Adatszerkezetek

6. hét – A halmaz adatszerkezet: ábrázolások és műveletek

Dr. Koczka Ferenc · Eszterházy Károly Egyetem · 2026

Ismétlés

- **Tömb:** közvetlen elérés O(1), beszúrás/törlés O(n)
- **Bináris keresés** rendezett tömbön: O(log n)
- **Rendezések:** buborék, kiválasztásos, beszúrásos – $O(n^2)$
- **Összefésülés** (3. hét, 5. feladat): két rendezett tömbből egy rendezett tömb, O(n + m)

A mai óra kérdései

- Hogyan ábrázolhatunk egy matematikai halmazt a számítógépen?
- Melyik ábrázolásban milyen gyorsak a halmazműveletek?

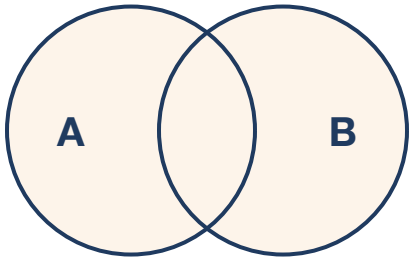
Matematikai háttér

Halmaz: különböző elemek **rendezetlen** gyűjteménye.

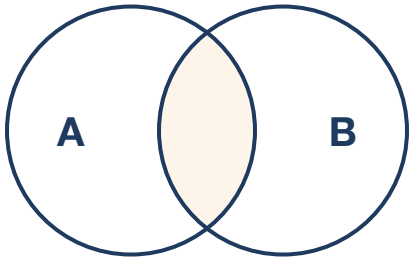
- Minden elem **legfeljebb egyszer** szerepel: {1, 2, 2} = {1, 2}
- A sorrend **nem számít**: {1, 2} = {2, 1}
- Jelölések: $x \in A$ (eleme) · $|A|$ (elemszám) · $\varnothing$ (üres halmaz) · **U** (alaphalmaz, univerzum)

Művelet	Jelölés	Jelentés
Eleme-e?	$x \in A$	x benne van-e A-ban
Részhalmaz	$A \subseteq B$	A minden eleme B-nek is eleme
Unió	$A \cup B$	A **vagy** B elemei
Metszet	$A \cap B$	A **és** B közös elemei
Különbség	$A \setminus B$	A azon elemei, amelyek **nincsenek** B-ben

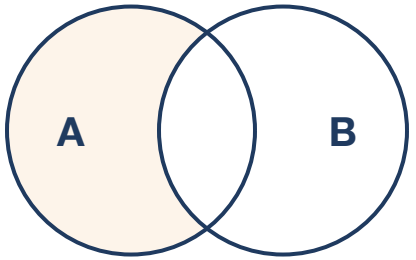
Halmazműveletek szemléltetése



A ∪ B (unió)



A ∩ B (metszet)



A \ B (különbség)

Példa – U = a hét napjai:

A = {H, Sze, Cs, Szo} (napok, amikor edzés van) · B = {K, Sze, Cs} (napok, amikor óra van)

A ∪ B = {H, K, Sze, Cs, Szo} · A ∩ B = {Sze, Cs} · A \ B = {H, Szo}

A halmaz mint adatszerkezet

Tulajdonság	Halmaz
Elemek	homogén
Kapcsolat az elemek között	**nincs** (nincs sorrend)
Méret	dinamikus (bővíthető, csökkenthető)
Ismétlődés	nincs

Ábrázolási lehetőségek (mind tömbön alapul):

1. **Logikai vektor** (karakterisztikus vektor, bitvektor)
2. **Rendezetlen** tömb (sorozat)
3. **Rendezett** tömb (sorozat)
4. Hash tábla (11. hét), keresőfa (15. hét)

Ábrázolás logikai vektorral

Logikai vektor

Ha az **alaphalmaz kicsi és ismert** ($U = \{u_1, \ldots, u_n\}$), minden halmaz egy **n** hosszú logikai vektor:

A[i] = igaz ⇔ $u_i \in A$

	H	K	Sze	Cs	P	Szo	V	
U	1	1	1	1	1	1	1	
A	1	0	1	1	0	1	0	A = {H, Sze, Cs, Szo}
B	0	1	1	1	0	0	0	B = {K, Sze, Cs}
A ∪ B	1	1	1	1	0	1	0	elemenként: vagy
A ∩ B	0	0	1	1	0	0	0	elemenként: és
A \ B	1	0	0	0	0	1	0	elemenként: és nem

Logikai vektor – műveletek

Pszeudokód

```
Függvény Eleme(A, i)          // O(1)
  Vissza: A[i]
Függvény vége

Eljárás Unió(A, B, C, n)       // O(n)
  Ciklus i ← 1-től n-ig
    C[i] ← A[i] vagy B[i]
  Ciklus vége
Eljárás vége

// Metszet: C[i] ← A[i] és B[i]
// Különbség: C[i] ← A[i] és nem B[i]

Függvény Része(A, B, n)        // A ⊆ B ?
  i ← 1
  Ciklus amíg i ≤ n és (nem A[i] vagy B[i])
    i ← i + 1
  Ciklus vége
  Vissza: i > n
Függvény vége
```

Java

```java
boolean[] in = new boolean[7];

BitVectorSet union(BitVectorSet o) {
    BitVectorSet r = new BitVectorSet();
    for (int i = 0; i < in.length; i++)
        r.in[i] = in[i] || o.in[i];
    return r;
}
```

Művelet	Lépésszám
eleme-e, bővítés, törlés	**O(1)**
unió, metszet, különbség	O(|U|)
részhalmaz	O(|U|)
elemszám	O(|U|)

Bitmaszk: egy bit = egy elem

Ha $|U| \le 64$, a teljes halmaz elfér **egyetlen egész számban**: az **i**-edik bit jelzi, hogy $u_i \in A$.

Művelet	Java	Go	Példa (A = 0101101, B = 0001110)
unió	`a | b`	`a | b`	0101111
metszet	`a & b`	`a & b`	0001100
különbség	`a & ~b`	`a &^ b`	0100001
eleme-e (i)	`(a >> i & 1) == 1`	`a>>i&1 == 1`	
részhalmaz	`(a & ~b) == 0`	`a&^b == 0`	
elemszám	`Integer.bitCount(a)`	`bits.OnesCount(a)`	4

Egy halmazművelet = **egyetlen processzorutasítás**. Alkalmazás: jogosultságok (olvasás/írás/futtatás), napok, beállítások.

Logikai vektor – előnyök és korlátok

Előnyök:

- Leggyorsabb: eleme-e **O(1)**, a halmazműveletek bitmaszkkal egy lépésben
- Egyszerű megvalósítás

Korlátok:

- Az alaphalmaznak **kicsinek és előre ismertnek** kell lennie
- A tárigény |U|-val arányos, nem az elemek számával
  pl. egész számok halmaza 0 … 10^9 között: 10^9 bit ≈ **125 MB** – akkor is, ha csak 3 elem van benne!
- Az elemeket egy **sorszámmal** kell azonosítani (H → 0, K → 1, …)

Ábrázolás rendezetlen tömbbel

Rendezetlen tömb

Az elemeket egy tömbben tároljuk, **tetszőleges sorrendben**, ismétlődés nélkül.

A = `[8, 2, 11, 5]` · B = `[9, 5, 14, 3, 11]`

- **Eleme-e?** lineáris keresés – O(n)
- **Bővítés:** előbb meg kell nézni, benne van-e már (O(n)), utána a végére – O(1)
- **Törlés:** megkeressük (O(n)), és az **utolsó elemet tesszük a helyére** – nem kell tolni, hiszen a sorrend nem számít!

> A tárigény az elemek számával arányos – nincs szükség alaphalmazra, akármilyen elemet tárolhatunk.

Rendezetlen tömb – műveletek

Részhalmaz: A ⊆ B

```
Függvény Része(A, n, B, m)
  i ← 1
  Ciklus amíg i ≤ n és Eleme(B, m, A[i])
    i ← i + 1
  Ciklus vége
  Vissza: i > n
Függvény vége
```

Metszet: A ∩ B

```
Eljárás Metszet(A, n, B, m, C, k)
  k ← 0
  Ciklus i ← 1-től n-ig
    Ha Eleme(B, m, A[i]) akkor
      k ← k + 1 ; C[k] ← A[i]
    Elágazás vége
  Ciklus vége
Eljárás vége
```

Unió: A minden eleme + B azon elemei, amelyek nincsenek A-ban

Különbség: A azon elemei, amelyek nincsenek B-ben

Művelet	Lépésszám
eleme-e	O(n)
bővítés	O(n)
törlés	O(n)
részhalmaz	**O(n · m)**
unió, metszet, különbség	**O(n · m)**

Minden elemre lineáris keresés a másik halmazban → négyzetes!

Ábrázolás rendezett tömbbel

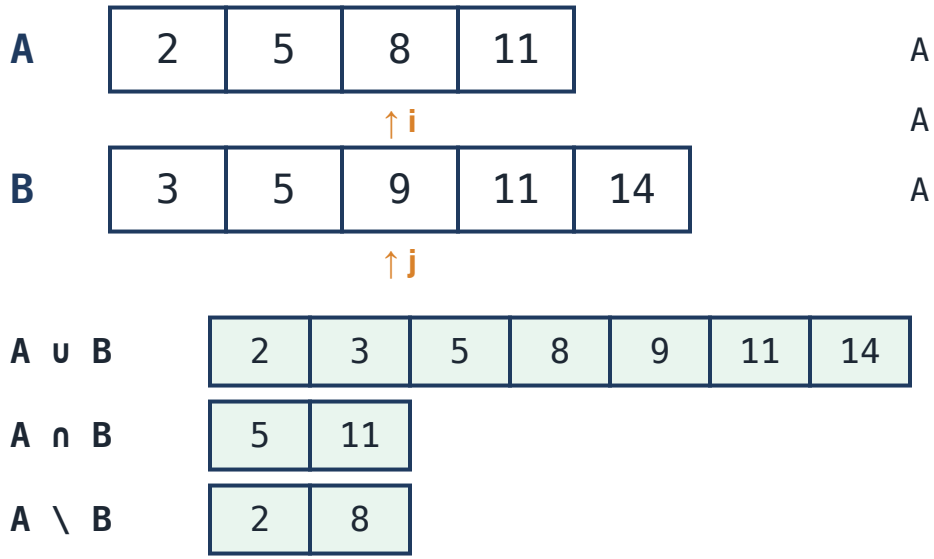
Rendezett tömb

Az elemeket **növekvő sorrendben** tároljuk, ismétlődés nélkül.

A = `[2, 5, 8, 11]` · B = `[3, 5, 9, 11, 14]`

- **Eleme-e? bináris keresés** – O(log n)
- **Bővítés / törlés:** a helyét bináris kereséssel megtaláljuk, de **tolni** kell – O(n)
- **Halmazműveletek:** a két rendezett tömb **összefésülése** – egyetlen közös bejárás

Összefésülés: a halmazműveletek alapja



`A[i] < B[j]` → `A[i]` kerül az eredménybe, `i ← i + 1`

`A[i] = B[j]` → egyszer kerül be, mindkettő lép

`A[i] > B[j]` → `B[j]` kerül be, `j ← j + 1`

mindhárom: egyetlen közös bejárás

`O(n + m)`

Unió összefésüléssel

Pszeudokód

```
Eljárás Unió(A, n, B, m, C, k)
  i ← 1 ; j ← 1 ; k ← 0
  Ciklus amíg i ≤ n és j ≤ m
    k ← k + 1
    Ha A[i] < B[j] akkor
      C[k] ← A[i] ; i ← i + 1
    különben ha A[i] > B[j] akkor
      C[k] ← B[j] ; j ← j + 1
    különben                  // egyenlők
      C[k] ← A[i] ; i ← i + 1 ; j ← j + 1
    Elágazás vége
  Ciklus vége
  Ciklus amíg i ≤ n            // A maradéka
    k ← k + 1 ; C[k] ← A[i] ; i ← i + 1
  Ciklus vége
  Ciklus amíg j ≤ m            // B maradéka
    k ← k + 1 ; C[k] ← B[j] ; j ← j + 1
  Ciklus vége
Eljárás vége
```

Java

```java
int i = 0, j = 0, k = 0;
while (i < a.length && j < b.length) {
    if (a[i] < b[j])      c[k++] = a[i++];
    else if (a[i] > b[j]) c[k++] = b[j++];
    else { c[k++] = a[i++]; j++; }
}
while (i < a.length) c[k++] = a[i++];
while (j < b.length) c[k++] = b[j++];
```

Metszet: csak az egyenlő ágban írunk C-be, a maradékot nem másoljuk.

Különbség: csak az `A[i] < B[j]` ágban írunk, és A maradékát másoljuk.

Részhalmaz összefésüléssel

```
Függvény Része(A, n, B, m)            // A ⊆ B ?
  i ← 1 ; j ← 1
  Ciklus amíg i ≤ n és j ≤ m és A[i] ≥ B[j]
    Ha A[i] = B[j] akkor i ← i + 1
    j ← j + 1
  Ciklus vége
  Vissza: i > n
Függvény vége
```

- Ha `A[i] < B[j]`, akkor A[i] **biztosan nincs** B-ben (B további elemei még nagyobbak) → nem részhalmaz
- Lépésszám: **O(n + m)**

Az ábrázolások összehasonlítása

Művelet	Logikai vektor	Rendezetlen tömb	Rendezett tömb
eleme-e	**O(1)**	O(n)	O(log n)
bővítés, törlés	**O(1)**	O(n)	O(n)
részhalmaz	O(|U|)	O(n · m)	**O(n + m)**
unió, metszet, különbség	O(|U|)	O(n · m)	**O(n + m)**
tárigény	|U| bit	n elem	n elem
feltétel	kicsi, ismert U	–	rendezhető elemek

Nincs „legjobb” ábrázolás: a választás az **alaphalmaz méretétől** és a **leggyakoribb műveletektől** függ.

Beépített halmazok

Java

```
Set<String> a = new HashSet<>();  // O(1) átl.
a.add("Wuhan");
a.contains("Wuhan");    // true
a.remove("Wuhan");

Set<String> u = new HashSet<>(a);
u.addAll(b);            // unió
u.retainAll(b);         // metszet
u.removeAll(b);         // különbség

TreeSet<Integer> t = new TreeSet<>(); // O(log n)
t.first(); t.last();    // rendezett!
```

Go – nincs set típus, map-pel

```
type Set map[string]struct{}

s := Set{}
s["Wuhan"] = struct{}{}      // bővítés
_, ok := s["Wuhan"]          // eleme-e
delete(s, "Wuhan")           // törlés

// gyakori rövidítés:
seen := map[int]bool{}
if !seen[x] {
    seen[x] = true
}
```

`HashSet` és Go `map`: **hash tábla** (11. hét) · `TreeSet`: **kiegyensúlyozott keresőfa** (15. hét)

Összefoglalás

- **Halmaz:** különböző elemek rendezetlen gyűjteménye; műveletek: $\in$, $\subseteq$, $\cup$, $\cap$, $\setminus$
- **Logikai vektor / bitmaszk:** kicsi, ismert alaphalmaz esetén a leggyorsabb – O(1), O(|U|)
- **Rendezetlen tömb:** egyszerű, de a halmazműveletek O(n · m)
- **Rendezett tömb:** bináris keresés O(log n), **összefésüléssel** a halmazműveletek O(n + m)
- Beépített: Java `HashSet` / `TreeSet`, Go `map[T]struct{}`

Következő hét: a verem (stack) – LIFO elv, tömbös megvalósítás, postfix kifejezések kiértékelése.

Ellenőrző kérdések

1. Az univerzum U = {1, 2, …, 8}. Írd fel logikai vektorként: A = {2, 3, 5, 7}, B = {1, 2, 3, 4}! Számold ki: A ∪ B, A ∩ B, A \ B!
2. Igaz vagy hamis: rendezetlen tömbös ábrázolásnál a metszet lépésszáma O(n + m).
3. Végezd el papíron az összefésülést: A = [1, 4, 6, 9], B = [2, 4, 9, 10]! Add meg A ∪ B, A ∩ B és A \ B értékét!
4. Miért nem célszerű logikai vektorral ábrázolni a hallgatói azonosítók (8 jegyű számok) halmazát?
5. Bitmaszkokkal: `a = 0b1010`, `b = 0b0110`. Mennyi `a & b`, `a | b`, `a & ~b`?

Gyakorlat

✎ Bemutató programok

Program	Java	Go	Mit mutat?
Logikai vektor	`BitVectorSet.java`	`h06/bitvectorset`	napok halmaza, bitmaszk-műveletek
Rendezetlen	`UnorderedSet.java`	`h06/unorderedset`	O(n · m) műveletek, törlés az utolsóval
Rendezett	`SortedSet.java`	`h06/sortedset`	bináris keresés, összefésülés
Beépített	`BuiltinSets.java`	`h06/builtin`	`HashSet` , `TreeSet` , Go `map`

✎ Feladatok

1. **Ismétlődések törlése:** készíts egy tömbből halmazt rendezéssel! (`[4,1,4,2,1] → [1,2,4]`)
2. **Szimmetrikus differencia:** A △ B = (A \ B) ∪ (B \ A) – rendezett tömbökön, egyetlen összefésüléssel!
3. **Diszjunktak-e?** Döntsd el O(n + m) lépésben, hogy két rendezett halmaznak nincs közös eleme!
4. **Hatványhalmaz:** írd ki egy **n** elemű halmaz összes részhalmazát bitmaszkokkal! Hány van?
5. **Különböző betűk:** hány különböző angol betű szerepel egy szövegben? (logikai vektor)
6. **Pangramma:** szerepel-e egy mondatban az angol ábécé összes betűje?
7. **Közös barátok:** két barátlista metszete beépített halmaz (map) segítségével.

Megoldások: `Exercises06.java`, `h06/exercises`

✎ Házi feladat

1. Valósíts meg egy `SortedSet` osztályt `add(x)` és `remove(x)` műveletekkel, amely mindig rendezett marad! (Tipp: bináris kereséssel keresd a helyet.)
2. **Multihalmaz** (zsák): egy elem többször is szerepelhet. Hogyan ábrázolnád logikai vektor helyett? Valósítsd meg az uniót és a metszetet (a darabszámok maximuma, illetve minimuma)!
3. Három rendezett halmaz metszetét számold ki **egyetlen** közös bejárással!
4. Egy 1 és 100 000 közötti számokból álló, 1 000 000 elemű tömbben hány **különböző** szám van? Oldd meg logikai vektorral, és hasonlítsd össze a rendezéses megoldással!