Gyakorlás

Dr. Szeghalmy Szilvia
Debreceni Egyetem, Informatikai Kar

Tömb

Pékáru láncolatunk folyamatosan figyeli a termékek fogadtatását a piacon.
A különböző sütemények nevét, jellegét és a vizsgált időszakra vonatkozó keresleti adatokat *n*>0 elemű tömbökben (*nev, jelleg, kereslet*) tároljuk. Az azonos pozíción lévő értékek összetartoznak.

Feladat: Írj függvényt, mely megjeleníti az összes olyan "sós" jellegű sütemény nevét, melyekre a kereslet meghaladta az *x* paraméterben kapott értéket.

```
def kiir( nev, jelleg, kereslet, n, x):
    for i in range(n):
        if jelleg[i] == "sós" and kereslet[i]>x:
            print(nev[i])
```

	0	1	...	n-1
nev	sajt roló			isler
jelleg	sós	sós	...	édes
kereslet	100	50		84

Tömb

Pékáru láncolatunk folyamatosan figyeli a termékek fogadtatását a piacon.
A különböző sütemények nevét, jellegét és a vizsgált időszakra vonatkozó keresleti adatokat *n*>0 elemű tömbökben (*nev, jelleg, kereslet*) tároljuk. Az azonos pozíción lévő értékek összetartoznak.

Feladat: Írj függvényt, mely meghatározza, hogy mennyi volt az átlagos kereslet a vizsgált időszakban az "édes" jellegű pékáruk iránt. Ha nem volt édes süti az adott időszakban, (-1)-es értéket adj vissza.

```
def atlagkereslet_edes(jelleg, kereslet, n ):
    db = 0
    s = 0
    for i in range(n):
        if jelleg[i] == "édes":
            s += kereslet[i]
            db += 1
    if db == 0:
       return -1
    return s/db
```

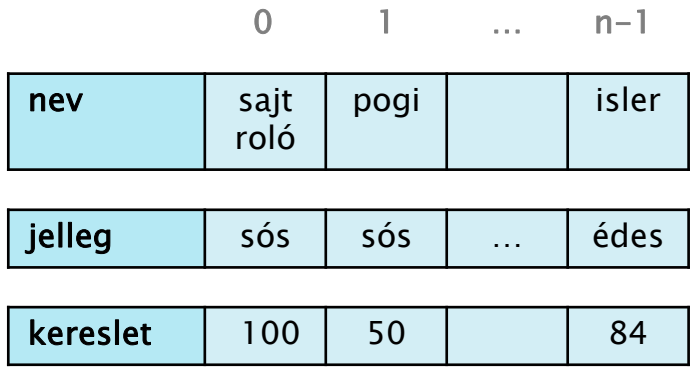
	0	1	...	n-1
nev	sajt roló	pogi		isler
jelleg	sós	sós	...	édes
kereslet	100	50		84

Tömb

Pékáru láncolatunk folyamatosan figyeli a termékek fogadtatását a piacon.
A különböző sütemények nevét, jellegét és a vizsgált időszakra vonatkozó keresleti adatokat *n*>0 elemű tömbökben (*nev, jelleg, kereslet*) tároljuk. Az azonos pozíción lévő értékek összetartoznak.

Feladat: Írj függvényt, mely meghatározza, hogy melyik volt a **legkeresettebb** "sós" jellegű pékáru a vizsgált időszakban.
(Tf. nincs holtverseny és forgalmaztunk sós terméket).

```
import math
def legkeresettebb_sos(nev, jelleg, kereslet, n ):
    max_ker = -math.inf
    max_suti =""
    for i in range(n):
        if kereslet[i] > max_ker and jelleg[i] == "sós":
           max_ker = kereslet[i]
           max_suti = nev[i]
    return max_suti
```



	0	1	...	n-1
nev	sajt roló	pogi		isler
jelleg	sós	sós	...	édes
kereslet	100	50		84

Tömb

Pékáru láncolatunk folyamatosan figyeli a termékek fogadtatását a piacon.
A különböző sütemények nevét, jellegét és a vizsgált időszakra vonatkozó keresleti adatokat *n*>0 elemű tömbökben (*nev, jelleg, kereslet*) tároljuk. Az azonos pozíción lévő értékek összetartoznak.

Feladat: Írj függvényt, mely visszaadja az *s* paraméterben megadott nevű pékáru keresletét. (a nevek egyediek)
Ha az *s* paraméterben kapott érték nem található, akkor 0 értéket adj vissza.

```
def suti_kereslete(nev, kereslet, n, s):
    for i in range(n):
        if nev[i] == s:
            return kereslet[i]
    return 0
```

	0	1	...	n-1
nev	sajt roló	pogi		isler
jelleg	sós	sós	...	édes
kereslet	100	50		84

Mátrix

	0. tárgy	1. tárgy	…	9. tárgy
0. diák	5	4		5
1. diák	3	2		2
…				
N-1. diák	2	1		4

Az elsős diákoknak 10 féle tantárgyuk van a suliban. A diákok év végi osztályzatait osztályonként egy-egy mátrixban tároljuk. A diákok számát N jelöli, ez osztályonként eltérhet. A mátrix egy cellája megadja, hogy mely diák, mely tárgyból milyen jegyet szerzett.

Feladat: Írj függvényt, mely visszaadja a matematika tantárgynál az átlagot.
A matematika a 0. tárgy a mátrixban. (N>0)

```
def matek_atlag(jegyek, N):   # jegyek matrix: N x 10
    s = 0
    for i in range(N):
       s += jegyek[i, 0]
    return s/N
```

Mátrix

	0. tárgy	1. tárgy	…	9. tárgy
0. diák	5	4		5
1. diák	3	2		2
…				
N-1. diák	2	1		4

Az elsős diákoknak 10 féle tantárgyuk van a suliban. A diákok év végi osztályzatait osztályonként egy-egy mátrixban tároljuk. A diákok számát N jelöli, ez osztályonként eltérhet. A mátrix egy cellája megadja, hogy mely diák, mely tárgyból milyen jegyet szerzett.

Feladat: Írj függvényt, mely megszámolja, hány 5-ös érdemjegyet szerzett az osztály!

```
def otosok_szama(jegyek, N):   # jegyek matrix: N x 10
    db = 0
    for i in range(N):
       for j in range(10):
           if jegyek[i, j] == 5:
              db += 1
    return db
```

Mátrix

nevek		0. tárgy	1. tárgy	…	9. tárgy
Éva	0. diák	5	4		5
Pál	1. diák	3	2		2
…	…				
Jenő	N-1. diák	2	1		4

Az elsős diákoknak 10 féle tantárgyuk van a suliban. A diákok év végi osztályzatait osztályonként egy-egy mátrixban tároljuk. A diákok számát N jelöli, ez osztályonként eltérhet. A mátrix egy cellája megadja, hogy mely diák, mely tárgyból milyen jegyet szerzett.

Feladat: Írj függvényt, mely megjeleníti a kitűnő tanulók neveit (mindenből 5-ös).
Az i. diák neve a nevek tömb i. pozícióján van.

```
def kitunok(jegyek, nevek, N):
    for i in range(N):
        jeles = True
        for j in range(10):
            if jegyek[i, j] < 5:
               jeles = False
               break
        if jeles:
           print( nevek[i] )
```

```
def kitunok(jegyek, nevek, N):
    for i in range(N):
        s = 0
        for j in range(10):
            s += jegyek[i, j]
        if s == 50:
           print( nevek[i] )
```

Mátrix

	0. falu	1. falu	2. falu	…	9. falu
0. falu	0	4	3	…	5
1. falu	4	0	6	…	2
2. falu	3	6	0	…	7
…	…	…	…	…	1
9. falu	5	2	7	1	0

Egy szimmetrikus mátrixban 10 falu légvonalban mért távolságát tároljuk (tavok).

Feladat: Mivel az i. és j. falu távolsága és j. és i. falu távolsága nyilván valóan ugyanannyi légvonalban mérve, felesleges duplán tárolni az információt, és a falu önmagától való távolsága sem érdekel bennünket. Írj függvényt, mely leképezi a szimmetrikus mátrix **főátló alatti** területét paraméterként kapott tömbbe. A tömbben van elég hely a tároláshoz.

```
def szimm_lekepez(tavok, tomb):   # tavok matrix: 10 x 10
    db = 0
    for i in range(1, 10):
        for j in range(i): # átlót kihagyjuk
            tomb[db] = tavok[i, j]
            db += 1
```

Ritka mátrix

sor (mérőállomás)	0	0	1	1	…
oszlop (év napja)	4	156	68	365	…
ertek (rengés erőssége)	2	4	1	2	…
gyakori elem: 0					

A világ N meteorológiai állomása az év minden napján méri a földmozgások erősségét. Az adatok egy N x M-es (Nx365 vagy Nx366) mátrixban lehetne rögzíteni, de mivel a legtöbb napon szerencsére nincs érdemleges földmozgás, ezért a tároláshoz **háromsoros reprezentációt** használunk (*n* elemű *reprez* tömb).
[(sor, oszlop, ertek), (sor, oszlop, ertek), …]

Feladat: Írj függvény, mely visszaadja, hogy hányszor mért 3-as erősségű földmozgást az *m*. (eredeti mátrix oszlopindexe) mérőállomás!

```
def kozepes_renges(reprez, n, m):
    db = 0
    for i in range(n):
        if reprez[i][1] == m and reprez[i][2] == 3:
            db += 1
    return db
```

```
def kozepes_renges(reprez, n, m):
    db = 0
    for i in range(n):
        allomas, nap, erosseg = reprez[i]
        if allomas == m and erosseg == 3:
            db += 1
    return db
```

Ritka mátrix

sor (mérőállomás)	0	0	1	1	…
oszlop (év napja)	4	156	68	365	…
ertek (rengés erőssége)	2	4	1	2	…
gyakori elem: 0					

A világ N meteorológiai állomása az év minden napján méri a földmozgások erősségét. Az adatok egy N x M-es (Nx365 vagy Nx366) mátrixban lehetne rögzíteni, de mivel a legtöbb napon szerencsére nincs érdemleges földmozgás, ezért a tároláshoz **háromsoros reprezentációt** használunk (*n* elemű *reprez* tömb).
[(sor, oszlop, ertek), (sor, oszlop, ertek), …]

Feladat: Írj függvény, mely átmásolja az *allomas* tömbbe azon mérőállomások sorszámát, melyek rengést mértek az év 100. napján. A tömbben van elég hely a mérőállomások sorszámának bemásolásához. A tömböt folytonosan töltsd fel.

```
def renges_100(reprez, n, allomas):
    db = 0
    for i in range(n):
        if reprez[i][2] == 100:
            allomas[db] = reprez[i][0]
            db += 1
```

```
def renges_100(reprez, n, allomas):
    db = 0
    for i in range(n):
        allomas, nap, erosseg = reprez[i]
        if nap == 100:
            allomas[db] = allomas
            db += 1
```

Ritka mátrix

sor (mérőállomás)	0	0	1	1	…
oszlop (év napja)	4	156	68	365	…
ertek (rengés erőssége)	2	4	1	2	…
gyakori elem: 0					

A világ N meteorológiai állomása az év minden napján méri a földmozgások erősségét. Az adatok egy N x M-es (Nx365 vagy Nx366) mátrixban lehetne rögzíteni, de mivel a legtöbb napon szerencsére nincs érdemleges földmozgás, ezért a tároláshoz **háromsoros reprezentációt** használunk (*n* elemű *reprez* tömb).
[(sor, oszlop, ertek), (sor, oszlop, ertek), …]

Feladat: Írj függvény, mely visszaadja, hogy a magyarországi mérőállomás (1-es sorszámú) milyen erősségű rengést mért az év *x.* napján.

```
def foldrenges_hazankban(reprez, n, x):
    for i in range(n):
        if reprez[i][0] == 1 and reprez[i][1] == x:
            return reprez[i][2]
    return 0 # gyakori elem, nem volt rengés
```

```
def foldrenges_hazankban(reprez, n, allomas):
    for i in range(n):
        allomas, nap, erosseg = reprez[i]
        if allomas == 1 and nap == x:
            return erosseg
    return 0
```

Halmaz

Bibircsóka bájitalait különböző halmazokba soroljuk be a bennük lévő alapanyagok alapján (*bekanyalas*, *torpkivonatos*, *sarkanyfuves*, stb.). A bájitalokat 0-tól kezdődő egész sorszámmal látjuk el: {0, 1, 2, …, *b-1*}, ahol b a bájitalok száma. Ez képezi az alaphalmazt a reprezentáció során.

bajital sorsz:	0	1	…	b-1
bekanyalas	0	1	…	1
sarkanyfuves	0	0		1
undormanyos	0	0		1

Feladat: Írj függvény, mely megkapja a békanyálat és sárkányfüvet tartalmazó bájitalok halmazának reprezentációját és visszaadja azon halmaz reprezentációját a *b* elemű *undormanyos* tömbbe, melyek mindkét összetevőt tartalmazzák.

```
def halmaz1(bekanyalas, sarkanyfuves, undormanyos, b):
    for i in range(b):
        undormanyos[i] = 1 if bekanyalas[i] and sarkanyfuves[i] else 0
```

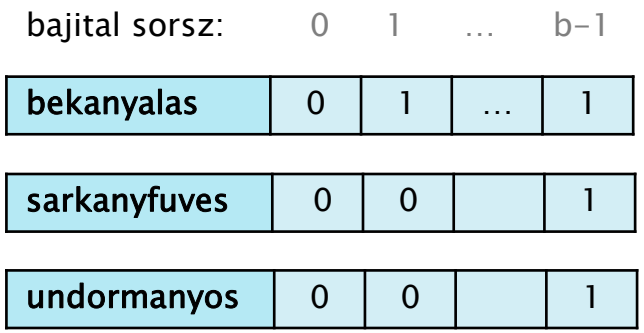
Halmaz

Bibircsóka bájitalait különböző halmazokba soroljuk be a bennük lévő alapanyagok alapján (*bekanyalas*, *torpkivonatos*, *sarkanyfuves*, stb.). A bájitalokat 0-tól kezdődő egész sorszámmal látjuk el: {0, 1, 2, …, *b-1*}, ahol b a bájitalok száma. Ez képezi az alaphalmazt a reprezentáció során.

Feladat: Írj függvény, mely megkapja a békanyálat tartalmazó bájitalok halmazának reprezentációját, és egy létező bájital sorszámot (sorsz). Adj vissza True értéket, ha a bájital tartalmaz békanyálat, különben adj vissza False értéket.

```
def halmaz2(bekanyalas, b, sorsz):
    # ez a feladatkiírás szerint teljesül, ha assert-et használasz,
    az debug módban jelzi, ha valaki mégis rosszul hívta meg a fv-t
    assert( 0<= sorsz < b)

    # algoritmus szempontjából ennyi (egyszerűsített eleme művelet)
    return bekanyalas[sorsz] > 0
```



Multihalmaz

Pékáru sütödénk egy multihalmazokban tárolja, hogy mely péksütiből éppen mennyi kiszállítható darab van, mennyi készült el, vagy épp mennyi került kiszállításra. A nyilvántartásnak minden kiszállításnál és minden sütés után frissülnie kell!

A péksütik azonosítója: {0, 1, 2, …, *n-1*}. Ez képezi az alaphalmazt a reprezentáció során.

Feladat: Írj függvényt, mely paraméterben megkapja a kiszállítható és a frissen elkészült sütemények multihalmazának reprezentációját. Frissítsd a kiszállítható péksütik multihalmazának reprezentációját!

```
def multihalmaz1( kiszallithato, kisutott, n):
    # multihalmaz unio, az eredményt a kiszallithatoba visszairva
    for i in range(n):
        kiszallithato[i] += kisutott[i]
```

péksüti sorsz:	0	1	…	n−1
kiszallithato	6	1	…	1
kisutott	4	2	…	1
kiszallithato	10	3	…	2

Multihalmaz

Pékáru sütödénk egy multihalmazokban tárolja, hogy mely péksütiből éppen mennyi kiszállítható darab van, mennyi készült el, vagy épp mennyi került kiszállításra. A nyilvántartásnak minden kiszállításnál és minden sütés után frissülnie kell!

A péksütik azonosítója: {0, 1, 2, …, *n-1*}. Ez képezi az alaphalmazt a reprezentáció során.

Feladat: Írj függvényt, mely paraméterben megkapja a kiszállítható és az éppen most kiszállított sütemények multihalmazának reprezentációját. Frissítsd a kiszállítható péksütik multihalmazának reprezentációját!

```
def multihalmaz2( kiszallithato, szallitmany, n):
    # A multihalmaz különbség művelete.
    # Normál esetben ellenőriznénk, hogy van-e értelme a kivonásnak, de
    # most elhagyható, hiszen fizikailag nem tudunk kiszállítani,
    # többet, mint amennyi süti éppen van
     for i in range(n):
         # if kiszállítható[i] > kisutott[i]:
             kiszallithato[i] -= kisutott[i]
         # else:
         #   kiszallithato[i] = 0
```

péksüti sorsz:	0	1	…	n−1
kiszallithato	6	1	…	1
szallitmany	4	1	…	1
kiszallithato	2	0	…	0

Rögzített sor

- Mi történik a sorral a műveletek hatására?

		0	1	2	3	4	5	
`kiinduló állapot:`	sor	1	6	5				
`put(2)`	sor	1	6	5	2			
`x = get()`	sor	6	5	2				x = 1
`y = access()`	sor	6	5	2				y = 6
`put(x+y)`	sor	6	5	2	7			
`get()`	sor	5	2	7				
`access()`	sor	5	2	7				
`put(-get())`	sor	2	7	-5				

Gyakorlás: sor

Vállalkozásunk beindításához tőkére volt szükségünk, ezért adományokat fogadtunk a leendő ügyfeleinktől. A első néhány olyan adományozónak, aki 10000Ft feletti adományt küldött, ingyenes terméket küldünk a gyártás beindulását követően. Az ő adatai egy sorba kerülnek.

Készíts függvényt, mely megkapja az adományozó nevét, címét és az utalás mértékét (Ft). Ha 10000Ft feletti adomány érkezett, pakold be a sorba az adományozót, ha még befér.

```
def bepakol(nev, cim, utalas):
    if utalas > 10000 and not full():
        put(Adomanyozo(nev, cím))
```

```
class Adomanyozo:
  def __init__(self, n, c):
     self.nev = n
     self.cim = c


Példa egy adományozó létrehozására
t = Adomanyozo("Kiss Ákos","4028
Debrecen, Kassai út 27.")
```

Az építőkockáink:

```
empty()
full()
access()
get()
put(adomanyozo)
```

Az access és get kivételt dob, ha a sor üres.

A put kivételt dob, ha a sor megtelt.

Gyakorlás: sor

Vállalkozásunk beindításához tőkére volt szükségünk, ezért adományokat fogadtunk a leendő ügyfeleinktől. A első néhány olyan adományozónak, aki 10000Ft feletti adományt küldött, ingyenes terméket küldünk a gyártás beindulását követően. Az ő adatai egy sorba kerülnek.

Cégünk végre megkezdte a gyártást. Az első terméket a legelső adományozóknak küldjük ki. Vedd ki a sorból és jelenítsd meg a standard kimeneten a címét.

```
def hovakuld():
     if not empty():
         print( get().cim )
```

```
class Adomanyozo:
  def __init__(self, n, c):
     self.nev = n
     self.cim = c


Példa egy adományozó létrehozására
t = Adomanyozo("Kiss Ákos","4028
Debrecen, Kassai út 27.")
```

Az építőkockáink:

```
empty()
full()
access()
get()
put(adoanyozo)
```

Az access és get kivételt dob, ha a sor üres.

A put kivételt dob, ha a sor megtelt.

Gyakorlás: sor

Vállalkozásunk beindításához tőkére volt szükségünk, ezért adományokat fogadtunk a leendő ügyfeleinktől. A első néhány olyan adományozónak, aki 10000Ft feletti adományt küldött, ingyenes terméket küldünk a gyártás beindulását követően. Az ő adatai egy sorba kerülnek.

Az első termékek gyártását sok másik követte. Most már az összes adományozónak el tudjuk küldeni az ajándékokat.

Pakold át a sorban szereplő összes adományozó címét egy tömbbe, hogy megcímezhessük a csomagokat. A *cimek* tömbben van elég hely.
A függvény térjen vissza a tömbbe bekerült címek számával.

```
def csomagok_cime(cimek):
    i = 0
    while not empty():
        cimek[i] = get()
        i += 1
    return i
```

```
class Adomanyozo:
  def __init__(self, n, c):
     self.nev = n
     self.cim = c


Példa egy adományozó létrehozására
t = Adomanyozo("Kiss Ákos","4028
Debrecen, Kassai út 27.")
```

Az építőkockáink:

```
empty()
full()
access()
get()
put(torp)
```

Az access és get kivételt dob, ha a sor üres.

A put kivételt dob, ha a sor megtelt.

Kérdés?