



# Algoritmuskélet 1. témakör

Pusztai Pál  
[pusztai@sze.hu](mailto:pusztai@sze.hu)

# Tartalom

- Az algoritmusok megadásánál alkalmazott jelölésmód
- Függvények növekedése, aszimptotikus jelölések
- Rendezések
  - Rendezés beszúrással
  - Rendezés összefésüléssel
  - Gyorsrendezés
  - Kupacrendezés
    - A kupac adatstruktúra és a kupacépítés
- Elsőbbségi sorok
  - Megvalósítás kupac adatszerkezettel
- Rendezések lineáris időben
  - A leszámláló, a számjegyes és az edényrendezés
- Mediánok és rendezett minták
  - A kiválasztási probléma



## Jelölésmód

- Az algoritmusokat **szubrutinok** (eljárások, függvények) formájában, **pszeudokóddal** adjuk meg, amelyben:
  - A **tagolás** a blokkszerkezetet tükrözi.
  - Egy sorba legfeljebb egy utasítást írunk.
  - Megengedjük a **többszörös értékadást**.
    - Az  $i \leftarrow j \leftarrow k$  értékadás ekvivalens a  $j \leftarrow k$ , és az ezt követő  $i \leftarrow j$  értékadással.
  - Egy **tömb** elemeire [ ] zárójellel hivatkozunk.
    - Pl:  $A[i]$  jelenti az  $A$  egydimenziós tömb  $i$ -edik elemét.
    - Pl:  $B[2, j+1]$  jelenti a  $B$  kétdimenziós tömb második sorának  $j+1$ -edik elemét.
    - Pl:  $A[1..j]$  jelenti az  $A[1], A[2], \dots, A[j]$  elemekből álló **résztömböt**.
  - Egy egydimenziós tömb megadása
    - Pl:  $A = \langle 5, 3, 8, 1, 9, 7, 2, 6, 4 \rangle$
  - Az összetett adatok (objektumok) esetén a **tulajdonságok** (mezők) elérése
    - Pl:  $hossz[A]$  jelenti az objektumként kezelt  $A$  tömb elemeinek számát.

## Jelölésmód

- További pszeudokód jelölések, megállapodások
  - Alsó és felső egészrész
    - Minden  $x$  valós szám esetén:  $x-1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x+1$ .
    - Minden  $n$  egész szám esetén:  $\lfloor n/2 \rfloor + \lfloor n/2 \rfloor = n$ .
  - Programmá írás
    - A **paraméterátadás** az egyszerű adatoknál értékszerinti, az összetett adatoknál címszerinti.
    - A **változók** a szubrutinok lokális munkaváltozói.
  - A **mutatók** konstansa: NIL (az ábrákon / jelöli).
  - A logikai műveletek **gyors kiértékelésűek**.
  - A **megjegyzéseket** /\* és \*/ jelek közé tesszük.



# Vezérlőszervezetek

## ■ Szekvencia

$T_1$   
 $T_2$   
 $\dots$   
 $T_n$

## ■ Elöltesztelő iteráció

**while**  $f$   
 $T$

## ■ Szelekció

**if**  $f_1$   $T_1$   
**else if**  $f_2$   $T_2$   
 $\dots$   
**else if**  $f_n$   $T_n$

## ■ Hátultesztelő iteráció

**repeat**  $T$   
**until**  $f$

## ■ Növekményes iteráció

**for**  $cv \leftarrow ke, ve, le$   
 $T$

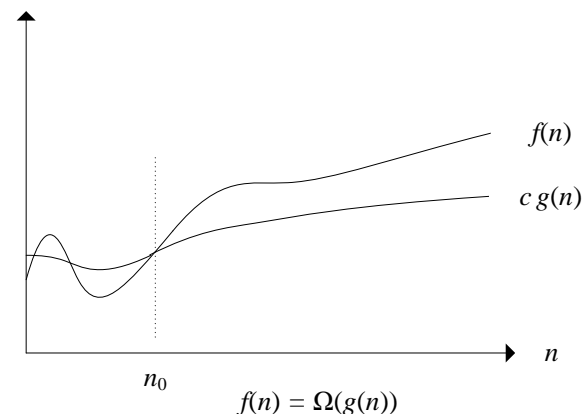
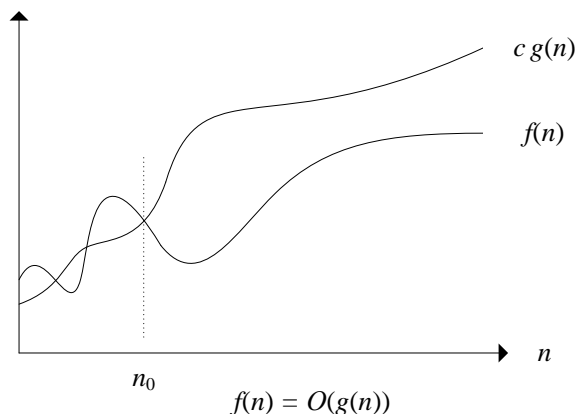
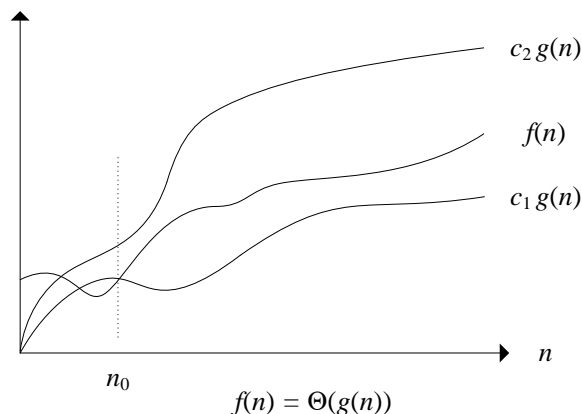


## Feladatok

- Írjuk ki a páros számokat 1-től 10-ig mindhárom ciklusfajtaival! A kiírást az alábbi utasítással végezzük!
  - Ki: kifejezéslista
  - Pl: Ki: "Az eredmény:", er



# Függvények növekedése



## Aszimptotikus jelölések

$\Theta(g(n)) = \{f(n): \text{léteznek } c_1, c_2 \text{ pozitív állandók és } n_0 \text{ úgy, hogy } n \geq n_0 \text{ esetén } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}.$

$O(g(n)) = \{f(n): \text{létezik } c \text{ pozitív állandó és } n_0 \text{ úgy, hogy } n \geq n_0 \text{ esetén } 0 \leq f(n) \leq c g(n)\}.$

$\Omega(g(n)) = \{f(n): \text{létezik } c \text{ pozitív állandó és } n_0 \text{ úgy, hogy } n \geq n_0 \text{ esetén } 0 \leq c g(n) \leq f(n)\}.$

$o(g(n)) = \{f(n): \text{minden } c \text{ pozitív állandóhoz létezik } n_0 > 0 \text{ úgy, hogy } n \geq n_0 \text{ esetén } 0 \leq f(n) < c g(n)\}.$

$\omega(g(n)) = \{f(n): \text{minden } c \text{ pozitív állandóhoz létezik } n_0 > 0 \text{ úgy, hogy } n \geq n_0 \text{ esetén } 0 \leq c g(n) < f(n)\}.$

$\Theta$ : aszimptotikusan **éles** korlát,  $o$ ,  $\omega$ : aszimptotikusan **nem** **éles** korlátok.

$O$ ,  $\Omega$ : **éles** vagy **nem** **éles** korlátok.

$\Theta(1)$ ,  $O(1)$  a konstans függvényeket jelölik.

# Függvények növekedése

## ■ Megjegyzés:

- Egy  $k$ -adfokú polinom  $\Theta(n^k)$  nagyságrendű.
- Minden  $c > 1$  és  $k > 1$  egész esetén  $n^k = o(c^n)$ , ill.  $c^n = \omega(n^k)$ .
- $\lg(n!) = \Theta(n \lg n)$





# Feladatok

## ■ Igazak-e vagy sem az alábbiak?

- $0.1n^2 - 3n + 1 = \Theta(n^2)$
- $2n = O(n^2)$
- $2n = \Omega(n^2)$
- $2n = o(n^2)$
- $2n = o(n)$
- $2n^2 = \omega(n)$
- $2^{n+1} = O(2^n)$
- $2^{2n} = O(2^n)$
- $n! = \Omega(n^n)$ .
- $n! = \Omega(2^n)$ .
- $f(n) = \Theta(g(n))$  akkor és csak akkor, ha  $f(n) = O(g(n))$  és  $f(n) = \Omega(g(n))$ .
- $f(n) = \Theta(g(n))$  akkor és csak akkor, ha  $g(n) = \Theta(f(n))$ .
- $f(n) = O(g(n))$  akkor és csak akkor, ha  $g(n) = \Omega(f(n))$ .
- $f(n) = o(g(n))$  akkor és csak akkor, ha  $g(n) = \Omega(f(n))$ .
- $o(f(n)) \cap \omega(f(n)) = \emptyset$



## Rendezés beszúrással

BESZÚRÓ-RENDEZÉS( $A$ )

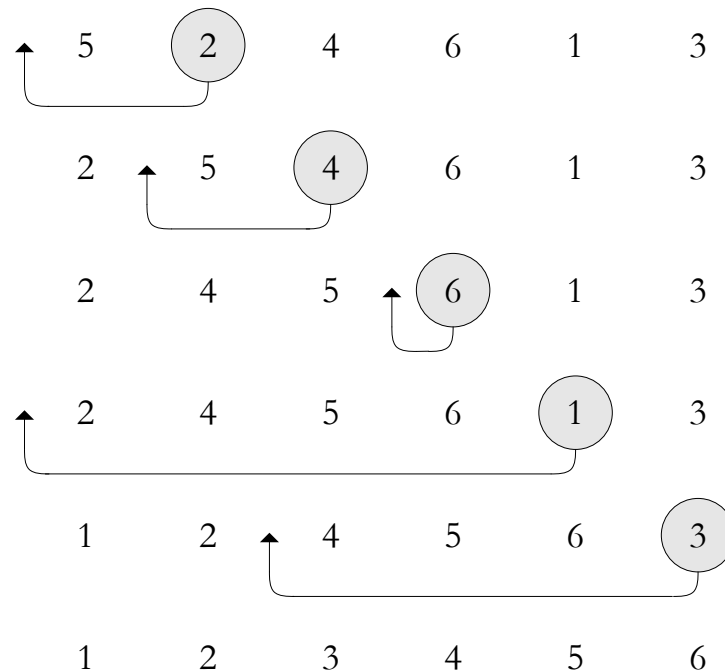
```
1  for  $j \leftarrow 2, \text{hossz}[A]$ 
2      /* Szúrjuk be az  $A[j]$  elemet az  $A[1..j-1]$  rendezett sorozatba */
3       $\text{kulcs} \leftarrow A[j]$ 
4       $i \leftarrow j-1$ 
5      while  $i > 0$  és  $A[i] > \text{kulcs}$ 
6           $A[i+1] \leftarrow A[i]$ 
7           $i \leftarrow i-1$ 
8       $A[i+1] \leftarrow \text{kulcs}$ 
```

**Hatékonyság:** Egy  $n$  elemű tömb esetén a futási idő  $O(n^2)$ .



# Rendezés beszúrással

Kezdeti sorozat



Rendezett sorozat

**A BESZÚRÓ-RENDEZÉS működése**



## Feladatok

- Milyen állapotokat állít elő a BESZÚRÓ-RENDEZÉS egy-egy elem beszúrása után az alábbi bemenő tömb esetén? Összesen hány darab elem-hátraléptetés történik?
  - $A = \langle 8, 2, 1, 5, 6, 9, 4, 3, 7 \rangle$



# Rendezés összefésüléssel

- „Oszd-meg-és-uralkodj” elvű algoritmus:
  - **Felosztás:** Az  $A[p, r]$  tömböt két „egyforma” elemszámú részre osztjuk.
  - **Uralkodás:** A résztömbök elemeit az összefésülő rendezés rekurzív hívásaival rendezzük.
  - **Összevonás:** A már rendezett résztömbök elemeit „összefésüljük” egy rendezett adatsorba.

ÖSSZEFÉSÜLŐ-RENDEZÉS( $A, p, r$ )

```

1  if  $p < r$ 
2       $q \leftarrow \lfloor (p+r)/2 \rfloor$ 
3      ÖSSZEFÉSÜLŐ-RENDEZÉS( $A, p, q$ )
4      ÖSSZEFÉSÜLŐ-RENDEZÉS( $A, q+1, r$ )
5      ÖSSZEFÉSÜL( $A, p, q, r$ )
    
```

**Megjegyzés:** A teljes  $A$  tömb rendezése az ÖSSZEFÉSÜLŐ-RENDEZÉS( $A, 1, \text{hossz}[A]$ ) hívással végzendő.

**Hatékonyság:** Egy  $n$  elemű tömb esetén a futási idő  $O(n \lg n)$ .

## Rendezés összefésüléssel

ÖSSZEFÉSÜL( $A, p, q, r$ )

```

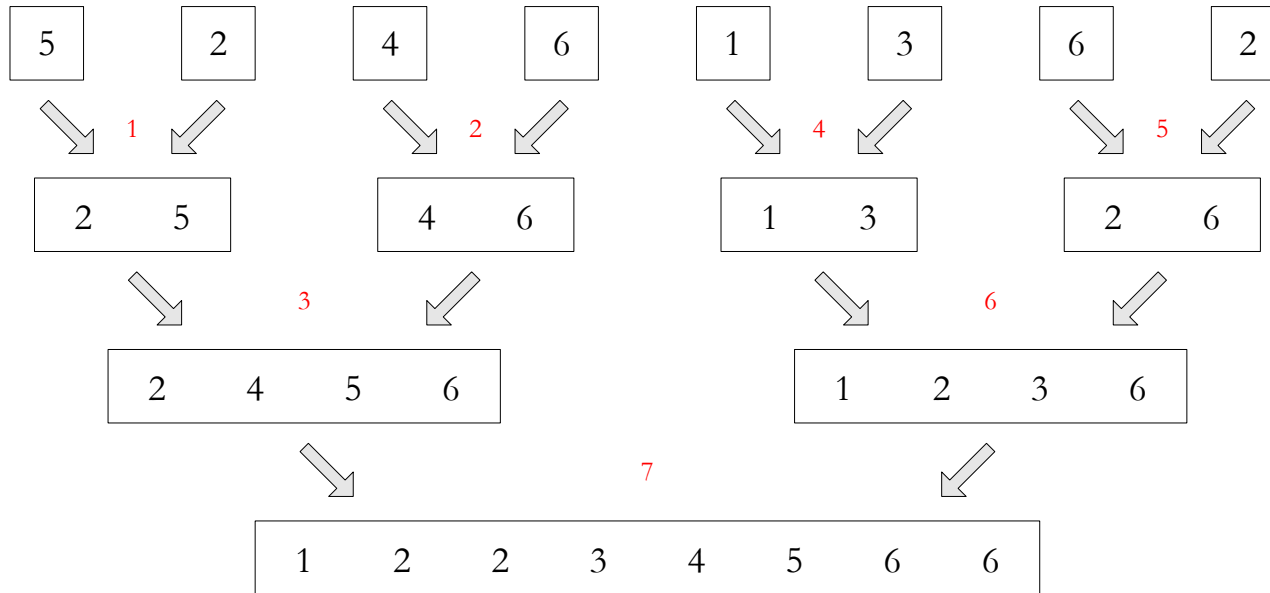
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  az  $L[1..n_1+1]$  és  $R[1..n_2+1]$  tömbök létrehozása
4  for  $i \leftarrow 1, n_1$ 
5       $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1, n_2$ 
7       $R[j] \leftarrow A[q + j]$ 
8   $L[n_1+1] \leftarrow \infty$ 
9   $R[n_2+1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p, r$ 
13     if  $L[i] \leq R[j]$ 
14          $A[k] \leftarrow L[i]$ 
15          $i \leftarrow i + 1$ 
16     else
17          $A[k] \leftarrow R[j]$ 
18          $j \leftarrow j + 1$ 
    
```

**Hatékonyság:** Egy  $n$  elemű tömbrészes esetén a futási idő  $O(n)$ .



# Rendezés összefésüléssel

Kezdeti sorozat

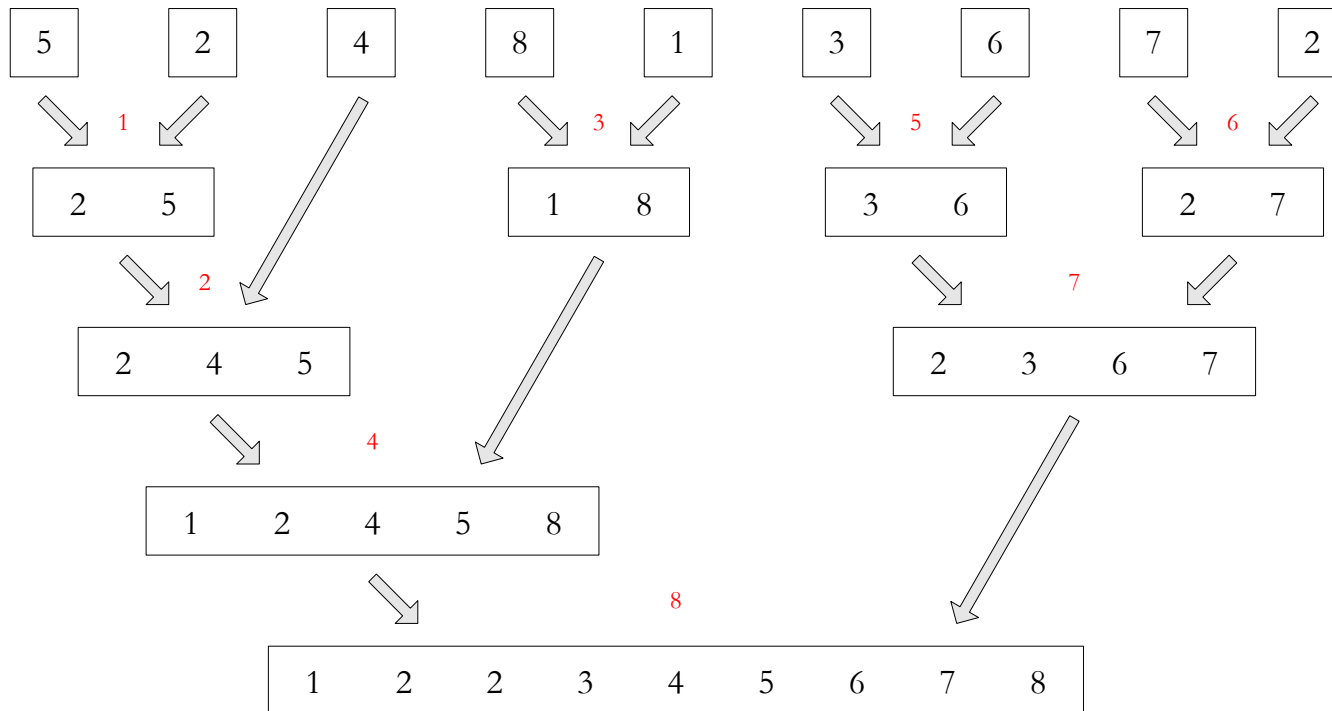


Rendezett sorozat

**Az ÖSSZEFÉSÜLŐ-RENDEZÉS működése**

# Rendezés összefésüléssel

Kezdeti sorozat



Rendezett sorozat

**Az ÖSSZEFÉSÜLŐ-RENDEZÉS működése**



## Feladatok

- Milyen részekre bontja az ÖSSZEFÉSÜLŐ-RENDEZÉS az alábbi bemenő tömböt a rendezés során, és azokból hogyan áll elő a rendezett adatsor (nyilas ábra)?
  - $A = \langle 8, 11, 5, 10, 2, 1, 9, 6, 7, 4, 3 \rangle$



# Gyorsrendezés

- Az „oszd-meg-és-uralkodj” lépések:
  - **Felosztás:** Az  $A[p, r]$  tömböt két (esetleg üres)  $A[p..q-1]$  és  $A[q+1..r]$  résztömbre osztjuk ahol  $x \leq A[q] \leq y$  teljesül,  $x \in A[p..q-1]$ ,  $y \in A[q+1..r]$ . A  $q$  meghatározása is része a felosztó eljárásnak.
  - **Uralkodás:** Az  $A[p..q-1]$  és  $A[q+1..r]$  résztömböket a gyorsrendezés rekurzív hívásával rendezzük.
  - **Összevonás:** Mivel a két tömböt helyben rendeztük, nincs szükség összevonásra, az egész  $A[p, r]$  tömb rendezett.

GYORSRENDEZÉS( $A, p, r$ )

```
1  if  $p < r$ 
2       $q \leftarrow \text{FELOSZT}(A, p, r)$ 
3      GYORSRENDEZÉS( $A, p, q-1$ )
4      GYORSRENDEZÉS( $A, q+1, r$ )
```

**Megjegyzés:** A teljes  $A$  tömb rendezése a GYORSRENDEZÉS( $A, 1, \text{hossz}[A]$ ) hívással végzendő.

**Hatékonyság:** Egy  $n$  elemű tömb esetén az **átlagos** futási idő  $O(n \lg n)$ .

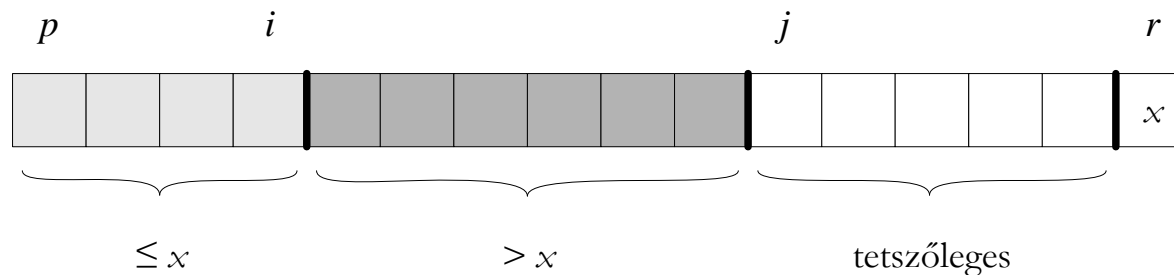


# Gyorsrendezés

FELOSZT( $A, p, r$ )

```

1   $x \leftarrow A[r]$ 
2   $i \leftarrow p-1$ 
3  for  $j \leftarrow p, r-1$ 
4      if  $A[j] \leq x$ 
5           $i \leftarrow i+1$ 
6           $A[i] \leftrightarrow A[j]$  csere
7   $A[i+1] \leftrightarrow A[r]$  csere
8  return  $i+1$ 
    
```



**A FELOSZT ciklusinvariáns tulajdonsága**

# Gyorsrendezés

$$i \quad p, j \quad r$$

2	8	7	1	3	5	6	4
---	---	---	---	---	---	---	---

$$p \quad i \quad j \quad r$$

2	1	3	8	7	5	6	4
---	---	---	---	---	---	---	---

$$p, i \quad j \quad r$$

2	8	7	1	3	5	6	4
---	---	---	---	---	---	---	---

$$p \quad i \quad j \quad r$$

2	1	3	8	7	5	6	4
---	---	---	---	---	---	---	---

$$p, i \quad j \quad r$$

2	8	7	1	3	5	6	4
---	---	---	---	---	---	---	---

$$p \quad i \quad r$$

2	1	3	8	7	5	6	4
---	---	---	---	---	---	---	---

$$p, i \quad j \quad r$$

2	8	7	1	3	5	6	4
---	---	---	---	---	---	---	---

$$p \quad i \quad r$$

2	1	3	4	7	5	6	8
---	---	---	---	---	---	---	---

$$p \quad i \quad j \quad r$$

2	1	7	8	3	5	6	4
---	---	---	---	---	---	---	---

A FELOSZT működése

# Gyorsrendezés

HOARE-FELOSZT( $A, p, r$ )

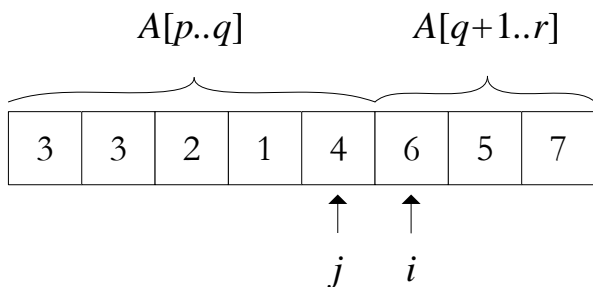
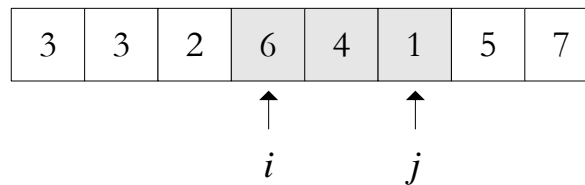
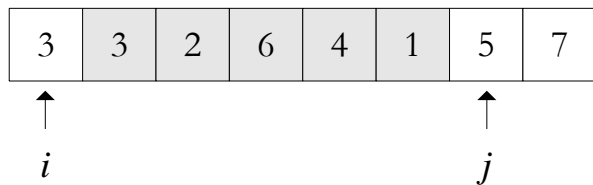
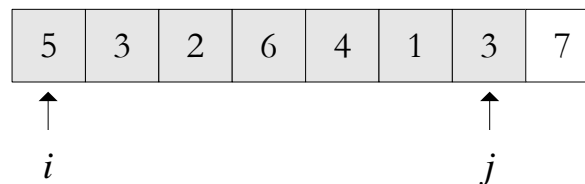
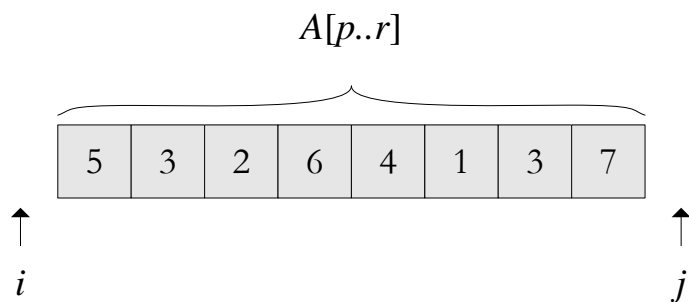
```

1   $x \leftarrow A[p]$ 
2   $i \leftarrow p-1$ 
3   $j \leftarrow r+1$ 
4  while IGAZ
5      repeat
6           $j \leftarrow j-1$ 
7      until  $A[j] \leq x$ 
8      repeat
9           $i \leftarrow i+1$ 
10     until  $A[i] \geq x$ 
11     if  $i < j$ 
12          $A[i] \leftrightarrow A[j]$  csere
13     else
14         return  $j$ 
    
```

**Megjegyzés:** Ez a gyorsrendezés eredeti felosztó algoritmus, amely C. A. R. Hoare-tól származik. Ennél a felosztásnál csak az  $x \leq y$ ,  $x \in A[p..q]$ ,  $y \in A[q+1..r]$  teljesül, ezért az ezt használó gyorsrendezés első rekurzív hívása GYORSRENDEZÉS( $A, p, q$ ).



# Gyorsrendezés



**A HOARE-FELOSZT működése**

# Gyorsrendezés

- **Probléma:** Bizonyos bemenetekre  $O(n^2)$  hatékonyság.
- **Megoldás:** Az őrszem (strázsa elem) véletlenszerű választásával jó átlagos viselkedést kapunk minden bemenetre.

VÉLETLEN-FELOSZT( $A, p, r$ )

- 1  $i \leftarrow \text{VÉLETLEN}(p, r)$
- 2  $A[r] \leftrightarrow A[i]$  csere
- 3 **return** FELOSZT( $A, p, r$ )

VÉLETLEN-GYORSRENDEZÉS( $A, p, r$ )

- 1 **if**  $p < r$
- 2      $q \leftarrow \text{VÉLETLEN-FELOSZT}(A, p, r)$
- 3     VÉLETLEN-GYORSRENDEZÉS( $A, p, q-1$ )
- 4     VÉLETLEN-GYORSRENDEZÉS( $A, q+1, r$ )

**Megjegyzés:** A VÉLETLEN( $a, b$ ) egy véletlenszerű egész számot ad az  $[a, b]$  intervallumból.



## Feladatok

- Milyen állapotot állít elő a FELOSZT( $A, 1, 9$ ) hívás az alábbi bemenő tömb esetén? Hány darab elemcsere történik? Milyen eredménnyel tér vissza a függvény?
  - $A = \langle 8, 2, 1, 5, 6, 9, 4, 3, 7 \rangle$
- Milyen állapotot állít elő a HOARE-FELOSZT( $A, 1, 9$ ) hívás az alábbi bemenő tömb esetén? Milyen eredménnyel tér vissza a függvény?
  - $A = \langle 8, 2, 1, 5, 6, 9, 4, 3, 7 \rangle$
- Milyen elemcserét végez a GYORSRENDEZÉS( $A, 1, 6$ ) hívás az alábbi bemenő tömb esetén?
  - $A = \langle 3, 6, 2, 4, 5, 1 \rangle$



# A kupac adatstruktúra

SZÜLŐ( $i$ )

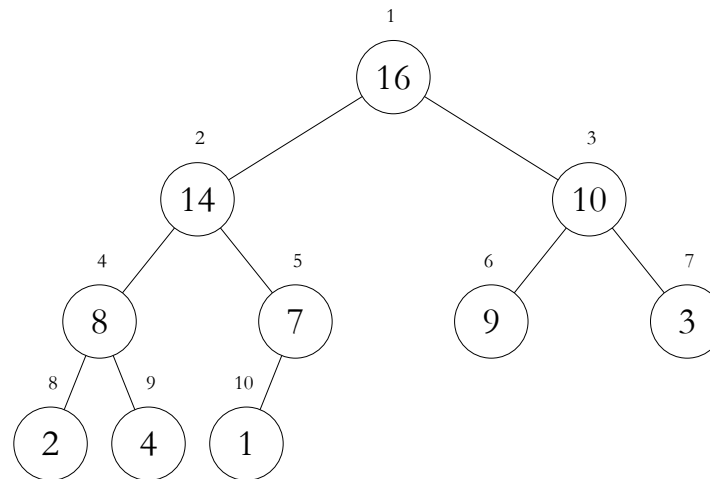
1    **return**  $\lfloor i/2 \rfloor$

BAL( $i$ )

1    **return**  $2i$

JOBB( $i$ )

1    **return**  $2i + 1$



**Maximum-kupactulajdonság:**

$A[\text{SZÜLŐ}(i)] \geq A[i], i > 1$

**Minimum-kupactulajdonság:**

$A[\text{SZÜLŐ}(i)] \leq A[i], i > 1$

	1	2	3	4	5	6	7	8	9	10
A	16	14	10	8	7	9	3	2	4	1

**Egy maximum-kupac mint  
bináris fa és mint tömb**

## Feladatok

- Teljesül-e a maximum-kupactulajdonság az alábbi tömb esetén és ha nem, hány elem-pár sérti azt meg?
  - $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 \rangle$
- Adjon meg egy minimum-kupactulajdonságú tömböt, amely az alábbi elemeket tartalmazza!
  - $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 \rangle$



## Segéd eljárás a kupacépítéshez

MAXIMUM-KUPACOL( $A, i$ )

```

1   $l \leftarrow \text{BAL}(i)$ 
2   $r \leftarrow \text{JOBBI}(i)$ 
3  if  $l \leq \text{kupac-méret}[A]$  és  $A[l] > A[i]$ 
4       $\text{legnagyobb} \leftarrow l$ 
5  else
6       $\text{legnagyobb} \leftarrow i$ 
7  if  $r \leq \text{kupac-méret}[A]$  és  $A[r] > A[\text{legnagyobb}]$ 
8       $\text{legnagyobb} \leftarrow r$ 
9  if  $\text{legnagyobb} \neq i$ 
10      $A[i] \leftrightarrow A[\text{legnagyobb}]$  csere
11     MAXIMUM-KUPACOL( $A, \text{legnagyobb}$ )
    
```

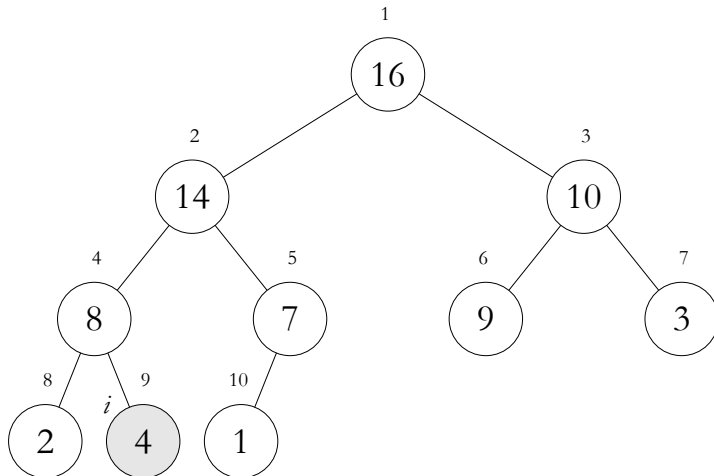
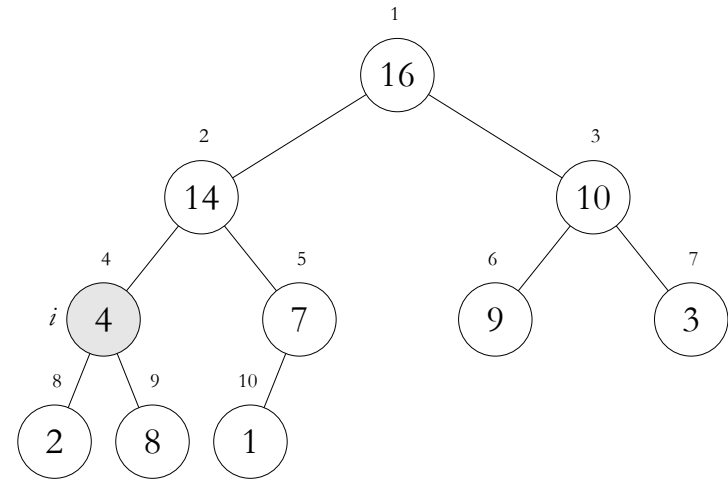
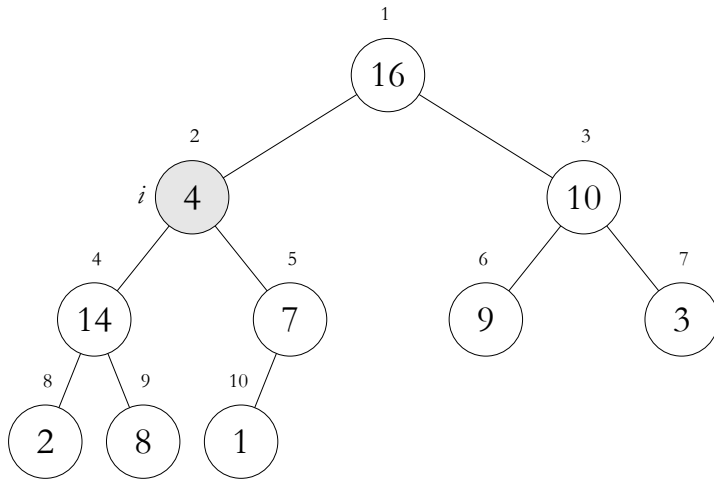
**Feltétel:** BAL( $i$ ) és JOBB( $i$ ) gyökerű részfák maximum-kupac szerkezetűek, de  $A[i]$  kisebb lehet a gyerekeinél, így megsértheti a maximum-kupactulajdonságot.

A MAXIMUM-KUPACOL feladata az  $A[i]$  érték „lefelé mozgatása” úgy, hogy az  $i$  gyökerű részfa maximum-kupaccá alakuljon.

**Hatékonyság:** Egy  $h$  magasságú fára a futási idő  $O(h)$ , azaz  $O(\lg n)$ .



# Segéd eljárás a kupacépítéshez



**A MAXIMUM-KUPACOL( $A, 2$ ) működése**

## Feladatok

- Milyen kupacot állít elő a  $\text{MAXIMUM-KUPACOL}(A, 3)$  hívás az alábbi tömb esetén?
  - $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0 \rangle$



## Kupackészítés

MAXIMUM-KUPACOT-ÉPÍT( $A$ )

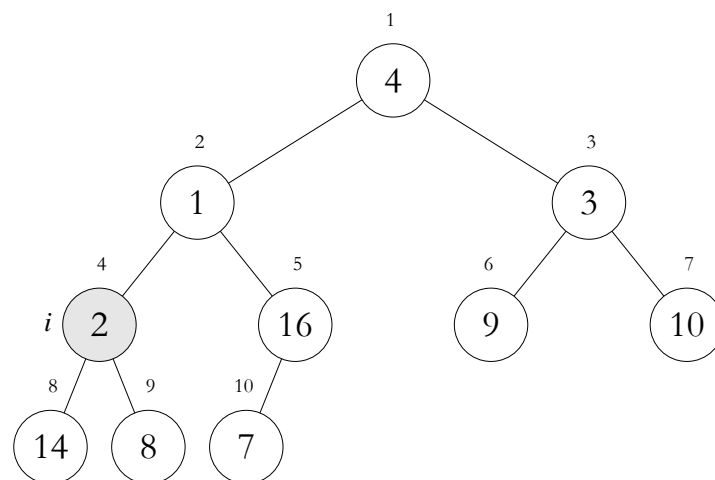
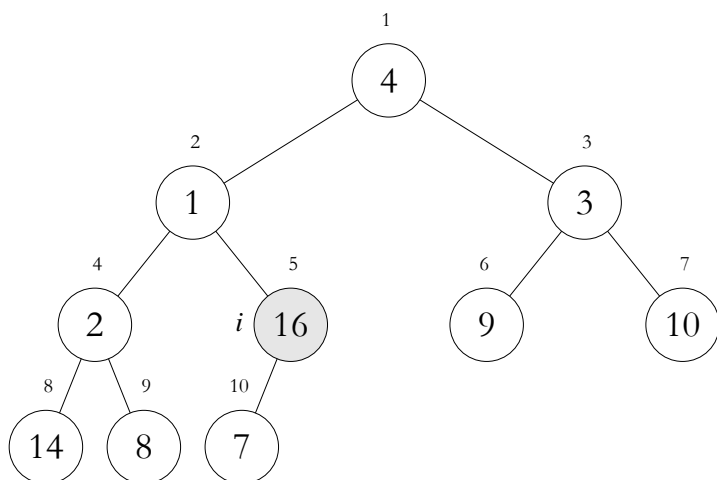
- 1  $kupac-méret[A] \leftarrow hossz[A]$
- 2 **for**  $i \leftarrow \lfloor hossz[A] / 2 \rfloor, 1, -1$
- 3     MAXIMUM-KUPACOL( $A, i$ )

**Hatékonyság:** Az egyszerűen becsült  $O(n \lg n)$  futási idő aszimptotikusan nem éles, bizonyítható az  $O(n)$  futási idő, azaz egy rendezetlen tömb lineáris idő alatt kupaccá alakítható.

# Kupackészítés

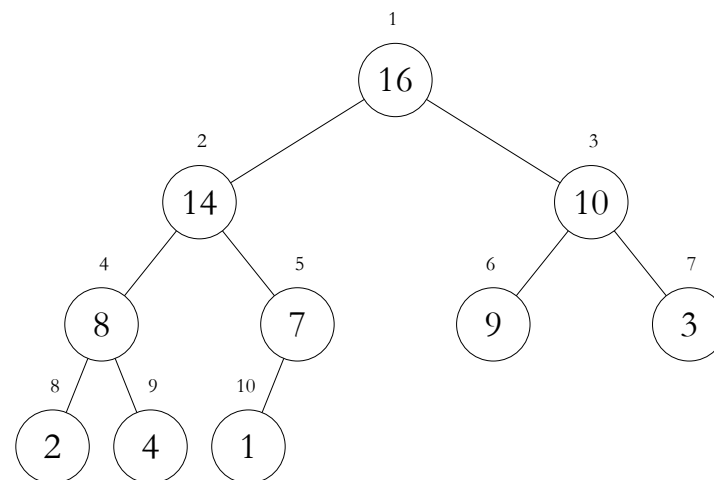
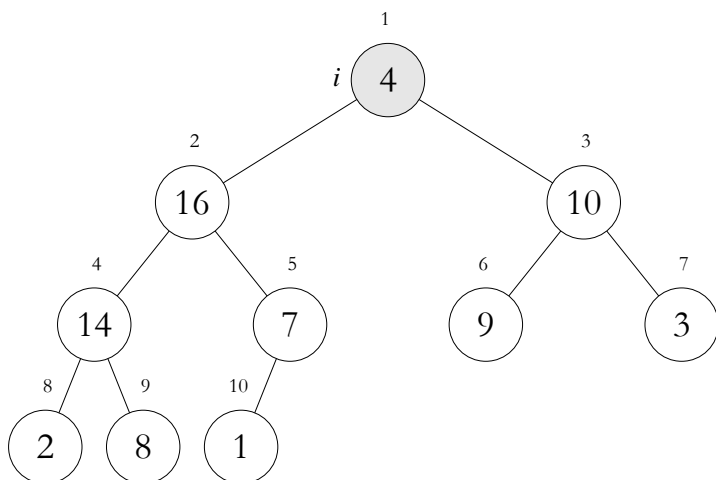
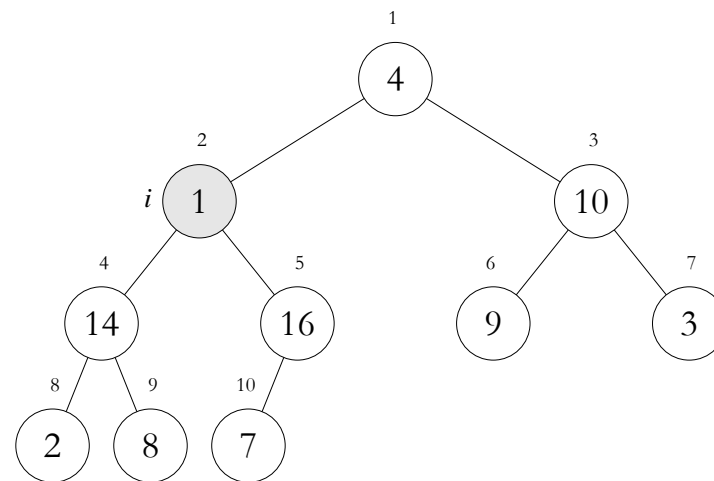
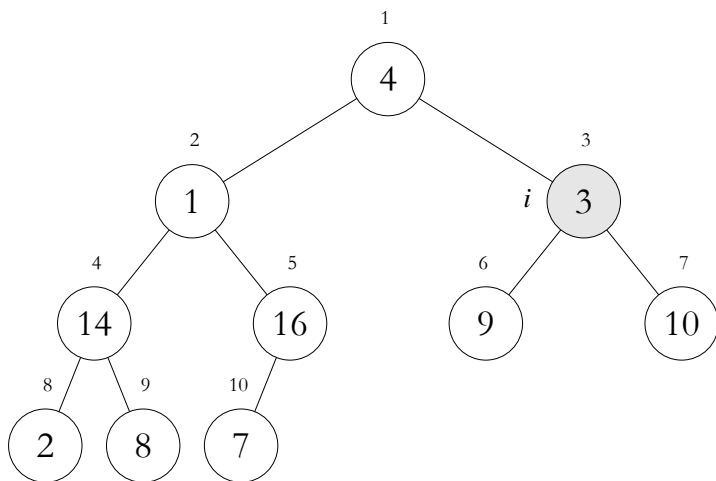
A

4	1	3	2	16	9	10	14	8	7
---	---	---	---	----	---	----	----	---	---



**A MAXIMUM-KUPACOT-ÉPÍT működése**

# Kupackészítés



**A MAXIMUM-KUPACOT-ÉPÍT működése**



## Feladatok

- Milyen kupacot épít a MAXIMUM-KUPACOT-ÉPÍT eljárás az alábbi bemenő tömb esetén?
  - $A = \langle 8, 2, 1, 5, 6, 9, 4, 3, 7 \rangle$



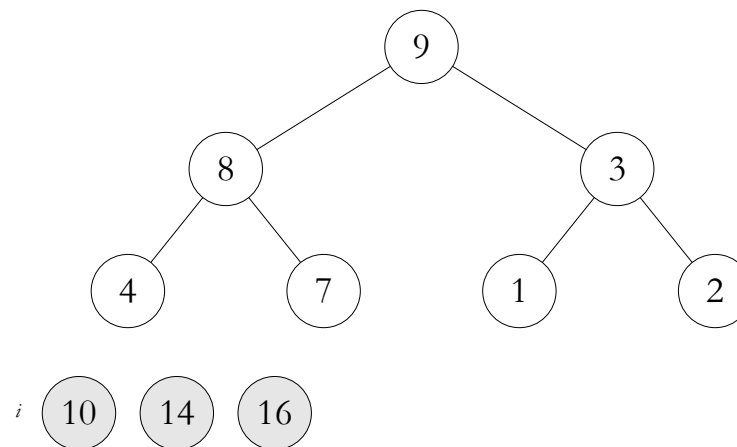
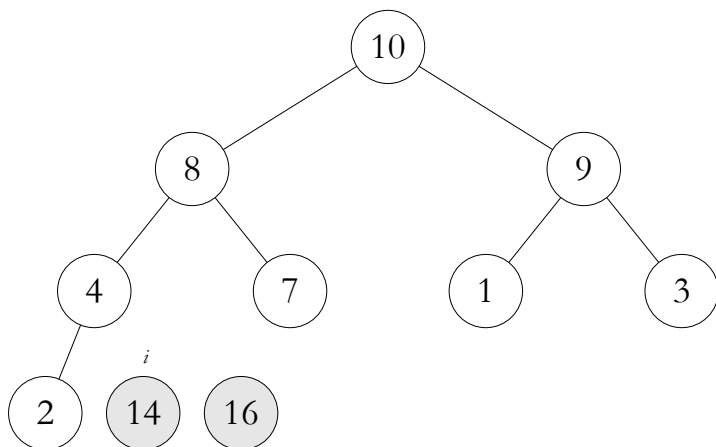
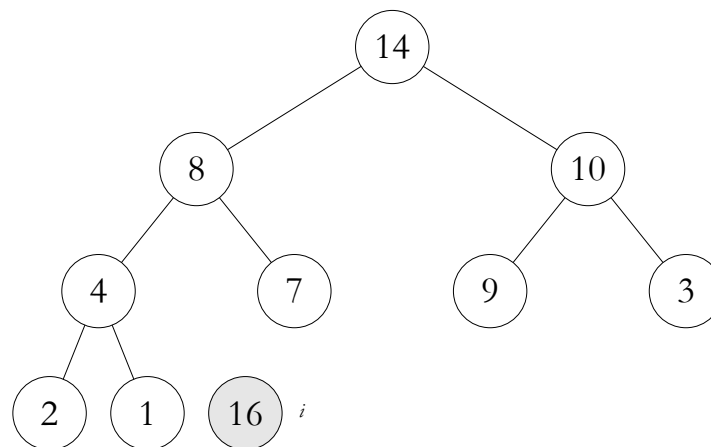
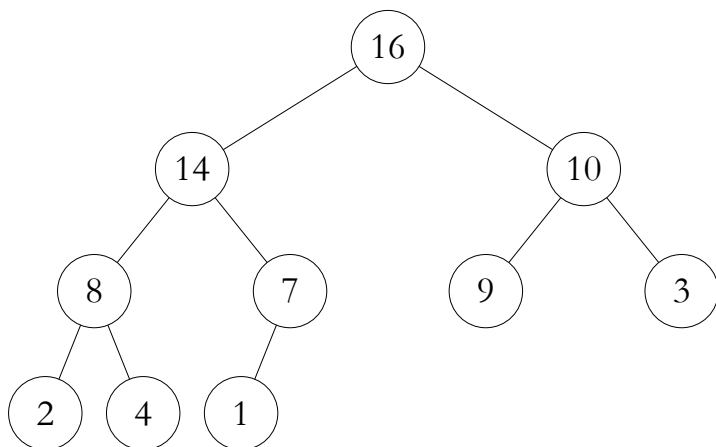
# Kupacrendezés

KUPACRENDEZÉS(A)

```
1  MAXIMUM-KUPACOT-ÉPÍT(A)
2  for  $i \leftarrow \text{hossz}[A], 2, -1$ 
3       $A[1] \leftrightarrow A[i]$  csere
4       $\text{kupac-méret}[A] \leftarrow \text{kupac-méret}[A] - 1$ 
5      MAXIMUM-KUPACOL(A, 1)
```

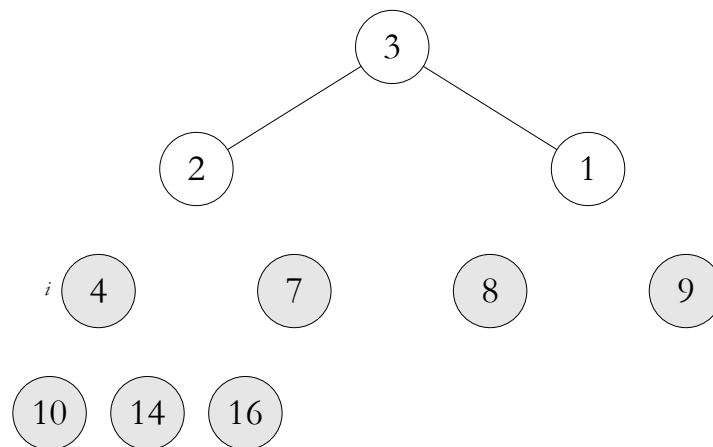
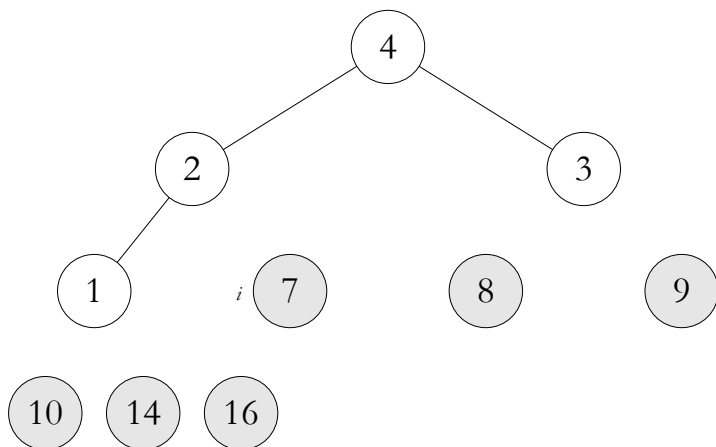
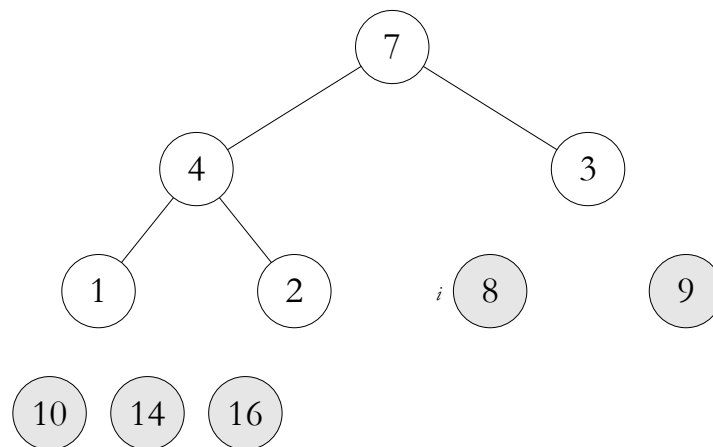
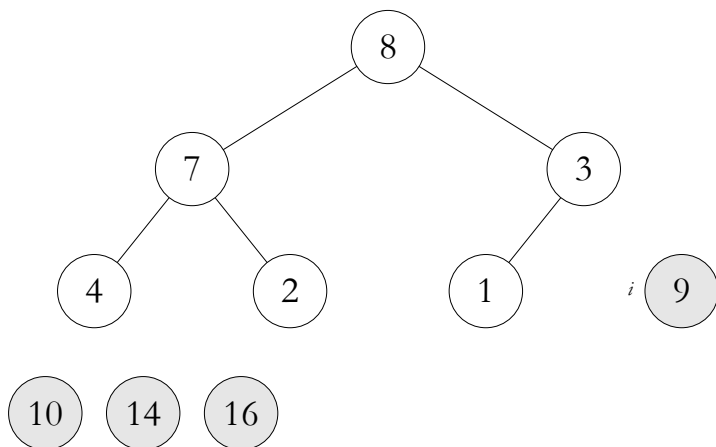
**Hatékonyság:** A MAXIMUM-KUPACOT-ÉPÍT  $O(n)$  és a MAXIMUM-KUPACOL  $O(\lg n)$  idejéből,  $O(n \lg n)$  futási időt kapunk.

# Kupacrendezés



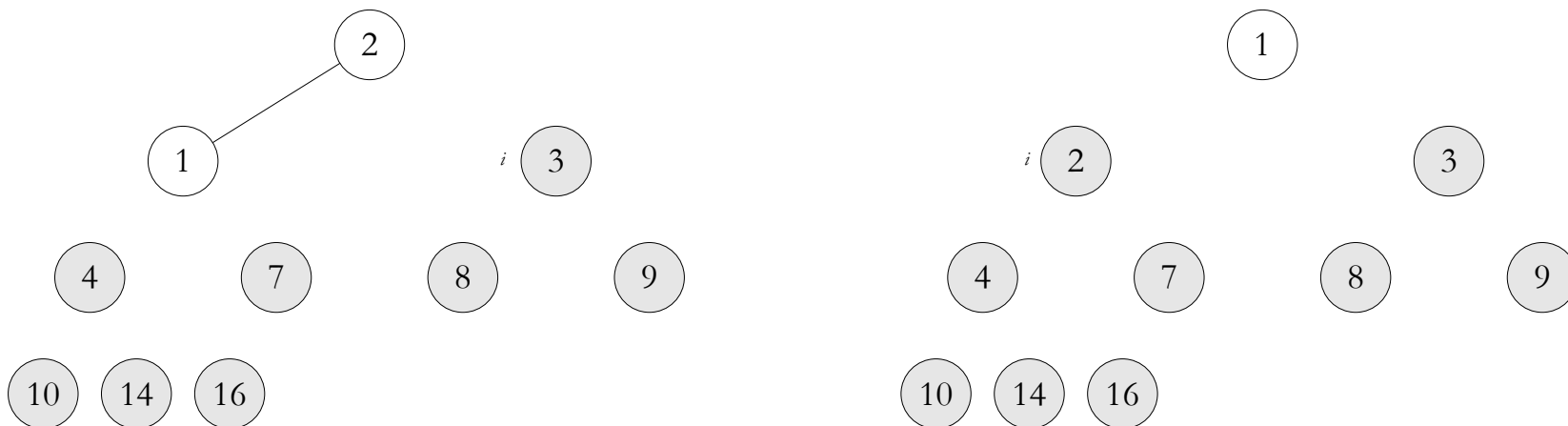
**A KUPACRENDEZÉS működése**

# Kupacrendezés



**A KUPACRENDEZÉS működése**

# Kupacrendezés



A

1	2	3	4	7	8	9	10	14	16
---	---	---	---	---	---	---	----	----	----

**A KUPACRENDEZÉS működése**

## Feladatok

- Milyen kupacot épít és milyen állapotokat állít elő a KUPACRENDEZÉS a ciklusmagjának egy-egy végrehajtása után az alábbi tömb esetén?
  - $A = \langle 8, 2, 1, 5, 6, 9, 4, 3, 7 \rangle$



## Elsőbbségi sorok

- **Elsőbbségi soron** egy olyan  $S$  halmazt értünk, amelynek minden eleméhez egy **kulcs** értéket rendelünk.
- A maximum-elsőbbségi sorokat kezelő **műveletek**:
  - $BESZÚR(S, x)$  egy  $x$  elemet hozzáad az  $S$  halmazhoz.
  - $MAXIMUM(S)$  megadja  $S$  legnagyobb kulcsú elemét.
  - $KIVESZ-MAXIMUM(S)$  megadja és törli  $S$  legnagyobb kulcsú elemét.
  - $KULCSOT-NÖVEL(S, x, k)$  megnöveli a  $x$  elem kulcsát, az új érték  $k$  lesz (amiről feltesszük, hogy legalább akkora, mint az  $x$  elem aktuális kulcsa).

KUPAC-MAXIMUMA( $A$ )

1    **return**  $A[1]$

**Hatékonyság:** A KUPAC-MAXIMUMA  $\Theta(1)$  idő alatt megvalósítja a MAXIMUM műveletet.



## Elsőbbségi sorok

KUPACBÓL-KIVESZ-MAXIMUM( $A$ )

```
1  if  $kupac-méret[A] < 1$ 
2      error „kupacméret alulcsordulás”
3   $max \leftarrow A[1]$ 
4   $A[1] \leftarrow A[kupac-méret[A]]$ 
5   $kupac-méret[A] \leftarrow kupac-méret[A] - 1$ 
6  MAXIMUM-KUPACOL( $A, 1$ )
7  return  $max$ 
```

**Hatékonyság:** A MAXIMUM-KUPACOL  $O(\lg n)$  ideje miatt a futási idő  $O(\lg n)$ .





## Elsőbbségi sorok

KUPACBAN-KULCSOT-NÖVEL( $A, i, kulcs$ )

```

1  if  $kulcs < A[i]$ 
2      error „az új kulcs kisebb, mint az eredeti”
3   $A[i] \leftarrow kulcs$ 
4  while  $i > 1$  és  $A[SZÜLŐ(i)] < A[i]$ 
5       $A[i] \leftrightarrow A[SZÜLŐ(i)]$  csere
6       $i \leftarrow SZÜLŐ(i)$ 
    
```

MAXIMUM-KUPACBA-BESZÚR( $A, kulcs$ )

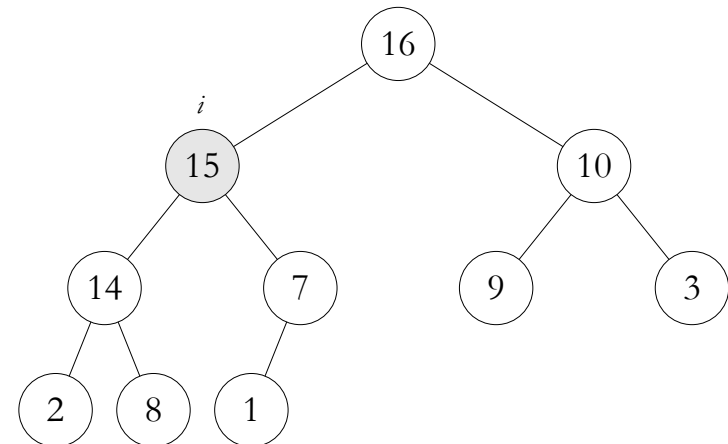
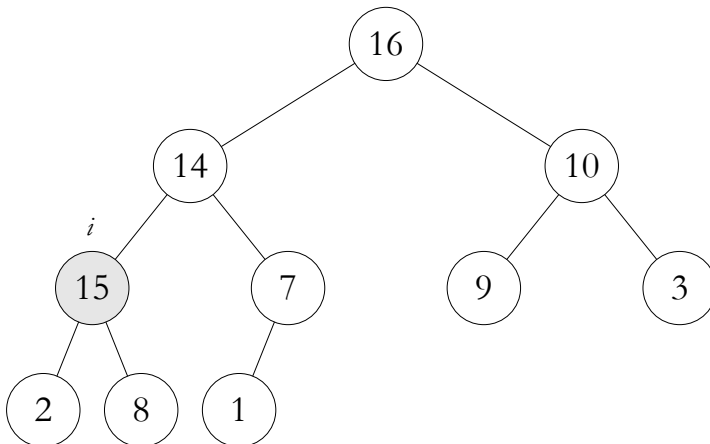
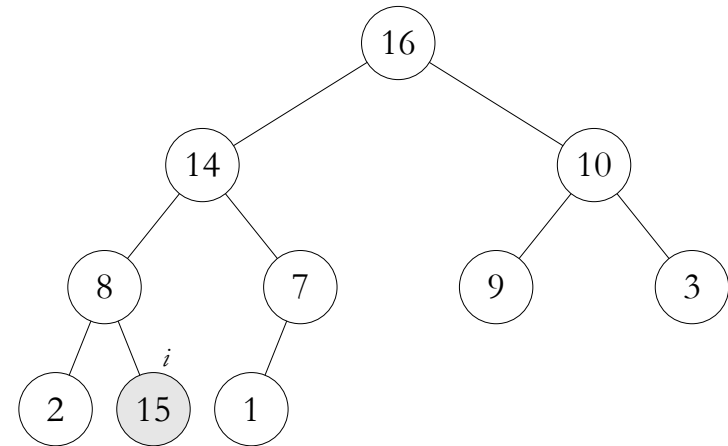
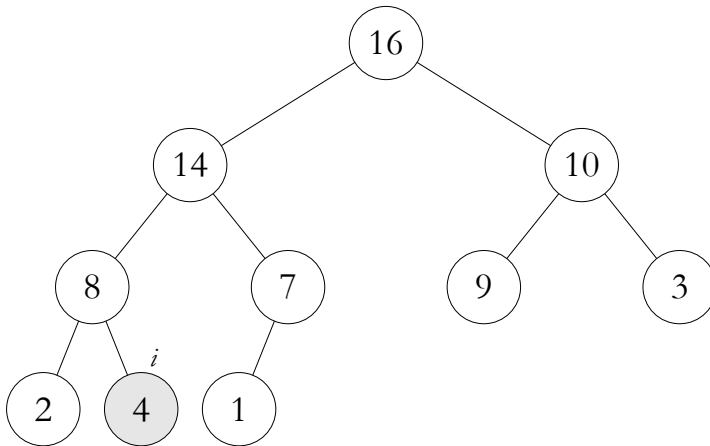
```

1   $kupac-méret[A] \leftarrow kupac-méret[A] + 1$ 
2   $A[kupac-méret[A]] \leftarrow -\infty$ 
3  KUPACBAN-KULCSOT-NÖVEL( $A, kupac-méret[A], kulcs$ )
    
```

**Hatékonyság:** Mindkét algoritmus futási ideje  $n$  elemű kupac esetén  $O(\lg n)$ .

**Összegezve:** Kupac adatszerkezetet használva az elsőbbségi sorok összes művelete  $n$  elemű halmazra  $O(\lg n)$  idő alatt elvégezhető.

# Elsőbbségi sorok



**A KUPACBAN-KULCSOT-NÖVEL(A, 9, 15) működése**

## Elsőbbségi sorok

MAXIMUM-KUPACOT-ÉPÍT'(A)

```
1  kupac-méret[A] ← 1
2  for i ← 2, hossz[A]
3      MAXIMUM-KUPACBA-BESZÚR(A, A[i])
```

**Hatékonyság:** Egy  $n$  elemű kupac ilyen felépítésének ideje  $O(n \lg n)$ .

## Feladatok

- Milyen kupacot állít elő a  $\text{MAXIMUM-KUPACBA-BESZÚR}(A, 20)$  hívás az alábbi kupacból?
  - $A = \langle 16, 15, 10, 8, 14, 9, 3, 2, 4, 1, 7 \rangle$
- Milyen kupacot épít a  $\text{MAXIMUM-KUPACOT-ÉPÍT'}$  eljárás az alábbi bemenő tömb esetén?
  - $A = \langle 8, 2, 1, 5, 6, 9, 4, 3, 7 \rangle$



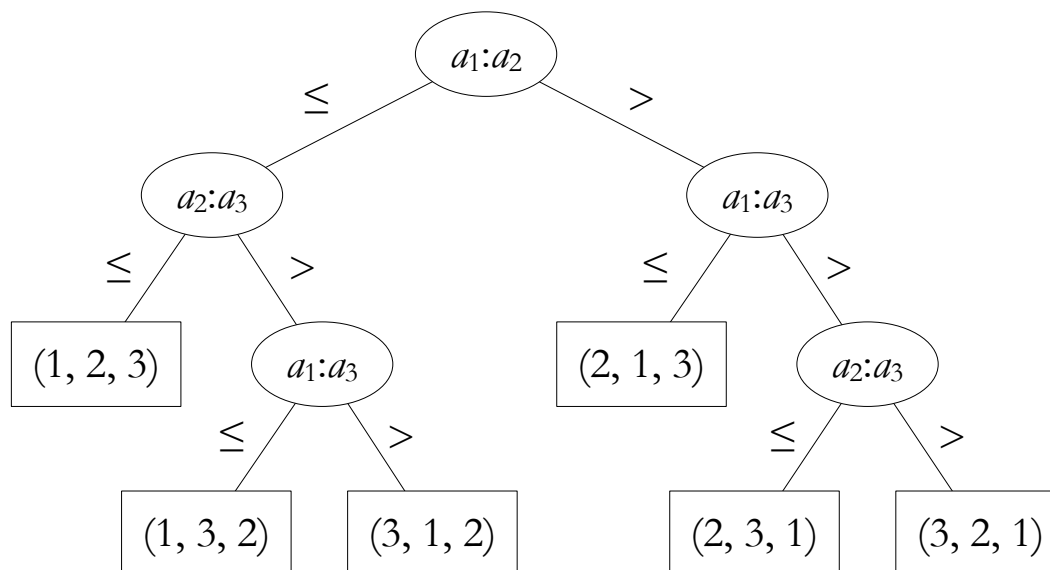
# Összehasonlító rendezések

Az összehasonlító rendezéseket tekinthetjük **döntési fának**, amelyek egy adott méretű bemeneti tömb rendezése során történt összehasonlításokat ábrázolja.

**Tétel:** Bármely összehasonlító rendezőalgoritmus a legrosszabb esetben  $\Omega(n \lg n)$  összehasonlítást végez.

**Megjegyzés:** A bizonyítás a  $\lg(n!) = \Theta(n \lg n)$  összefüggésen alapul (lásd: jegyzet).

**Következmény:** A kupacrendezés és az összefésüléses rendezés aszimptotikusan optimális összehasonlító rendezések.



**Döntési fa három elem beszúrásos rendezéséhez**

## Rendezés lineáris időben

LESZÁMLÁLÓ-RENDEZÉS( $A, B, k$ )

```
1  for  $i \leftarrow 0, k$ 
2       $C[i] \leftarrow 0$ 
3  for  $j \leftarrow 1, \text{hossz}[A]$ 
4       $C[A[j]] \leftarrow C[A[j]] + 1$ 
5  /*  $C[i]$  azoknak az elemeknek a száma, amelyek értéke  $i$  */
6  for  $i \leftarrow 1, k$ 
7       $C[i] \leftarrow C[i] + C[i-1]$ 
8  /*  $C[i]$  azoknak az elemeknek a száma, amelyek értéke kisebb vagy egyenlő mint  $i$  */
9  for  $j \leftarrow \text{hossz}[A], 1, -1$ 
10      $B[C[A[j]]] \leftarrow A[j]$ 
11      $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

**Feltétel:** A rendezendő  $n$  elem mindegyike 0 és  $k$  közötti egész szám, ahol  $k$  egy egész szám.

**Hatékonyság:** Ha  $k = O(n)$ , a rendezés futási ideje  $\Theta(n)$ .

**Stabilitás:** A leszámoló rendezés **stabil**, mert az azonos értékű elemek ugyanabban a sorrendben jelennek meg a kimeneti tömbben, mint ahogyan a bemeneti tömbben szerepeltek.



# Rendezés lineáris időben

	1	2	3	4	5	6	7	8
<i>A</i>	2	5	3	0	2	3	0	3

	0	1	2	3	4	5
<i>C</i>	2	0	2	3	0	1

	1	2	3	4	5	6	7	8
<i>B</i>							3	

	0	1	2	3	4	5
<i>C</i>	2	2	4	6	7	8

	1	2	3	4	5	6	7	8
<i>B</i>		0				3	3	

	0	1	2	3	4	5
<i>C</i>	1	2	4	5	7	8

	0	1	2	3	4	5
<i>C</i>	2	2	4	7	7	8

	1	2	3	4	5	6	7	8
<i>B</i>		0					3	

	0	1	2	3	4	5
<i>C</i>	1	2	4	6	7	8

	1	2	3	4	5	6	7	8
<i>B</i>	0	0	2	2	3	3	3	5

## A LESZÁMLÁLÓ-RENDEZÉS működése

## Feladatok

- Mi lesz a  $C$  segédtömb tartalma a LESZÁMLÁLÓ-RENDEZÉS eljárásban azután, hogy három elem már bekerült az eredménytömbbe?
  - $A = \langle 6, 5, 2, 6, 1, 3, 6, 2, 7, 5 \rangle$
- Mi történik akkor, ha az algoritmus 9. sorát az alábbira cseréljük?  
9    **for**  $j \leftarrow 1, \text{hossz}[A]$





# Rendezés lineáris időben

SZÁMJEGYES-RENDEZÉS( $A, d$ )

- 1 **for**  $i \leftarrow 1, d$
- 2       stabil algoritmussal rendezzük az  $A$  tömböt az  $i$ -edik számjegy szerint

**Feltétel:** Az  $A$  tömb minden egyes eleme  $d$  jegyű, ahol az első számjegy a legkisebb helyiértékű számjegy és a  $d$ -edik számjegy a legnagyobb helyiértékű számjegy.

329	720	720	329
457	355	329	355
657	436	436	436
839	⇒ 457	⇒ 839	⇒ 457
436	657	355	657
720	329	457	720
355	839	657	839
	↑	↑	↑

**A SZÁMJEGYES-RENDEZÉS működése**



## Feladatok

- Hányadik helyre kerül a „kés” szó a SZÁMJEGYES-RENDEZÉS egyes menetei után az alábbi bemenő szavak (és magyar ábécé szerinti rendezés esetén)?
  - lap, tép, tea, gép, lop, cél, láb, fél, kés, fel, lép

## Rendezés lineáris időben

EDÉNY-RENDEZÉS( $A$ )

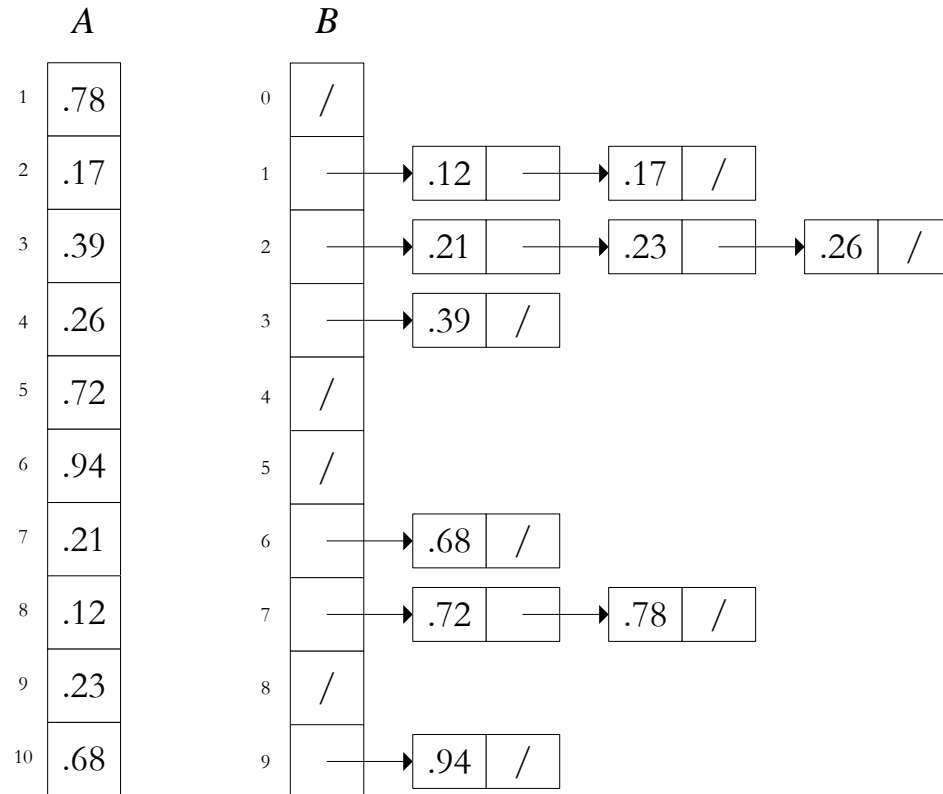
```
1   $n \leftarrow \text{hossz}[A]$ 
2  for  $i \leftarrow 1, n$ 
3      szúrjuk be az  $A[i]$  elemet a  $B[\lfloor nA[i] \rfloor]$  listába
4  for  $i \leftarrow 0, n-1$ 
5      rendezzük a  $B[i]$  listát beszúrásos rendezéssel
6  fűzzük össze sorban a  $B[0], B[1], \dots, B[n-1]$  listákat
```

**Feltétel:** A rendezendő  $n$  elem véletlenszerű eloszlású elemei a  $[0, 1)$  intervallumnak.

**Hatékonyság:** Az edényrendezés várható futási ideje  $\Theta(n)$ . Az 5. sor várható végrehajtási ideje  $\Theta(1)$ .



# Rendezés lineáris időben



**Az EDÉNY-RENDEZÉS működése ( $n=10$ )**

## Feladatok

- Melyik elem kerül az EDÉNY-RENDEZÉS végén a legtöbb elemet tartalmazó edény „végére” az alábbi bemenő tömb esetén?
  - $A = \langle 0.65, 0.52, 0.23, 0.68, 0.12, 0.38, 0.61, 0.29, 0.72, 0.53 \rangle$

# Rendezések

Algoritmus	Legrosszabb eset	Átlagos/várható eset
Beszűrő rendezés	$\Theta(n^2)$	$\Theta(n^2)$
Összefésülő rendezés	$\Theta(n \lg n)$	$\Theta(n \lg n)$
Kupacrendezés	$O(n \lg n)$	-
Gyorsrendezés	$\Theta(n^2)$	$\Theta(n \lg n)$ (várható)
Leszámláló rendezés	$\Theta(n+k)$	$\Theta(n+k)$
Számjegyes rendezés	$\Theta(d(n+k))$	$\Theta(d(n+k))$
Edényrendezés	$\Theta(n^2)$	$\Theta(n)$ (átlagos)

**A rendező algoritmusok futási ideje**



## Mediánok és rendezett minták

Egy  $n$  elemű halmaz esetén a **rendezett minta**  $i$ -edik eleme a halmaz  $i$ -edik legkisebb eleme.

Pl. Egy  $n$  elemű halmaz **minimuma** a halmaz rendezett mintájának első eleme ( $i = 1$ ), míg a halmaz **maximuma** a rendezett minta  $n$ -edik eleme ( $i = n$ ).

A **medián** lényegében a halmaz „középpontja”. Ha  $n$  páratlan, akkor a medián egyedi, és az  $i = (n+1)/2$  pozícióban jelenik meg. Ha  $n$  páros, akkor két medián létezik, amelyek az  $i = n/2$  (**alsó medián**), és az  $i = n/2+1$  (**felső medián**) pozíciókban jelennek meg.

MINIMUM(A)

```

1   $min \leftarrow A[1]$ 
2  for  $i \leftarrow 2, hossz[A]$ 
3      if  $A[i] < min$ 
4           $min \leftarrow A[i]$ 
5  return  $min$ 
    
```

**Megjegyzés:** A halmazt az  $A$  tömbbel ábrázoltuk, ahol  $hossz[A] = n$ .

## Kiválasztás átlagosan lineáris időben

### ■ A kiválasztási probléma

**Bemenet:** az  $n$  (különböző) számból álló  $A$  halmaz és egy  $i$  szám, amelyre fennáll, hogy  $1 \leq i \leq n$ .

**Kimenet:** az  $x \in A$  elem, amelyik nagyobb, mint az  $A$  pontosan  $i-1$  másik eleme.

VÉLETLEN-KIVÁLASZT( $A, p, r, i$ )

```

1  if  $p = r$ 
2      return  $A[p]$ 
3   $q \leftarrow$  VÉLETLEN-FELOSZT( $A, p, r$ )
4   $k \leftarrow q - p + 1$ 
5  if  $i = k$ 
6      return  $A[q]$           /* Az őrszem elem az eredmény */
7  else if  $i < k$ 
8      return VÉLETLEN-KIVÁLASZT( $A, p, q-1, i$ )
9  else
10     return VÉLETLEN-KIVÁLASZT( $A, q+1, r, i-k$ )
    
```

**Hatékonyság:** A legrosszabb esetben  $\Theta(n^2)$ , de a várható futási idő  $\Theta(n)$ .

**Megjegyzés:** Létezik  $O(n)$  idejű megoldás is, azaz amelyik legrosszabb esetben is lineáris idejű.  
(Hasonlóan rekurzív, de mediánok használatával jó felosztást biztosít.)





## Feladatok

- Hogyan íránk meg a VÉLETLEN-KIVÁLASZT iteratív változatát?
- Milyen  $i$  értékeket generáljon a VÉLETLEN-FELOSZT ahhoz, hogy a VÉLETLEN-KIVÁLASZT algoritmus legrosszabb végrehajtását kapjuk az alábbi tömb minimumának meghatározásakor?
  - $A = \langle 2, 7, 5, 6, 1, 4, 3 \rangle$

