

# Algoritmuselmélet

Keresés, minimumkeresés, rendezés

Katona Gyula Y. / Vizer Máté

Számítástudományi és Információelméleti Tanszék  
Budapesti Műszaki és Gazdaságtudományi Egyetem

# Keresés, ha nincs összehasonlítás

## Feladat

Adott az  $U$  halmaz véges  $S = \{s_1, s_2, \dots, s_{n-1}, s_n\}$  részhalmaza és  $s \in U$ . El akarjuk dönteni, hogy igaz-e  $s \in S$ , és ha igen, akkor melyik  $i$ -re teljesül  $s_i = s$ .

Hány lépés kell?

Itt egy lépés: **igaz-e, hogy  $s_i = s$ ?**

Válasz: **igen** vagy **nem**.

Legrosszabb esetben minden elemet végig kell nézni  $\implies$   
 **$n$  lépés kell legrosszabb esetben.**

# Ha bármely két elem összehasonlítható

## Példák:

- $\mathbb{Z}$  az egész számok halmaza.  $A <$  rendezés a nagyság szerinti rendezés.
- Az  $abc$  betűinek  $\Sigma$  halmaza;  $a <$  rendezést az  $abc$ -sorrend adja. Az  $x$  betű kisebb, mint az  $y$  betű, ha  $x$  előbb szerepel az  $abc$ -sorrendben, mint  $y$ .
- A  $\Sigma$  betűiből alkotott szavak  $\Sigma^*$  halmaza a szótárszerű vagy **lexikografikus** rendezéssel.  $\Rightarrow$  legyen  $X = x_1 x_2 \cdots x_k$  és  $Y = y_1 y_2 \cdots y_l$  két szó.  
Az  $X$  kisebb mint  $Y$ , ha vagy  $l > k$  és  $x_i = y_i$  ha minden  $i = 1, 2, \dots, k$  esetén;  
vagy pedig  $x_j < y_j$  teljesül a legkisebb olyan  $j$  indexre, melyre  $x_j \neq y_j$ . **Tehát például  $kar < karika$  és  $bor < bot$ .**

## Keresés, ha van összehasonlítás

Itt összehasonlítás: **Mi a viszonya  $s$ -nek és  $s_i$ -nek?**

Válasz:  $s_i = s$  vagy  $s_i < s$  vagy  $s_i > s$ .

## Lineáris keresés

Sorban mindegyik elemmel összehasonlítjuk.

**Költség a legrosszabb esetben:**  $n$ , mert lehet, hogy pont az utolsó volt.

## Segít, ha rendezett az input?

**Barkochba játék:** gondolk egy számot 1 és 100 között, hány eldöntendő kérdésből lehet kitalálni?

### Feladat

Adott az  $U$  halmaz véges  $S = \{s_1 < s_2 < \dots < s_{n-1} < s_n\}$  részhalmaza és  $s \in U$ .

*Összehasonlításokkal akarjuk eldönteni, hogy igaz-e  $s \in S$ , és ha igen, akkor melyik  $i$ -re teljesül  $s_i = s$ .*

*Hány összehasonlítás kell?*

# Bináris keresés

**Oszd meg és uralkodj:** először a középső  $s_i$ -vel hasonlítunk:  $i = \lceil \frac{n+1}{2} \rceil$   
Hasonló feladatot kapunk egy  $S_1$  halmazra, amire viszont  $|S_1| \leq |S|/2$ .  
És így tovább:

$$|S_2| \leq \frac{|S|}{4}, |S_3| \leq \frac{|S|}{2^3}, \dots |S_k| \leq \frac{|S|}{2^k}$$

Pl. keressük meg, benne van-e 21 az alábbi sorozatban!

15, 22, 25, 37, 48, 56, 70, 82 (1)

15, 22, 25, 37, 48, 56, 70, 82 (2)

15, 22, 25, 37, 48, 56, 70, 82 (3)

15, 22, 25, 37, 48, 56, 70, 82 (4)

# Bináris keresés

## Legfeljebb hány összehasonlítás kell?

Megválaszthatjuk  $k$ -t úgy, hogy  $2^k \leq n < 2^{k+1}$  teljesüljön.

Ekkor  $|S_k| \leq \frac{|S|}{2^k} = \frac{n}{2^k} < 2$ , mivel  $|S_k|$  egész szám, ezért  $|S_k| \leq 1$ .  
Tehát  $k$  összehasonlítás után legfeljebb egy elem jöhet még szóba.  
Ezért már csak legfeljebb egy további összehasonlítás kell.

$$2^k \leq n \implies k \leq \lfloor \log_2 n \rfloor$$

$$k + 1 \leq \lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n + 1) \rceil$$

Tehát az összehasonlítások száma legfeljebb:  $\lceil \log_2(n + 1) \rceil$

Nyilván például egy nem sikeres keresésnél ennyire szükségünk is van, így a bináris keresés lépésszáma:  $\lceil \log_2(n + 1) \rceil$

## Van-e jobb, mint a bináris keresés?

Természetesen merül fel a kérdés, hogy tudunk-e jobb kereső algoritmust találni a bináris keresésnél, ha az input egy rendezett tömb? A válasz az, hogy nem.

### Tétel

*A bináris keresés optimális, azaz nincs olyan kereső algoritmus, ami csak összehasonlításokat használ egy rendezett tömbben és a lépésszáma kevesebb mint  $\lceil \log_2(n + 1) \rceil$ .*

### Bizonyítás.

*Tegyük fel, hogy lenne ilyen algoritmusunk. Az ellenség (E) fog mutatni egy rendezett tömböt (minden  $n$ -re egyet), amire algoritmusunk lépésszáma  $\lceil \log_2(n + 1) \rceil$  lesz.*

*Az ellenség az algoritmusunk kérdéseire mindig eldönti, hogy mit válaszoljon és közben az  $x_1, \dots, x_n$  rendezett tömb egyre nagyobb részét fedi fel a következő módon:*

# Van-e jobb, mint a bináris keresés?

## Bizonyítás.

*Általános lépés:*

*Ha az algoritmus  $x_i$ -vel hasonlít, akkor  $E$  megnézi, hogy hány  $x_i$ -nél kisebb és hány nagyobb olyan cella van, ami még tartalmazhatja a keresett  $s$  elemet (amit még nem zárt ki az algoritmus).*

- *Ha több ilyen cella van  $x_i$  után, akkor  $E$  azt válaszolja, hogy  $s$  nagyobb, mint  $x_i$  (ekkor  $E$  kitölti az  $x_i$  előtti szóba jövő cellákat és az  $x_i$  celláját  $s$ -nél kisebb számok növekvő sorozatával, hogy azok a már korábban kitöltött cellákkal együtt is növekvő sorozatot alkossanak),*
- *különben meg azt válaszolja, hogy  $s$  kisebb, mint  $x_i$  (ekkor  $E$  kitölti az  $x_i$  utáni szóba jövő cellákat és  $x_i$  celláját tetszőleges  $s$ -nél nagyobb számok növekvő sorozatával, hogy azok a már korábban kitöltött cellákkal együtt is növekvő sorozatot alkossanak).*



# Van-e jobb, mint a bináris keresés?

Legyen  $n = 8$  és  $s = 12$ .

Mivel  $\lceil \log_2(8 + 1) \rceil = 4$ , ezért 4 kérdés elég.

Tegyük fel, hogy 3 kérdéssel is lehetne.

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	$s$ nagyobb $s$ kisebb $s$ kisebb
1	2	3	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$	
1	2	3	$x_4$	$x_5$	$x_6$	88	100	
1	2	3	$x_4$	33	45	88	100	

A 3 kérdés után még nem lehet tudni, hogy  $s = x_4$  teljesül-e, hiszen  $x_4$  lehet 12 is vagy pl. 15 is.

# Van-e jobb, mint a bináris keresés?

## Tétel

A bináris keresés optimális, azaz nincs olyan kereső algoritmus, ami csak összehasonlításokat használ egy rendezett tömbben és a lépésszáma kevesebb mint  $\lceil \log_2(n+1) \rceil$ .

## Bizonyítás.

Ha  $E$  így válaszol és egy kérdés előtt volt  $x$  cella, ahol  $s$  lehetett (ennyit nem zárt még ki az algoritmus), akkor a kérdés után marad még legalább  $(x-1)/2$  cella, ahol  $s$  lehet. (A „-1” az  $s_i$  miatt van, az már biztos nem jöhet szóba).

$\implies$  2 kérdés után legalább  $\frac{\frac{n-1}{2}-1}{2} = \frac{n}{2^2} - \frac{1}{2} - \frac{1}{2^2}$  marad.

$\implies$   $k$  kérdés után is marad még  $\frac{n}{2^k} - \frac{1}{2} - \dots - \frac{1}{2^k}$  lehetőség.

Tehát teljesülnie kell  $\frac{n}{2^k} - \frac{1}{2} - \dots - \frac{1}{2^k} \leq 1$ -nek.

Vagyis  $n \leq 2^k + 2^{k-1} + \dots + 1 = 2^{k+1} - 1$ .  $\implies \lceil \log_2(n+1) \rceil - 1 \leq k$ .

Ha még van egy lehetséges elem, akkor még +1 egy kérdés.

# Rendezés

## Feladat

Adott az  $U$  rendezett halmaz véges  $S = \{s_1, s_2, \dots, s_{n-1}, s_n\}$  részhalmaza. Összehasonlításokkal rendezzük az  $S$  elemeit a rendezés szerint növekvő sorrendbe, azaz keressünk olyan  $\sigma$  permutációt, hogy  $s_{\sigma(1)} < s_{\sigma(2)} < \dots < s_{\sigma(n)}$ .

**Input:** tömb, láncolt lista, (vagy bármi)

**Output:** általában, mint az input

**Lépések:** elemek mozgatása, cseréje, összehasonlítása

A rendezés önmagában is előforduló feladat, de előjön, mint hasznos adatstruktúra is. **Rendezett tömbben keresni (pl. telefonkönyv).**

- Hány összehasonlítás kell a legrosszabb esetben?
- Hány összehasonlítás kell átlagos esetben?
- Hány csere kell a legrosszabb esetben?
- Mennyi plusz tárhely szükséges?

**Input:**  $A[1 : n]$  (rendezetlen) tömb

Ha valamely  $i$ -re  $A[i] > A[i + 1]$ , akkor a két cella tartalmát kicseréljük. A tömb elejéről indulva, közben cserélgetve eljutunk a tömb végéig. Ekkor a legnagyobb elem  $A[n]$ -ben van. Ismételjük ezt az  $A[1 : n - 1]$  tömbre, majd az  $A[1 : n - 2]$  tömbre, stb.

**procedure** buborék

*(\* az  $A[1 : n]$  tömböt növekvően (nem csökkenően) rendezi \*)*

**for** ( $j = n - 1, j > 0, j := j - 1$ ) **do**

**for** ( $i = 1, i \leq j, i := i + 1$ ) **do**

        { ha  $A[i + 1] < A[i]$ , akkor cseréljük ki őket. }

**összehasonlítások száma:**  $n - 1 + n - 2 + \dots + 1 = \frac{n(n-1)}{2}$

**cserék száma:**  $\leq \frac{n(n-1)}{2}$

Animáció

Video tánc: Buborék rendezés

# Beszúrásos rendezés

Ha az  $A[1 : k]$  résztömb már rendezett, akkor szúrjuk be a következő elemet,  $A[k + 1]$ -et, **lineáris** vagy **bináris** kereséssel, majd a következőt ebbe, stb.

	lineáris	bináris
összehasonlítás	$\frac{n(n-1)}{2}$	$\sum_{k=1}^{n-1} \lceil \log_2(k+1) \rceil$
mozgatás	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$

# Bináris beszúrásos rendezés lépésszáma

$$K := \lceil \log_2 2 \rceil + \lceil \log_2 3 \rceil + \dots + \lceil \log_2 n \rceil \leq n \lceil \log_2 n \rceil$$

**Jobb becslés:** használjuk fel, hogy  $\lceil \log_2 k \rceil \leq 1 + \log_2 k$

$$K < n - 1 + \log_2 2 + \dots + \log_2 n = n - 1 + \log_2(n!)$$

**Felhasználva a Stirling formulát:**  $n! \sim (n/e)^n \sqrt{2\pi n}$  kapjuk, hogy

$$\log_2 n! \sim n(\log_2 n - \log_2 e) + \frac{1}{2} \log_2 n + \log_2 \sqrt{2\pi} \sim n(\log_2 n - 1,442)$$

Ezért  $K \leq n(\log_2 n - 0,442)$  elég nagy  $n$ -re.

Animáció

Video tánc: Beszúrásos rendezés

## Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha barchoba-ban kellene kitalálni, hogy az elemek melyik sorrendje (permutációja) az igazi sorrend.

Kezdetben  $n!$  lehetséges sorrend jön szóba.

Két elemet összehasonlítva a válasz két részre osztja a sorrendeket.

Ha pl. azt kapjuk, hogy  $x < y$ , akkor az olyan sorrendek, amelyekben  $x$  hátrébb van  $y$ -nál, már nem jönnek szóba.

Ha az ellenség megint úgy válaszol, hogy minél több sorrend maradjon meg, akkor  $k$  kérdés után még szóba jön  $\frac{n!}{2^k}$  sorrend.

Ha  $\frac{n!}{2^k} > 1$ , nem tudjuk megadni a rendezést.  $\implies$

### Tétel

*Nincs olyan összehasonlítás alapú rendező módszer, ami  $n$  elem rendezésekor bármely bemenet esetén kevesebb mint  $\log_2(n!)$  összehasonlítást használ.*

Az előbbi számításból kijön, hogy  $\log_2(n!) \geq n \log_2 n - c \cdot n$

## Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:

$A[1 : k]$  és  $B[1 : l]$  rendezett tömbök  $\rightarrow C[1 : k + l]$  rendezett tömb

Nyilván  $C[1] = \min\{A[1], B[1]\}$ , pl.  $A[1]$ ,

ezt rakjuk át  $C$ -be és töröljük  $A$ -ból.

$C[2] = \min\{A[2], B[1]\}$ , stb.

Ha elfogynak az egyik tömbben/listában stb-ben az elemek, a másik végét odamásoljuk a közös tömb/lista stb, végére.



## Példa

<i>A</i>	<i>B</i>	<i>C</i>
12, 15, 20, 31	13, 16, 18	
15, 20, 31	13, 16, 18	12,
15, 20, 31	16, 18	12, 13
20, 31	16, 18	12, 13, 15
20, 31	18	12, 13, 15, 16
20, 31		12, 13, 15, 16, 18
31		12, 13, 15, 16, 18, 20
		12, 13, 15, 16, 18, 20, 31

**összehasonlítások száma:**  $k + l - 1$ , ahol  $k, l$  a két tömb hossza

# Összefésüléses rendezés

**Alapötlet:** Rendezzük külön a tömb első felét, majd a második felét, végül fésüljük össze.  
Ezt csináljuk rekurzívan.

$$\text{MSORT}(A[1 : n]) := \\ \text{MERGE}(\text{MSORT}(A[1 : \lceil n/2 \rceil]), \text{MSORT}(A[\lceil n/2 \rceil + 1 : n])).$$

Hogy elvarrjuk a rekurzió alját, legyen  $\text{MSORT}(A[i, i])$  az üres utasítás.

# Összehasonlítások száma

Jelöljük  $T(n)$ -el a lépésszámot  $n$  hosszú tömb rendezésekor. Az egyszerűség kedvéért tegyük fel, hogy  $n = 2^k$ .

$$T(n) \leq n - 1 + 2T(n/2),$$

$$T(n) \leq n - 1 + 2(n/2 - 1 + 2T(n/4)) = n - 1 + 2(n/2 - 1) + 4T(n/4).$$

$$T(n) \leq n - 1 + 2(n/2 - 1) + 4(n/4 - 1) + \dots + 2^{k-1}(n/2^{k-1} - 1) \leq n \lceil \log_2 n \rceil.$$

Felhasználva, hogy  $T(1) = 0$ .

Az összefésüléssel rendezés konstans szorzó erejéig optimális.

**Mozgatások száma:**  $2n \lceil \log_2 n \rceil$

**Tárigény:**  $2n$  cella (bonyolultabban megcsinálva elég  $n + konst.$ )

Animáció

Video animáció: Összefésüléssel rendezés

Video animáció: Összefésüléssel rendezés táncban

# Példa összefésüléses rendezésre

2	<sub>3</sub>	8	<sub>2</sub>	7	<sub>4</sub>	5	<sub>1</sub>	6	<sub>6</sub>	4	<sub>5</sub>	1	<sub>7</sub>	3
2		8	<sub>2</sub>	5		7	<sub>1</sub>	4		6	<sub>5</sub>	1		3
2		5		7		8	<sub>1</sub>	1		3		4		6
1		2		3		4		5		6		7		8

# Gyorsrendezés

[C. A. R. Hoare, 1960]

*oszd meg és uralkodj*: véletlen  $s$  elem a tömbből  $\rightarrow$  PARTÍCIÓ( $s$ )  $\rightarrow$

<i>s-nél kisebb elemek</i>	$s$	<i>s-nél nagyobb elemek</i>
----------------------------	-----	-----------------------------

GYORSREND( $A[1 : n]$ )

1. Válasszunk egy véletlen  $s$  elemet az  $A$  tömbből.
2. PARTÍCIÓ( $s$ ); az eredmény legyen az  $A[1 : k - 1]$ ,  $A[k : k]$ ,  $A[k + 1 : n]$  felbontás.
3. GYORSREND( $A[1 : k - 1]$ ); GYORSREND( $A[k + 1 : n]$ ).

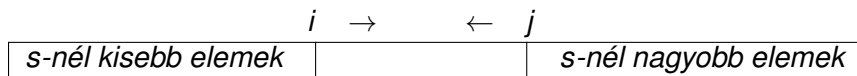
Véletlen elemnek választhatjuk mindig a tömb első helyén állót.

# A PARTÍCIÓ(s) működése

Legyen  $i := 2, j := n$ ,

- $i$ -t növeljük, amíg  $A[i] < s$  teljesül, utána
- $j$ -t csökkentjük, amíg  $A[j] > s$

$\implies$



Ha mindkettő megáll (nem lehet továbblépni), és  $i < j$ , akkor  $A[i] > s$  és  $A[j] < s \implies$

Kicseréljük  $A[i]$  és  $A[j]$  tartalmát, majd az  $i$ -t tovább növeljük, amíg  $A[i] < s$  teljesül, stb. . . . Ha a két mutató összeér (már nem teljesül  $i < j$ ), akkor kicseréljük  $A[1] = s$ -et és  $A[i - 1]$ -et

**PARTÍCIÓ lépésszáma:**  $\Theta(n)$

**GYORSREND lépésszáma legrosszabb esetben:**  $\Theta(n^2)$

**GYORSREND lépésszáma átlagos esetben:**

$1,39n \log_2 n + O(n) = O(n \log n)$

Animáció

# Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.

Pl. ismerjük az elemek számát, belső szerkezetét.

## Ládarendezés (binsort)

Legyen  $U = \{u_1 < u_2 < \dots < u_m\}$  egy  $m$  elemű halmaz, aminek elemei összehasonlíthatóak és tudjuk a nagyság szerinti sorrendjüket. A legegyszerűbb példa az  $\in \{1, \dots, m\}$  halmaz. Tudjuk, hogy  $A[1 : n]$  elemei az  $U$  halmazból kerülnek ki.

$\implies$  Lefoglalunk egy  $U$  elemeivel indexelt  $B$  tömböt ( $m$  db ládát), először mind üres.

**Első fázis:** végigolvassuk az  $A$ -t, és az  $s = A[i]$  elemet a  $B[s]$  lista végére fűzzük.

$\implies$  konzervatív rendezés, azaz az egyenlő elemek sorrendjét megtartja.

## Ládarendezés

**Példa:** Tegyük fel, hogy a rendezendő  $A[1 : 7]$  tömb elemei 0 és 9 közötti egészek:

A : 

5	3	1	5	6	9	6
---	---	---	---	---	---	---

B : 

	1		3		5 5	6 6			9
--	---	--	---	--	-----	-----	--	--	---

**Második fázis:** elejétől a végéig  $U$  elemeinek növekvő sorrendjében végigmegyünk  $B$ -n, és a  $B[i]$  listák tartalmát visszaírjuk  $A$ -ba.

B : 

	1		3		5 5	6 6			9
--	---	--	---	--	-----	-----	--	--	---

A : 

1	3	5	5	6	6	9
---	---	---	---	---	---	---

**Lépésszám:**  $B$  létrehozása  $O(m)$ , első fázis  $O(n)$ , második fázis  $O(n + m)$ , összesen  $O(n + m)$ .

Ez gyorsabb, mint az általános alsó korlát, ha pl.  $m \leq cn$ .

Animáció



## Radix rendezés

A kulcsok összetettek, több komponensből állnak,  $t_1 \dots t_k$  alakú szavak, ahol a  $t_i$  komponens az  $L_i$  rendezett típusból való, legyen  $|L_i| = s_i$ , a rendezés lexicografikus.

**Példa:** Legyen  $(U, <)$  a *huszadik századi dátumok* összessége az időrendnek megfelelő rendezéssel.

$$L_1 = \{1900, 1901, \dots, 1999\}, \quad s_1 = 100.$$

$$L_2 = \{\text{január, február, \dots, december}\}, \quad s_2 = 12.$$

$$L_3 = \{1, 2, \dots, 31\}, \quad s_3 = 31.$$

A dátumok rendezése éppen az  $L_i$  típusokból származó lexicografikus rendezés lesz.

# Radix rendezés

- Rendezzük a sorozatot az utolsó, a  $k$ -edik komponens szerint ládarendezéssel.
- A kapottat rendezzük a  $k - 1$ -edik komponens szerint ládarendezéssel.
- stb.

Fontos, hogy a ládarendezésnél, az elemeket a lédában mindig a lista végére tettük. Így ha két azonos kulcsú elem közül az egyik megelőzi a másikat, akkor a rendezés után sem változik a sorrendjük.

→ Az ilyen rendezést **konzervatív** rendezésnek nevezzük.

## Miért működik a radix jól?

Ha  $X < Y$ , az első  $i - 1$  tag megegyezik, de  $x_i < y_i$ , akkor az  $i$ -edik komponens rendezésekor  $X$  előre kerül.

A láderendezés konzervatív  $\implies$  később már nem változik a sorrendjük.

Példa:

1969.01.18.	1969.01.01.	1955.12.18.	1955.01.18.	1918.12.18.
-------------	-------------	-------------	-------------	-------------

Napok szerint rendezve:

1969.01.01.	1969.01.18.	1955.12.18.	1955.01.18.	1918.12.18.
-------------	-------------	-------------	-------------	-------------

Hónapok szerint rendezve:

1969.01.01.	1969.01.18.	1955.01.18.	1955.12.18.	1918.12.18.
-------------	-------------	-------------	-------------	-------------

Évek szerint rendezve:

1918.12.18.	1955.01.18.	1955.12.18.	1969.01.01.	1969.01.18.
-------------	-------------	-------------	-------------	-------------

# Radix rendezés

**Lépésszám:**  $k$  ládarendezés összköltsége:  $O(kn + \sum_{i=1}^k s_i)$

Ez lehet gyorsabb az általános korlátnál

- $c, k$  állandók és  $s_i \leq cn$   
 $\implies O(kn + \sum_{i=1}^k cn) = O(k(c+1)n) = O(n)$ .  
pl. az  $[1, n^{10} - 1]$  intervallumból való egészek rendezése
- $k = \log n, s_i = 2 \implies O(n \log n + 2 \log n) = O(n \log n)$ .

Animáció