

Halmaz, multihalmaz adatszerkezet

Dr. Szeghalmy Szilvia
Debreceni Egyetem, Informatikai Kar

A halmaz

- ▶ Matematikai halmaz fogalom
- ▶ Tulajdonságok
 - Homogén
 - Statikus (alaphalmaz mérete rögzített) és dinamikus is lehet
 - Az adatelemek között nincs kapcsolat, **nincs sorrend**.
 - Véges (ez eltér a matek órától)
- ▶ Reprezentáció: folytonos/szétszórt
- ▶ Folytonos reprezentáció:
 - Alaphalmaz megadása, rögzített elemsorrenddel
 - Kényelmi szempontból a halmaznál: $0, 1, \dots, n-1$ (ezentúl röviden $[0, n-1]$)
 - Karakterisztikus függvény

$$f: [0, n - 1] \rightarrow \{0, 1\}$$

Halmaz: reprezentáció

Feladat: Add meg a $\{2, 3\}$ halmaz reprezentációját a $[0, 3]$ egész számokból álló zárt tartomány felett.

	0	1	2	3
reprez	0	0	1	1

Feladat: Add meg a $\{2, 1, 4\}$ halmaz reprezentációját a $\{0, 1, \dots, 5\}$ alaphalmaz felett.

	0	1	2	3	4	5
reprez	0	1	1	0	1	0

Halmaz: reprezentáció

Feladat: Add meg a $[0, 1, 0, 1]$ reprezentációkhoz tartozó halmazt.

$A = \{ 1, 3 \}$

	0	1	2	3
reprez	0	1	0	1

Feladat: Add meg a $[1, 0, 1, 1, 0, 1]$ reprezentációkhoz tartozó halmazt.

$A = \{ 0, 2, 3, 5 \}$

	0	1	2	3	4	5
reprez	1	0	1	1	0	1

Műveletek

▶ Létrehozás:

- Üres halmaz létrehozása

```
Arep = np.zeros(alaphalmaz_elemszama, np.uint8)
```

- Adott halmaz létrehozása

```
Brep = np.array(reprezentacio, np.uint8)
```

pl: $B = \{2, 4\}$ halmaz reprezentációja a $[0, 4]$ egészszámokból álló tartomány felett:

```
Brep = np.array([0, 0, 1, 0, 1], np.uint8)
```

▶ Elérés: nincs, helyette eleme-e művelet van

▶ Bejárás: -

▶ Módosítás:

- Bővítés: unió művelettel
- Törlés: különbség művelettel
- Cseré: nincs, de szimulálható: pl. x cseréje y -ra: $A = A \setminus \{x\} \cup \{y\}$

▶ Keresés: -

▶ Rendezés: -

▶ Feldolgozó műveletek: a halmazműveleteken alapulnak

Műveletek: eleme-e

Feladat: Írj függvényt, mely egy A halmaz reprezentációját (n elemű *Arep* tömb) és egy e egész számot kap meg paraméterként. A függvény adjon vissza True értéket, ha az e eleme az A halmaznak, különben adjon vissza False értéket.

```
def eleme_e(Arep, n, e ):
    return 0 <= e < n and Arep[e] == 1
```

```
Arep = np.array([1, 0, 1, 1], np.uint8) # A: {0, 2, 3}
n = len(Arep)
print(eleme_e(Arep, n, 3)) # True
print(eleme_e(Arep, n, 1)) # False
print(eleme_e(Arep, n, 10)) # False
print(eleme_e(Arep, n, -2)) # False
```

	0	1	2	3
Arep	1	0	1	1

Műveletek: unio

Feladat: Írj függvényt, mely egy A és egy B halmaz reprezentációját (n elemű $Arep$ és $Brep$ tömb) kapja meg paraméterként. A függvény állítsa elő az $A \cup B$ halmaz reprezentációját az n elemű $Erep$ tömbbe.

```
def unio(Arep, Brep, Erep, n):  
    for i in range(n):  
        if Arep[i] or Brep[i]:  
            Erep[i] = 1  
        else:  
            Erep[i] = 0
```

```
Arep = np.array([0, 1, 0, 1], np.uint8) # A: {1, 3}  
Brep = np.array([0, 0, 1, 1], np.uint8) # B: {2, 3}  
n = len(Arep)  
Erep = np.empty(n, np.uint8)           # E: {?}   
unio(Arep, Brep, Erep, n)              # E: {1, 2, 3}
```

	0	1	2	3
Arep	0	1	0	1
Brep	0	0	1	1
Erep	0	1	1	1

Műveletek: metszet

Feladat: Írj függvényt, mely egy A és egy B halmaz reprezentációját (n elemű $Arep$ és $Brep$ tömb) kapja meg paraméterként. A függvény állítsa elő az $A \cap B$ halmaz reprezentációját az n elemű $Erep$ tömbbe.

```
def metszet(Arep, Brep, Erep, n):  
    for i in range(n):  
        if Arep[i] and Brep[i]:  
            Erep[i] = 1  
        else:  
            Erep[i] = 0
```

```
Arep = np.array([0, 1, 0, 1], np.uint8) # A: {1, 3}  
Brep = np.array([0, 0, 1, 1], np.uint8) # B: {2, 3}  
n = len(Arep)  
Erep = np.empty(n, np.uint8)           # E: {?}   
metszet(Arep, Brep, Erep, n)           # E: {3}
```

	0	1	2	3
Arep	0	1	0	1
Brep	0	0	1	1
Erep	0	0	0	1

Műveletek: különbség

Feladat: Írj függvényt, mely egy A és egy B halmaz reprezentációját (n elemű $Arep$ és $Brep$ tömb) kapja meg paraméterként. A függvény állítsa elő az $A \setminus B$ halmaz reprezentációját az n elemű $Erep$ tömbbe.

```
def AminusB(Arep, Brep, Erep, n):  
    for i in range(n):  
        if Arep[i] and not Brep[i]:  
            Erep[i] = 1  
        else:  
            Erep[i] = 0
```

```
Arep = np.array([0, 1, 0, 1], np.uint8) # A: {1, 3}  
Brep = np.array([0, 0, 1, 1], np.uint8) # B: {2, 3}  
n = len(Arep)  
Erep = np.empty(n, np.uint8)           # E: {?}  
AminusB(Arep, Brep, Erep, n)           # E: {1}
```

	0	1	2	3
Arep	0	1	0	1
Brep	0	0	1	1
Erep	0	1	0	0

Műveletek: komplementer

Feladat: Írj függvényt, mely egy A halmaz reprezentációját (n elemű $Arep$) kapja meg paraméterként. A függvény állítsa elő az A komplementer halmazát az n elemű $komp$ tömbbe.

(Megj.: A használt reprezentáció miatt az elemszám (n) meghatározza az alaphalmazt: $0, 1, \dots, n-1$.)

```
def komplementer(Arep, Erep, n):  
    for i in range(n):  
        Erep[i] = 1-Arep[i]
```

```
Arep = np.array([0, 1, 0, 1], np.uint8) # A: {1, 3}  
n = len(Arep)  
Erep = np.empty(n, np.uint8)           # E: {?}   
komplementer(Arep, Erep, n)           # E: {0, 2}
```

	0	1	2	3
Arep	0	1	0	1
Erep	1	0	1	0

Számosság

Feladat: Írj függvényt, mely egy A halmaz reprezentációját (n elemű $Arep$) kapja meg paraméterként. A függvény adja vissza a halmaz számosságát.

```
def szamossag(Arep, n):  
    # az egyesek darabszáma  
    db = 0  
    for i in range(n):  
        db += Arep[i]  
    return db
```

```
Arep = np.array([0, 1, 0, 1], np.uint8) # A: {1, 3}  
n = len(Arep)  
print(szamossag(Arep, n))    # 2
```

	0	1	2	3
Arep	0	1	0	1

Teszteléshez

Feladat: Írj függvényt, mely egy halmazreprezentációt kap meg paraméterként. Jelenítsd meg a halmaz elemeit!

```
def kiir(Arep, n):  
    for i in range(n):  
        if Arep[i]:  
            print(i)
```

	0	1	2	3
Arep	0	0	1	1

```
Arep = np.array([0, 0, 1, 1], np.uint8) # A: {2, 3}  
n = len(Arep)  
kiir (Arep, n)    # 2, 3
```

Feldolgozás

Feladat: Írj függvényt, mely megkap egy halmaz adatszerkezetet. Jelenítsd meg a halmaz elemei az alábbi függvények segítségével:

```
halmaz.base_size(): Visszaadja az alaphalmaz elemeinek a számát  
halmaz.eleme_e(x): True értéket ad vissza, ha x a halmaz  
                    eleme, különben False értéket
```

```
def kiir(halmaz):  
    n = halmaz.base_size()  
    # alaphalmaz elemek: 0, 1, ..., n-1  
    for i in range(n):  
        if halmaz.eleme_e(i):  
            print(i)
```

```
halmaz = Halmaz(4)  
kiir( halmaz )
```

```
class Halmaz:  
    def __init__(self, n):  
        self._reprez = np.zeros(n, np.uint8)  
  
    def eleme_e(self, x):  
        return 0<=x<n and self._reprez[x]  
  
    # az alaphalmaz elemszáma  
    def base_size(self):  
        return len(self._reprez)
```

Megj.: Ez a rész olyan programozási eszközöket tartalmaz, melynek ismerete még nem elvárt. Elég annyit megértened, hogy bizonyos esetekben nem lesz hozzáférése a reprezentációhoz, ezért muszáj lesz a műveleteket használnod.

A multihalmaz

- ▶ A matematikai multihalmaz fogalom
- ▶ Miben tér el a halmaztól? Egy elem többször is szerepelhet.
- ▶ Tulajdonságok:
 - Homogén
 - Statikus és dinamikus is lehet
 - Az adatelemek között nincs kapcsolat, **nincs sorrend**.
 - Véges

A multihalmaz

▶ Reprezentáció: folytonos/szétszórt

▶ Folytonos reprezentáció:

- Alaphalmaz megadása, rögzített elemsorrenddel:

$0, 1, \dots, n-1$ (ezentől röviden tartományként írjuk fel $[0, n-1]$)

- Karakterisztikus függvény:

$$f: [0, n - 1] \rightarrow \mathbb{Z}$$

- Az alaphalmaz elemeihez azt az egész számot rendeli hozzá, ami megadja, hogy **hányszor szerepel** az alaphalmazelem a multihalmazban.

Multihalmaz: reprezentáció

Feladat: Add meg a $\{1, 1, 2, 1, 2, 3\}$ multihalmaz reprezentációját a $[0, 3]$ egész számokból álló zárt tartomány felett.

	0	1	2	3
reprez	0	3	2	1

Feladat: Add meg a $\{2, 1, 4, 4, 0, 0, 4, 1\}$ multihalmaz reprezentációját a $\{0, 1, \dots, 5\}$ alaphalmaz felett.

	0	1	2	3	4	5
reprez	2	2	1	0	3	0

Multihalmaz: reprezentáció

Feladat: Add meg a $[0, 3, 0, 1]$ reprezentációkhoz tartozó multihalmazt.

$A = \{ 1, 1, 1, 3 \}$

	0	1	2	3
reprez	0	3	0	1

Feladat: Add meg a $[4, 0, 1, 1, 0, 2]$ reprezentációkhoz tartozó multihalmazt.

$A = \{ 0, 0, 0, 0, 2, 3, 5, 5 \}$

	0	1	2	3	4	5
reprez	4	0	1	1	0	2

Műveletek

▶ Létrehozás:

- Üres multihalmaz létrehozása
`Arep = np.zeros(alaphalmaz_elemszama, int)`
- Adott multihalmaz létrehozása
`Brep = np.array(reprezentacio, int)`

pl: $B = \{2, 2, 2, 3, 4, 4\}$ multihalmaz reprezentációja a $[0, 4]$ egészszámból álló tartomány felett:

```
Brep = np.array([0, 0, 3, 1, 2], int)
```

▶ Elérés: nincs, helyette eleme-e/hányszor szerepel művelet van

▶ Bejárás: -

▶ Módosítás:

- Bővítés: unió művelettel
- Törlés: különbség művelettel
- Csere: nincs, de szimulálható: pl. x cseréje y -ra: $A = A \setminus \{x\} \cup \{y\}$

▶ Keresés: -

▶ Rendezés: -

▶ Feldolgozó műveletek: a multihalmazműveleteken alapulnak

Műveletek: eleme-e

Feladat: Írj függvényt, mely egy A **multihalmaz** reprezentációját (n elemű $Arep$ tömb) és egy e egész számot kap meg paraméterként. A függvény adjon vissza True értéket, ha az e eleme az A multihalmaznak, különben adjon vissza False értéket.

```
def eleme_e(Arep, n, e ):
    return 0 <= e < n and Arep[e]
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}
n = len(Arep)
print(eleme_e(Arep, n, 2)) # True
print(eleme_e(Arep, n, 1)) # False
print(eleme_e(Arep, n, 10)) # False
print(eleme_e(Arep, n, -2)) # False
```

	0	1	2	3
Arep	2	0	3	1

Műveletek: hányszor eleme?

Feladat: Írj függvényt, mely egy A **multihalmaz** reprezentációját (n elemű A_{rep} tömb) és egy e egész számot kap meg paraméterként. A függvény adja vissza, hogy az e elem hányszor szerepel a multihalmazban.

```
def hanyszor_szerel (Arep, n, e ):
    if 0 <= e < n:
        return Arep[e]
    return 0
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}
n = len(Arep)
print(hanyszor_szerel (Arep, n, 2)) # 3
print(hanyszor_szerel (Arep, n, 1)) # 0
print(hanyszor_szerel (Arep, n, 10)) # 0
print(hanyszor_szerel (Arep, n, -2)) # 0
```

	0	1	2	3
Arep	2	0	3	1

Műveletek: unió

Feladat: Írj függvényt, mely egy A és egy B **multihalmaz** reprezentációját (n elemű $Arep$ és $Brep$ tömb) kapja meg paraméterként. A függvény állítsa elő az $A \cup B$ multihalmaz reprezentációját az n elemű $Erep$ tömbbe.

```
def unio(Arep, Brep, Erep, n):  
    for i in range(n):  
        Erep[i] = Arep[i] + Brep[i]
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}  
Brep = np.array([3, 0, 1, 1], int) # B: {0, 0, 0, 2, 3}  
n = len(Arep)  
Erep = np.empty(n, np.uint8)      # E: {?}   
unio(Arep, Brep, Erep, n)         # E: {0,0,0,0,0,2,2,2,2,3,3}
```

	0	1	2	3
Arep	2	0	3	1
Brep	3	0	1	1
Erep	5	0	4	2

Műveletek: metszet

Feladat: Írj függvényt, mely egy A és egy B **multihalmaz** reprezentációját (n elemű $Arep$ és $Brep$ tömb) kapja meg paraméterként. A függvény állítsa elő az $A \cap B$ multihalmaz reprezentációját az n elemű $Erep$ tömbbe.

```
def metszet(Arep, Brep, Erep, n):  
    for i in range(n):  
        if Arep[i] < Brep[i]:  
            Erep[i] = Arep[i]  
        else:  
            Erep[i] = Brep[i]
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}  
Brep = np.array([3, 0, 1, 1], int) # B: {0, 0, 0, 2, 3}  
n = len(Arep)  
Erep = np.empty(n, int)          # E: {?}  
metszet(Arep, Brep, Erep, n)     # E: {0,0,2,3}
```

	0	1	2	3
Arep	2	0	3	1
Brep	3	0	1	1
Erep	2	0	1	1

Műveletek: különbség

Feladat: Írj függvényt, mely egy A és egy B **multihalmaz** reprezentációját (n elemű $Arep$ és $Brep$ tömb) kapja meg paraméterként. A függvény állítsa elő az $A \setminus B$ multihalmaz reprezentációját az n elemű $Erep$ tömbbe.

```
def AminusB(Arep, Brep, Erep, n):  
    for i in range(n):  
        if Arep[i] > Brep[i]:  
            Erep[i] = Arep[i]-Brep[i]  
        else:  
            Erep[i] = 0
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}  
Brep = np.array([3, 0, 1, 1], int) # B: {0, 0, 0, 2, 3}  
n = len(Arep)  
Erep = np.empty(n, np.uint8)      # E: {?}  
AminusB(Arep, Brep, Erep, n)      # E: {2, 2}
```

	0	1	2	3
Arep	2	0	3	1
Brep	3	0	1	1
Erep	0	0	2	0

Számosság

Feladat: Írj függvényt, mely egy A multihalmaz reprezentációját (n elemű $Arep$) kapja meg paraméterként. A függvény adja vissza a multihalmaz számosságát.

```
def szamossag(Arep, n):  
    db = 0  
    for i in range(n):  
        db += Arep[i]  
    return db
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}  
n = len(Arep)  
print(szamossag(Arep, n))      # 6
```

	0	1	2	3
Arep	2	0	3	1

Teszteléshez

Feladat: Írj függvényt, mely egy A multihalmaz reprezentációját (n elemű $Arep$) kapja meg paraméterként és megjeleníti a multihalmaz elemeit. Minden elemet annyiszor jelenítsen meg, ahányszor az szerepel a multihalmazban.

```
def kiir(Arep, n):  
    for i in range(n):  
        for j in range(Arep[i]):  
            print(i)
```

```
Arep = np.array([2, 0, 3, 1], int) # A: {0, 0, 2, 2, 2, 3}  
n = len(Arep)  
kiir(Arep, n)      # 0, 0, 2, 2, 2, 3
```

	0	1	2	3
Arep	2	0	3	1