Gráfok
2. előadás

Szélességi bejárás alkalmazásai

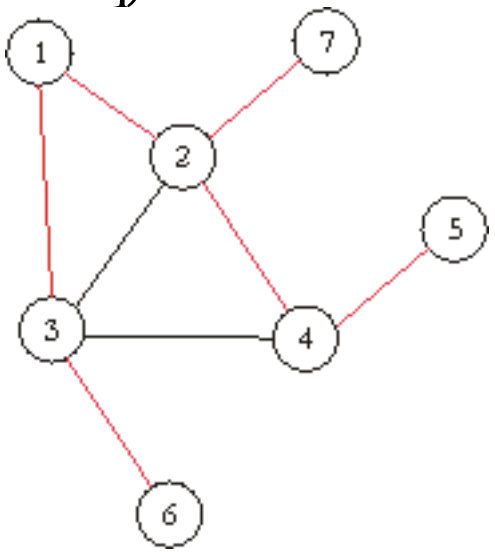
Legrövidebb utak minden pontba (p→q)

- 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.

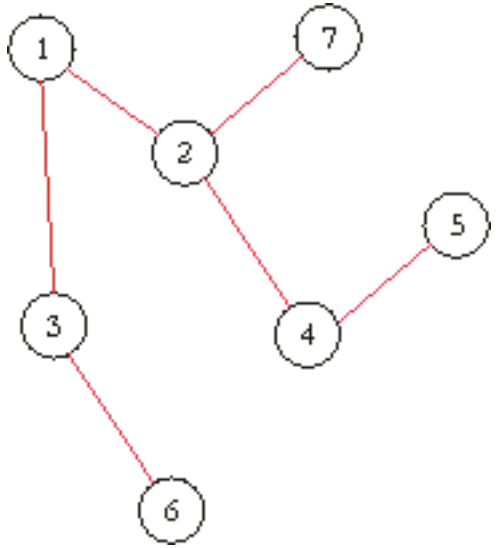
Probléma: így a két útnak lehet közös szakasza!

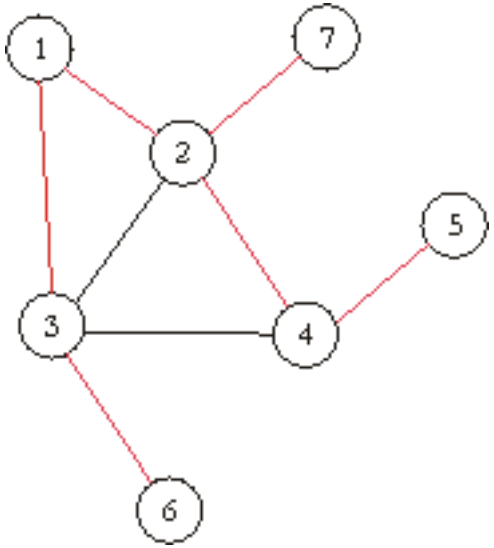


Szélességi bejárás alkalmazásai

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.
```

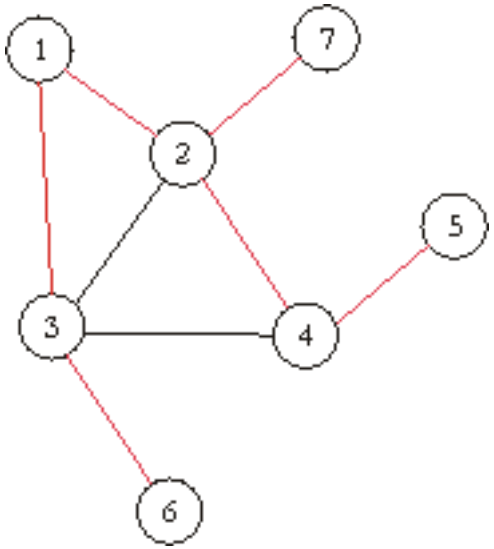
Bejárás csúcslista esetén.

Szélességi bejárás alkalmazásai

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

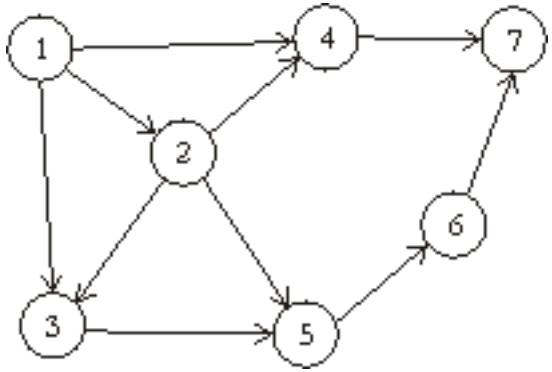
Topologikus rendezés hálóban

Háló: irányított körmentes gráf, egyetlen forrással és nyelővel

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

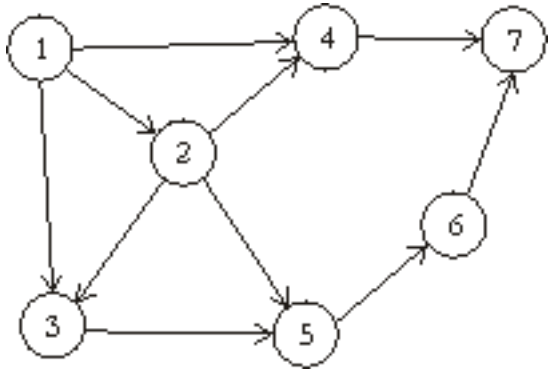
Alapötlet: A 0 befokúak kerüljenek be a sorba (nem biztos, hogy a legrégebbi szürke)!

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.
```

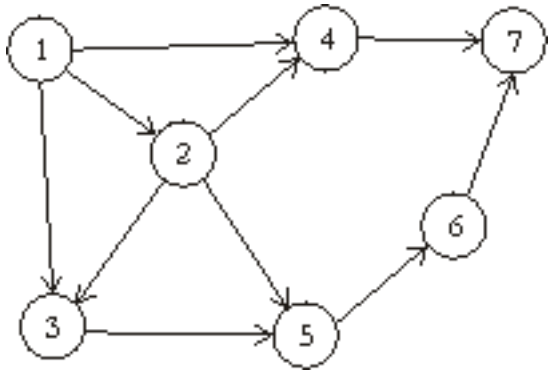


Szélességi bejárás alkalmazásai

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!



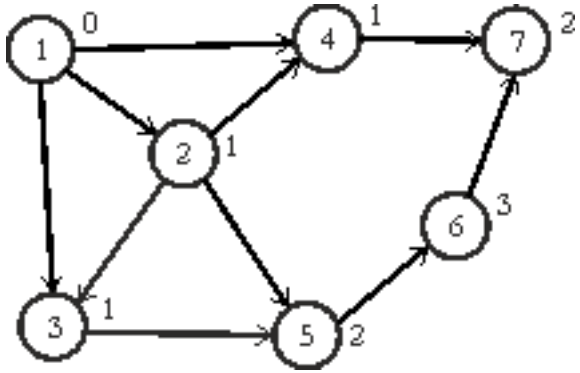
Szélességi bejárás alkalmazásai

```
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

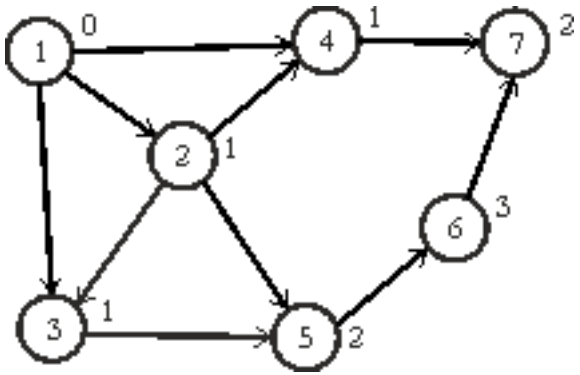
Legrövidebb utak topologikus rendezés esetén

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.
```



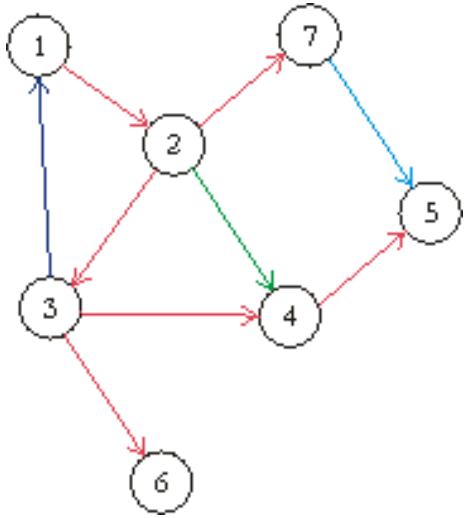
Megjegyzés: +1 helyett +élhossz(hely(i),j) is lehetne!

Mélységi bejárás alkalmazásai

Körmentes-e egy irányított gráf?

Alapötlet: Ha a bejárás során nincs visszamutató él, akkor a gráf körmentes.

```
Körmentes?(p):
   km:=igaz; Honnan(p):=p
   Mélységi bejárás(p,km)
   Körmentes?:=km
Eljárás vége.
```



Feltétel: erősen összefüggő vagy összefüggő (a példában) az 1. pontból. Az előremutató és a kereszt-élek nem okoznak kört (fekete pontba vezetnek).

Mélységi bejárás alkalmazásai

```
Mélységi bejárás(p,km):
  Szín(p):=szürke
  Ciklus i∈Ki(p)
   Ha Szín(i)=fehér
    akkor Ha km akkor Mélységi bejárás(i,km)
    különben Ha szín(i)=szürke akkor km:=hamis
  Ciklus vége
  Szín(p):=fekete
Eljárás vége.
```

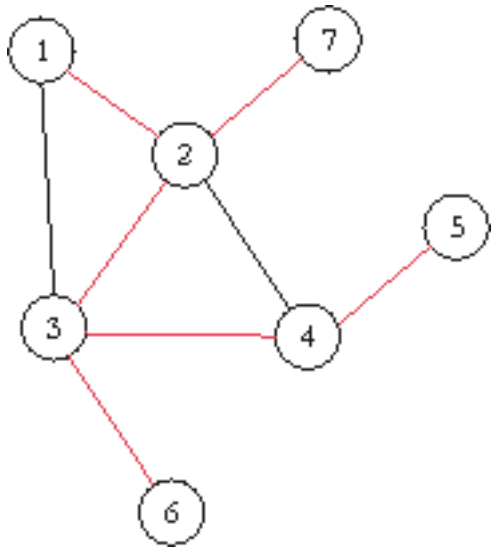
Mélységi bejárás alkalmazásai

Körmentes-e egy irányítatlan összefüggő gráf?

Alapötlet: Ha a bejárás során nincs visszamutató él, akkor a gráf körmentes.

```
Körmentes?(p):
   km:=igaz; Honnan(1):=1
   Mélységi bejárás(1,km)
   Körmentes?:=km
Eljárás vége.
```

Itt előremutató él és kereszt-él nincs.

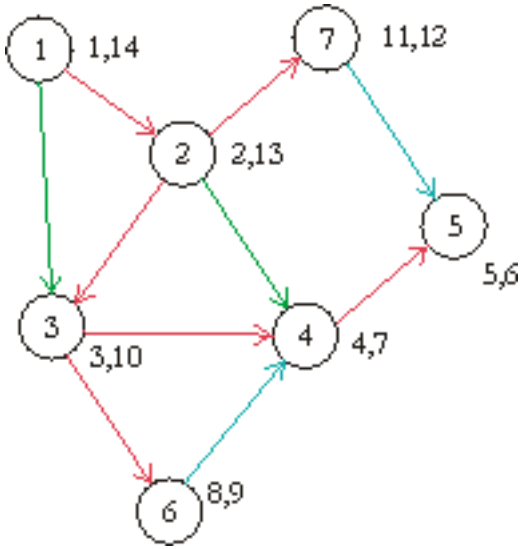


Mélységi bejárás alkalmazásai

Topologikus rendezés

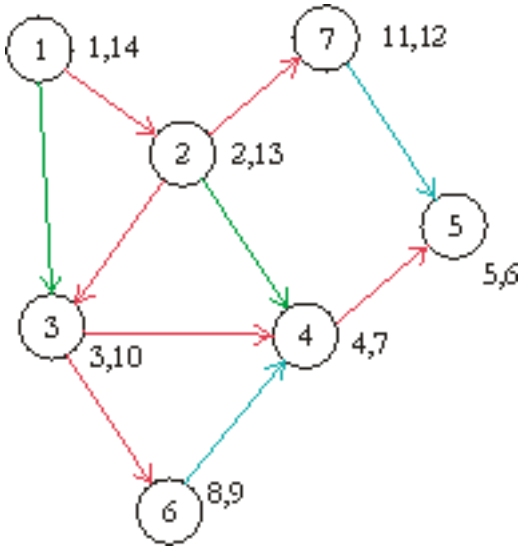
Alapötlet: Egy irányított gráf mélységi bejárásakor az elhagyási idők szerinti sorrend pontosan a topologikus sorrend ellentettje – elhagyáskor tegyük a pontokat egy verembe!

```
Topologikus rendezés(p):
  Mélységi bejárás_v(p)
  Veremkiírás
Függvény vége.
```



Mélységi bejárás alkalmazásai

```
Mélységi bejárás_v(p):
   Szín(p):=szürke
   Ciklus i∈Ki(p)
     Ha Szín(i)=fehér akkor Mélységi bejárás_v(i)
   Ciklus vége
   Szín(p):=fekete; Verembe(p)
Eljárás vége.
```



Mélységi bejárás alkalmazásai

Euler séta

Olyan út, amely az A pontból a B pontba vezet és a gráf minden élén pontosan egyszer halad át.

Csak olyan gráfra van Euler séta, amelyben A és B foka páratlan, a többi ponté pedig páros. Ha a gráf ilyen, akkor biztosan van Euler séta!

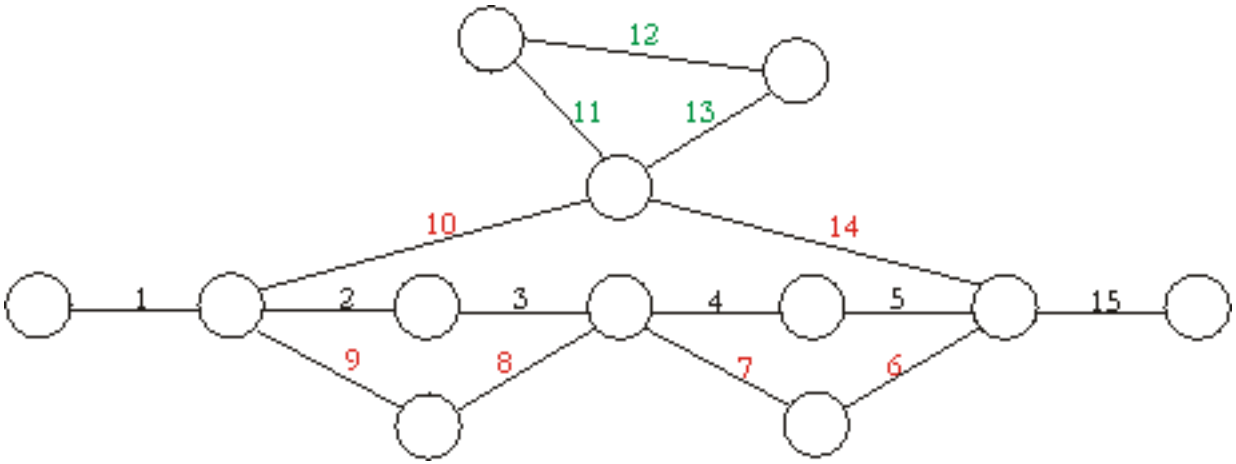
Megjegyzés: Euler körről beszélünk A=B, ekkor A foka is páros.

Mélységi bejárás alkalmazásai

- Keressünk egy utat A-ból B-be!
- A kimaradó élek biztos olyan utakon vannak, amelyek az (A,B) út adott pontjából indulnak és oda is érkeznek.
- A mélységi bejáráskor engedjük meg korábbi pontok újabb elérését – nem kellenek a színek!
- A mélységi bejárás során ugyanazt az élt kétszer nem használhatjuk – előre haladáskor töröljük az éleket!

Mélységi bejárás alkalmazásai

```
Mélységi bejárás(p):
  Ciklus i∈Ki(p)
    Éltörlés(p,i); Mélységi bejárás(i)
  Ciklus vége
  Verembe(p)
Eljárás vége.
```



Gráfalgoritmusok

Elérési (összefüggőségi) mátrix meghatározása (tranzitív lezárt) – Warshall:

$$E(i,j)=\begin{cases} igaz & ha \quad Vanút?(i,j) \\ hamis & egyébként \end{cases}$$

Emlékeztető – csúcsmátrix:

$$Cs(i,j)=\begin{cases} igaz & ha \quad Vanél?(i,j) \\ hamis & egyébként \end{cases}$$

Gráfalgoritmusok

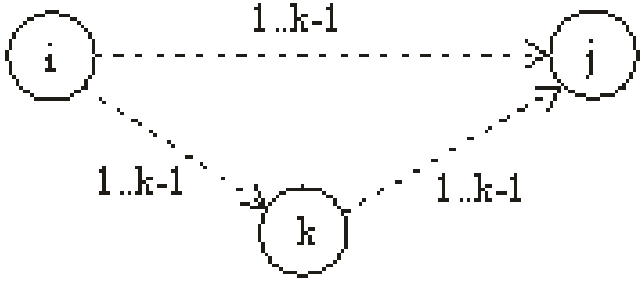
Ötlet:

- A csúcsmátrix azon utakat tartalmazza, amelyeknek nincs közbülső pontja.
- Ezt a mátrixot Pontszám lépésben transzformáljuk úgy, hogy egyre újabb és újabb *pontot iktatunk közbe* közvetítő pontként.
- A Pontszám. lépés után jutunk épp a keresett *E*-hez, hiszen így már az összes pont előfordulhat közbeiktatott pontként.

Gráfalgoritmusok

- Kiindulunk a csúcsmátrixból: $E^0 = Cs$
- Olyan utak, amelyek az első k ponton mennek keresztül:

$$E_{i,j}^{k} = E_{i,j}^{k-1} \vee \left(E_{i,k}^{k-1} \wedge E_{k,j}^{k-1} \right)$$



- A végeredmény:

$$E = E^{Pontszám}$$

Megjegyzés: A k felső indexre nincs szükség: $E_{i,k}^{k} = E_{i,k}^{k-1}$

Gráfalgoritmusok

```
Elérési mátrix(Cs,E):
  E:=Cs
  Ciklus k=1-től Pontszám-ig
    Ciklus i=1-től Pontszám-ig
      Ciklus j=1-től Pontszám-ig
        E(i,j):=E(i,j) vagy (E(i,k) és E(k,j))
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Gráfalgoritmusok

Elérési mátrix módosítása új él felvétele miatt:

- Az u-ból elérhető lesz a v.
- Minden i pontból elérhető lesz minden j pont, ha u elérhető volt i-ből és v-ből elérhető volt j.
- Minden i pontból elérhető lesz v, ha u elérhető volt i-ből.
- Minden j pont elérhető lesz u-ból, ha v-ből elérhető volt j.

Ez egy online algoritmus.

Gráfalgoritmusok

```
Elérési mátrix(E,u,v):
  E(u,v):=igaz
  Ciklus i=1-től Pontszám-ig
    Ha E(i,u) akkor E(i,v):=igaz
                    Ciklus j=1-től Pontszám-ig
                      E(i,j):=E(i,j) vagy E(v,j)
                    Ciklus vége
    Ha E(v,i) akkor E(u,i):=Igaz
  Ciklus vége
Eljárás vége.
```

Gráfalgoritmusok

Távolság mátrix meghatározása (Floyd-Warshall)

$$E(i,j)=\begin{cases} Úthossz(i,j) & ha \quad Vanút?(i,j) \\ +\infty & egyébként \end{cases}$$

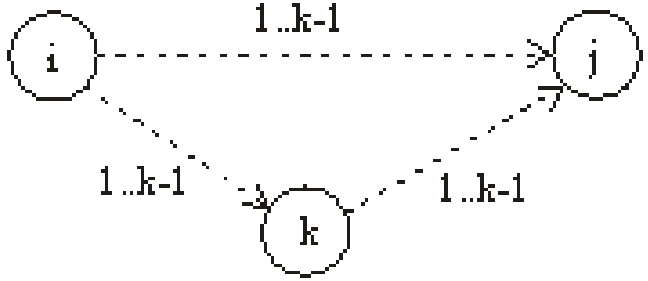
Emlékeztető – csúcsmátrix:

$$Cs(i,j)=\begin{cases} Élhossz(i,j) & ha \quad Vanél?(i,j) \\ +\infty & egyébként \end{cases}$$

Gráfalgoritmusok

- Kiindulunk a csúcsmátrixból: $T^0 = Cs$
- Olyan utak, amelyek az első k ponton mennek keresztül:

$$T_{i,j}^{k} = \min\left(T_{i,j}^{k-1}, T_{i,k}^{k-1} + T_{k,j}^{k-1}\right)$$



- A végeredmény:

$$T = T^{Pontszám}$$

Megjegyzés: A k felső indexre nincs szükség.

Gráfalgoritmusok

```
Távolság mátrix(Cs,T):
  T:=Cs
  Ciklus k=1-től Pontszám-ig
    Ciklus i=1-től Pontszám-ig
      Ciklus j=1-től Pontszám-ig
        Ha T(i,j)>T(i,k)+T(k,j)
           akkor T(i,j):=T(i,k)+T(k,j)
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Gráfalgoritmusok

Merre mennek a legrövidebb utak?

- Minden (i,j) pontpárra az (i,j) út vagy i-t követő, vagy j-t megelőző vagy éppen egy közbülső k pontját kell tárolni.

```
Start(T,Első)
  Ciklus i=1-től Pontszám-ig
    Ciklus j=1-től Pontszám-ig
      Ha VanÉl(i,j) akkor Első(i,j):=j
                     különben Első(i,j):=0
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Gráfalgoritmusok

```
Távolság mátrix(Cs,T):
  T:=Cs
  Ciklus k=1-től Pontszám-ig
    Ciklus i=1-től Pontszám-ig
      Ciklus j=1-től Pontszám-ig
        Ha T(i,j)>T(i,k)+T(k,j)
           akkor T(i,j):=T(i,k)+T(k,j)
                  Első(i,j):=Első(i,k)
      Ciklus vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

```
Utolsó(i,j):=Utolsó(k,j)
```

vagy

```
Közép(i,j):=k
```

Gráfalgoritmusok

Az út kiírása:

```
Útkiírás(A,B):
  Ha T(A,B)<+∞ akkor
     Ki: A; i:=A
     Ciklus amíg i≠B és i>0
       i:=első(i,B); Ki: i
     Ciklus vége
  Elágazás vége
Eljárás vége.
```

Gráfalgoritmusok

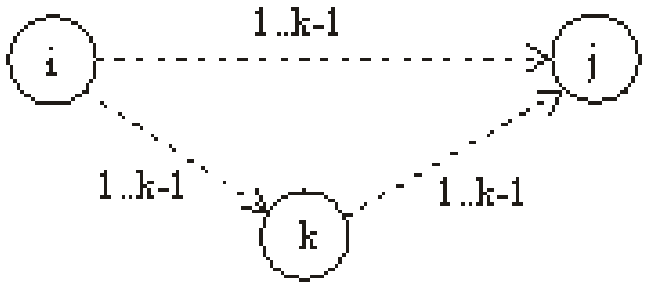
Feladatvariációk:

- Minden pontpárra a közöttük levő utak leghosszabb éleiből a legrövidebb:

$$T_{i,j}^{k} = \min\left(T_{i,j}^{k-1}, max\left(T_{i,k}^{k-1}, T_{k,j}^{k-1}\right)\right)$$

- Minden pontpárra a közöttük levő utak legrövidebb éleiből a leghosszabb:

$$T_{i,j}^{k} = \max\left(T_{i,j}^{k-1}, min\left(T_{i,k}^{k-1}, T_{k,j}^{k-1}\right)\right)$$



Távolság mátrix alkalmazásai

Legelszigeteltebb pont – az a pont, amelyhez a legközelebbi szomszédja a lehető legtávolabb van.

```
Legelszigeteltebb(Cs,p):
  Távolság mátrix(Cs,T); p:=1
  Ciklus i=1-től Pontszám-ig
    Min(i):=1
    Ciklus j=2-től Pontszám-ig
      Ha T(i,j)<T(i,Min(i)) akkor Min(i):=j
    Ciklus vége
    Ha T(i,Min(i))>T(p,Min(p)) akkor p:=i
  Ciklus vége
Eljárás vége.
```

Távolság mátrix alkalmazásai

Középpont – az a pont, amelytől a többiek átlagos távolsága a lehető legkisebb.

```
Legelszigeteltebb(Cs,p):
  Távolság mátrix(Cs,T); p:=1
  Ciklus i=1-től Pontszám-ig
    Táv(i):=0
    Ciklus j=1-től Pontszám-ig
      Táv(i):=Táv(i)+T(i,j)
    Ciklus vége
    Ha Táv(i)<Táv(p) akkor p:=i
  Ciklus vége
Eljárás vége.
```

Alternatív definíció: az a pont, amitől a legtávolabbi a lehető legközelebb van.

Gráfok
2. előadás vége