Adatszerkezetek

5. hét – Rendezések I.: buborék-, kiválasztásos és beszúrásos rendezés

Dr. Koczka Ferenc · Eszterházy Károly Egyetem · 2026

Ismétlés

- **Bináris keresés:** O(log n) – de csak **rendezett** tömbön!
- Tömbben az elemek mozgatása költséges: beszúrás, törlés **O(n)**
- Stringek összehasonlítása: **lexikografikus** sorrend

A mai óra kérdései

- Hogyan állíthatunk sorba egy tömböt?
- Mennyi összehasonlítás és mozgatás kell hozzá?
- Mikor melyik egyszerű rendezést válasszuk?

A rendezés feladata

Adott: egy **n** elemű sorozat, amelynek elemei összehasonlíthatók (≤ reláció).

Feladat: az elemek átrendezése úgy, hogy A[1] ≤ A[2] ≤ … ≤ A[n].

Miért fontos?

- **Gyors keresés:** rendezett tömbön bináris keresés – O(log n)
- **Csoportosítás:** az azonos elemek egymás mellé kerülnek (ismétlődések, gyakoriság)
- **Megjelenítés:** névsor, rangsor, időrend
- Becslések szerint a számítógépek futásidejének jelentős része rendezéssel telik

Fogalmak

- **Összehasonlítás:** két elem viszonyának vizsgálata (`A[i] < A[j]`) – a rendezések fő költsége
- **Mozgatás / csere:** elemek áthelyezése – egy csere 3 értékadás
- **Helyben rendezés (in-place):** csak O(1) segédtár kell
- **Stabil rendezés:** az egyenlő kulcsú elemek **megtartják** eredeti sorrendjüket
- **Összehasonlító rendezés:** csak összehasonlításokkal dönt az elemek sorrendjéről

> Ma három **egyszerű**, $O(n^2)$ lépésszámú rendezést tanulunk. A 13. héten jönnek a gyorsabb, O(n log n)-es módszerek.

Buborékrendezés

Buborékrendezés – az ötlet

- Haladjunk végig a tömbön, és hasonlítsuk össze a **szomszédos** elemeket
- Ha rossz sorrendben vannak, **cseréljük** meg őket
- Egy menet után a **legnagyobb** elem a tömb végére kerül („felbuborékol”)
- Ismételjük a rendezetlen részen

Buborékrendezés: menetenként a szomszédos párok cseréje



Buborékrendezés – algoritmus

Pszeudokód

```
Eljárás BuborékRendezés(A, n)
  Ciklus i ← n-től 2-ig visszafelé
    volt_csere ← hamis
    Ciklus j ← 1-től i - 1-ig
      Ha A[j] > A[j + 1] akkor
        csere(A[j], A[j + 1])
        volt_csere ← igaz
      Elágazás vége
    Ciklus vége
    Ha nem volt_csere akkor
      kilépés a ciklusból
  Ciklus vége
Eljárás vége
```

Java

```java
for (int i = n - 1; i >= 1; i--) {
    boolean swapped = false;
    for (int j = 0; j < i; j++)
        if (a[j] > a[j + 1]) {
            int t = a[j];
            a[j] = a[j + 1];
            a[j + 1] = t;
            swapped = true;
        }
    if (!swapped) break;
}
```

Go

```go
if a[j] > a[j+1] {
    a[j], a[j+1] = a[j+1], a[j]
    swapped = true
}
```

Buborékrendezés – elemzés

Eset	Összehasonlítás	Csere
Legjobb (már rendezett)	n – 1	0
Legrosszabb (fordított)	n(n – 1) / 2	n(n – 1) / 2
Átlagos	~ n^2 / 2	~ n^2 / 4

- Lépésszám: **O(n^2)** · rendezett tömbön a `volt_csere` jelzővel **O(n)**
- **Helyben** rendez, **stabil** (egyenlő elemeket nem cserél)
- A cserék száma = az **inverziók** száma (a rossz sorrendben lévő párok)

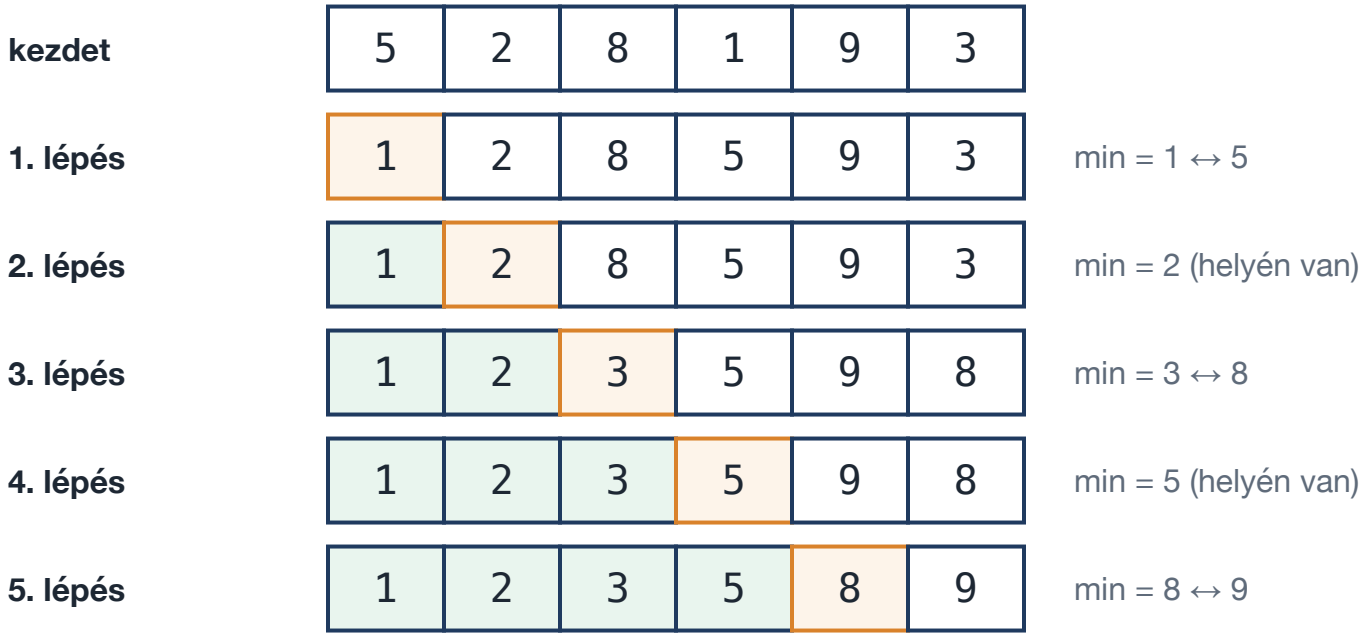
> A gyakorlatban a leglassabb az egyszerű rendezések közül – sok cserét végez.

Kiválasztásos rendezés

Kiválasztásos rendezés – az ötlet

- Keressük meg a **legkisebb** elemet, és cseréljük az **első** helyre
- A maradék (rendezetlen) részben keressük meg a legkisebbet, és tegyük a **második** helyre
- … és így tovább

Kiválasztásos rendezés: a legkisebb a rendezetlen rész elejére



Kiválasztásos rendezés – algoritmus

Pszeudokód

```
Eljárás KiválasztásosRendezés(A, n)
  Ciklus i ← 1-től n - 1-ig
    min ← i
    Ciklus j ← i + 1-től n-ig
      Ha A[j] < A[min] akkor
        min ← j
      Elágazás vége
    Ciklus vége
    csere(A[i], A[min])
  Ciklus vége
Eljárás vége
```

A belső ciklus egy **minimumkiválasztás** (2. hét: maximumkiválasztás).

Java

```java
for (int i = 0; i < n - 1; i++) {
    int min = i;
    for (int j = i + 1; j < n; j++)
        if (a[j] < a[min]) min = j;
    int t = a[i];
    a[i] = a[min];
    a[min] = t;
}
```

Go

```go
for i := 0; i < len(a)-1; i++ {
    min := i
    for j := i + 1; j < len(a); j++ {
        if a[j] < a[min] {
            min = j
        }
    }
    a[i], a[min] = a[min], a[i]
}
```

Kiválasztásos rendezés – elemzés

Eset	Összehasonlítás	Csere
Minden esetben	n(n – 1) / 2	n – 1

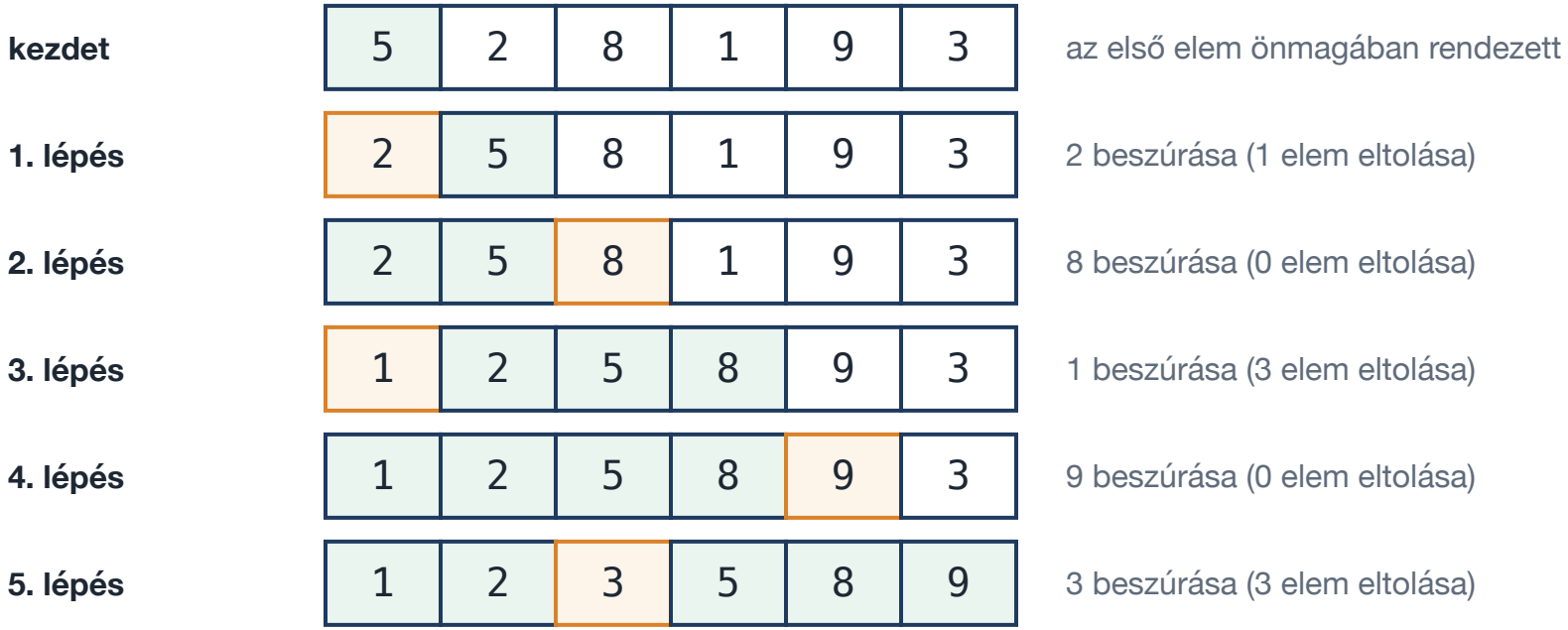
- Lépésszám: **mindig $O(n^2)$** – a rendezett tömbön is végignézi a maradékot
- Csak **n – 1 csere** – akkor előnyös, ha a mozgatás nagyon drága (nagy rekordok)
- **Helyben** rendez
- **Nem stabil:** a távoli csere átugorhat egy egyenlő elemet
  `[5a, 5b, 2]` → `[2, 5b, 5a]`

Beszúrásos rendezés

Beszúrásos rendezés – az ötlet

- Úgy rendezünk, ahogy kártyalapokat a kézben: a következő lapot **beszúrjuk** a helyére a már rendezett lapok közé
- A tömb eleje rendezett, a következő elemet addig toljuk balra, amíg a helyére nem ér

Beszúrásos rendezés: a következő elem beszúrása a rendezett részbe



Beszúrásos rendezés – algoritmus

Pszeudokód

```
Eljárás BeszúrásosRendezés(A, n)
  Ciklus i ← 2-től n-ig
    x ← A[i]
    j ← i - 1
    Ciklus amíg j ≥ 1 és A[j] > x
      A[j + 1] ← A[j]
      j ← j - 1
    Ciklus vége
    A[j + 1] ← x
  Ciklus vége
Eljárás vége
```

Nem cserél, hanem **tol**: egy lépés egy értékadás.

Java

```java
for (int i = 1; i < n; i++) {
    int x = a[i];
    int j = i - 1;
    while (j >= 0 && a[j] > x) {
        a[j + 1] = a[j];
        j--;
    }
    a[j + 1] = x;
}
```

Go

```go
for i := 1; i < len(a); i++ {
    x, j := a[i], i-1
    for j >= 0 && a[j] > x {
        a[j+1] = a[j]
        j--
    }
    a[j+1] = x
}
```

Beszúrásos rendezés – elemzés

Eset	Összehasonlítás	Mozgatás
Legjobb (már rendezett)	$n - 1$	$2(n - 1)$
Legrosszabb (fordított)	$n(n - 1) / 2$	$\sim n^2 / 2$
Átlagos	$\sim n^2 / 4$	$\sim n^2 / 4$

- Lépésszám: **$O(n^2)$**, rendezett vagy **majdnem rendezett** tömbön **$O(n)$**
- **Helyben** rendez, **stabil** (`A[j] > x` , nem `≥` !)
- **Online:** az elemeket érkezésük sorrendjében is rendezheti
- Kis tömbökön ($n < 20$) a leggyorsabb – a profi rendezések (Java, Go könyvtár) is ezt használják a kis részekre

Összehasonlítás

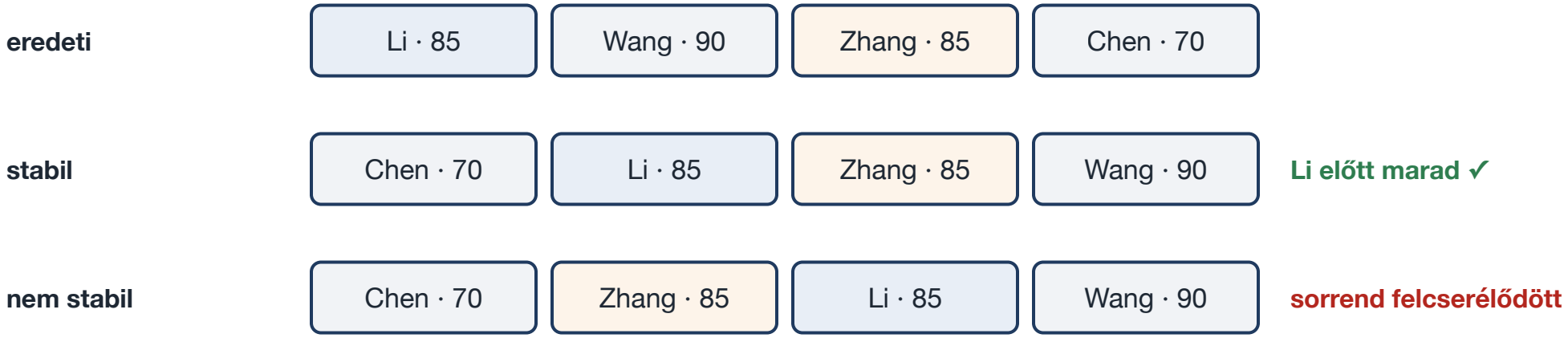
Mérési eredmények (n = 1000)

Összehasonlítások / mozgatások száma (`SortCompare` program):

	véletlen	rendezett	fordított
buborék	499 004 / 750 423	**999 / 0**	499 500 / 1 498 500
kiválasztásos	499 500 / **2 997**	499 500 / 2 997	499 500 / 2 997
beszúrásos	**251 585** / 252 589	**999 / 1 998**	499 500 / 501 498

> Mozgatás = értékadás (egy csere 3 értékadás). Figyeld meg: a kiválasztásos rendezés **mindig ugyanannyi** lépést tesz, a beszúrásos rendezett adaton **lineáris**.

Stabilitás



- **Miért fontos?** Ha előbb név, aztán pontszám szerint rendezünk **stabil** rendezéssel, az azonos pontszámúak névsorban maradnak
- Stabil: buborék, beszúrásos · Nem stabil: kiválasztásos

Melyiket válasszuk?

	Buborék	Kiválasztásos	Beszúrásos
Legrosszabb eset	$O(n^2)$	$O(n^2)$	$O(n^2)$
Legjobb eset	$O(n)$	$O(n^2)$	$O(n)$
Cserék / mozgatások	sok	**kevés (n – 1)**	közepes
Stabil	igen	nem	igen
Helyben	igen	igen	igen

- Általában: **beszúrásos** – egyszerű, majdnem rendezett adaton gyors
- Drága mozgatás esetén: **kiválasztásos**
- Buborék: főleg oktatási célra

> Nagy **n**-re egyik sem jó: $n = 10^6$ esetén ~10^{12} lépés. Megoldás: **összefésülő** és **gyorsrendezés** (13. hét), $O(n \log n)$.

Beépített rendezések

Java

```java
int[] a = {5, 2, 8, 1};
Arrays.sort(a);   // primitív: Dual-Pivot Quicksort

String[] s = {"Wuhan", "Eger"};
Arrays.sort(s);   // objektum: TimSort (stabil)

list.sort(Comparator
    .comparing(Student::score));
```

Go

```go
a := []int{5, 2, 8, 1}
slices.Sort(a)     // pdqsort

sort.Strings(s)

slices.SortStableFunc(st,
    func(x, y Student) int {
        return x.Score - y.Score
    })
```

A gyakorlatban a könyvtári rendezést használjuk – de **tudnunk kell, hogyan működik**, és mikor milyen gyors.

Összefoglalás

- **Buborék:** szomszédos cserék, menetenként a legnagyobb a végére · $O(n^2)$, stabil
- **Kiválasztásos:** minimumot az elejére · mindig $O(n^2)$, csak $n - 1$ csere, nem stabil
- **Beszúrásos:** következő elem beszúrása a rendezett részbe · $O(n^2)$, rendezett adaton $O(n)$, stabil
- Mindhárom **helyben** rendez
- **Stabilitás:** az egyenlő kulcsúak eredeti sorrendje megmarad

Következő hét: a halmaz adatszerkezet – logikai vektor, rendezetlen és rendezett ábrázolás, halmazműveletek.

Ellenőrző kérdések

1. Rendezd papíron a `[4, 1, 3, 2]` tömböt mindhárom módszerrel! Írd fel az állapotokat lépésenként!
2. Igaz vagy hamis: a kiválasztásos rendezés rendezett tömbön O(n) lépésben végez.
3. Melyik rendezés végzi a legkevesebb cserét? Hányat?
4. Igaz vagy hamis: a buborékrendezés stabil.
5. Hány összehasonlítást végez a beszúrásos rendezés egy fordított sorrendű, 10 elemű tömbön?
6. Mi történik, ha a beszúrásos rendezésben `A[j] > x` helyett `A[j] ≥ x` szerepel?

Gyakorlat

✎ Bemutató programok

Program	Java	Go	Mit mutat?
Egyszerű rendezések	`SimpleSorts.java`	`h05/simplesorts`	mindhárom rendezés lépésenkénti kiírással
Összehasonlítás	`SortCompare.java`	`h05/sortcompare`	összehasonlítások, mozgatások, idő különböző bemeneteken

Futtasd a `SortCompare` programot! Melyik rendezés függ a bemenettől, és melyik nem? Ha **n** duplázódik, hányszorosára nő a futási idő?

✎ Feladatok

1. **Rendezett-e?** Döntsd el O(n) lépésben, hogy egy tömb növekvően rendezett-e!
2. **Csökkenő sorrend:** írd át a beszúrásos rendezést csökkenő sorrendre!
3. **Stringek:** rendezz egy városnevekből álló tömböt kiválasztásos rendezéssel!
4. **Rekordok:** rendezd a hallgatókat (név, pontszám) pontszám szerint **stabilan**!
5. **k legkisebb:** add meg egy tömb **k** legkisebb elemét a kiválasztásos rendezés első **k** lépésével! Mennyi a lépésszám?
6. **Számláló rendezés:** rendezd a vizsgajegyeket (1–5) összehasonlítás nélkül, egy 6 elemű számlálótömbbel! Mi a lépésszám?
7. **Inverziók:** számold meg, hány rossz sorrendű pár van egy tömbben! Mi köze ennek a buborékrendezéshez?

Megoldások: `Exercises05.java`, `h05/exercises`

✎ Házi feladat

1. **Koktélrendezés:** a buborékrendezés változata, amely felváltva balról jobbra és jobbról balra halad. Valósítsd meg, és mérd meg, hány összehasonlítást spórol a `[2, 3, 4, 5, 1]` tömbön!
2. **Bináris beszúrásos rendezés:** a beszúrás helyét bináris kereséssel keresd meg (2. hét)! Hogyan változik az összehasonlítások és a mozgatások száma?
3. Mutass példát, amely igazolja, hogy a kiválasztásos rendezés **nem stabil**!
4. Rendezd a hallgatókat **elsődlegesen pontszám** szerint csökkenően, **azonos pontszám esetén név** szerint! (Tipp: két stabil rendezés egymás után, vagy összetett összehasonlítás.)