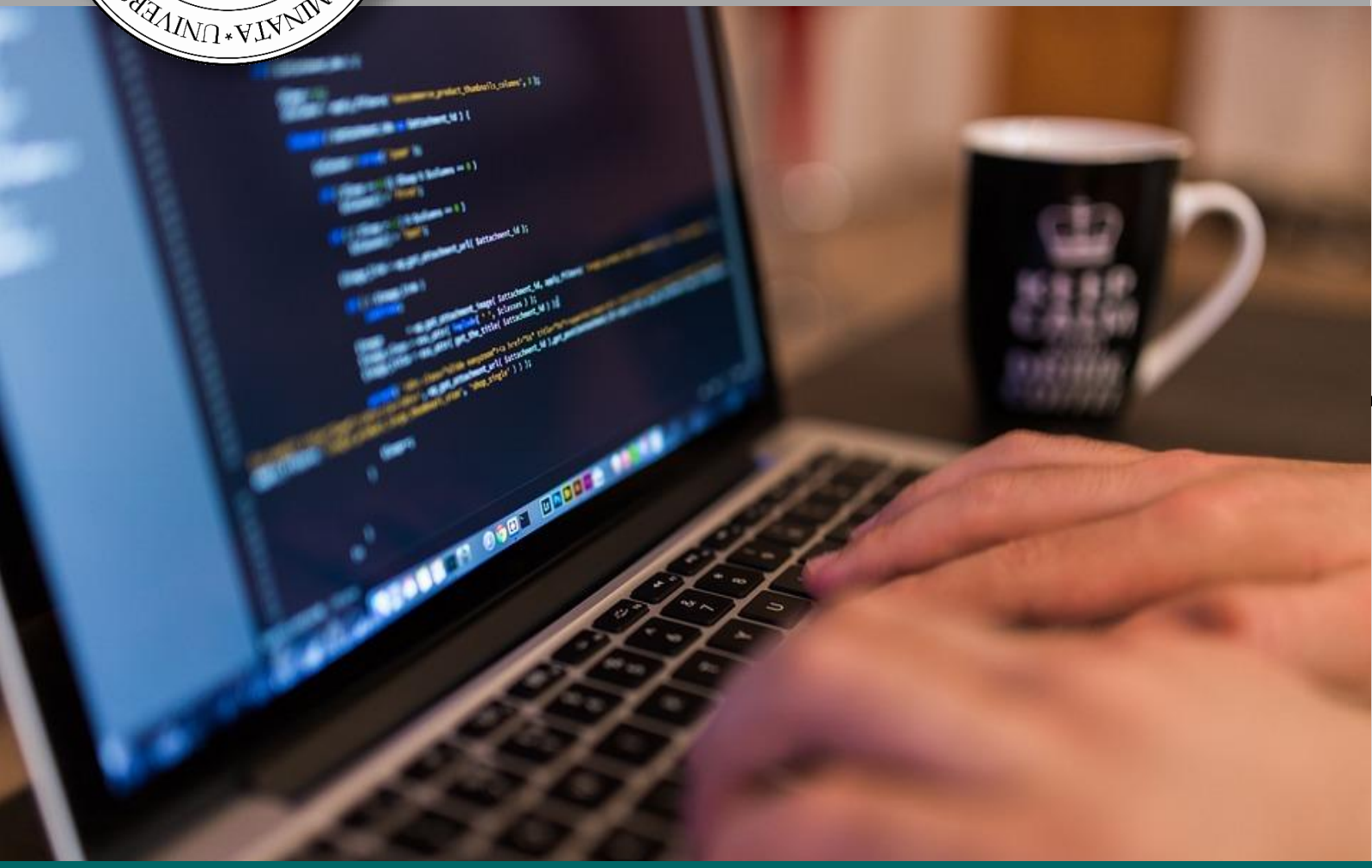
Belépő a tudás közösségébe

Informatika szakköri segédanyag



Visszalépéses maximumkiválasztás

Heizlerné Bakonyi Viktória, Horváth Győző, Menyhárt László, Szlávi Péter, Törley Gábor, Zsakó László

Szerkesztő: Abonyi-Tóth Andor, Zsakó László

A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2017-ben.

Eötvös Loránd Tudományegyetem
Informatikai Kar

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

1. Munkásfelvétel: N állás – N jelentkező

Egy vállalkozás `N` különböző állásra keres munkásokat. Pontosan N jelentkező érkezett, ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne.

Legyen `F[i,j]` értéke 0, ha az `i.` jelentkező a `j.` munkához nem ért, illetve `F[i,j]=A>0`, ha ért hozzá és `A` forintot szeretne kapni érte.

Állások:	1.	2.	3.	4.	5.
1. jelentkező:	100	0	0	**100**	0
2. jelentkező:	0	**200**	0	0	0
3. jelentkező:	**200**	100	0	0	0
4. jelentkező:	0	0	**200**	0	400
5. jelentkező:	500	0	400	0	**200**

A vállalkozás vezetője azt szeretné, ha az összes állást betöltenék, de úgy, hogy számára ez a lehető legkisebb költséggel járna.

A következőt gondolta ki: első lépésként állítsuk elő az összes lehetséges állásbetöltést (ha van egyáltalán olyan). Ez azt jelenti, hogy minden jelentkezőhöz hozzárendelünk egy állás sorszámot két feltétellel:

- olyan állást választhatunk számára, amihez ért (`F(i,j)>0`);
- olyan állást választhatunk számára, amit még nem adtunk másnak (`Volt(i,j,x)` függvény értéke hamis).

```
Megoldás(N,F,Db,Y):
  Összes_állás(1,N,F,Db,Y,x)
Eljárás vége.
```

Az `Összes_állás` eljárás első paramétere az aktuálisan vizsgált jelentkező `i` sorszáma legyen, a második paramétere pedig a jelentkezők (és egyben állások) `N` száma!

A megoldásban használt fontos változók:

- `x` – az aktuálisan számolt megoldás: `x[i]` az `i.` jelentkezőnek adott munka sorszáma
- `Db` – a megoldások száma;
- `Y` – a megoldásokat tartalmazó vektor.

```
Összes_állás(i,N,F,Db,Y,x):
  Ha i>N akkor Db:=Db+1; Y[Db]:=x
  különben Ciklus j=1-től N-ig
             Ha nem Volt(i,j,x) és F[i,j]>0
                akkor x[i]:=j; Összes_állás(i+1,N,F,Db,Y,x)
           Ciklus vége
  Elágazás vége
Eljárás vége.

Volt(i,j,x):
  k:=1
  Ciklus amíg k<i és x[k]≠j
    k:=k+1
  Ciklus vége
  Volt:=(k<i)
Függvény vége.
```

Ezután nincs más hátra, mint az `Y` vektorban összegyűlt megoldásokból kiválasztani a vállalkozó számára leggazdaságosabbat. Nagyon hamar kiderülhet azonban, hogy túlságosan sok elem lehet az `Y` vektorban.

A megoldási ötlet: felesleges az összes lehetséges megoldást tárolni, elég csupán minden lépés után a leggazdaságosabbat.

A megoldásban használt fontos változók:

- `x` – az aktuálisan számolt megoldás: `X[i]` az `i`. jelentkezőnek adott munka sorszáma
- `Y` – a legjobb megoldás.

```
Legjobb_állás(i,N,F,Y,x):
  Ha i>N akkor Ha Költség(N,F,x)<Költség(N,F,Y) akkor Y:=x
               Elágazás vége
  különben Ciklus j=1-től N-ig
             Ha nem Volt(i,j,x) és F[i,j]>0
                akkor x[i]:=j; Legjobb_állás(i+1,N,F,Y,x)
           Ciklus vége
  Elágazás vége
Eljárás vége.

Költség(N,F,x):
  s:=0
  Ciklus i=1-től N-ig
    s:=s+F[i,x[i]]
  Ciklus vége
  Költség:=s
Függvény vége.
```

Itt egy kicsi probléma léphet fel: az első megoldást mivel hasonlítjuk? Vegyünk fel egy új változót, ami az eddigi legjobb megoldás költségét tartalmazza! Állítsuk ennek az értékét a program elején a legnagyobb egész számra! Ha egy megoldást találunk, akkor az ennél biztosan jobb lesz, azaz lecserélhetjük rá.

```
Megoldás(N,F,Maxkölt,Y):
  Maxkölt:=+∞
  Legjobb_állás(1,N,F,Maxkölt,Y,x)
Eljárás vége.
```

A megoldásban használt fontos változók:

- `x` – az aktuálisan számolt megoldás: `x[i]` az `i`. jelentkezőnek adott munka sorszáma
- `Maxkölt` – az eddigi legjobb megoldás költsége;
- `Y` – a legjobb megoldás.

```
Legjobb_állás(i,N,F,Maxkölt,Y,x):
  Ha i>N akkor Ha Költség(N,F,x)<Maxkölt
                  akkor Y:=x; Maxkölt:=Költség(N,F,x)
               Elágazás vége
  különben Ciklus j=1-től N-ig
             Ha nem Volt(i,j,x) és F[i,j]>0
             akkor x[i]:=j; Legjobb_állás(i+1,N,F,Maxkölt,Y,x)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

Előfordulhat természetesen, hogy a feladatnak egyáltalán nincs megoldása, azaz legjobb megoldás sincs:

Állások:	1.	2.	3.	4.	5.
1. jelentkező:	0	0	100	**100**	0
2. jelentkező:	0	200	0	0	**100**
3. jelentkező:	200	**100**	0	0	0
4. jelentkező:	0	0	**200**	400	0
5. jelentkező:	0	0	400	200	0

Már csak egy apróságra gondolhatunk: ha van egy megoldásunk és a most készülő megoldásról látszik, hogy már biztosan rosszabb lesz – többe fog kerülni –, akkor azt már nem érdemes tovább vinni.

Legyen az eljárás paramétere az eddigi költség, s az eljárást csak akkor folytassuk, ha még nem érjük el a korábban kiszámolt maximális költséget. Emiatt nem a megoldások elkészültekor kell számolni költséget, hanem menet közben, folyamatosan.

A megoldást is módosítanunk kell, a `Legjobb_állás` eljárásnak új paramétere lesz, az aktuális előtti választások `költ` költsége.

A megoldásban használt fontos változók:

- `x` – az aktuálisan számolt megoldás: x[`i`] az `i`. jelentkezőnek adott munka sorszáma
- `költ` – az aktuálisan számolt megoldás költsége;
- `Maxköltt` – az eddigi legjobb megoldás költsége;
- `Y` – a legjobb megoldás.

```
Megoldás(N,F,Maxköltt,Y):
  Maxköltt:=+∞
  Legjobb állás(1,0,N,F,Maxköltt,Y,x)
Eljárás vége.

Legjobb_állás(i,költ,N,F,Maxköltt,Y,x):
  Ha i>N akkor Ha költ<Maxköltt akkor Y:=x; Maxköltt:=költ
               Elágazás vége
  különben Ciklus j=1-től N-ig
             Ha nem Volt(i,j,x) és F[i,j]>0
                és költ+F[i,j]<Maxköltt
                akkor x[i]:=j
                      Legjobb_állás(i+1,költ+F[i,j],
                                    N,F,Maxköltt,Y,x)
    Ciklus vége
  Elágazás vége
Eljárás vége.
```

Minta kódok

C++	cpp/1_Munkasfelvetel_N_allas_N_jelentkezo/fel adat.cpp
C#	cs/1_Munkasfelvetel_N_allas_N_jelentkezo/feladat.cs
Java	java/1_Munkasfelvetel_N_allas_N_jelentkezo/feladat.java
Pascal	pas/1_Munkasfelvetel_N_allas_N_jelentkezo/feladat.pas
Python	py/1_Munkasfelvetel_N_allas_N_jelentkezo/feladat.py

2. Munkásfelvétel: N állás – M jelentkező

Egy vállalkozás N különböző állásra keres munkásokat. A hirdetésre M jelentkező érkezett (M<N), ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne. Legyen `F(i,j)` értéke 0, ha az `i`. jelentkező a `j`. munkához nem ért, illetve `F(i,j)=A>0`, ha ért hozzá és `A` forintot szeretne kapni érte.

Állások:	1.	2.	3.	4.	5.
1. jelentkező:	100	0	0	**100**	0
2. jelentkező:	0	**200**	0	0	0
3. jelentkező:	**200**	100	0	0	0
4. jelentkező:	0	0	**200**	0	400

A vállalkozás vezetője azt szeretné, ha az összes jelentkezőt fel tudná venni, de úgy, hogy számára ez a lehető legkisebb költséggel járna.

A megoldás nagyon hasonló az előzőhöz: itt nem akkor van kész egy megoldás, ha N jelentkezőnek adtunk munkát, hanem akkor, ha az M jelentkezőnek adtunk munkát:

```
Megoldás(N,M,F,Maxkölt,Y):
  Maxkölt:=+∞
  Legjobb állás(1,0,N,M,F,Maxkölt,Y,x)
Eljárás vége.

Legjobb_állás(i,költ,N,M,F,Maxkölt,Y,x):
  Ha i>M akkor Ha költ<Maxkölt akkor Y:=x; Maxkölt:=költ
               Elágazás vége
  különben Ciklus j=1-től N-ig
             Ha nem Volt(i,j,x) és F[i,j]>0
                és költ+F[i,j]<Maxkölt
                akkor x[i]:=j
                       Legjobb_állás(i+1,költ+F[i,j],
                                     N,M,F,Maxkölt,Y,x)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

Egy vállalkozás N különböző állásra keres munkásokat. A hirdetésre M jelentkező érkezett (M>N), ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne.

Legyen `F(i,j)` értéke 0, ha az `i`. jelentkező a `j`. munkához nem ért, illetve `F(i,j)=A>0`, ha ért hozzá és `A` forintot szeretne kapni érte.

A megoldási ötlet: ne jelentkezőhöz keressünk állást, hanem álláshoz jelentkezőt! Így a feladat megoldása az előzőével majdnem megegyezik, csupán a két index szerepét kell felcserélni.

Állások:	1.	2.	3.	4.
1. jelentkező:	100	0	0	**100**
2. jelentkező:	0	**200**	0	0
3. jelentkező:	**200**	100	0	0
4. jelentkező:	0	0	**200**	0
5. jelentkező:	500	0	400	0

```
Legjobb_állás(i,költ,N,M,F,Maxköltt,Y,x):
  Ha i>N akkor Ha költ<Maxköltt akkor Y:=x; Maxköltt:=költ
              Elágazás vége
  különben Ciklus j=1-től M-ig
             Ha nem Volt(j,i,x) és F[j,i]>0
                és költ+F[j,i]<Maxköltt
                akkor x[j]:=i
                      Legjobb_állás(i+1,költ+F[i,j],
                                    N,M,F,Maxköltt,Y,x)
          Ciklus vége
  Elágazás vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Visszalépéses keresés
Feladat	5. Munka

3. Munkásfelvétel: N állás – N jelentkező, nem lehet mind felvenni

Egy vállalkozás N különböző állásra keres munkásokat. Pontosan N jelentkező érkezett, ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne. Legyen `F(i,j)` értéke 0, ha az `i`. jelentkező a `j`. munkához nem ért, illetve `F(i,j)=A>0`, ha ért hozzá és `A` forintot szeretne kapni érte.

Állások:	1.	2.	3.	4.	5.
1. jelentkező:	100	0	0	**100**	0
2. jelentkező:	0	**200**	0	0	0
3. jelentkező:	**200**	100	0	0	0
4. jelentkező:	0	0	**200**	0	400
5. jelentkező:	500	400	0	0	0

A vállalkozás vezetője azt szeretné, ha az összes állást betöltenék, de úgy, hogy számára ez a lehető legkisebb költséggel járna. Nem biztos azonban, hogy ez lehetséges. Akkor is érdekes lehet azonban egy olyan megoldás, amiben a lehető legtöbb munkát adhatjuk ki.

A megoldás ötlete: Vezessünk be egy N+1. fiktív állást, amihez azt gondoljuk, hogy mindenki ért! Legyen ennek az ára nagyobb, mint minden más összeg a táblázatunkban (`maxért`)! Az N+1. állásra engedjük meg, hogy többen is válasszák!

Állások:	1.	2.	3.	4.	5.	6.
1. jelentkező:	100	0	0	**100**	0	1000
2. jelentkező:	0	**200**	0	0	0	1000
3. jelentkező:	**200**	100	0	0	0	1000
4. jelentkező:	0	0	**200**	0	400	1000
5. jelentkező:	500	400	0	0	0	**1000**

Belátható, hogy a leggazdaságosabb megoldásban ekkor a lehető legkevesebb fiktív állás lesz, azaz a legtöbb állást tudjuk betölteni.

```
Legjobb_állás(i,költ,N,F,maxért,Maxköltség,Y,x):
  Ha i>N akkor Ha költ+maxért<Maxköltség
                  akkor Y:=x; Maxköltség:=költ+maxért
               Elágazás vége
  különben Ciklus j=1-től N-ig
             Ha nem Volt(i,j,x) és F[i,j]>0
                és költ+F[i,j]<Maxköltség
                akkor x[i]:=j
                      Legjobb_állás(i+1,költ+F[i,j],
                                    N,F,maxért,Maxköltség,Y,x)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	
Téma	
Feladat	

4. L üzlet – K pékség

Egy üzletlánc L üzlete K pékségtől rendelhet kenyeret. Megadjuk, hogy az egyes üzletek mennyi kenyérre tartanak igényt, és azt, hogy az egyes pékségek mennyit sütnek naponta. Továbbá adott az is, hogy az egyes üzletek mely pékségekkel állnak kapcsolatban. Az üzletek csak egyetlen egy pékségtől rendelhetnek (az adott napon). Ha ismerjük azt is, hogy az egyes pékségek hány forintért adják a kenyeret, akkor megadhatjuk, hogy hova honnan szállítsák a kenyeret úgy, hogy az az üzletláncnak a lehető legkevesebbe kerüljön!

Pékségek:	1.	2.	3.	4.	Igény
1. üzlet:	**igaz**	hamis	hamis	hamis	200
2. üzlet:	**igaz**	hamis	hamis	hamis	100
3. üzlet:	hamis	**igaz**	hamis	hamis	300
4. üzlet:	igaz	hamis	**igaz**	hamis	200
5. üzlet:	hamis	hamis	hamis	**igaz**	200
6. üzlet:	hamis	hamis	igaz	**igaz**	100
7. üzlet:	**igaz**	hamis	igaz	hamis	100

Van	400	600	400	400
Ár	200	300	250	100

Ez a feladat is visszalépéses maximumkiválasztás, első lépésként azonban próbáljuk megfogalmazni az összes megoldás előállítását!

Adatok:

- `Kapcs[i,j]` – igaz, ha az i. üzletbe a j. pékségből lehet szállítani
- `Van[j]` – a j. pékségben előállított kenyérmennyiség
- `Igény[i]` – az i. üzlet kenyérigénye
- `Ár[j]` – a j. pékségben előállított kenyerek ára

A megoldásban az `i`. üzletbe lehet a `j`. üzletből szállítani, ha van közöttük kapcsolat és a `j`. pékségben a kisebb sorszámú üzletek által elvitt kenyérmennyiség után még maradt annyi, amennyi az `i`. üzletnek kell.

```
Megoldás(L,K):
  Összes_rendelés(L,K,1)
Eljárás vége.
```

```
Összes_rendelés(L,K,i):
  Ha i>L akkor Db:=Db+1; Y[Db]:=x
  különben Ciklus j=1-től K-ig
             Ha vankenyér(i,j) és Kapcs[i,j]
                akkor x[i]:=j; Összes_rendelés(L,K,i+1)
           Ciklus vége
  Elágazás vége
Eljárás vége.

vankenyér(i,j):
  s:=Van[j]
  Ciklus k=1-től i-1-ig
    Ha Y[k]=j akkor s:=s-Igény[k]
  Ciklus vége
  vankenyér:=s≥Igény[i]
Eljárás vége.
```

Ezt ezután már nem túl nehéz átfogalmazni maximumkiválasztásra:

```
Megoldás(L,K):
  Maxkölt:=+∞
  Legjobb_rendelés(L,K,1)
Eljárás vége.

Legjobb_rendelés(L,K,i):
  Ha i>L akkor Ha Költség(x)<Maxkölt akkor Y:=x
                                           Maxkölt:=Költség(x)
               Elágazás vége
  különben Ciklus j=1-től K-ig
             Ha vankenyér(i,j) és Kapcs[i,j]
                akkor x[i]:=j; Legjobb_rendelés(L,K,i+1)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

A költségszámítás egy összegzés – ha az `i`. üzlet az `X[i]`. pékségtől rendel kenyeret, akkor annak az ára `Ár[X[i]]*Igény[i]`.

```
Költség(x):
  s:=0
  Ciklus i=1-től L-ig
    s:=s+Ár[X[i]]*Igény[i]
  Ciklus vége
  Költség:=s
Függvény vége.
```

A költségszámítást itt is elvégezhetnénk menet közben, megtakarítva ezzel a költség függvény megírását:

```
Megoldás(L,K):
  Maxkölt:=+∞
  Legjobb_rendelés(L,K,1,0)
Eljárás vége.
```

```
Összes_rendelés(L,K,i,költ):
  Ha i>L akkor Ha költ<Maxkölt akkor Y:=x
```

```
                                        Maxkölt:=Költség(x)
              Elágazás vége
  különben Ciklus j=1-től K-ig
             Ha vankenyér(i,j) és Kapcs[i,j]
                akkor x[i]:=j; z:=költ+Ár[j]*Igény[i]
                      Összes_rendelés(L,K,i+1,z)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	
Téma	
Feladat	