Soros táblázat adatszerkezet

Dr. Szeghalmy Szilvia
Debreceni Egyetem, Informatikai Kar

A soros táblázat

- Az 1 dimenziós tömb kiterjesztése
- Tulajdonságok
  - Homogén
  - Asszociatív
  - **Dinamikus**
  - Elemek összetettek:

    kulcs (egyedi azonosító)

    érték
- Reprezentáció: folytonos
  - Tömb
  - Az elemek érkezési sorrendben kerülnek be
  - Az elemek a tömbben folytonosan állnak

cszam	cnev	bar (Ft)	ear (Ft)
"XAZ"	"alma (Jonathán)"	100	400
"WEW"	"körte (Vilmos)"	200	800
"WER"	"alma (Starking)"	120	350
…			
…			

Műveletek: Létrehozás

- A szerkezetet alakítjuk ki a problémához igazodva
- példa:

```
class Cikk:
    def __init__(self, csz, cnev, bar, ear):
            self.cszam = csz      # cikkszám, kulcs
            self.cnev = cnev      # cikk megnevezése
            self.bar = bar        # beszerzési ár
            self.ear = ear        # eladási ár
            … # dátum, mennység, mértékegység, stb.

MAX_MERET = 1000
tabla = np.empty(MAX_MERET, dtype = 'object')
elemszam = 0
```

Műveletek: elérés, bejárás

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

- **Elérés**: index alapján közvetlen (0<= i <**elemszam**)

```
tabla[i]  # a táblázatban szereplő i. cikk
```

- **Bejárás:** minden elem elérése pontosan egyszer

  Pl.: jelenítsük meg az összes elem cikkszámát és nevét.

```
def kiir_cszam_es_nev():
    for i in range(elemszam):
        print(tabla[i].cszam, tabla[i].cnev)
```

Műveletek: keresés

- **Lineáris keresés rendezetlen elemekre**

Pl.: Add vissza, hogy a *csz* cikkszámú termék hol van a táblázatban. Ha nincs benne, -1 értéket adj vissza.

```
def hol_van( csz ):
    for i in range(elemszam):
        if tabla[i].cszam == csz:
            return i
    return -1
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Műveletek: bővítés

- **Csak ha van hely**
- **Csak ha az adott kulcs még nem létezik a táblázatban**
- Használd a *hol_van(cszam)* függvényt!

```
def bovit( uj_cikk ):
    global elemszam
    # van hely?
    if elemszam < MAX_MERET:
        # van ilyen kulcs?
        if hol_van( uj_cikk.cszam ) == -1: #nincs
            tabla[elemszam] = uj_cikk
            elemszam
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Műveletek: törlés

- **Csak ha van elem**
- **Csak ha van adott kulcsú elem**
  (ez a szigorúbb feltétel, elég ezt nézni)
- Használd a *hol_van(cszam)* függvényt!

```
def torol( torlendo_cszam ):
    global elemszam
    idx = hol_van( torlendo_cszam )
    if idx != -1:  # találat
        # írd felül az utolsóval:
        tabla[idx] = tabla[elemszam-1]
        # "töröld" az utolsót:
        elemszam -= 1
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Műveletek: csere

- **Csak ha van elem**
- **Csak ha van adott kulcsú elem (szigorúbb felt, ezt elég nézni)**
- Használd a *hol_van(cszam)* függvényt!

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Pl.: írjuk felül az adott cikkszámú termék beszerzési árát a paraméterben kapott értékre.

```
def csere( cszam, uj_bar ):
    idx = hol_van( torlendo_cszam )
    if idx != -1:  # találat
        tabla[idx].bar = uj_bar
```

Gyakorlás

- Jelenítsd meg azon termékek cikkszámát és nevét, ahol a beszerzési ár meghaladja az eladási árat.

```
def rossz_ar():
    for i in range(elemszam):
        if tabla[i].bar > tabla[i].ear:
           print(tabla[i].cszam, tabla[i].cnev)
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás

- Számold ki a termékek **átlagos eladási ár**át!
- Ha nincs egyetlen termék sem, akkor -1 értéket adj vissza.

```
def atlag():
    if elemszam == 0:
       return -1

    s = 0
    for i in range(elemszam):
        s += tabla[i].ear
    return s / elemszam
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás

- Add vissza annak a terméknek a cikkszámát, amelyiknek a legnagyobb az **eladási** ára. (Tf.h.: nincs holtverseny.)
- Ha nincs termék a táblázatban, akkor üressztringet adj vissza.

```
import math
def legdragabb_cikk_cikkszama():
    max_ear = -math.inf
    max_cszam = ""  # ha nincs, ezt adjuk vissza

    for i in range(elemszam):
        if max_ear < tabla[i].ear:
            max_ear = tabla[i].ear
            max_cszam = tabla[i].cszam

    return max_cszam
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás

- Módosítsd az "XXX" cikkszámú termék nevét "egres"-re.
- A megoldáshoz használhatod a hol_van(cszam) függvényt, mely vagy az adott cikkszámú termék táblázatbeli helyét vagy ha nincs, -1-es értéket ad vissza.

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

```
def nev_csere():
    idx = hol_van("XXX")
    if idx != -1: # találat
        tabla[idx].cnev = "egres"
```

```
vagy pl.:
def nev_csere():
    for i in range(elemszam):
       if tabla[i].cszam == "XXX":
          tabla[i].cnev = "egres"
          break # felesleges tovább nézni, hiszen az XXX kulcs, csak 1x szerepelhet
```

Gyakorlás

- Emeld meg az összes táblázatban szereplő almák eladási árát 20%-kal.
- A részsztring kereséshez használhatod az *in* operátort.

```
def aremeles():
    for i in range(elemszam):
        if "alma" in tabla[i].cnev:
            tabla[i].ear *= 1.2
```

```
# LÉTREHOZÁS:
class Cikk:
   def __init__(self,csz,cnev,bar,ear):
        self.cszam = csz
        self.cnev = cnev
        self.bar = bar
        self.ear = ear
        …

MAX_MERET=1000
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás: HALLGATÓK

- A táblázatban ezúttal hallgatók hiányzásait tároljuk.
- **Összetett kulcs** van: a neptunkód és a gyakorlat együtt nézve egyedi.

- Írj függvényt, mely a neptunkód és a gyakorlat neve alapján visszaadja, hogy az adott hallgató hol szerepel a táblázatban. Ha a kulcs nem létezik, -1 értéket adj vissza.

```
def hol_van2(nk, gyak):
    for i in range(elemszam):
        if tabla[i].neptun == nk and
           tabla[i].gyakorlat == gyak:
           return i
    return -1
```

```
# LÉTREHOZÁS:
class Hallgato:
   def __init__(self, nk, gyak, hsz ):
        self.neptun = nk
        self.gyakorlat = gyak
        self.hianyzasok_szama = hsz

MAX_MERET=150
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás: HALLGATÓK

- Az "XXXXXX" neptunkódú hallgató nem jelent meg az "Adatszerk" gyakorlaton. Növeld meg a hiányzásai számát.
- A megoldás során használhatod az előbb megírt hol_van2(nk, gyak) függvényt.

```
def hianyzas_bejegyzese():
    idx = hol_van2("XXXXXX", "Adatszerk")
    if idx != -1:
       tabla[idx].hianyzasok_szam += 1
```

```
# LÉTREHOZÁS:
class Hallgato:
   def __init__(self, nk, gyak, hsz ):
        self.neptun = nk
        self.gyakorlat = gyak
        self.hianyzasok_szama = hsz

MAX_MERET=150
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás: HALLGATÓK

- Írj függvényt, mely egy hallgató neptunkódja alapján kilistázza, hogy mely tárgyakból hiányzott már túl sokat.
- Emlékeztetőül: 3 megengedett, 4 már nem.

```
def alairas_megtagadva(nk):
    for i in range(elemszam):
        if tabla[i].neptun == nk and
           tabla[i].hianyzasok_szama > 3:
           print(tabla[i].gyakorlat)
```

```
# LÉTREHOZÁS:
class Hallgato:
   def __init__(self, nk, gyak, hsz ):
        self.neptun = nk
        self.gyakorlat = gyak
        self.hianyzasok_szama = hsz

MAX_MERET=150
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```

Gyakorlás: HALLGATÓK

- Írj függvényt, mely átmásolja azon hallgatók neptunkódját egy tömbbe, akik **"Adatszerk"** tárgyból **pontosan háromszor hiányoztak**.
- A tömb létezik (tomb paraméter), és van elég hely benne.
- Folytonosan legyen feltöltve.

```
def neptunkodot_masol(tomb):
    db = 0
    for i in range(elemszam):
        if tabla[i].gyakorlat == "Adatszerk" and
           tabla[i].hianyzasok_szama == 3:
             tomb[db] = tabla[i].neptun
             db += 1
```

```
# LÉTREHOZÁS:
class Hallgato:
   def __init__(self, nk, gyak, hsz ):
        self.neptun = nk
        self.gyakorlat = gyak
        self.hianyzasok_szama = hsz

MAX_MERET=150
tabla=np.empty(MAX_MERET,dtype='object')
elemszam=0
```