



Belépő a tudás közösségébe

Informatika szakköri segédanyag



Rendezések

**Bende Imre, Heizlerné Bakonyi Viktória, 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 2018-ban.

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

Az alapfeladat egy N elemű sorozat nagyság szerinti sorba rendezése. A sorozat elemei olyanok, amelyekre a $<, \leq$ relációk léteznek. Feltesszük a feladatok mindegyikében, hogy a sorozathoz létezik indexelés művelet, s ezt a megoldásban ki is használjuk. Az eredmény a módszerek jelentős részében helyben keletkezik, az eredeti sorrend elvész.

Konkrét feladatok ismertetése és megoldása helyett most egy számpéldát fogunk végignézni az egyes módszereknél. E számsorozat - egy kivétellel - a következő lesz: 5, 3, 9, 1, 7.

Változók:

N : **Egész** [a feldolgozandó sorozat elemei száma]
 X : **Tömb**(1..N:H_Elementípus) [a feldolgozandó sorozat]

Az egyes módszereket összehasonlítjuk tárigény (a rendezésben résztvevő tároló helyek száma), valamint végrehajtási idő (hasonlítások száma, mozgatók száma) szerint. A helyfoglalásba nem számítjuk bele a ciklusváltozókat, a végrehajtási időbe pedig ezek növelését, vizsgálatát, hiszen ezek mérete és ideje nem függ a rendezendő elemek típusától.

A hasonlítások és a mozgatók száma néha nem függ a rendezendő elemek értékétől, a legtöbb esetben azonban igen, így csak minimális és maximális számokról beszélhetünk.

1. Minimum-kiválasztásos rendezés

Az első módszer a következő ötletre épül. Hasonlítsuk össze az első elemet a sorozat összes többi mögötte levő elemével, s cseréljük meg közülük a legkisebbel! Ezzel elérhetjük, hogy a sorozat első helyére a legkisebb elem kerül. Folytassuk ugyanezen elven a sorozat második elemével, utoljára pedig az utolsóelőttivel.

Rendezés (N, X):

```
Ciklus i=1-től N-1-ig
  min:=i
  Ciklus j=i+1-től N-ig
    Ha X(min)>X(j) akkor min:=j
  Ciklus vége
  Csere(X(i),X(min))
Ciklus vége
Eljárás vége.
```

Itt a számpéldát a külső ciklus ciklusfeltételének kiértékelése előtt vizsgáljuk:

$i=1 \Rightarrow$	5, 3, 9, 1, 7
$i=2 \Rightarrow$	1, 3, 9, 5, 7
$i=3 \Rightarrow$	1, 3, 9, 5, 7
$i=4 \Rightarrow$	1, 3, 5, 9, 7

Helyfoglalás: $N+1$

Hasonlítások száma: $N*(N-1)/2$

Mozgatók száma: $3*(N-1)$

Minta kódok

C++ [cpp/minkiv/feladat.cpp](#)

C# [cs/minkiv/feladat.cs](#)

Java [java/minkiv/feladat.java](#)

Pascal [pas/minkiv/feladat.pas](#)

Python [py/minkiv/feladat.py](#)



2. Buborékos rendezés

Most egy másik rendezési alapelvet vizsgálunk meg: hasonlítsuk egymással a szomszédos elemeket, s ha a sorrendjük nem jó, akkor cseréljük meg őket!

Ezzel a módszerrel egy cikluslépés lefutása alatt a legnagyobb elem biztosan a sorozat végére kerül, s ezen kívül a nagyobb értékű elemek hátrafelé, a kisebbek pedig előre felé mozdulnak el (innen van a buborékmódszer elnevezés).

Figyeljük tehát minden menetben a legutolsó csere helyét, s a következő menetben csak addig rendezzünk!

Rendezés (N, X) :

```
i:=N
Ciklus amíg i≥2
  cs:=0
  Ciklus j=1-től i-1-ig
    Ha X(j)>X(j+1) akkor Csere(X(j),X(j+1)); cs:=j
  Ciklus vége
  i:=cs
Ciklus vége
Eljárás vége.
```

Újra a belső ciklusnál nézzük a konkrét rendezési példát:

i=5 ⇒	5, 3, 9, 1, 7	Helyfoglalás:	N+1
	3, 5, 9, 1, 7	Hasonlítások száma:	N-1 - N*(N-1)/2
	3, 5, 9, 1, 7	Mozgatások száma:	0 - 3*N*(N-1)/2
	3, 5, 1, 9, 7		
	3, 5, 1, 7, 9		
i=4 ⇒	3, 5, 1, 7, 9		
	3, 5, 1, 7, 9		
	3, 1, 5, 7, 9		
	3, 1, 5, 7, 9		
i=2 ⇒	1, 3, 5, 7, 9		
	1, 3, 5, 7, 9		

A hasonlítások számában javulást látunk, az igazi javulás azonban nem a minimális és a maximális, hanem az átlagos végrehajtási időben van.

Minta kódok

C++ [cpp/buborek/feladat.cpp](#)

C# [cs/buborek/feladat.cs](#)

Java [java/buborek/feladat.java](#)

Pascal [pas/buborek/feladat.pas](#)

Python [py/buborek/feladat.py](#)



3. Beillesztéses rendezés

Újabb rendezési elvvel ismerkedünk meg. Az eddigi módszerek mindegyike olyan volt, hogy a sorozatot felosztotta egy már kész, rendezett szakaszra, s a rendezést a másik szakasz elemei között

folytatta. Másik jellemzőjük volt, hogy a rendezés elkezdéséhez már az összes elemnek rendelkezésre kellett állnia.

Most a kártyakeveréshez hasonló elvből indulunk ki: egyetlen elem mindig rendezett; ha van egy rendezett részsorozatunk, akkor abba a nagyság szerinti helyére illesszük be a soron következő elemet! A beillesztendő nem tesszük azonnal be a sorozatba, hanem csak a többieket tologatjuk hátra, s a beillesztendő csak a végén tesszük a helyére.

Rendezés (N, X) :

```
Ciklus i=2-től N-ig
  j:=i-1; y:=X(i)
  Ciklus amíg j>0 és X(j)>y
    X(j+1):=X(j); j:=j-1
  Ciklus vége
  X(j+1):=y
Ciklus vége
Eljárás vége.
```

Az elemek mozgatása miatt e példában a kivett elem visszahelyezéséig egyes elemek kétszer szerepelnek.

i=2	⇒	5, 3, 9, 1, 7
		5, 5, 9, 1, 7
i=3	⇒	3, 5, 9, 1, 7
i=4	⇒	3, 5, 9, 1, 7
		3, 5, 1, 9, 7
		3, 1, 5, 9, 7
		3, 3, 5, 9, 7
i=5	⇒	1, 3, 5, 9, 7
		1, 3, 5, 9, 9

Helyfoglalás: $N+1$

Hasonlítások száma: $N-1 - N*(N-1)/2$

Mozgatások száma: $2*(N-1) - 2*(N-1)+N*(N-1)/2$

Minta kódok

C++ [cpp/beilleszt/feladat.cpp](#)

C# [cs/beilleszt/feladat.cs](#)

Java [java/beilleszt/feladat.java](#)

Pascal [pas/beilleszt/feladat.pas](#)

Python [py/beilleszt/feladat.py](#)



4. Szétoztó rendezés

A három alaprendezés, s azok javításai után speciális rendezésekkel foglalkozunk, amelyekben előfeltételként speciális megszorításaink lesznek.

A legelsőben feltesszük, hogy a rendezendő elemek olyan rekordok, amelyek kulcsmezője (vagy egy abból kiszámolt számérték) 1 és N közötti természetes szám lehet, s nincs két azonos kulcsú rekord.

A kulcsmező itt egyértelműen megadja azt a helyet, ahova az elemet tenni kell, így semmi hasonlításra nincs szükség. A módszerhez azonban egy újabb tömbre van szükség, ahova az eredményt elhelyezzük.

Változók:

N : **Egész** [a feldolgozandó sorozat elemei száma]
 X, Y: **Tömb**(1..N:H_Elemtípus) [a két sorozat]

Rendezés (N, X, Y) :

Ciklus i=1-től N-ig
 Y(X(i).kulcs) :=X(i)
Ciklus vége

Eljárás vége.

Itt a speciális feltétel miatt meg kell változtatnunk a példaszorozatot: 3, 2, 5, 1, 4 (most csak a kulcsmező értékét tároljuk). A feladat másik specialitása miatt pedig két sorozatot kell adnunk, a bemenetet és az eredményt. Az eredmény még nem kitöltött tagjait jelöli.

Bemenet

3, 2, 5, 1, 4

Kimenet

i=1 ⇒ ⊗, ⊗, ⊗, ⊗, ⊗
 i=2 ⇒ ⊗, ⊗, 3, ⊗, ⊗
 i=3 ⇒ ⊗, 2, 3, ⊗, ⊗
 i=4 ⇒ ⊗, 2, 3, ⊗, 5
 i=5 ⇒ 1, 2, 3, ⊗, 5
 i=6 ⇒ 1, 2, 3, 4, 5

Helyfoglalás: 2*N

Hasonlítások száma: 0

Mozgatások száma: N

Kulcsmezőindexelés: N

Minta kódok

C++ [cpp/szetoszt/feladat.cpp](#)

C# [cs/szetoszt/feladat.cs](#)

Java [java/szetoszt/feladat.java](#)

Pascal [pas/szetoszt/feladat.pas](#)

Python [py/szetoszt/feladat.py](#)



5. Számlálva szétosztó rendezés

Egy kicsit kevesebbet követel előfeltételként a következő rendezés: itt az elemek kulcsmezője lehet az $[1, N]$ -nél szélesebb intervallumban is és nem szükséges feltétel a kulcsok különbözősége.

Itt első lépésként számoljuk le minden lehetséges kulcsértékre, hogy hány ilyen értékű elem van! Mivel a kulcsértékekkel indexelhetünk, ezért a számlálás egyetlen indexeléssel elvégezhető minden elemre.

Másodjára minden lehetséges kulcsértékhez határozzuk meg a nála kisebb vagy egyenlő kulcsú rekordok számát!

Harmadik lépésként minden elemet közvetlenül a helyére helyezhetünk a fenti információ alapján.

```

Rendezés (N,M,X,Y) :
  DB(1..M) := (0,...,0)
  Ciklus i=1-től N-ig
    DB(X(i).kulcs) := DB(X(i).kulcs) + 1
  Ciklus vége
  Első(1) := 1
  Ciklus j=2-től M-ig
    Első(j) := Első(j-1) + DB(j-1)
  Ciklus vége
  Ciklus i=1-től N-ig
    Y(Első(X(i).kulcs)) := X(i)
    Első(X(i).kulcs) := Első(X(i).kulcs) + 1
  Ciklus vége
Eljárás vége.

```

Itt a speciális rendezés miatt újabb példaszorozatot vizsgálunk: 5, 3, 5, 1, 7. A bemenet mellett először a DB vektor értékeit közöljük, majd az eredményt.

Bemenet

5, 3, 5, 1, 7

DB:

i=1 ⇒ 0, 0, 0, 0, 0, 0, 0
i=2 ⇒ 0, 0, 0, 0, 1, 0, 0
i=3 ⇒ 0, 0, 1, 0, 1, 0, 0
i=4 ⇒ 0, 0, 1, 0, 2, 0, 0
i=5 ⇒ 1, 0, 1, 0, 2, 0, 0
i=6 ⇒ 1, 0, 1, 0, 2, 0, 1

A második ciklusban:

DB:

j=2 ⇒ 1, 0, 1, 0, 2, 0, 1
j=3 ⇒ 1, 1, 1, 0, 2, 0, 1
j=4 ⇒ 1, 1, 2, 0, 2, 0, 1
j=5 ⇒ 1, 1, 2, 2, 2, 0, 1
j=6 ⇒ 1, 1, 2, 2, 4, 0, 1
j=7 ⇒ 1, 1, 2, 2, 4, 4, 1
1, 1, 2, 2, 4, 4, 5

Kimenet

i=1 ⇒ ⊗, ⊗, ⊗, ⊗, ⊗
i=2 ⇒ ⊗, ⊗, ⊗, 5, ⊗
i=3 ⇒ ⊗, 3, ⊗, 5, ⊗
i=4 ⇒ ⊗, 3, 5, 5, ⊗
i=5 ⇒ 1, 3, 5, 5, ⊗
i=6 ⇒ 1, 3, 5, 5, 7

Helyfoglalás: $2*N+M*\epsilon$

Hasonlítások száma: 0

Mozgatások száma: N

Kulcsmező-indexelés: $5*N$

Minta kódok

C++ cpp/szamszet/feladat.cpp

C# cs/szamszet/feladat.cs

Java java/szamszet/feladat.java

Pascal pas/szamszet/feladat.pas

Python py/szamszet/feladat.py



Feladatok programozási tételekre a Nemes Tihamér OITV-ből és az Informatika OKTV-ből

1. feladat

Egy futóversenyen a versenyzőket (N db) rajtszám szerint (a rajtszám 1-től N -ig fut) percenként indítják. A célba érkezési listán időrendben megadják, hogy melyik rajtszámú versenyző mikor érkezett célba.

Írj programot, amely beolvassa a versenyzők számát ($1 \leq N \leq 100$), a célba érkezett versenyzők számát ($1 \leq M \leq N$), majd M darab versenyző sorszámot ($1 \leq \text{sorszám} \leq N$) és célba érkezési időt (0 és 10 000 közötti egész számok, növekvő sorrendben). A program ezek alapján készítse el az eredménylistát, valamint adja meg, hogy kik nem érkeztek célba.

Példa

Bemenet

```
N=6, M=4
sorszám: 2 idő: 10
sorszám: 4 idő: 12
sorszám: 5 idő: 12
sorszám: 1 idő: 15
```

Kimenet

```
1. helyezett: 5
2. helyezett: 2 4
4. helyezett: 1

Nem érkezett célba: 3 6
```

Jelölje $v(i) = \text{igaz}$, ha az i -edik versenyző célba ért! Kezdetben a v vektor minden elem legyen hamis, amit célba érésnél állítunk igazra! A célba ért versenyzők futási idejét a beérkezési időből és a rajtszámából számolhatjuk ki.

Ezután rendezni kell futási idő szerint, majd jöhet a kiírás, de figyelni kell a holtversenyre is!

```
Verseny(n,m):
  v(i):=(hamis, ..., hamis)
  Ciklus i=1-től m-ig
    Be: t(i).sor; v(t(i).sor):=igaz
    Be: j; t(i).idő:=j-t(i).sor
  Ciklus vége
  Ciklus i=1-től m-1-ig
    min:=i
    Ciklus j=i+1-től m-ig
      Ha t(j).idő<t(min).idő akkor min:=j
    Ciklus vége
  s:=t(i); t(i):=t(min); t(min):=s
  Ciklus vége
  j:=t(1).idő-1
  Ciklus i=1-től m-ig
    Ha j<t(i).idő akkor Ki: i, '. helyezett:'
    Ki: t(i).sor; j:=t(i).idő
  Ciklus vége
  Ha m<n akkor Ki: 'Nem érkezett célba:'
    Ciklus i=1-től n-ig
      Ha nem v(i) akkor Ki: i
    Ciklus vége
```

Eljárás vége.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Programozási tételek: rendezések
Feladat	4. Futóverseny

2. feladat

Egy pontozásos versenyen N ($1 \leq N \leq 100$) résztvevő indul. A versenyzőket M ($3 \leq M \leq 10$) pontozó pontozza. A pontozók azonban befolyásolhatnák a verseny végeredményét (a saját országbeli versenyzőknek nagyobb, a más országbeli versenyzőknek pedig kisebb pontot adva, ezért a verseny szervezői úgy döntöttek, hogy az M pontszám közül a legkisebbet és a legnagyobbat egy versenyzőnél sem veszik figyelembe, azaz mindenkit $M-2$ pontszám összegével értékelnek.

Készíts programot, amely

- kiírja minden versenyzőre, hogy mely pontozók pontszámát hagyják ki;
- kiszámítja és kiírja minden versenyző pontszámát;
- kiírja a verseny végeredményét (a versenyzőket pontszám szerint csökkenő sorrendben, holtverseny esetén azonos helyezési számmal)!

Példa

Bemenet

```
N=4, M=4
1. versenyző pont: 8 2 6 6
2. versenyző pont: 5 5 5 5
3. versenyző pont: 4 5 6 7
4. versenyző pont: 5 4 6 5
```

Kimenet

```
Kihagyott pontozók:
1. versenyző: 1 2
2. versenyző: 1 2
3. versenyző: 1 4
4. versenyző: 2 3
Pontszámok:
1. versenyző: 12 pont
2. versenyző: 10 pont
3. versenyző: 11 pont
4. versenyző: 10 pont
Sorrend:
1. helyezett: 1. versenyző 12 pont
2. helyezett: 3. versenyző 11 pont
3. helyezett: 2. versenyző 10 pont
3. helyezett: 4. versenyző 10 pont
```

- NEM RENDEZÉS, de a rendezendők kiszámításához szükséges;
- NEM RENDEZÉS, de a rendezendők kiszámításához szükséges;
- ez már egyszerű rendezés (bármely módszer jó lenne), holtverseny esetén azonos helyezési számkiírásával.

Figyelni kell arra, hogy rendezésnél az elemek elmozdulnak a helyükről, azaz a sorszámuk nem azonos az indexükkel!


```

Verseny(n,m):
  Ciklus i=1-től n-ig
    Ha pont(i,1) ≥ pont(i,2) akkor max(i):=1; min(i):=2
    különben max(i):=2; min(i):=1
    Ciklus j=3-től m-ig
      Ha pont(i,j) > pont(i,max(i)) akkor max(i):=j
      Ha pont(i,j) < pont(i,min(i)) akkor min(i):=j
    Ciklus vége
  Ciklus vége
  Ciklus i=1-től n-ig
    op(i):=-pont(i,max(i))-pont(i,min(i))
    Ciklus j=1-től m-ig
      op(i):=op(i)+pont(i,j)
    Ciklus vége
    s(i):=i
  Ciklus vége
  Ciklus i=1-től n-1-ig
    Ciklus j=i+1-től n-ig
      Ha op(i) < op(j) akkor Csere(op(i),op(j))
      Csere(s(i),s(j))
    Ciklus vége
  Ciklus vége
  x:=1
  Ciklus i=1-től n-ig
    Ki: x, '. helyezett:', s(i), '. versenyző ', op(i), ' pont'
    Ha op(i) > op(i+1) akkor x:=i+1
  Ciklus vége
Eljárás vége.

```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Programozási tételek összeépítése
Feladat	61. Pontozásos verseny eredménye

3. feladat

Az Olimpiai Játékokon M ország vesz részt ($1 \leq M \leq 100$), N versenyszámban versenyeznek a résztvevők ($1 \leq N \leq 1000$). Minden versenyszámban 1 arany-, 1 ezüst-, valamint 1 vagy 2 bronzérmes adnak ki (kieséses versenyek esetén a döntőbe jutásért küzdők közül mindkét vesztes bronzérmes kap). Az országokat 1 és M közötti sorszámmal azonosítjuk.

Készíts programot, amely az eredmények ismeretében előállítja az olimpia éremtáblázatát! Az éremtáblázat aranyérmek száma szerint csökkenő sorrendű legyen. Azonos aranyérem szám esetén a több ezüst-, azonos ezüstérem szám esetén a több bronzérmes döntse el! Ha mindhárom éremből ugyanannyi van, akkor a kisebb sorszámu legyen elől!

Példa

Bemenet

```
M=5, N=3
1. szám: A - 2, E - 3, B - 2
2. szám: A - 1, E - 2, B - 3
3. szám: A - 5, E - 2, B - 2,3
```

Kimenet

```
2. ország: 1 A, 2 E, 2 B
1. ország: 1 A
5. ország: 1 A
3. ország: 1 E, 2 B
```

Számoljuk ki az egyes országok éremszámait, majd az éremszám szerint rendezzük csökkenő sorrendbe őket!

```
olimpia(N,s,M,érem):
  érem(i,j):=(0,...,0)
  Ciklus i=1-től M-ig
    érem(i,0):=i
  Ciklus vége
  Ciklus i=1-től N-ig
    érem(s(i,1),1):=érem(s(i,1),1)+1
    érem(s(i,2),2):=érem(s(i,2),2)+1
    érem(s(i,3),3):=érem(s(i,3),3)+1
    Ha s(i,4)>0 akkor érem(s(i,4),3):=érem(s(i,4),3)+1
  Ciklus vége
  Ciklus i=1-től M-1-ig
    max:=i
    Ciklus j=i+1-től M-ig
      Ha nagyobb(j,max) akkor max:=j
    Ciklus vége
    Csere(érem(max),érem(i))
  Ciklus vége
Eljárás vége.

nagyobb(i,j):
  nagy:=hamis
  Ha érem(i,1)>érem(j,1) akkor nagy:=igaz
  különben ha érem(i,1)=érem(j,1) akkor
    Ha érem(i,2)>érem(j,2) akkor nagy:=igaz
    különben ha érem(i,2)=érem(j,2) akkor
      Ha érem(i,3)>érem(j,3)
        akkor nagy:=igaz
  nagyobb:=nagy
Függvény vége.
```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Programozási tételek: rendezések
Feladat	12. Olimpia

4. feladat

Egy piacon N egymást követő napon árulnak almát. Arra vagyunk kíváncsiak minden napon, hogy az addigi napok közül mely K napon lehetett a legolcsóbban almát venni!

Írj programot, amely beolvassa a napok N számát ($1 \leq N \leq 100$), és K értékét ($1 \leq K \leq 10$), majd ezután egyesével olvassa az egyes napokon az alma árát, s minden beolvasás után kiírja azon K nap sorszámát növekvő sorrendben, amelyeken a legolcsóbban lehetett almát venni! Amíg nem volt K nap, addig a program ne írjon ki semmit!

Példa

Bemenet	Kimenet
N=10, K=4	1 2 3 4
1. nap: 80	1 2 3 4
2. nap: 70	1 2 3 6
3. nap: 75	2 3 6 7
4. nap: 90	2 3 6 7
5. nap: 100	2 3 6 7, de a 2 3 6 9 is jó megoldás
6. nap: 60	2 3 6 7, de a 2 3 6 9 is jó megoldás
7. nap: 77	
8. nap: 80	
9. nap: 77	
10. nap: 90	

Ha még nem volt K nap, akkor sorba rendezve gyűjtjük az árakat és a napsorszámokat ($T(i)$.ár, illetve $t(i)$.sorszám). Ha már volt K nap, akkor ha a K -adik legolcsóbbnál olcsóbb az alma, akkor beillesztjük a tároltak közé.

```

Almák(n, k):
  üres(h)
  Ciklus i=1-től n-ig
    Be: ár
    Ha i≤k akkor j:=i-1
      Ciklus amíg j>0 és ár<t(j).ár
        t(j+1):=t(j); j:=j-1
      Ciklus vége
      t(j+1).ár:=ár; t(j+1).sorszám:=i
      Halmazba(h, i)
    különben ha ár<t(k).ár
      akkor j:=k-1; halmazból(h, t(k).sorszám)
      Ciklus amíg j>0 és ár<t(j).ár
        t(j+1):=t(j); j:=j-1
      Ciklus vége
      t(j+1).ár:=ár; t(j+1).sorszám:=i
      Halmazba(h, i)
    Ha i≥k akkor Ciklus j=1-től i-ig
      Ha eleme(j, h) akkor Ki: j
      Ciklus vége
  Ciklus vége
Eljárás vége.

```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Interaktív feladatok, online algoritmusok
Feladat	3. Legolcsóbb almák

5. feladat

Egy piacon M árus N egymást követő napon árul almát. Az árusok különböző napokon kezdenek almát árulni, s ettől kezdve, amíg más árat nem adnak, ugyanazon az áron adják az almát. Arra vagyunk kíváncsiak minden napon, hogy aznap mely K árustól lehet a legolcsóbban almát venni!

Írj programot, amely megadja minden napra, hogy aznap mely K árustól lehet a legolcsóbban almát venni, ha van aznap egyáltalán K árus!

A bemenet első sorában az árusok M száma ($1 \leq M \leq 100$), a napok N száma ($1 \leq N \leq 1000$), a K értéke ($1 \leq K \leq M$) és az árusok adatsorainak S száma ($1 \leq S \leq 1000$) van egy-egy szóközzel elválasztva. A további sorok egy-egy árus érkezését írják le. Mindegyik sorban három szám van egy-egy szóközzel elválasztva: az érkezés napja ($1 \leq \text{nap} \leq N$, a sorok eszerint növekvő sorrendben jönnek), az árus sorszáma ($1 \leq \text{sorszám} \leq M$) és az általa árult alma ára attól a naptól kezdve ($0 < \text{ár} \leq 1000$).

A kimenet pontosan N sort kell írni, az i -edik sorba az i -edik napon legolcsóbb K árus **sorszámat**, növekvő sorrendben, egy-egy szóközzel elválasztva. Ha aznap nem árult almát K árus, akkor a sorba egyetlen 0-t kell kiírni.

Példa

Bemenet	Kimenet	Magyarázat
M=6 N=8 K=3 S=9	0	az első napon nincs 3 árus
1. adatsor: 1 1 100	2 3 6	
2. adatsor: 1 2 90	1 3 6	
3. adatsor: 2 6 80	1 3 6	a negyedik napon nem jött újabb árus
4. adatsor: 2 3 70	1 3 6	az ötödik napon nem jött újabb árus
5. adatsor: 2 5 120	1 3 6	a hatodik napon nem jött újabb árus
6. adatsor: 3 1 60	3 4 6	
7. adatsor: 3 4 100	3 4 6	
8. adatsor: 7 1 120		
9. adatsor: 7 4 75		a nyolcadik napon nem jött újabb árus

Naponta figyeljük a K legolcsóbb árust. A bemeneten előreolvasunk egy sort, a következő árus érkezését. Azon a napon van teendőnk, amely napon jött árus (több is lehet egymás után).

Ekkor meg kell néznünk, hogy árult-e már korábban. Ha szerepelt, akkor el kell távolítani: ki kell szedni a legolcsóbb K halmazából (h), valamint az árak szerint rendezett tömbből is ki kell hagyni! Ha már több, mint K árusunk van, akkor a $K+1$ -ediket kell bevinnünk a legolcsóbbak közé!

Ezután az új árat be kell szúrni a rendezett tömbbe, s ha a legolcsóbb K közé kerül, akkor a h halmazba is fel kell venni. Ha a halmazban már volt K elem, akkor a legnagyobbat ki kell hagyni.

Minden nap végén meg kell nézni, hogy van-e K árus, s ha igen, akkor a h halmaz elemeit kell kiírni!

```

Almák(n,m,k,s):
  Be:nap,sorszám,ár; sdb:=1
  db:=0; üres(h)
  Ciklus ns=1-től n-ig
    Ciklus amíg nap=ns
      j:=1
      Ciklus amíg j≤db és t(j).sorszám≠sorszám
        j:=j+1
      Ciklus vége
      Ha j≤db akkor
        Ha j≤k akkor Halmazból(h,t(j).sorszám)
          Ha db>k akkor Halmazba(h,t(k+1).sorszám)
        Ciklus amíg j≤db
          t(j):=t(j+1); j:=j+1
        Ciklus vége
        db:=db-1
      Elágazás vége
      j:=db
      Ciklus amíg j>0 és ár<t(j).ár
        t(j+1):=t(j); j:=j-1
      Ciklus vége
      t(j+1).ár:=ár; t(j+1).sorszám:=sorszám; db:=db+1
      Ha j+1≤k akkor Halmazba(h,sorszám)
        Ha db>k akkor Halmazból(h,t(k+1).sorszám)
      Ha sdb<s akkor Be:nap,sorszám,ár; sdb:=sdb+1
        különben nap:=n+1
      Ciklus vége
      Ha db≥k akkor Ciklus i=1-től m-ig
        Ha eleme(i,h) akkor Ki:i
        Ciklus vége
        különben Ki:0
      Ciklus vége
Eljárás vége.

```

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Kezdő
Téma	Programozási tételek: rendezések
Feladat	8. Legolcsóbb almák