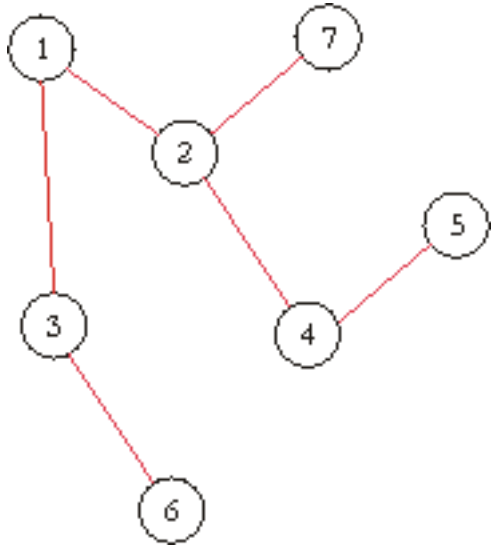
Probléma: így a két útnak lehet közös szakasza!

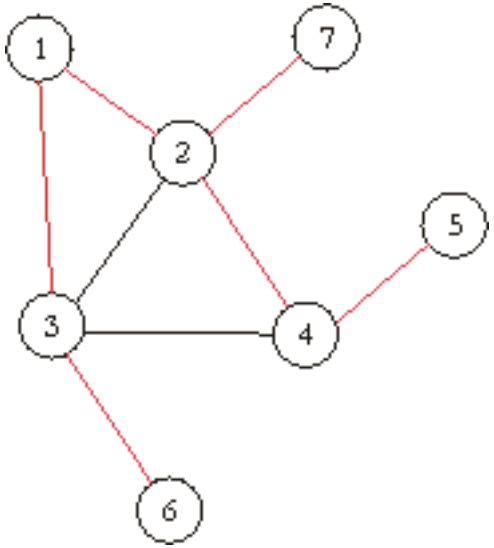


Szélességi bejárás alkalmazásai

Körmentes-e egy irányítatlan gráf?

Alapötlet: Ha a bejárás során minden szürke pontból csak fehér pontba vezet él, akkor a gráf körmentes.





Szélességi bejárás alkalmazásai

```
Körmentes?(p):
  Szín(p):=szürke; Sorba(p); Honnan(p):=p; km:=igaz
  Ciklus amíg nem üresSor? és km
    Sorból(p); Szín(p):=fekete
    Ciklus i∈Ki(p)
      Ha Szín(i)=fehér
       akkor Sorba(i); Szín(i):=szürke; Honnan(i):=p
      különben ha Honnan(p)≠i akkor km:=hamis
    Ciklus vége
  Ciklus vége
  Körmentes?:=km
Eljárás vége.
```

Szélességi bejárás alkalmazásai

Egy irányítatlan gráf páros gráf-e?

- A pontokat két osztályba soroljuk: a kezdőponttól páros, illetve páratlan távolságra levőkre.
- Szélességi bejárás az 1. csúcsból – ha minden él csak párosból páratlanba, vagy páratlanból párosba megy, akkor a gráf páros gráf.

Szélességi bejárás

```
Szélességi bejárás(p,jó):
  Szín(p):=szürke; Sorba(p); páros(p):=igaz; jó:=igaz
  Ciklus amíg nem üresSor? és jó
    Sorból(p); Szín(p):=fekete
    Ciklus i∈Ki(p)
      Ha Szín(i)=fehér
         akkor Sorba(i); Szín(i):=szürke
               páros(i):=nem páros(p)
      különben ha páros(i)=páros(p) akkor jó:=hamis
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Bejárás csúcslista esetén.

Szélességi bejárás alkalmazásai

Erősen összefüggő-e egy irányított gráf?

- Szélességi bejárás az 1. csúcsból – elérhető-e minden pont az első csúcsból.
- A fordított gráf előállítása (már a gráf beolvasásánál előállítható).
- Ebben újabb szélességi bejárás az 1. csúcsból – elérhető-e az első csúcs az összes többi csúcsból.

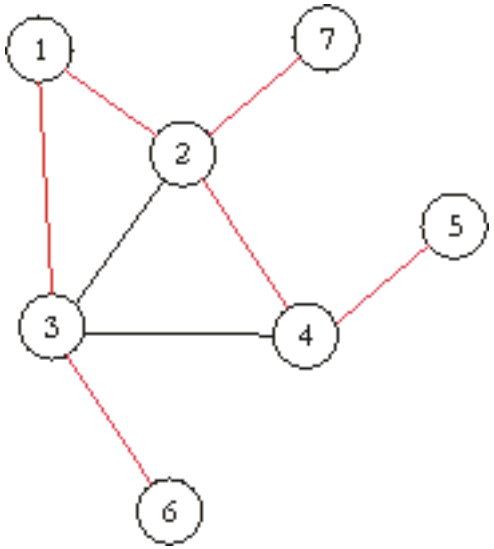
Ha mindkét bejárásnál elérjük az összes pontot, akkor a gráf erősen összefüggő.

Szélességi bejárás alkalmazásai

Legrövidebb utak száma

Alapötlet: fehér pont, illetve szürke pont esetén külön számítás.

- Fehér pontba annyi legrövidebb út vezet, amennyi a szürkébe, ahonnan jöttünk.
- Szürke pontba annyival több legrövidebb út vezet, amennyi a szürkébe, ahonnan jöttünk, ha a távolság így is minimális.



Szélességi bejárás alkalmazásai

```
Legrövidebb utak száma(p):
  Szín(p):=szürke; Sorba(p); Táv(p):=0; Db(p):=1
  Ciklus amíg nem üresSor?
    Sorból(p); Szín(p):=fekete
     Ciklus i∈Ki(p)
       Ha Szín(i)=fehér
            akkor Sorba(i); Szín(i):=szürke
                  Táv(i):=Táv(p)+1; Db(i):=Db(p)
       különben Ha Táv(i)=Táv(p)+1
            akkor Db(i):=Db(i)+Db(p)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Megjegyzés: A p pontból fekete pontba is vezethet él, de az éppen a p őse.

Szélességi bejárás alkalmazásai

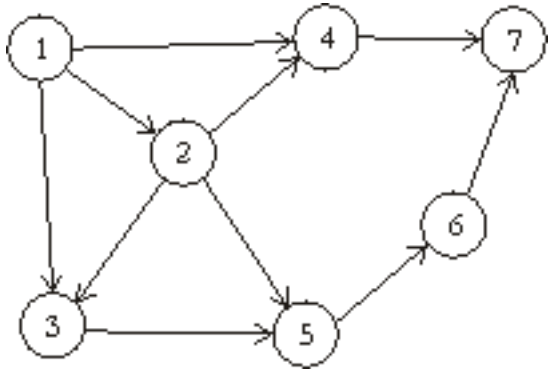
Topologikus rendezés hálóban

Háló: irányított körmentes gráf, egyetlen forrással és nyelővel

Pontok olyan sorba rendezése, hogy az élek csak a rendezés szerinti irányban haladjanak!

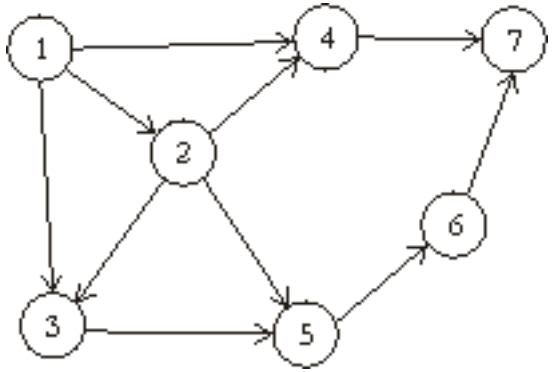
Alapötlet: A 0 befokúak kerüljenek be a sorba (nem biztos, hogy a legrégebbi szürke)!

Megjegyzés: az algoritmus bármely körmentes irányított gráfra működik.



Szélességi bejárás alkalmazásai

```
Topologikus rendezés(p):
  Sorba(p); t:=1; hely(t):=p
  Ciklus amíg nem üresSor?
    Sorból(p)
    Ciklus i∈Ki(p)
      Befok(i):=Befok(i)-1
      Ha Befok(i)=0 akkor Sorba(i)
                          t:=t+1; hely(t):=i
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

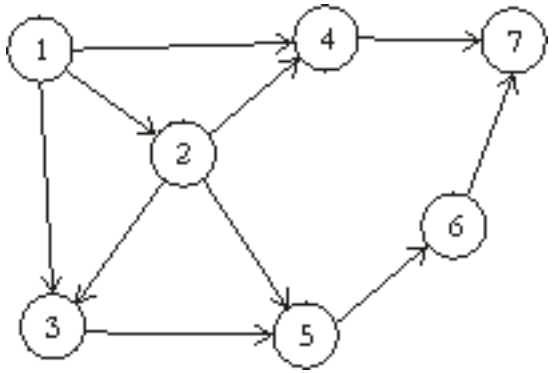


Szélességi bejárás alkalmazásai

Topologikus rendezés irányított körmentes gráfban

Pontok olyan sorba rendezése, hogy az élek csak a rendezés szerinti irányban haladjanak!

Alapötlet: A 0 befokúak kerüljenek be a sorba, már kezdetben is!



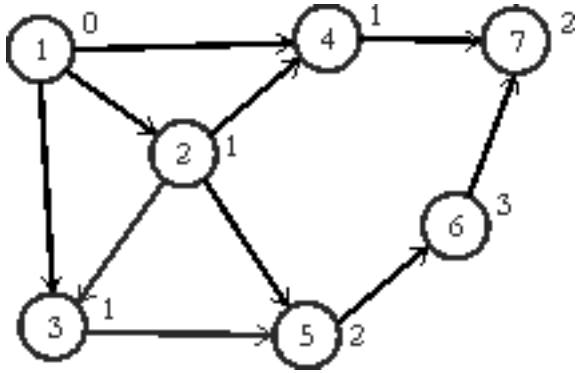
Szélességi bejárás alkalmazásai

```
Topologikus rendezés(p):
  t:=0
  Ciklus i=1-től Pontszám-ig
    Ha befok(i)=0 akkor Sorba(i); t:=t+1
                        hely(t):=p
  Ciklus vége
  Ciklus amíg nem üresSor?
    …
Eljárás vége.
```

Szélességi bejárás alkalmazásai

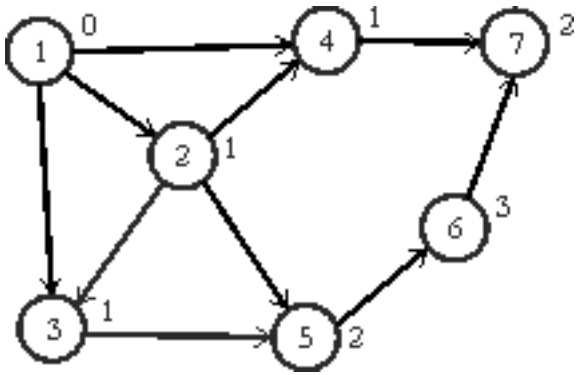
Legrövidebb utak topologikus rendezés esetén

Alapötlet: Távolság becslés a topologikus rendezés sorrendjében – ha egy ponthoz elérünk, akkor már minden bemenő élét feldolgoztuk → ismerjük a legkisebb távolságát is.



Szélességi bejárás alkalmazásai

```
Legrövidebb utak(p):
  Topologikus rendezés(p)
  Táv:=(+∞,…,+∞); Honnan:=(0,...,0)
  Ciklus i=1-től Pontszám-ig
    Ciklus j∈Ki(hely(i))
      Ha Táv(j)>Táv(hely(i))+1
         akkor Táv(j):=Táv(hely(i))+1; Honnan(j):=i
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

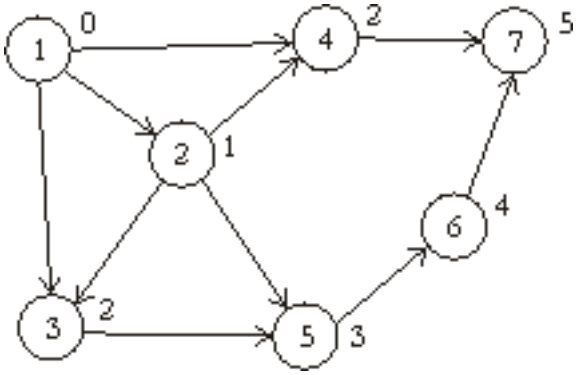


Megjegyzés: +1 helyett +élhossz(hely(i),j) is lehetne!

Szélességi bejárás alkalmazásai

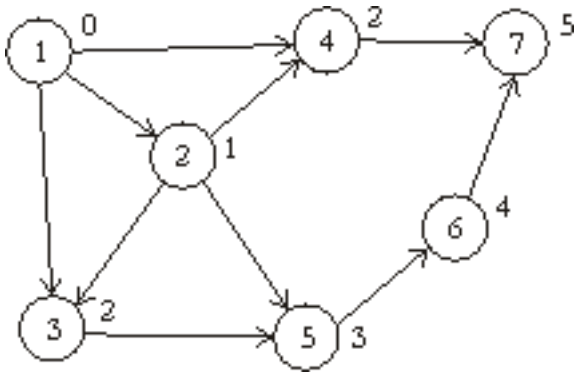
Leghosszabb utak topologikus rendezés esetén

Alapötlet: Távolság becslés a topologikus rendezés sorrendjében – ha egy ponthoz elérünk, akkor már minden bemenő élét feldolgoztuk → ismerjük a legnagyobb távolságát is.



Szélességi bejárás alkalmazásai

```
Leghosszabb utak(p):
  Topologikus rendezés(p)
  Táv:=(0,…,0) ; Honnan:=(0,...,0)
  Ciklus i=1-től Pontszám-ig
    Ciklus j∈Ki(hely(i))
      Ha Táv(j)<Táv(hely(i))+1
         akkor Táv(j):=Táv(hely(i))+1; Honnan(j):=i
    Ciklus vége
  Ciklus vége
Eljárás vége.
```



Megjegyzés: +1 helyett +élhossz(hely(i),j) is lehetne!

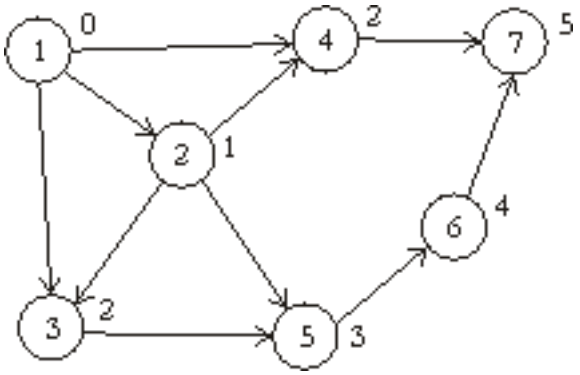
Szélességi bejárás alkalmazásai

Kritikus út, tartalék idők

A leghosszabb utak számolása közben előállított Honnan vektor alapján megadhatunk egy leghosszabb utat.

A nem leghosszabb úton levő pontok esetén beszélhetünk ún. tartalék időről: mennyivel növelhetjük a távolságukat a kezdőponttól, hogy a leghosszabb út hossza ne változzon!

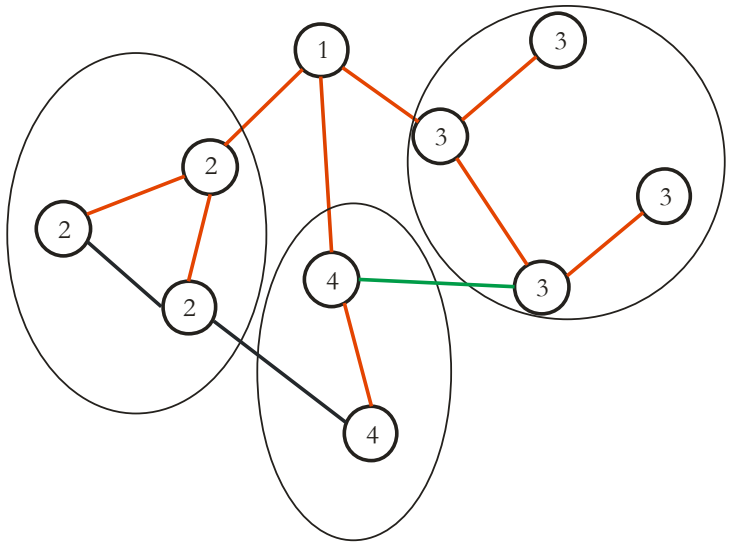
A végpontból indított visszafelé legnagyobb távolságok számolásával ez meghatározható.



Szélességi bejárás alkalmazásai

Adott ponton átmenő legrövidebb kör

- Induljunk ki az adott pontból!
- Ha megy át rajta kör, akkor az a szélességi feszítőfában biztosan két különböző gyereke felé indul el.
- Minden ponthoz tároljuk, hogy a kezdőpont melyik gyerekéből jutottunk oda!
- A kört okozó él az ezek közül valamelyik.



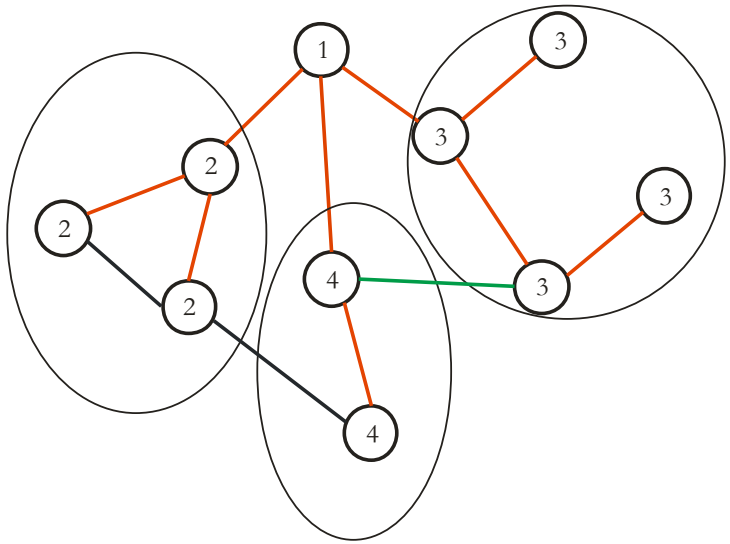
Szélességi bejárás alkalmazásai

```
Szélességi bejárás(p,V,Min1,Min2):
  Szín(p):=szürke; Sorba(p); Honnan(p):=p; Táv(p):=0
  Min:=+∞
  Ciklus amíg nem üresSor?
    Sorból(q); Szín(q):=fekete
    Ciklus i∈Ki(q)
      Ha Szín(i)=fehér
        akkor Sorba(i); Szín(i):=szürke; Honnan(i):=q
              Táv(i):=Táv(q)+1
              Ha p=q akkor Ős(i)=i
                   különben Ős(i):=Ős(q)
      …
```

Gyökérelem gyerekének saját maga az őse, a többieké azonos a szülő ősével.

Szélességi bejárás alkalmazásai

```
…
különben Ha Szín(i)=szürke és Ős(q)≠Ős(i)
              és Táv(i)+Táv(Q)+1<Min
          akkor Min:=Táv(i)+Táv(Q)+1
                Min1:=q; Min2:=i
   Ciklus vége
 Ciklus vége
Eljárás vége.
```



Keresztél, aminek a végpontjaihoz különböző ősök tartoznak.

Szélességi bejárás alkalmazásai

Elkerülhetetlen pont hálóban

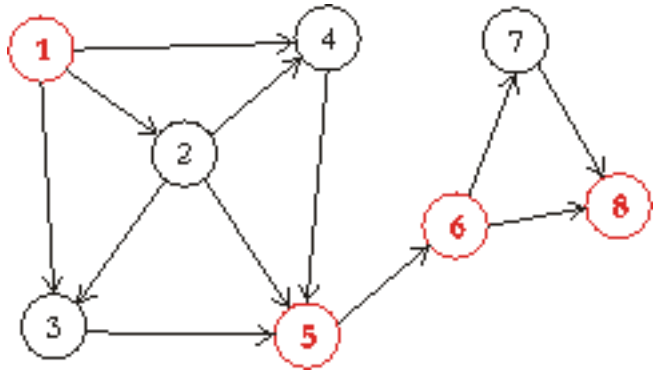
Alapötlet:

- A bejárás során a szürke pontokból vezet még ki feldolgozatlan él.
- Ha a bejárás során egyetlen 0 befokú szürke pont van, akkor az a pont elkerülhetetlen.
- A sorba csak a 0-befokú szürkék kerülnek be.
- A kezdő- és a végpont biztos elkerülhetetlen.

Általános irányított körmentes gráf esete?

Szélességi bejárás alkalmazásai

```
Szélességi bejárás(p):
 Sorba(p); Szdb:=1; Szín(p):=szürke; db:=0
  Ciklus amíg nem üresSor?
    Sorból(p); Szdb:=Szdb-1; Szín(p):=fekete
    Ha Szdb=0 akkor db:=db+1; Elk(db):=p
    Ciklus i∈Ki(p)
      Befok(i):=Befok(i)-1
      Ha Befok(i)=0 akkor Sorba(i)
      Ha Szín(i)=fehér akkor Szín(i):=szürke
                             Szdb:=Szdb+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```



Szélességi bejárás alkalmazásai

Elkerülhetetlen él hálóban

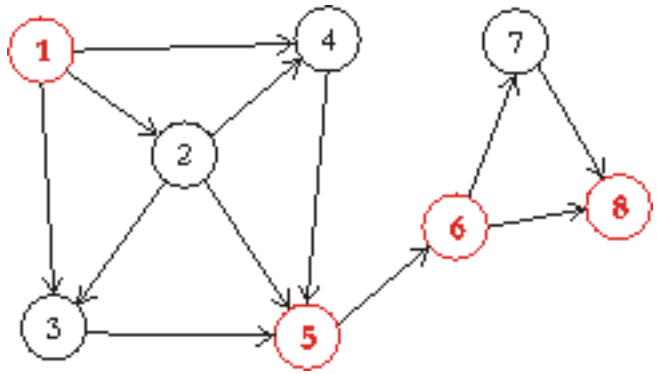
Alapötlet:

- A bejárás során a szürke pontokból vezet még ki feldolgozatlan él.
- Ha a bejárás során egyetlen 0 befokú szürke pont van, akkor az a pont elkerülhetetlen.
- Ha elkerülhetetlen pontból egy él vezet ki, akkor az az él elkerülhetetlen.

Általános irányított körmentes gráf esete?

Szélességi bejárás alkalmazásai

```
Szélességi bejárás(p):
 Sorba(p); Szdb:=1; Szín(p):=szürke; db:=0
  Ciklus amíg nem üresSor?
    Sorból(p); Szdb:=Szdb-1; Szín(p):=fekete
    Ha Szdb=0 és Szomszédpontokszáma(p)=1
         akkor db:=db+1; Elk(db):=p
    Ciklus i∈Ki(p)
      Befok(i):=Befok(i)-1
      Ha Befok(i)=0 akkor Sorba(i)
      Ha Szín(i)=fehér akkor Szín(i):=szürke
                             Szdb:=Szdb+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```



Szélességi bejárás alkalmazásai

Legrövidebb utak súlyozott gráfban

- A szélességi bejárás megadja a legrövidebb utat, ha az út hosszán a benne szereplő élek számát értjük.
- Ha az élek összhosszát, akkor azonban nem.
- Ha minden él pozitív hosszúságú, akkor a kezdőponthoz legközelebbi pontba biztosan ismerjük a legrövidebb út hosszát.
- A többi pontra pedig ismerjük az odavezető út hosszának egy felső korlátját.

Mi a helyzet a negatív élekkel?

Szélességi bejárás alkalmazásai

Legrövidebb utak súlyozott gráfban

Abból a szürke pontból lépjünk tovább, ami a legközelebb van a kezdőponthoz – prioritási sor legelső eleme.

A belőle elérhető pontokra számoljunk új felső korlátot:

- a fehér pontokra a szürke távolságát megnöveljük az élhosszal – bekerül a prioritási sorba;
- a szürke pontokra javítjuk a becslést az új távolsággal, ha lehetséges – mozog a prioritási sorban előre.

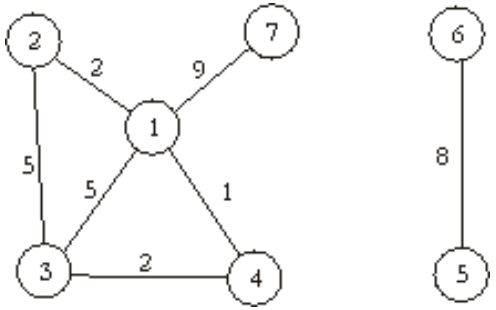
Szélességi bejárás alkalmazásai

```
Szélességi bejárás(p):
  Szín(p):=szürke; PrSorba(p); Táv(p):=0
  Ciklus amíg nem üresPrSor?
    PrSorból(p); Szín(p):=fekete
    Ciklus i∈Ki(p)
      Ha Szín(i)=fehér
         akkor Táv(i):=Táv(p)+Élhossz(p,i)
               PrSorba(i); Szín(i):=szürke
      különben ha Táv(i)>Táv(p)+Élhossz(p,i)
         akkor Táv(i):=Táv(p)+Élhossz(p,i)
               PrSorbanMozgat(i)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

PrSorbanMozgat helyett lehetne PrSorba, mert mindenki csak előre mozoghat. Ekkor a PrSorból átlépi a már kivett elemeket.

Dijkstra algoritmus

Dijkstra – legrövidebb utak



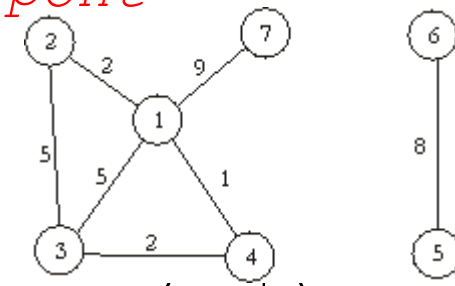
Alapötlet (pozitív élhosszak esetén):

- Az összes pont legyen szürke!
- A kezdőpont távolsága önmagától 0, a többi pont távolsága pedig $+\infty$ – becsült felső korlát – azaz mindenki legyen a prioritási sorban, a kezdőpont legelöl, a többiek bárhol!
- Vegyük a kezdőponthoz legközelebbi szürkét!
- Az ő távolsága biztos jó, módosítsuk a belőle kivezető éleken levő pontok távolságát, ha szükséges!

Azaz ez egy *mohó stratégia*!

Dijkstra algoritmus

```
LegrövidebbUtak(p):
  Táv():=(+∞,…,+∞); Szín():=(szürke,…,szürke)
  Honnan(p):=p; Táv(p):=0; PRSOR:=összes pont
  Ciklus i=1-től Pontszám-1-ig
    PrSorból(p); Szín(p):=fekete
    Ciklus k∈Ki(p)
      Ha Szín(k)≠fekete és
                          Táv(k)>Táv(p)+Élhossz(p,k)
        akkor Táv(k):=Táv(p)+Élhossz(pont,i)
              Honnan(k):=p; PrSorbanElőre(k)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```



Legyenek a pontok egy kupaccal ábrázolt prioritásai sorban, a Táv vektor értékei alapján!

Bellmann-Ford algoritmus

Bellmann-Ford – legrövidebb utak

Alapötlet (negatív élhosszak esetén, de nincs negatív összsúlyú kör):

- Az összes pont legyen szürke!
- A kezdőpont távolsága önmagától 0, a többi pont távolsága pedig $+\infty$ – becsült felső korlát!
- a pontok száma-1-szer nézi végig az összes élt, s ha kell, csökkenti az egyes pontok távolságát a kezdőponttól.

Bellmann-Ford algoritmus

```
LegrövidebbUtak(p):
  Táv():=(+∞,…,+∞); Honnan(p):=p; Táv(p):=0
  Ciklus i=1-től PontSzám-1-ig
    Ciklus j=1-től PontSzám-ig
      Ciklus k∈Ki(j)
        Ha Táv(k)>Táv(j)+ÉlHossz(j,k)
           akkor Táv(k):=Táv(j)+ÉlHossz(j,k)
                 Honnan(k):=j
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Bellmann-Ford algoritmus

Bellmann-Ford – helyesség belátása

Tegyük fel, hogy a legrövidebb út a $x=p_0, p_1, p_2, \ldots, p_k=y$, pontokon keresztül vezet x-ből y-ba.

Ekkor a külső ciklus k lépése alatt meghatározhatjuk y legkisebb távolságát.

Ezt teljes indukcióval láthatjuk be: k=0-ra p_0 távolsága helyes; ha k=i-1-re már minden pont távolsága helyes, akkor az i-edik lépésben a p_{i-1}-ből p_i-be vezető élt is megvizsgáljuk, s a feltevésünk szerint ez egy legrövidebb út p_i-be, tehát a távolságot meghatároztuk.

Gráfok
3. előadás vége