Algoritmuselmélet

Keresés, minimumkeresés

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

5. előadás

Rendezési reláció

Legyen U egy halmaz, és $<$ egy kétváltozós reláció U-n. Ha $a, b \in U$ és $a < b$, akkor azt mondjuk, hogy a kisebb, mint b.

Rendezési reláció

Legyen U egy halmaz, és $<$ egy kétváltozós reláció U-n. Ha $a, b \in U$ és $a < b$, akkor azt mondjuk, hogy a kisebb, mint b.

A $<$ reláció egy rendezés, ha teljesülnek a következők:
1. $a \not< a$ minden $a \in U$ elemre ($<$ irreflexív);
2. Ha $a, b, c \in U$, $a < b$, és $b < c$, akkor $a < c$ ($<$ tranzitív);
3. Tetszőleges $a \neq b \in U$ elemekre vagy $a < b$, vagy $b < a$ fennáll ($<$ teljes).

Rendezési reláció

Legyen U egy halmaz, és $<$ egy kétváltozós reláció U-n. Ha $a, b \in U$ és $a < b$, akkor azt mondjuk, hogy a kisebb, mint b.

A $<$ reláció egy rendezés, ha teljesülnek a következők:
1. $a \not< a$ minden $a \in U$ elemre ($<$ irreflexív);
2. Ha $a, b, c \in U$, $a < b$, és $b < c$, akkor $a < c$ ($<$ tranzitív);
3. Tetszőleges $a \neq b \in U$ elemekre vagy $a < b$, vagy $b < a$ fennáll ($<$ teljes).

Ha $<$ egy rendezés U-n, akkor az $(U, <)$ párt rendezett halmaznak nevezzük.

Rendezési reláció

Példák:

- $\mathbb{Z}$ az egész számok halmaza. A $<$ rendezés a nagyság szerinti rendezés.

Rendezési reláció

Példák:

- $\mathbb{Z}$ az egész számok halmaza. A $<$ rendezés a nagyság szerinti rendezés.
- Az *abc* betűinek Σ halmaza; a $<$ rendezést az *abc-sorrend* adja. Az *x* betű kisebb, mint az *y* betű, ha *x* előbb szerepel az *abc*-sorrendben, mint *y*.

Rendezési reláció

Példák:

- $\mathbb{Z}$ az egész számok halmaza. A $<$ rendezés a nagyság szerinti rendezés.
- Az *abc* betűinek Σ 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 Σ betűiből alkotott szavak Σ^* 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 $i = 1, 2, \ldots, k$ 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 rendezetlen halmazban

Feladat

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

Keresés rendezetlen halmazban

Feladat

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

Itt összehasonlítás: Igaz-e, hogy $s_i = s$?

Keresés rendezetlen halmazban

Feladat

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

Itt összehasonlítás: Igaz-e, hogy $s_i = s$?
Válasz: Igen vagy nem.

Keresés rendezetlen halmazban

Feladat

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

Itt összehasonlítás: Igaz-e, hogy $s_i = s$?
Válasz: Igen vagy nem.

Legrosszabb esetben minden elemet végig kell nézni

Keresés rendezetlen halmazban

Feladat

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

Itt összehasonlítás: Igaz-e, hogy $s_i = s$?
Válasz: Igen vagy nem.

Legrosszabb esetben minden elemet végig kell nézni $\Longrightarrow$
n összehasonlítás kell legrosszabb esetben.

Keresés rendezetlen halmazban

Feladat

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

Itt összehasonlítás: Igaz-e, hogy $s_i = s$?
Válasz: Igen vagy nem.

Legrosszabb esetben minden elemet végig kell nézni $\Longrightarrow$
n összehasonlítás kell legrosszabb esetben.
$n/2$ összehasonlítás kell átlagosan.

Keresés rendezett halmazban

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

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

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

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

Itt összehasonlítás: Mi a viszonya s-nek és s_i-nek?
Válasz: $s_i = s$ vagy $s_i < s$ vagy $s_i > s$.

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

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.

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

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.

Keresés rendezett halmazban

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

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1 < s_2 < \ldots < s_{n-1} < s_n\}$ 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?

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.
Költség átlagos esetben esetben: $(n/2) + 1$.

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.
És így tovább:

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

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.
És így tovább:

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

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

$$15, 22, 25, 37, \boxed{48}, 56, 70, 82 \tag{1}$$

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.
És így tovább:

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

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

$$15, 22, 25, 37, \boxed{48}, 56, 70, 82 \quad (1)$$

$$15, 22, \boxed{25}, 37, 48, 56, 70, 82 \quad (2)$$

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.
És így tovább:

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

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

$$15, 22, 25, 37, \boxed{48}, 56, 70, 82 \quad (1)$$

$$15, 22, \boxed{25}, 37, 48, 56, 70, 82 \quad (2)$$

$$15, \boxed{22}, 25, 37, 48, 56, 70, 82 \quad (3)$$

Bináris keresés

Oszd meg és uralkodj: először a középső s_i-vel hasonlítunk.
Hasonló feladatot kapunk egy S_1 halmazra, amire viszont $|S_1| \leq |S|/2$.
És így tovább:

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

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

$$15, 22, 25, 37, \boxed{48}, 56, 70, 82 \quad (1)$$

$$15, 22, \boxed{25}, 37, 48, 56, 70, 82 \quad (2)$$

$$15, \boxed{22}, 25, 37, 48, 56, 70, 82 \quad (3)$$

$$\boxed{15}, 22, 25, 37, 48, 56, 70, 82 \quad (4)$$

Bináris keresés

Addig kell csinálni, amíg $|S_k| = 1$ lesz. Innen $1 = |S_k| \leq \frac{n}{2^k}$.

Bináris keresés

Addig kell csinálni, amíg $|S_k| = 1$ lesz. Innen $1 = |S_k| \leq \frac{n}{2^k}$.
$\Longrightarrow 2^k \leq n \Longrightarrow k \leq \lfloor \log_2 n \rfloor$

Bináris keresés

Addig kell csinálni, amíg $|S_k| = 1$ lesz. Innen $1 = |S_k| \leq \frac{n}{2^k}$.
$\Longrightarrow 2^k \leq n \Longrightarrow k \leq \lfloor \log_2 n \rfloor$
Ez $k+1$ összehasonlítás volt. $\Longrightarrow k+1 \leq \lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n+1) \rceil$

Bináris keresés

Addig kell csinálni, amíg $|S_k| = 1$ lesz. Innen $1 = |S_k| \leq \frac{n}{2^k}$.
$\Longrightarrow 2^k \leq n \Longrightarrow k \leq \lfloor \log_2 n \rfloor$
Ez $k+1$ összehasonlítás volt. $\Longrightarrow k+1 \leq \lfloor \log_2 n \rfloor + 1 = \lceil \log_2(n+1) \rceil$

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni.

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).
Az ellenség úgy válaszol, hogy minél több lehetőség maradjon, így el tudja érni, hogy legalább $\frac{n-1}{2}$ marad.

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).
Az ellenség úgy válaszol, hogy minél több lehetőség maradjon, így el tudja érni, hogy legalább $\frac{n-1}{2}$ marad.

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

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).
Az ellenség úgy válaszol, hogy minél több lehetőség maradjon, így el tudja érni, hogy legalább $\frac{n-1}{2}$ marad.

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

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

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).
Az ellenség úgy válaszol, hogy minél több lehetőség maradjon, így el tudja érni, hogy legalább $\frac{n-1}{2}$ marad.

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

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

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

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).
Az ellenség úgy válaszol, hogy minél több lehetőség maradjon, így el tudja érni, hogy legalább $\frac{n-1}{2}$ marad.

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

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

Tehát teljesülnie kell $\frac{n}{2^k} - \frac{1}{2} - \cdots - \frac{1}{2^k} \leq 1$-nek.
Vagyis $n \leq 2^k + 2^{k-1} + \ldots + 1 = 2^{k+1} - 1$.

Bináris keresés

Tétel

Ez optimális, nincs olyan kereső algoritmus, ami minden esetben kevesebb mint $\lceil \log_2(n+1) \rceil$ *kérdést használ.*

Bizonyítás.

Az ellenség nem is gondol egy számra, csak mindig úgy válaszol, hogy minél többet kelljen kérdezni. Ha egy kérdést felteszek, és az igen válasz után mondjuk szóba jön x lehetőség, akkor a nem esetén szóba jön még $n - x - 1$ lehetőség. (A „-1” az $s = s_i$ válasz miatt van).
Az ellenség úgy válaszol, hogy minél több lehetőség maradjon, így el tudja érni, hogy legalább $\frac{n-1}{2}$ marad.

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

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

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

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

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

Minimumkeresés

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$ részhalmaza.
Összehasonlításokkal keressük meg az S minimális elemét, azaz egy olyan s_i elemet, hogy minden $i \neq j$ esetén $s_i < s_j$.

Minimumkeresés

Feladat

Adott az $(U, <)$ rendezett halmaz véges $S = \{s_1, s_2, \ldots, s_{n-1}, s_n\}$ részhalmaza.
Összehasonlításokkal keressük meg az S minimális elemét, azaz egy olyan s_i elemet, hogy minden $i \neq j$ esetén $s_i < s_j$.

- Hány összehasonlítás kell a legrosszabb esetben?

Minimumkeresés

Tétel

n elem közül a minimális kiválasztásához legrosszabb esetben $n - 1$ összehasonlítás kell.

Minimumkeresés

Tétel

n elem közül a minimális kiválasztásához legrosszabb esetben $n - 1$ összehasonlítás kell.

Bizonyítás.

$n - 1$ összehasonlítás mindig elég: Rendezzünk kiesés versenyt, mindig a kisebb elemet megtartva egy-egy összehasonlítás után.

Minimumkeresés

Tétel

n elem közül a minimális kiválasztásához legrosszabb esetben $n - 1$ összehasonlítás kell.

Bizonyítás.

$n - 1$ összehasonlítás mindig elég: Rendezzünk kiesés versenyt, mindig a kisebb elemet megtartva egy-egy összehasonlítás után. Mivel „mindenki pontosan egyszer kap ki a győztest kivéve”, ez $n - 1$ összehasonlítást igényel.

Minimumkeresés

Tétel

n elem közül a minimális kiválasztásához legrosszabb esetben $n - 1$ összehasonlítás kell.

Bizonyítás.

$n - 1$ összehasonlítás mindig elég: Rendezzünk kiesés versenyt, mindig a kisebb elemet megtartva egy-egy összehasonlítás után. Mivel „mindenki pontosan egyszer kap ki a győztest kivéve”, ez $n - 1$ összehasonlítást igényel.

$n - 1$ összehasonlításnál kevesebb nem mindig elég: Legyenek az elemek egy gráf pontjai, ha kettőt összehasonlítottunk, húzzunk közöttük élet.

Minimumkeresés

Tétel

n elem közül a minimális kiválasztásához legrosszabb esetben $n - 1$ összehasonlítás kell.

Bizonyítás.

$n - 1$ összehasonlítás mindig elég: Rendezzünk kiesés versenyt, mindig a kisebb elemet megtartva egy-egy összehasonlítás után. Mivel „mindenki pontosan egyszer kap ki a győztest kivéve”, ez $n - 1$ összehasonlítást igényel.

$n - 1$ összehasonlításnál kevesebb nem mindig elég: Legyenek az elemek egy gráf pontjai, ha kettőt összehasonlítottunk, húzzunk közöttük élet. Amíg a gráf nem összefüggő, bármely komponensében lehet a minimális elem.

Minimumkeresés

Tétel

n elem közül a minimális kiválasztásához legrosszabb esetben $n-1$ összehasonlítás kell.

Bizonyítás.

$n-1$ összehasonlítás mindig elég: Rendezzünk kiesés versenyt, mindig a kisebb elemet megtartva egy-egy összehasonlítás után. Mivel „mindenki pontosan egyszer kap ki a győztest kivéve”, ez $n-1$ összehasonlítást igényel.

$n-1$ összehasonlításnál kevesebb nem mindig elég: Legyenek az elemek egy gráf pontjai, ha kettőt összehasonlítottunk, húzzunk közöttük élet. Amíg a gráf nem összefüggő, bármely komponensében lehet a minimális elem.

Ha a gráf már összefüggő, akkor legalább $n-1$ éle van, tehát kell ennyi összehasonlítás. □

Algoritmuselmélet

Rendezés, buborék, beszúrásos, összefésüléses, kupacos, láda, radix

Katona Gyula Y.

Számítástudományi és Információelméleti Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

6. előadás

Rendezés

Feladat

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

Rendezés

Feladat

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

Input: tömb, láncolt lista, (vagy bármi)

Rendezés

Feladat

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

Input: tömb, láncolt lista, (vagy bármi)
Output: általában, mint az input

Rendezés

Feladat

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

Input: tömb, láncolt lista, (vagy bármi)
Output: általában, mint az input
Lépések: elemek mozgatása, cseréje, összehasonlítása

Rendezés

Feladat

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

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.

Rendezés

Feladat

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

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 halmazban könnyebb keresni (pl. telefonkönyv).

- Hány összehasonlítás kell a legrosszabb esetben?

Rendezés

Feladat

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

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 halmazban könnyebb keresni (pl. telefonkönyv).

- Hány összehasonlítás kell a legrosszabb esetben?
- Hány összehasonlítás kell átlagos esetben?

Rendezés

Feladat

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

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 halmazban könnyebb keresni (pl. telefonkönyv).

- Hány összehasonlítás kell a legrosszabb esetben?
- Hány összehasonlítás kell átlagos esetben?
- Hány csere kell a legrosszabb esetben?

Rendezés

Feladat

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

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 halmazban könnyebb keresni (pl. telefonkönyv).

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

Buborék-rendezés

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.

Buborék-rendezés

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.

Buborék-rendezés

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.

Buborék-rendezés

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: $n-1+n-2+\ldots+1 = \frac{n(n-1)}{2}$

cserék száma: $\leq \frac{n(n-1)}{2}$

Buborék-rendezés

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: $n-1+n-2+\ldots+1 = \frac{n(n-1)}{2}$

cserék száma: $\leq \frac{n(n-1)}{2}$

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.

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

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	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$

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	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$
átlagos összehasonlítás	$\frac{n(n-1)}{4}$	$\sum_{k=1}^{n-1} \lceil \log_2(n+1) \rceil$

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	$\frac{(n+2)(n-1)}{2}$	$\frac{(n+2)(n-1)}{2}$
átlagos összehasonlítás	$\frac{n(n-1)}{4}$	$\sum_{k=1}^{n-1} \lceil \log_2(n+1) \rceil$
átlagos mozgatás	$\frac{n^2}{4}$	$\frac{n^2}{4}$

Bináris beszúrásos rendezés lépésszáma

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

Bináris beszúrásos rendezés lépésszáma

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

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

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

Bináris beszúrásos rendezés lépésszáma

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

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

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

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

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

Bináris beszúrásos rendezés lépésszáma

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

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

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

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.

Alsó becslés összehasonlítás alapú rendezésre

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

Alsó becslés összehasonlítás alapú rendezésre

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

Kezdetben $n!$ lehetséges sorrend jön szóba.

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha barochba-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.

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha barochba-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.

Alsó becslés összehasonlítás alapú rendezésre

Ugyanaz, mintha barochba-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. $\Longrightarrow$

Tétel

Minden összehasonlítás alapú rendező módszer n elem rendezésekor legalább $\log_2(n!)$ *összehasonlítást használ.*

Összefésüléses rendezés

Összefésülés (MERGE):

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

Összefésüléses rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:
$A[1 : k]$ és $B[1 : l]$ rendezett tömbök $\longrightarrow$ $C[1 : k + l]$ rendezett tömb

Összefésüléses rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:
$A[1:k]$ és $B[1:l]$ rendezett tömbök $\longrightarrow$ $C[1:k+l]$ rendezett tömb
Nyilván $C[1] = \min\{A[1], B[1]\}$, pl. $A[1]$,

Összefésüléses rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:
$A[1:k]$ és $B[1:l]$ rendezett tömbök $\longrightarrow$ $C[1:k+l]$ rendezett tömb
Nyilván $C[1] = \min\{A[1], B[1]\}$, pl. $A[1]$,
ezt rakjuk át C-be és töröljük A-ból.

Összefésüléses rendezés

Összefésülés (MERGE):

Két már rendezett sorozat (tömb, lista, stb.) tartalmának egy sorozatba való rendezése:
$A[1:k]$ és $B[1:l]$ rendezett tömbök $\longrightarrow$ $C[1:k+l]$ 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.

Példa

A	B	C
$12, 15, 20, 31$	$13, 16, 18$	
$15, 20, 31$	$13, 16, 18$	$12,$

Példa

A	B	C
$12, 15, 20, 31$	$13, 16, 18$	
$15, 20, 31$	$13, 16, 18$	$12,$
$15, 20, 31$	$16, 18$	$12, 13$

Példa

A	B	C
$12, 15, 20, 31$	$13, 16, 18$	
$15, 20, 31$	$13, 16, 18$	$12,$
$15, 20, 31$	$16, 18$	$12, 13$
$20, 31$	$16, 18$	$12, 13, 15$

Példa

A	B	C
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

Példa

A	B	C
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

Példa

A	B	C
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

Példa

A	B	C
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.

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 $n = 2^k$.

Összehasonlítások száma

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

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

Összehasonlítások száma

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

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

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

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.
Az összefésüléses rendezés konstans szorzó erejéig optimális.

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.
Az összefésüléses rendezés konstans szorzó erejéig optimális.
Mozgatások száma: $2n\lceil \log_2 n \rceil$

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.
Az összefésüléses rendezés konstans szorzó erejéig optimális.
Mozgatások száma: $2n\lceil \log_2 n \rceil$
Tárigény: $2n$ cella (bonyolultabban megcsinálva elég $n + konst.$)

Összehasonlítások száma

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

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

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

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

Felhasználva, hogy $T(1) = 0$.
Az összefésüléses rendezés konstans szorzó erejéig optimális.
Mozgatások száma: $2n\lceil \log_2 n \rceil$
Tárigény: $2n$ cella (bonyolultabban megcsinálva elég $n + konst.$)

Példa összefésüléses rendezésre

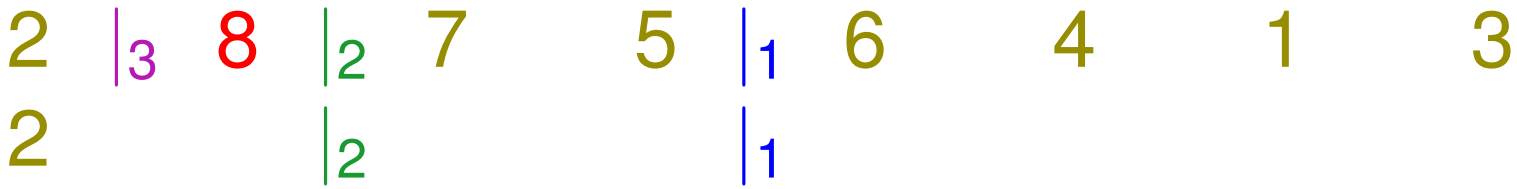
2 8 7 5 6 4 1 3

Példa összefésüléses rendezésre

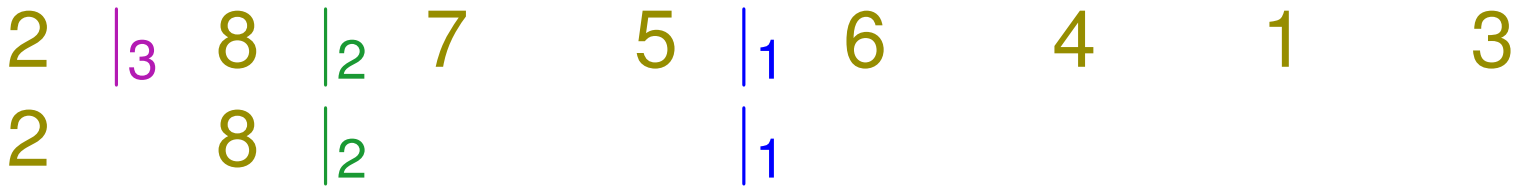
Példa összefésüléses rendezésre

Példa összefésüléses rendezésre

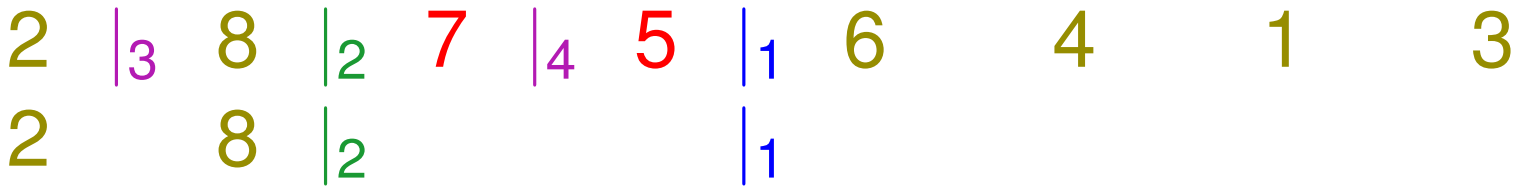
Példa összefésüléses rendezésre



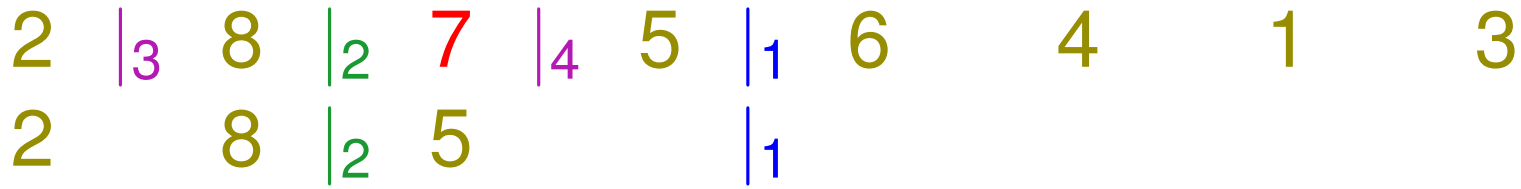
Példa összefésüléses rendezésre



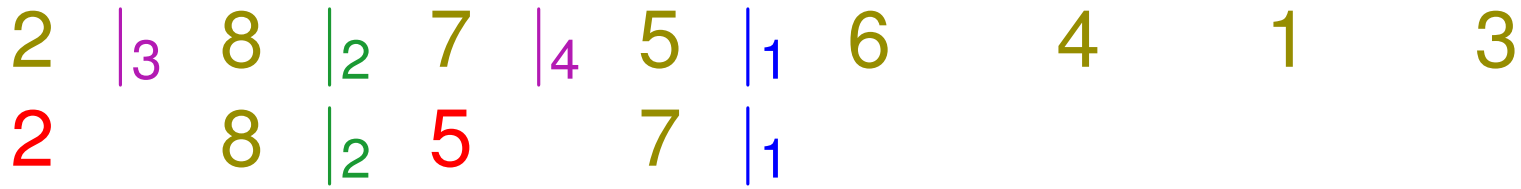
Példa összefésüléses rendezésre



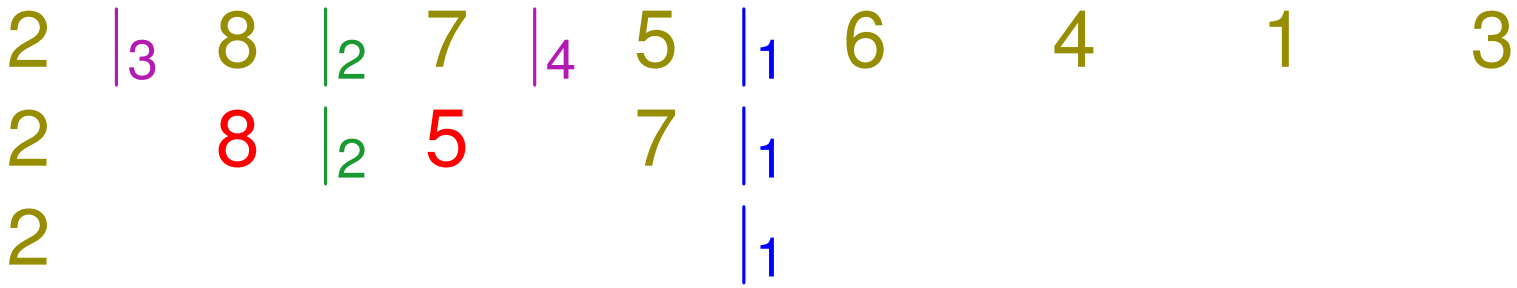
Példa összefésüléses rendezésre



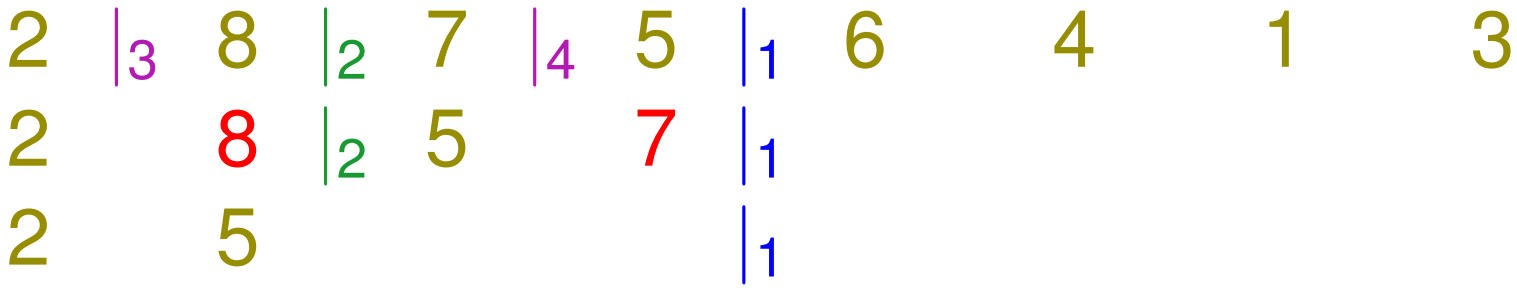
Példa összefésüléses rendezésre



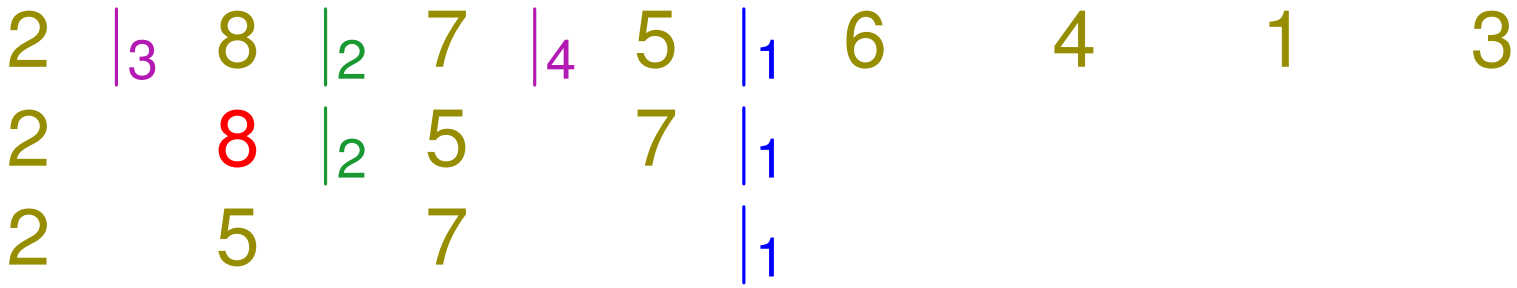
Példa összefésüléses rendezésre



Példa összefésüléses rendezésre



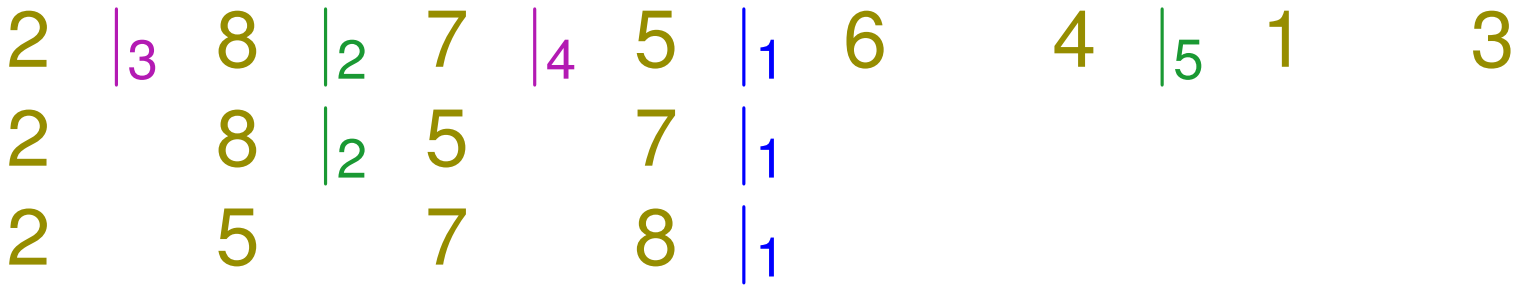
Példa összefésüléses rendezésre



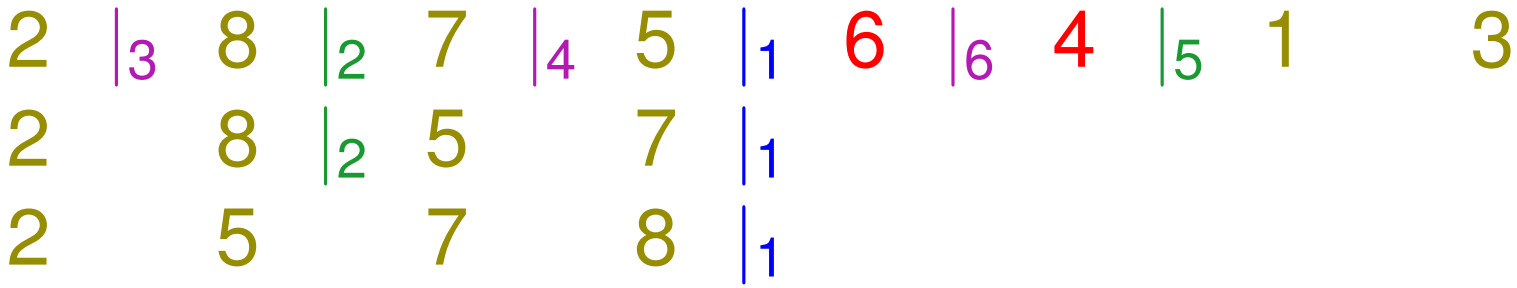
Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6		4		1		3
2		8	$\vert_2$	5		7	$\vert_1$							
2		5		7		8	$\vert_1$							

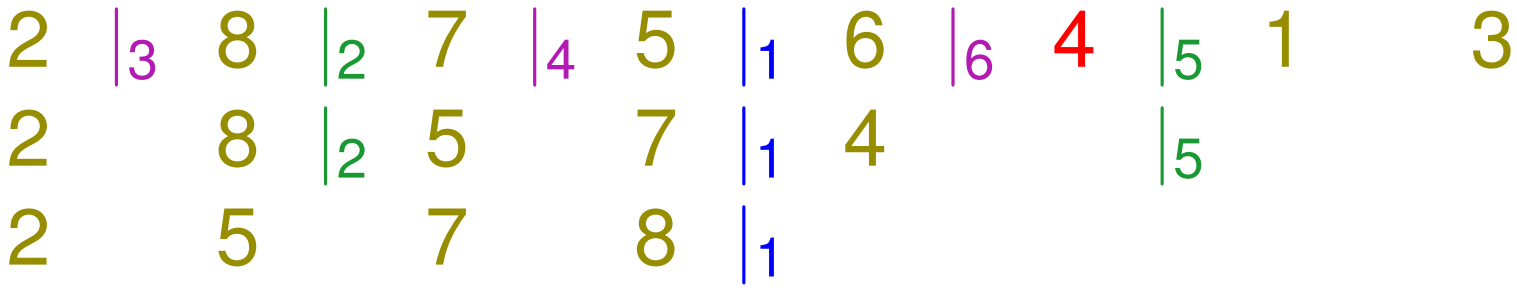
Példa összefésüléses rendezésre



Példa összefésüléses rendezésre



Példa összefésüléses rendezésre



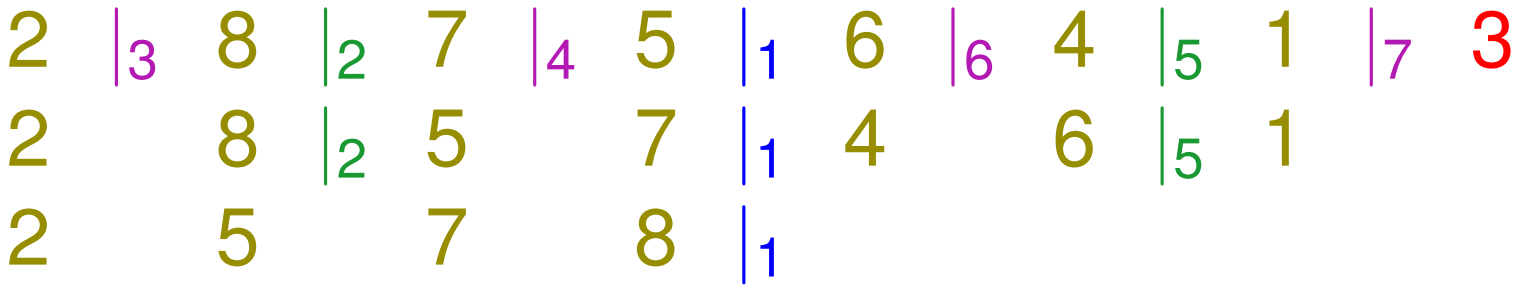
Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$		
2		5		7		8	$\vert_1$						

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$		
2		5		7		8	$\vert_1$						

Példa összefésüléses rendezésre



Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1	3
2		5		7		8	$\vert_1$						

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1	3
2		5		7		8	$\vert_1$	1					

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3				

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1	3
2		5		7		8	$\vert_1$	1		3		4	

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1	3
2		5		7		8	$\vert_1$	1		3		4	6

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1	3
2		5		7		8	$\vert_1$	1		3		4	6
1													

Példa összefésüléses rendezésre

2	$\|_3$	8	$\|_2$	7	$\|_4$	5	$\|_1$	6	$\|_6$	4	$\|_5$	1	$\|_7$	3
2		8	$\|_2$	5		7	$\|_1$	4		6	$\|_5$	1		3
2		5		7		8	$\|_1$	1		3		4		6
1		2												

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3		4		6
1		2		3										

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3		4		6
1		2		3		4								

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3		4		6
1		2		3		4		5						

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3		4		6
1		2		3		4		5		6				

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$	3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1		3
2		5		7		8	$\vert_1$	1		3		4		6
1		2		3		4		5		6		7		

Példa összefésüléses rendezésre

2	$\vert_3$	8	$\vert_2$	7	$\vert_4$	5	$\vert_1$	6	$\vert_6$	4	$\vert_5$	1	$\vert_7$ 3
2		8	$\vert_2$	5		7	$\vert_1$	4		6	$\vert_5$	1	3
2		5		7		8	$\vert_1$	1		3		4	6
1		2		3		4		5		6		7	8

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.
[J. W. J. Williams és R. W. Floyd, 1964]

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.
[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.
[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

Legjobb ismert rendező algoritmus.
Pontos implementációval:
$2n\lfloor \log_2 n \rfloor + 3n$ (összehasonlítások száma) és $n\lfloor \log_2 n \rfloor + 2,5n$ (cserék száma).

A kupacos rendezés

Először kupacot építünk, utána n darab MINTÖR adja nem csökkenő sorrendben az elemeket.
[J. W. J. Williams és R. W. Floyd, 1964]

Költség: $O(n) + O(n \log_2 n) = O(n \log_2 n)$

Legjobb ismert rendező algoritmus.
Pontos implementációval:
$2n\lfloor \log_2 n \rfloor + 3n$ (összehasonlítások száma) és $n\lfloor \log_2 n \rfloor + 2,5n$ (cserék száma).

Gyorsrendezés

[C. A. R. Hoare, 1960]

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

s-nél kisebb elemek	s	$\ldots$	s	*s-nél nagyobb elemek*

Gyorsrendezés

[C. A. R. Hoare, 1960]
oszd meg és uralkodj: véletlen s elem a tömbből $\longrightarrow$ PARTÍCIÓ(s) $\longrightarrow$

s-nél kisebb elemek	s	$\ldots$	s	*s-nél nagyobb elemek*

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

Véletlen elemnek választhatjuk mindig a tömb első helyén állót.

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

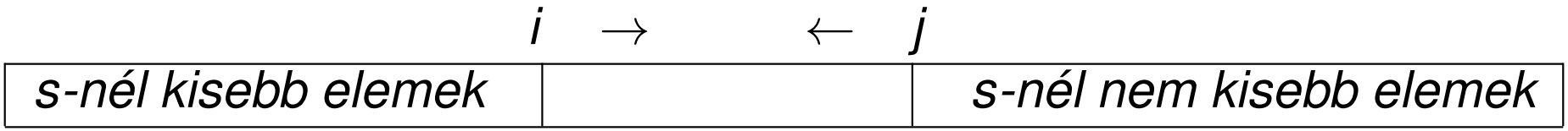
- i-t növeljük, amíg $A[i] < s$ teljesül

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

- i-t növeljük, amíg $A[i] < s$ teljesül
- j-t csökkentjük, amíg $A[j] \geq s$

$\Longrightarrow$

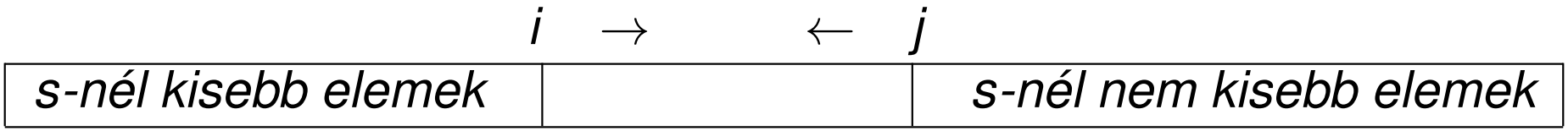


A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

- i-t növeljük, amíg $A[i] < s$ teljesül
- j-t csökkentjük, amíg $A[j] \geq s$

$\Longrightarrow$



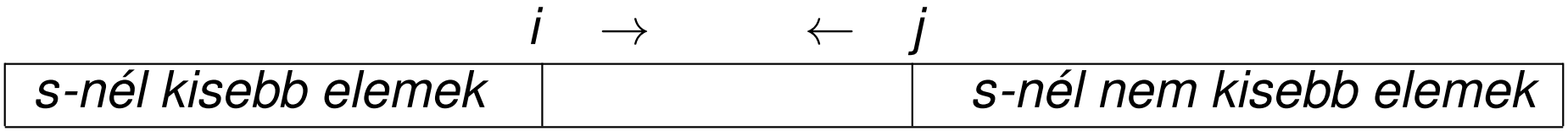
Ha mindkettő megáll (nem lehet továbblépni), és $i < j$, akkor $A[i] \geq s$ és $A[j] < s$

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

- i-t növeljük, amíg $A[i] < s$ teljesül
- j-t csökkentjük, amíg $A[j] \geq s$

$\Longrightarrow$



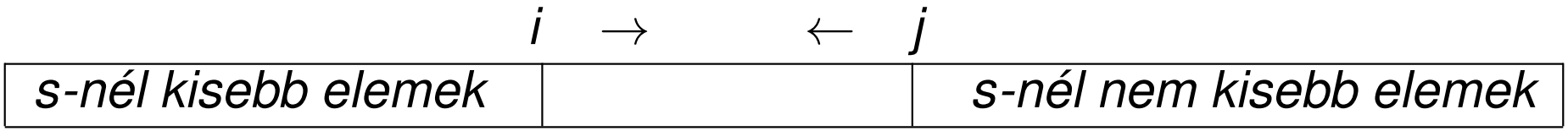
Ha mindkettő megáll (nem lehet továbblépni), és $i < j$, akkor $A[i] \geq s$ és $A[j] < s \Longrightarrow$
Kicseréljük $A[i]$ és $A[j]$ tartalmát, majd $i := i + 1$ és $j := j - 1$. Ha a két mutató összeér (már nem teljesül $i < j$), akkor s előfordulásait a felső rész elejére mozgatjuk.

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

- i-t növeljük, amíg $A[i] < s$ teljesül
- j-t csökkentjük, amíg $A[j] \geq s$

$\Longrightarrow$



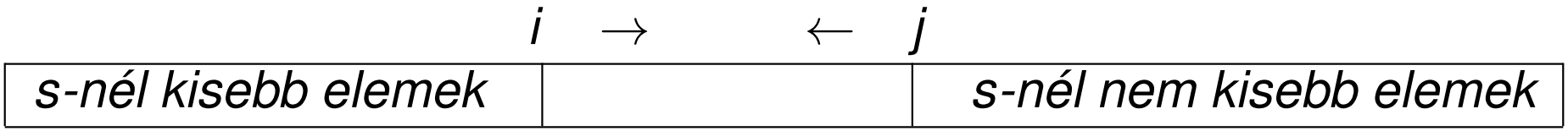
Ha mindkettő megáll (nem lehet továbblépni), és $i < j$, akkor $A[i] \geq s$ és $A[j] < s \Longrightarrow$
Kicseréljük $A[i]$ és $A[j]$ tartalmát, majd $i := i + 1$ és $j := j - 1$. Ha a két mutató összeér (már nem teljesül $i < j$), akkor s előfordulásait a felső rész elejére mozgatjuk.
PARTÍCIÓ lépésszáma: $O(n)$
GYORSREND lépésszáma *legrosszabb esetben*: $O(n^2)$
GYORSREND lépésszáma *átlagos esetben*:
$1,39n \log_2 n + O(n) = O(n \log n)$

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,

- i-t növeljük, amíg $A[i] < s$ teljesül
- j-t csökkentjük, amíg $A[j] \geq s$

$\Longrightarrow$



Ha mindkettő megáll (nem lehet továbblépni), és $i < j$, akkor $A[i] \geq s$ és $A[j] < s \Longrightarrow$
Kicseréljük $A[i]$ és $A[j]$ tartalmát, majd $i := i + 1$ és $j := j - 1$. Ha a két mutató összeér (már nem teljesül $i < j$), akkor s előfordulásait a felső rész elejére mozgatjuk.
PARTÍCIÓ lépésszáma: $O(n)$
GYORSREND lépésszáma *legrosszabb esetben*: $O(n^2)$
GYORSREND lépésszáma *átlagos esetben*:
$1,39n \log_2 n + O(n) = O(n \log n)$

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.

Pl. ismerjük az elemek számát, belső szerkezetét.

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.

Pl. ismerjük az elemek számát, belső szerkezetét.

Ládarendezés (binsort)

Tudjuk, hogy $A[1 : n]$ elemei egy m elemű U halmazból kerülnek ki, pl. $\in \{1, \ldots, m\}$

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.

PI. ismerjük az elemek számát, belső szerkezetét.

Ládarendezés (binsort)

Tudjuk, hogy $A[1:n]$ elemei egy m elemű U halmazból kerülnek ki, pl. $\in \{1, \ldots, m\}$

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

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.
Pl. ismerjük az elemek számát, belső szerkezetét.

Ládarendezés (binsort)

Tudjuk, hogy $A[1:n]$ elemei egy m elemű U halmazból kerülnek ki, pl. $\in \{1, \ldots, m\}$
$\Longrightarrow$ 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.

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.
Pl. ismerjük az elemek számát, belső szerkezetét.

Ládarendezés (binsort)

Tudjuk, hogy $A[1:n]$ elemei egy m elemű U halmazból kerülnek ki, pl. $\in \{1, \ldots, m\}$
$\Longrightarrow$ 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.

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

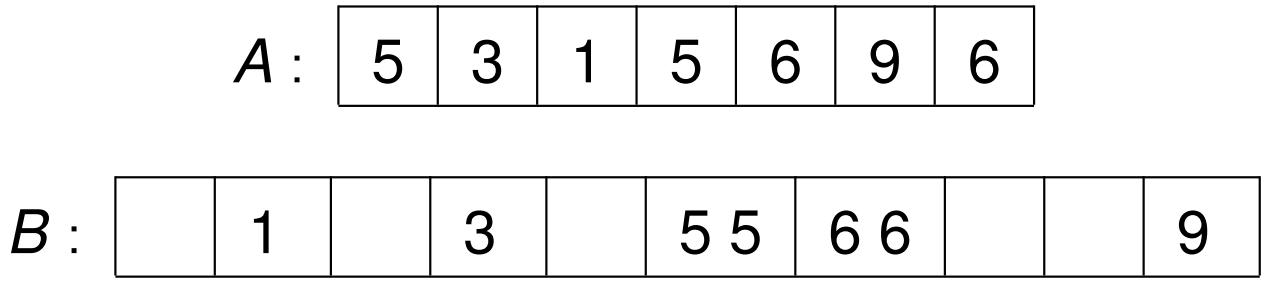
Ládarendezés

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

A :	5	3	1	5	6	9	6

Ládarendezés

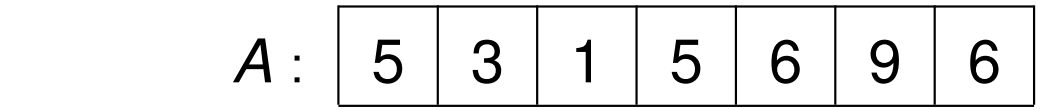
Példa: Tegyük fel, hogy a rendezendő $A[1:7]$ tömb elemei 0 és 9 közötti egészek:

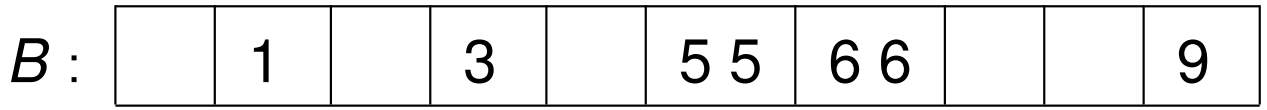


Második fázis: elejétől a végéig növő sorrendben végigmegyünk B-n, és a $B[i]$ listák tartalmát visszaírjuk A-ba.

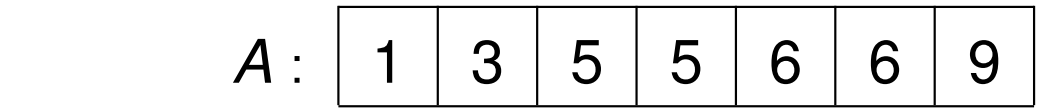
Ládarendezés

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



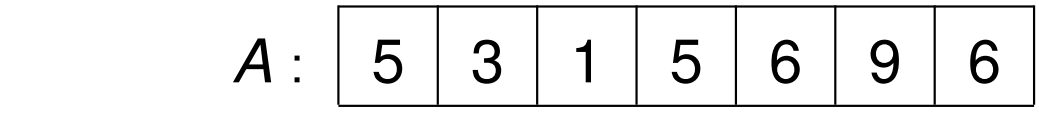


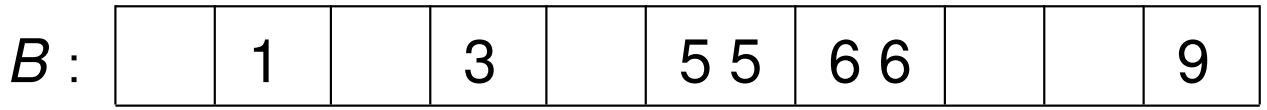
Második fázis: elejétől a végéig növő sorrendben végigmegyünk B-n, és a $B[i]$ listák tartalmát visszaírjuk A-ba.



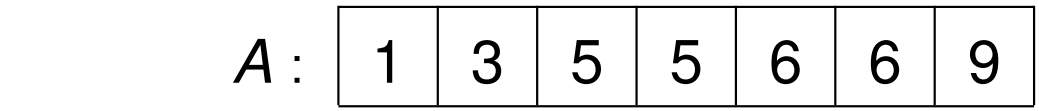
Ládarendezés

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





Második fázis: elejétől a végéig növő sorrendben végigmegyünk B-n, és a $B[i]$ listák tartalmát visszaírjuk A-ba.



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

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 növő sorrendben 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. $m \leq cn$.

Radix rendezés

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

Radix rendezés

A kulcsok összetettek, több komponensből állnak, $t_1 \ldots t_k$ alakú szavak, ahol a t_i komponens az L_i rendezett típusból való, legyen $|L_i| = s_i$, a rendezés lexikografikus.
Példa: Legyen $(U, <)$ a *huszadik századi dátumok* összessége az időrendnek megfelelő rendezéssel.

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

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

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

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

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

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

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

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

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

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

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

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

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

A dátumok rendezése éppen az L_i típusokból származó lexikografikus rendezés lesz.

Radix rendezés

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

Radix rendezés

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

Fontos, hogy a ládarendezésnél, az elemeket a ládában 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.

Radix rendezés

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

Fontos, hogy a ládarendezésnél, az elemeket a ládában 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.
$\longrightarrow$ 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.

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 $\Longrightarrow$ később már nem változik a sorrendjük.

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 $\Longrightarrow$ később már nem változik a sorrendjük.

Példa:

1969.01.18.	1969.01.01.	1955.12.18.	1955.01.18.	1918.12.18.

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 $\Longrightarrow$ 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.

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 $\Longrightarrow$ 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.

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 $\Longrightarrow$ 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: $O(kn + \sum_{i=1}^{k} s_i)$

Radix rendezés

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

Ez lehet gyorsabb az általános korlátnál

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

Radix rendezés

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

Ez lehet gyorsabb az általános korlátnál

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

Radix rendezés

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

Ez lehet gyorsabb az általános korlátnál

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