Gráfok
5. előadás

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

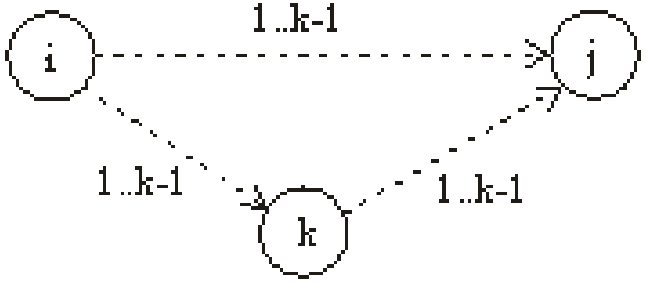
Ötlet:

- 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)

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

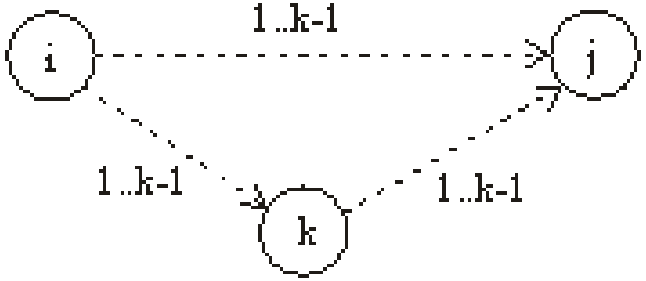
Emlékeztető – csúcsmátrix:

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

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.
```

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.

Minimális költségű feszítőfa

Feszítőfa: Egy irányítatlan gráf azon részgráfja, amely *fa* (körmentes, összefüggő) és *maximális* (tartalmazza a gráf összes pontját).

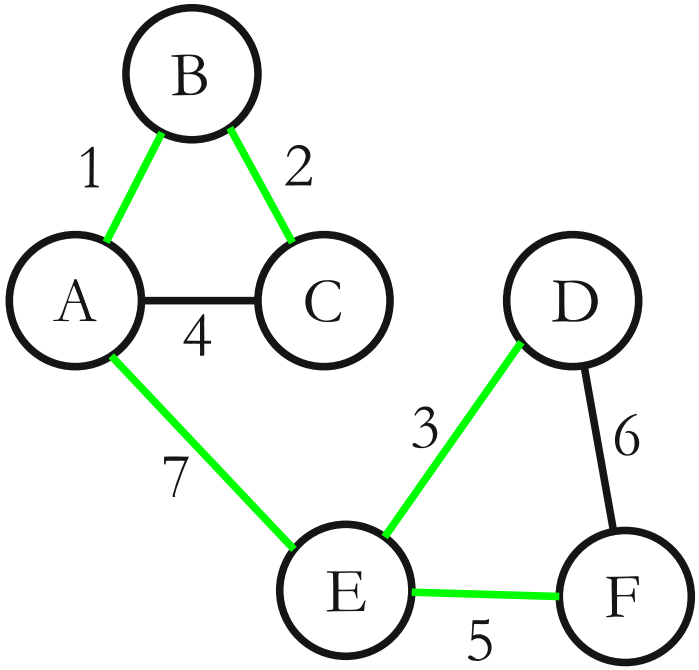
Minimális költségű feszítőfa: Súlyozott gráf azon feszítőfája, amely éleinek összköltsége *minimális*.

Megjegyzés: A szélességi és a mélységi bejárás is egy-egy feszítőfát határoz meg, de nem feltétlenül – sőt általában nem – minimális költségűt.

Ha a gráf nem összefüggő, akkor feszítő erdőről beszélhetünk.

Minimális költségű feszítőfa
Kruskal algoritmus

Ötlet: (mohó stratégiával)



Élvizsgálat sorrendje: 1,2,3,4,5,6,7.

Minimális költségű feszítőfa Kruskal algoritmus

Ötlet: (mohó stratégiával)

- a feszítőfa kezdetben álljon Pontszám darab feszítő erdőből (mindegyikben 1-1 pont lesz);
- vegyük az éleket hosszuk szerint növekvő sorrendben;
- ha egy él a feszítő erdő két különböző feszítőfáját köti össze, akkor biztosan eleme a feszítőfának (mert nincs nála rövidebb, ami a két fát összeköti);
- az ilyen fákat egyesítsük és vegyük fel az élt a feszítőfa élei közé!

Legyen Fa(i) az i pontot tartalmazó feszítőfa azonosítója!

Műveletigény: $O(\text{Élszám} * \log_2(\text{Élszám}))$

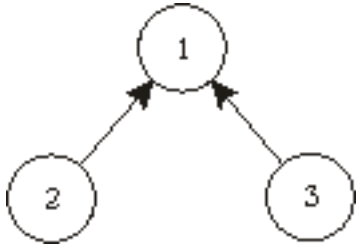
Minimális költségű feszítőfa
Kruskal algoritmus

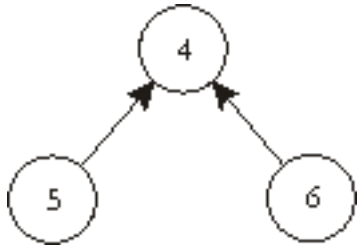
```
MinimálisFeszítőfa(F):
  Üres(F)
  Ciklus i=1-től PontSzám-ig
    szülő(i):=i
  Ciklus vége
  ÉlekRendezéseHosszSzerint(G)
  Ciklus e=1-től ÉlSzám-ig
    i:=Él(e,1); j:=Él(e,2)
    Ha Fa(i)≠Fa(j) akkor F:=F∪(i,j); Egyesít(i,j)
  Ciklus vége
Eljárás vége.
```

A Fa(i) gyakori művelet, annyiszor használjuk, ahány éle van a gráfnak, az Egyesít pedig ritkább, annyiszor használjuk, ahány pontunk van, azaz az előbbinek kell nagyon hatékonynak lenni.

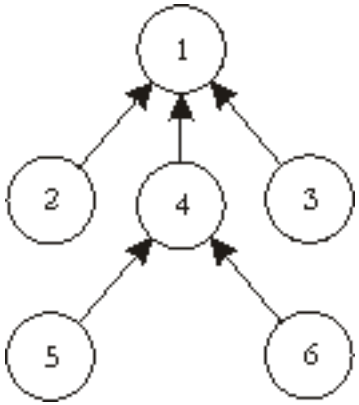
Diszjunkt halmazfelbontás

Építsünk egy erdőt, amelyben azonos fában vannak az azonos részhalmazban levő elemek:

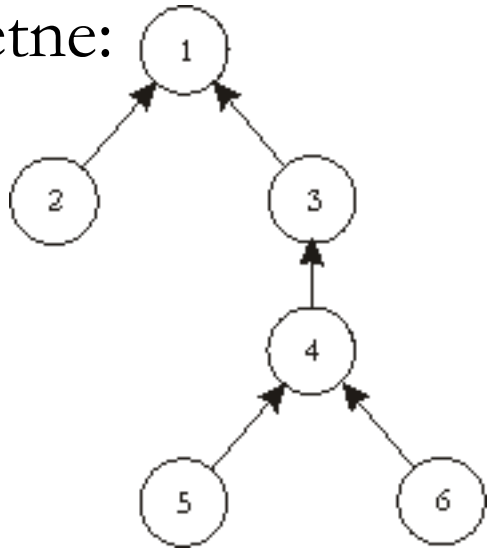




Egyesít(1,4) hatása:



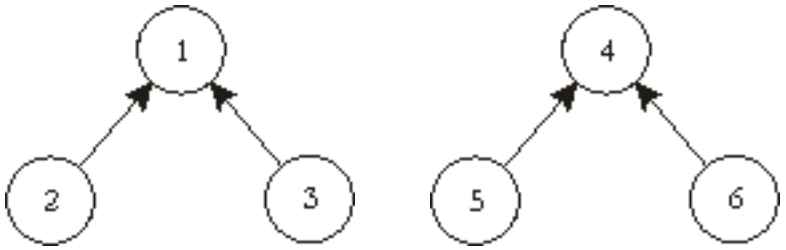
Egyesít(3,4) hatása lehetne:



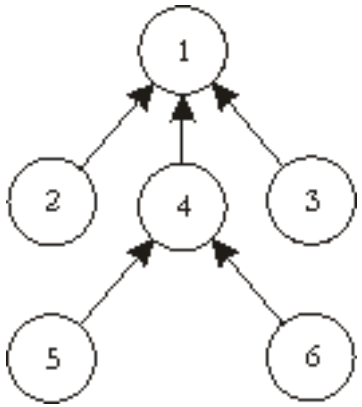
Mi lehetne Egyesít(3,5)?

Diszjunkt halmazfelbontás

A megoldás: egyesítéskor mindkét fában menjünk a legfelső elemhez és ott hajtsuk végre az egyesítést!



Egyesít(3,5) hatása:



Diszjunkt halmazfelbontás

```
Egyesít(u,v):
  Ciklus amíg u≠szülő(u)
    u:=szülő(u)
  Ciklus vége
  Ciklus amíg v≠szülő(v)
    v:=szülő(v)
  Ciklus vége
  szülő(v):=u
Eljárás vége.
```

```
Fa(u):
  Ciklus amíg u≠szülő(u)
    u:=szülő(u)
  Ciklus vége
  Fa:=u
Függvény vége.
```

Diszjunkt halmazfelbontás

Amikor a fában felfelé megyünk, akkor még érdemes a megtett utat tömöríteni, azaz minden bejárt pontot a gyökérhez csatolni:

```
Fa(u):
  v:=u
  Ciklus amíg v≠szülő(v)
    v:=szülő(v)
  Ciklus vége
  Ciklus amíg u≠szülő(u)
    w:=szülő(u); szülő(u):=v; u:=w
  Ciklus vége
  Fa:=u
Függvény vége.
```

Ilyenkor az egyesítés is egyszerűbb lehet, nem kell hozzá ciklus!

Diszjunkt halmazfelbontás

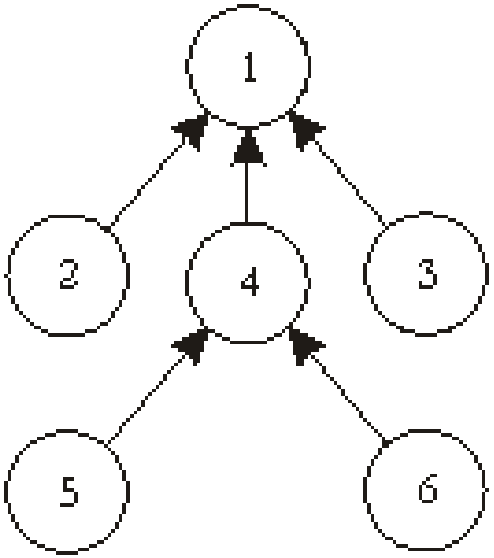
Ugyanez az úttömörítés rekurzívan:

```
Fa(u):
   Ha u≠szülő(u) akkor F:=Fa(szülő(u))
                       szülő(u):=F; Fa:=F
                 különben Fa:=u
Függvény vége.
```

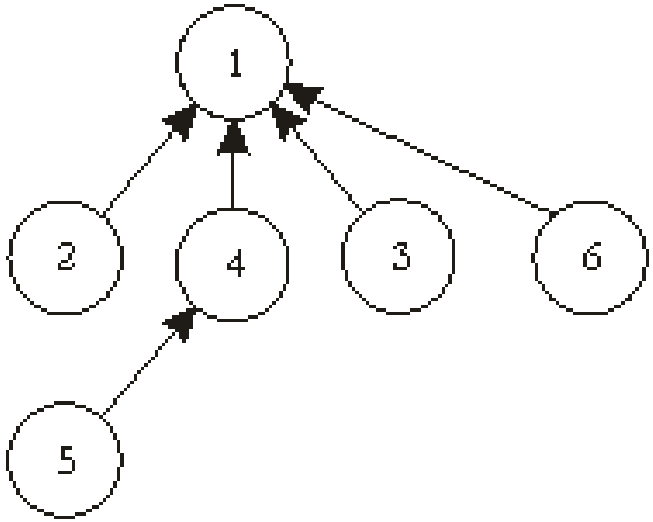
Egyesítésként legfeljebb 1 távolságra vagyunk a gyökértől:

```
Egyesít(u,v):
   Ha u≠szülő(u) akkor u:=szülő(u)
   Ha v≠szülő(v) akkor v:=szülő(v)
   szülő(v):=u
Eljárás vége.
```

Diszjunkt halmazfelbontás



Fa(6) hatása

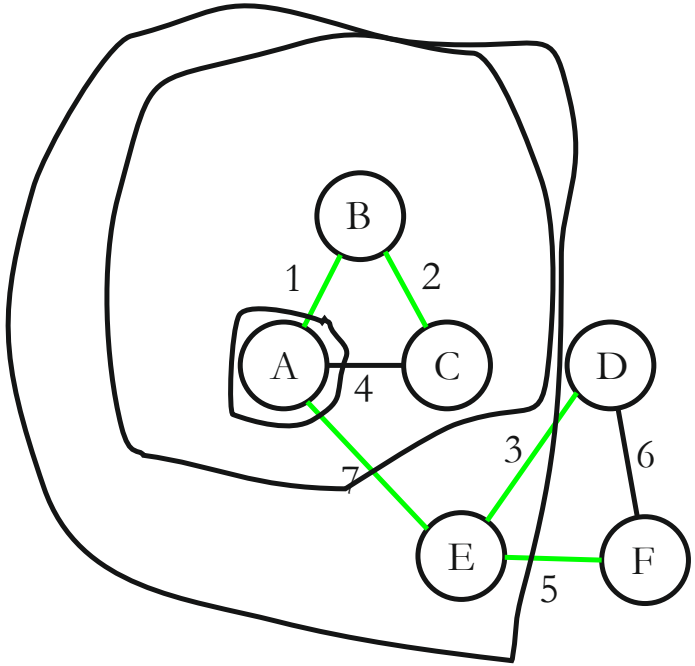


A fa magassága így kisebb lehet, ami gyorsíthatja az algoritmust!

Megjegyzés: a két művelet miatt hívják ezt a típust *Unió-Holvan* típusnak is.

Minimális költségű feszítőfa
Prim algoritmus

Ötlet: (mohó stratégiával)



Élsorrend: 1,2,7,3,5.

Minimális költségű feszítőfa
Prim algoritmus

Ötlet: (mohó stratégiával)

- a gráf pontjait 2 halmazba soroljuk, a feszítőfában bent levő és a még azon kívül levő pontok halmazára;
- a két halmaz közötti legrövidebb él biztosan eleme a feszítőfának (mert a legközelebbi pontba vezet);
- vegyük fel az első halmazhoz legközelebbi pontot a feszítőfa pontjai közé és számoljuk újra a szomszédjai távolságát!

Tegyük a második halmazbeli pontokat egy prioritásai sorba, az első halmaztól való távolságuk sorrendjében!

Műveletigény:

$O((\text{Élszám}+\text{Pontszám})*\log_2(\text{Pontszám}))$

Minimális költségű feszítőfa
Prim algoritmus

```
MinimálisFeszítőfa(Honnan):
  Táv(1..PontSzám):=+∞; PrSorba az összes pont
  p:=1; Honnan(p):=p;
  Ciklus i=1-től PontSzám-1-ig
    PrSorból(p); Táv(p):=0
    Ciklus j=1-től SzomszédPontokSzáma(p)-ig
      s:=SzomszédPont(p,j)
      Ha Táv(s)>0 és ÉlHossz(p,s)<Táv(s)
           akkor Táv(s):=ÉlHossz(p,s)
                 Honnan(s):=p; PrSorbanElőre(s)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

Online feszítőfa

Ötlet:

- kezdetben minden pont külön feszítőfában van;
- olvassuk egyesével a gráf éleit;
- ha különböző feszítőfabeli pontokat köt össze, akkor egyesítsük a két feszítőfát;
- ha azonos feszítőfabeli pontokat köt össze, akkor a köztük levő út leghosszabb élét cseréljük le rá, amennyiben rövidebb nála!

Legyen Fa(i) az i pontot tartalmazó feszítőfa azonosítója!

Online feszítőfa

```
MinimálisFeszítőfa(F):
  Üres(F); Fa():=(1,2,…,Pontszám)
  Ciklus e=1-től ÉlSzám-ig
    Be: i,j,Hossz(i,j)
    Ha Fa(i)≠Fa(j) akkor F:=F∪(i,j); Egyesít(i,j)
    különben Útkeresés(i,j, li,lj)
              Ha Hossz(i,j)<Hossz(li,lj)
                 akkor F:=F-(li,lj); F:=F∪(i,j)
  Ciklus vége
Eljárás vége.
```

Maximális párosítás páros gráfokban

Definíció: egy gráfot páros gráfnak nevezünk, ha a pontjai két (A és B) halmazba sorolhatók úgy, hogy él csak e két halmaz között vezet.

Ha egy gráfról tudjuk, hogy páros gráf, akkor a pontjai bármelyik bejárással két ilyen halmazra bonthatóak – ahova lépünk, azt mindig az ellenkező halmazba tesszük.

Ha a gráf nem páros, akkor mindkét bejárásnál olyan szürke pontba kell visszalépünk, amely az adott ponttal azonos halmazba tartozna.

Maximális párosítás páros gráfokban

```
Párosítás(A):
  Szín():=(fehér,...,fehér)
  Ciklus i=1-től PontSzám-ig
    Ha Szín(i)=fehér akkor Bejárás(i,igaz)
  Ciklus vége
Eljárás vége.
Bejárás(i,log):
  Szín(i):=szürke; A(i):=log
  Ciklus j∈Ki(i)
    Ha Szín(j)=fehér akkor Bejárás(j,nem log)
  Ciklus vége
Eljárás vége.
```

Maximális párosítás páros gráfokban

Feladat: Adjunk meg egy páros gráfban maximális párosítást!

Megoldás: magyar módszer.

Ha egy A-beli pontból tudunk olyan élt választani, amely még a párosításban nem szereplő B-beli pontba vezet, akkor ezt az élt felvesszük a maximális párosításba.

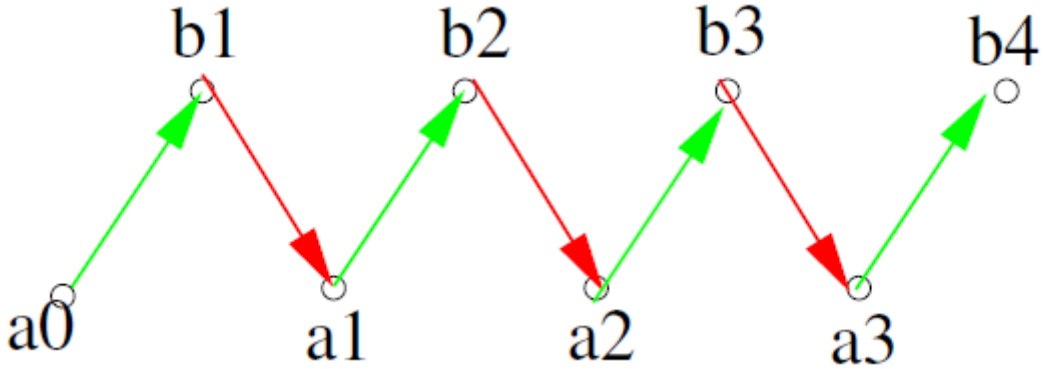
Ha csak olyan B-belibe vezet él, ami már szerepel a párosításban, akkor keresünk a gráfban ún. alternáló utat! Ha van, akkor ez valamelyik B-belibe vezető éllel kezdődik, onnan a korábban megtalált párosítás alapján egyértelműen egy A-belibe vezet. Ebből az A-beliből újra valamelyik B-belibe...

Maximális párosítás páros gráfokban

Ha olyan B-belihez érünk, amely nem volt még benne a párosításban, akkor az ide vezető úton minden párosítást megszüntetünk és bevesszük párosításba az úton levő nem párosított éleket – mivel eggyel több olyan élünk volt, ami nem szerepelt párosításban.

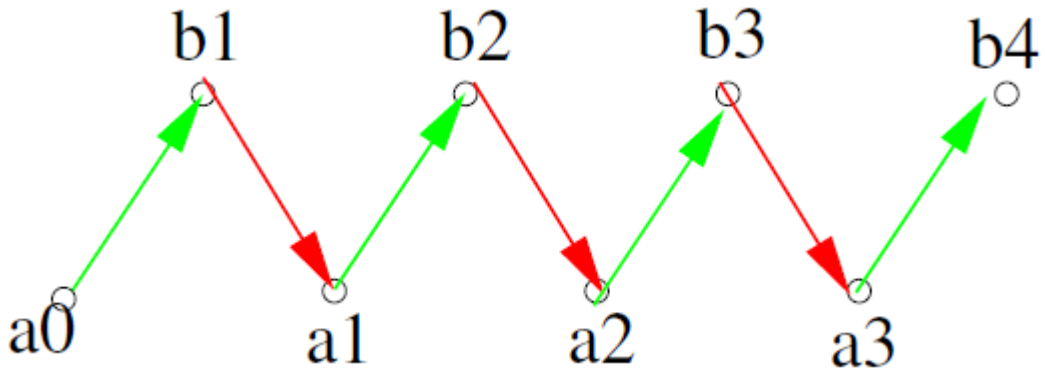
Ha nem találunk ilyen alternáló utat, akkor az adott A-beli pont nem lesz benne egy maximális párosításban.

Azaz a megoldás egy módosított mélységi bejárás.



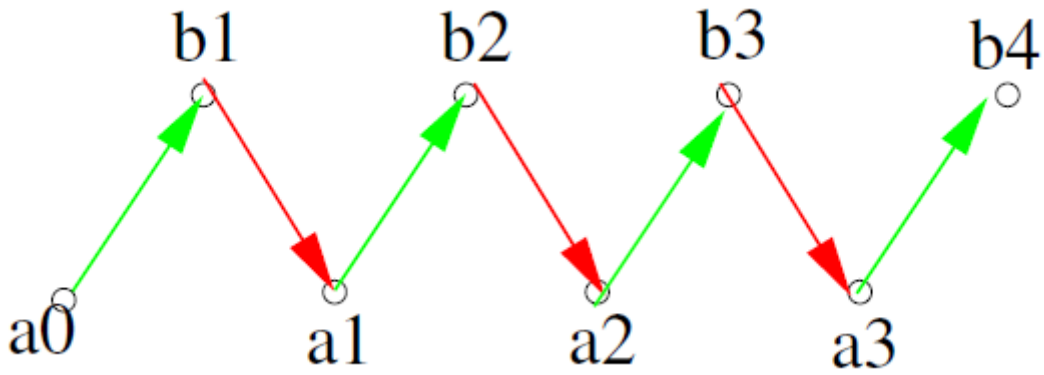
Maximális párosítás páros gráfokban

```
MaximálisPárosítás(A):
  Pár():=(1,...,PontSzám); db:=0;
  Ciklus i=1-től PontSzám-ig
    Ha A(i) és Pár(i)=i
       akkor Szín:=(fehér,...,fehér)
             L:=növelőút(i)
             Ha L akkor db:=db+1
  Ciklus vége
Eljárás vége.
```



Maximális párosítás páros gráfokban

```
növelőút(i):
  Szín(i):=szürke; k:=1; L:=hamis
  Ciklus amíg nem L és k≤Szomszédpontokszáma(i)
    j:=szomszéd(i,k)
    Ha Szín(j)=fehér akkor
         Ha Pár(j)=j akkor L:=igaz
                           Pár(i):=j; Pár(j):=i
         különben Szín(j):=szürke
                  L:=növelőút(Pár(j))
                  Ha L akkor Pár(i):=j; Pár(j):=i
    k:=k+1
  Ciklus vége
  növelőút:=L
Függvény vége.
```



Gráfok
5. előadás vége