Rendezések 3.

11. előadás

Batcher-féle rendezés

Batcher-féle rendezés

- Batcher-féle páros-páratlan összefésüléses rendezés
- Jelentősége: ez a MergeSort olyan változata, amelyben a lépések jelentős része párhuzamosan végezhető el.
  - Ha párhuzamos processzorokon hajtanánk végre, akkor azt tapasztalnánk, hogy $\Theta(\log_2 n)^2)$ idő alatt futna le, szemben a MergeSort $\Theta(n * \log_2 n)$ idejével

Összefuttatásos rendezés

- MergeSort(S)

Length(S) > 1	
Szétvág(S, S_1, S_2)	SKIP
MergeSort(S_1) MergeSort(S_2)	
Összefuttat(S_1, S_2, S)	

Összefuttatásos rendezés – tömbre

- MergeSort(A, k, v)

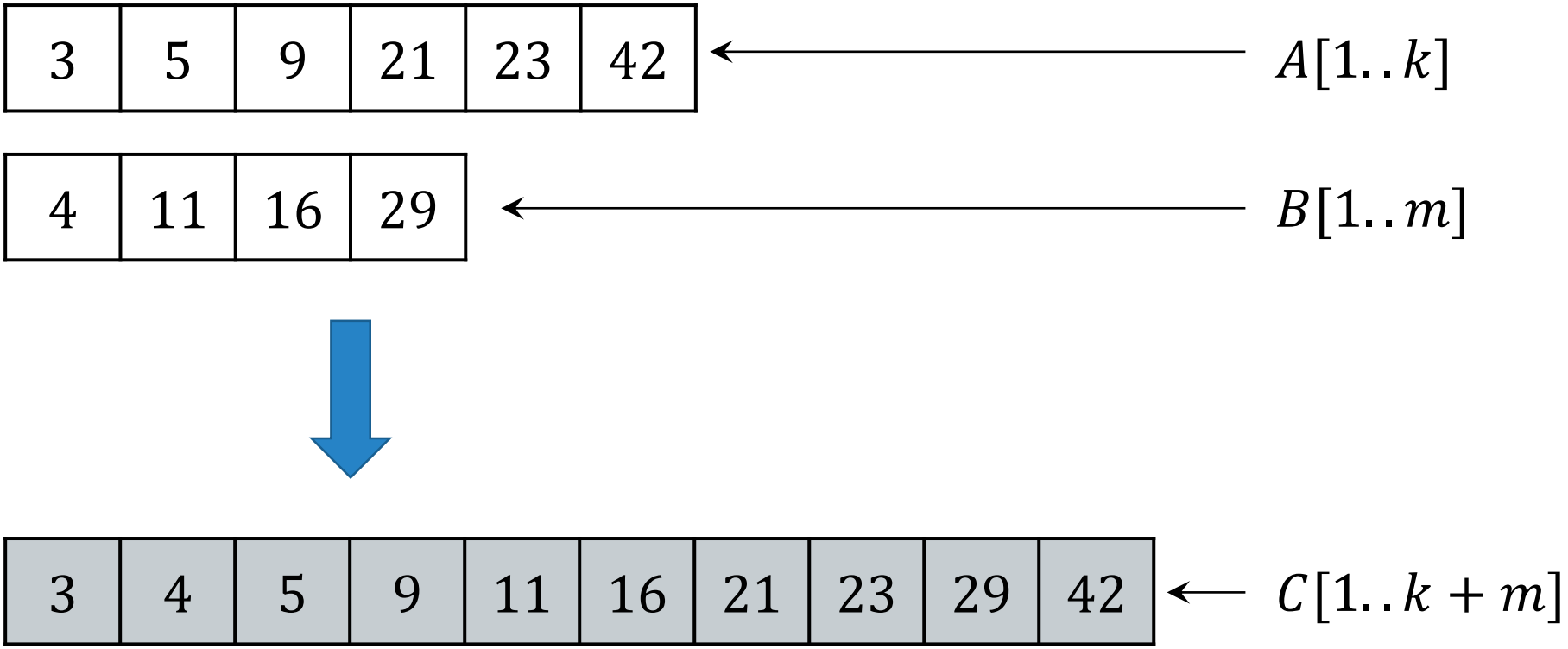
$k < v$	
$h \leftarrow \frac{k+v}{2}$ Szétvág(S, S_1, S_2)	SKIP
MergeSort(A, k, h) MergeSort$(A, h+1, v)$	
Összefuttat$(A[k, h], A[h+1, v], A[k, v])$	

- A külső hívás
  - MergeSort$(A, 1, n)$
- Az összefuttatásnál szükséges egy segédtömb használata

Mergesort

- Hatékonyságelemzés:
- Műveletigény:
  - Ha n a két sorozat együttes hossza:
    - $\text{MÖö}_{\text{sszefuttat}}(n) = n - 1$
    - $\text{MÖ}_{\text{MS}}(n) \leq (n-1) * \log_2 n = \Theta(n \log_2 n)$

Mergesort



Mergesort

$A = a_1 < a_2 < \cdots < a_k$

$B = b_1 < b_2 < \cdots < b_m$

$C = c_1 < c_2 < \cdots < c_{m+k}$

Batcher-féle

- Ez lesz a PPMerge
  - Egy tömb(részlet) két rendezett feléből összefésüléssel előállítja a tömb(részlet) rendezett tartalmát
  - Ez is rekurzív eljárás, csak kicsit másképp

Batcher-féle rendezés

- Az új összefésülés:
  1. A páratlan és B páros indexű elemeit fésüli össze U-ba
  2. A páros és B páratlan indexű elemeit fésüli össze V-be

$A = a_1 < a_3 < a_5 < \cdots$

$B = b_2 < b_4 < b_6 < \cdots$

$U = u_1 < u_2 < u_3 < u_4 < \cdots$

$A = a_2 < a_4 < a_6 < \cdots$

$B = b_1 < b_3 < b_5 < \cdots$

$V = v_1 < v_2 < v_3 < v_4 < \cdots$

- Tegyük fel, hogy $k = m$ vagy $k = m + 1$

Batcher-féle rendezés

3. Az U és V sorozatokat nem kell összefésülni, hanem elég csak az egymás alatti u_i, v_i párokat 1-1 összehasonlítással a helyes sorrendbe rakni
   - Így $\{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}$, ... -ból kialakul a helyes sorrend (Ezt be fogjuk látni.)

- Párhuzamosítás: 1. és 2. párhuzamosan végezhető.

Batcher-féle rendezés – példa

A	1	4	5	7	11	12	14	20
B	2	3	6	10	13	15	16	17

- 1. menet: A páratlan és B páros indexű elemei

A	1	5	11	14
B	3	10	15	17

U	1	3	5	10	11	14	15	17

Batcher-féle rendezés – példa

A	1	4	5	7	11	12	14	20
B	2	3	6	10	13	15	16	17

- 2. menet: A páros és B páratlan indexű elemei

A	4	7	12	20
B	2	6	13	16

V	2	4	6	7	12	13	16	20

Batcher-féle rendezés

- $\{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}$ párosítás

U	1	3	5	10	11	14	15	17
V	2	4	6	7	12	13	16	20

1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20

Batcher-féle rendezés

- Rekurzió
  - A 2-2 rövidebb sorozat összefésülését ugyanezzel az eljárással végezzük. Ez rekurzív hívással valósul meg.
  - Ehhez meg kell adni a legkisebb k és m értéket, amelyre már nem hívja az eljárás önmagát:
    - Egy 2 hosszú tömböt még szétvágunk két 1 hosszú részre és azokra meghívjuk az összefésülőt, ekkor $k = 1$ és $m = 1$.
    - Egy 1 hosszú tömböt azonban már nem fésülünk össze, hanem felismerjük, hogy az már rendezett, így nulla (0) összehasonlítást igényel, ekkor $k = 1$, $m = 0$.

Batcher-féle rendezés

- Állítás: A PP_Merge eljárás helyes.
  - Belátjuk, hogy a leírt eljárás azt a helyes rendezett sorozatot eredményezi, amelyet $C = c_1 < c_2 < \cdots < c_{m+k}$-val jelöltünk
  - Esetszétválasztással gondoljuk meg:
    $$c_1 = \min\{u_1, v_1\}, \qquad c_2 = \max\{u_1, v_1\}$$
  - Általában, ha $1 \leq i \leq \left\lfloor \frac{k+m}{2} \right\rfloor$:
    $$c_{2i-1} = \min\{u_i, v_i\}, \qquad c_{2i} = \max\{u_i, v_i\}$$

Batcher-féle rendezés

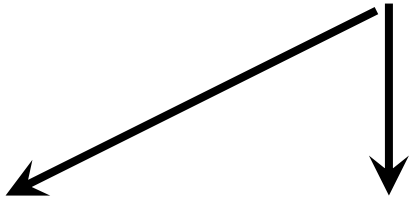
- Vegyük figyelembe a kiegyensúlyozott szétvágást is, azaz azt, hogy $k = m$ vagy $k = m + 1$ esetén $\left\lfloor \frac{k+m}{2} \right\rfloor = m$
- Ha $k + m$ páratlan, akkor még azt is be kell látni, hogy c_{k+m} is helyesen képződik, hiszen erre akkor a képlet nem vonatkozik.
- Bizonyítás:
  1. Belátjuk, hogy a C sorozatban bármely páros indexű tagig elmenve $c_1, \ldots c_{2k}$ között ugyanannyi u_i szerepel, mint v_j vagyis az U és a V sorozat szinte „cipzár"-szerűen építi a C-t.

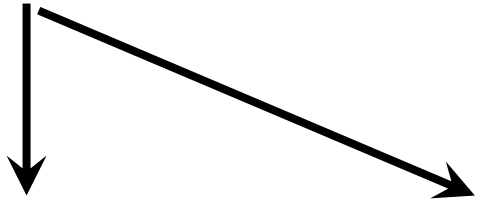
Batcher-féle rendezés

- A C sorozat a rendezett A és B sorozatok összefésülésével adódik.
- Tegyük fel, hogy
  - A-ból az $a_1, \ldots a_s$ elemek,
  - B-ből a $b_1, \ldots, b_{2k-s}$ elemek jönnek összefésüléssel a
  - C sorozat első $2k$ elemébe

Batcher-féle rendezés

- $\{c_1, \dots c_{2k}\} = \{a_1, \dots a_s\} \cup \{b_1, \dots b_{2k-s}\}$





U-ba kerülnek a páratlan indexű elemek, ezek száma: $\lceil s/2 \rceil$	V-be kerülnek a páros indexű elemek, ezek száma: $\lfloor s/2 \rfloor$	U-ba kerülnek a páros indexű elemek, ezek száma: $\lfloor (2k-s)/2 \rfloor$	V-be kerülnek a páratlan indexű elemek, ezek száma: $\lceil (2k-s)/2 \rceil$

Batcher-féle rendezés

- Az állítás ekkor egyenértékű azzal, hogy:
- $\lceil s/2 \rceil + \lfloor (2k - s)/2 \rfloor = \lfloor s/2 \rfloor + \lceil (2k - s)/2 \rceil$?
  - Ez nyilván igaz, ha s páros, hiszen ekkor minden tag egész,
  - Ha s páratlan, akkor a kérdés a következő egyenlőség fennáll-e (igen)
  - $\frac{s+1}{2} + \frac{2k-(s+1)}{2} = \frac{(s-1)}{2} + \frac{2k-(s-1)}{2}$

Batcher-féle rendezés

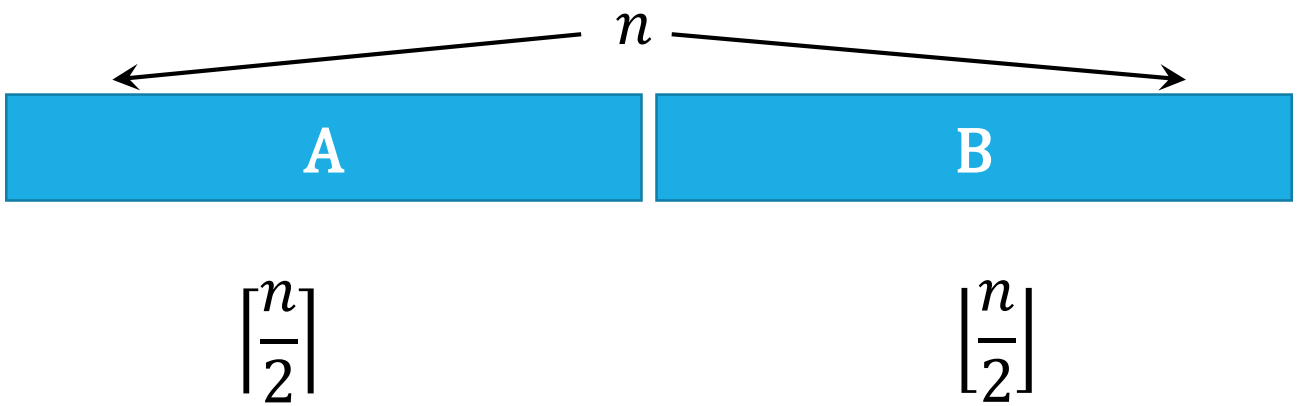
2. Eszerint:
   - $\{c_1, \dots c_{2i}\} = \{u_1, \dots u_i\} \cup \{v_1, \dots v_i\}$
   - $\{c_1, \dots c_{2(i-1)}\} = \{u_1, \dots u_{i-1}\} \cup \{v_1, \dots v_{i-1}\}$
   - A két halmaz kivonásával:
     - $\{c_{2i-1}, c_{2i}\} = \{u_i, v_i\}$
   - Így, mivel $c_{2i-1} < c_{2i}$, tehát az eredeti állítás fennáll.

- Tehát
  - U és V már egy párhuzamos lépésben összefésülhető.

Batcher-féle rendezés

- Hatékonyságelemzés:
  - Párhuzamos költséget számolunk, ez azt jelenti, hogy akárhány összehasonlítás is csak 1-nek számít, ha párhuzamosan végezzük!

1. A PP_Merge rekurzív eljárást egy n méretű, két (közel) egyenlő méretű, rendezett két félből álló tömbre hajtjuk végre.



Batcher-féle rendezés

- A két félből párhuzamosan lehet képezni az U és V sorozatokat, itt tehát $\lceil n/2 \rceil$-t vesszük alapul. Mivel ugyanezt az eljárást használjuk az U és V előállítására is, ez a költségszámítás képletében is rekurziót ad.
- Továbbá, az U és V tömbökből egyetlen párhuzamos összehasonlítással kapjuk a C eredményt.
  - Így az egyenlet: $\text{MÖ}_{\text{PPM}}(n) \leq \text{MÖ}_{\text{PPM}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1$
  - Itt egyébként elegendő MÖ helyett Ö-t írni, mert ez az eljárás fix összehasonlítás számmal dolgozik. Szokásos jelölés még:
    - $T(n) \leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1\,, T(1) = 0$

Batcher-féle rendezés

- Ennek megoldása úgy történik általánosan is, ahogy egy konkrét értékre:
  - például $n = 21$-re:
  - $T(21) \leq T(11) + 1 \leq T(6) + 1 + 1 \leq T(3) + 1 + 1 + 1 \leq T(2) + 1 + 1 + 1 + 1 \leq T(1) + 1 + 1 + 1 + 1 + 1 = = 0 + 1 + 1 + 1 + 1 + 1 = 5 = \lceil \log_2 21 \rceil$
  - $T(21) \leq \lceil \log_2 21 \rceil$
  - Ha v olyan egész, amire $2^v < n \leq 2^{v+1}$, akkor a rekurzív egyenletet éppen $v + 1$-szer alkalmazzuk, mire eljutunk $T(1)$-ig
    - Így $v + 1$ db egyest adunk össze, vagyis $T(n) \leq \lceil \log_2 n \rceil$
  - Ez tehát a párhuzamos összefésülés költsége.

Batcher-féle rendezés

2. A párhuzamos rendező eljárás szerkezete pontosan az, ami a bevezetőben felidézett MergeSort-é, csak más összefuttatást alkalmaz.

Batcher-féle rendezés - tömbökre

- BatcherSort(S, p, q)

$p < q$	
$r \leftarrow \lfloor (p+q)/2 \rfloor$	SKIP
$\text{BatcherSort}(S, p, r)$ $\text{BatcherSort}(S, r+1, q)$	
$\text{PP\_Merge}(S[p \dots r], S[r+1, \dots q], S[p, \dots q])$	

- Külső hívása
  - BatcherSort(S,1,n)

Batcher-féle rendezés

2. A párhuzamos összehasonlítás számra az egyenlet:

- $Ö_{\text{Batcher}}(n) \leq Ö_{\text{Batcher}}(\lceil n/2 \rceil) + \lceil \log_2 n \rceil$
- Egyszerűbb jelöléssel:
  - $T(n) \leq T(\lceil n/2 \rceil) + \lceil \log_2 n \rceil$
- Ha ezt is kifejtjük, akkor az előzőhöz hasonlóan azt kapjuk, hogy $\lceil \log_2 n \rceil$ számú tag lesz; ezek a tagok azonban csökkennek:
  - $T(n) \leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \lceil \log_2 n \rceil \leq$
    $T\left(\left\lceil \frac{n}{2^2} \right\rceil\right) + \left\lceil \log_2 \left(\left\lceil \frac{n}{2} \right\rceil\right) \right\rceil + \lceil \log_2 n \rceil \leq \cdots$
    $\leq T(1) + 1 + 2 + \cdots (\lceil \log_2 n \rceil - 1) + \lceil \log_2 n \rceil =$
    $\frac{(\lceil \log_2 n \rceil + 1) * \lceil \log_2 n \rceil}{2} \approx \frac{(\log_2 n)^2}{2} = \Theta(\log_2 n)^2$

Batcher-féle rendezés

- Az előzőekben felhasználtuk, hogy
- $\lceil \log_2(\lceil n/2 \rceil) \rceil = \lceil \log_2 n \rceil - 1$, ezt be is látjuk:
  - Ha $n = 2p$, akkor igaz.
  - Ha $2p < n \leq 2p + 1$, akkor
  - $2p - 1 < \frac{n}{2} \leq 2p$,
  - $2p - 1 < \left\lceil \frac{n}{2} \right\rceil \leq 2p$
  - $p - 1 < \log_2 \left\lceil \frac{n}{2} \right\rceil \leq p$, így
  - $\left\lceil \log_2 \left(\left\lceil \frac{n}{2} \right\rceil \right) \right\rceil = \lceil \log_2 n \rceil - 1$

Leszámláló rendezés

Leszámláló rendezés

- Tegyük fel, hogy van n db bemeneti elem, s ezek mindegyike 1 és k közötti egész szám
- Az alapötlet: meghatározzuk minden egyes x bemeneti elemre azoknak az elemeknek a számát, amelyek kisebbek, mint az x
- Ezután x-et közvetlenül a saját pozíciójára tudom helyezni
- Legyen a bemenet az $A[1..n]$ tömb, a kimenet a $B[1..n]$ tömb
  - Mindkettő hossza: $\text{hossz}[A] = \text{hossz}[B] = n$
- Szükség van még egy $C[1..k]$ tömbre átmeneti munkaterületként

Leszámláló rendezés

1. Végigmegyünk az A-n, és ha egy elem értéke i, akkor megnöveljük $C[i]$ értékét eggyel.
2. Minden i-re $1..k$ között meghatározzuk, hogy hány olyan bemeneti elem van, amelyiknek az értéke $\leq$ i (összegzés C-n)
3. Minden i-re $n..1$ között $A[i]$-t betesszük B megfelelő pozíciójába – ezt a C-ből állapítjuk meg

- Ha betettük, akkor a $C[A[i]]$ értékét csökkentjük, így a következő vele egyenlő elem már elé kerül, vagyis így stabil lesz a rendezés, az egyenlő elemeknél megtartja az eredeti sorrendet

Leszámláló rendezés

- Az A tömb elemeit leszámláljuk a C tömbbe
  - A C-ben az i helyen az i-vel egyenlő elemek száma szerepel
  - Végigmegyünk az A-n és ha egy elem értéke i, akkor a $C[i]$ értéket megnöveljük
- A

3	6	4	1	3	4	1	4

- C

2	0	2	3	0	1

Leszámláló rendezés

- Minden i-re $1 \ldots k$ között meghatározzuk, hogy hány olyan bemeneti elem van, amelyiknek az értéke $\leq i$ (összegzés C-n)
- A

3	6	4	1	3	4	1	4

- C

2	2	4	7	7	8

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

2	2	4	7	7	8

- B

						4	

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

2	2	4	6	7	8

- B

	1					4	

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

1	2	4	6	7	8

- B

	1				4	4	

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

1	2	4	5	7	8

- B

	1		3		4	4	

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

1	2	3	5	7	8

- B

1	1		3		4	4	

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

0	2	3	5	7	8

- B

1	1		3	4	4	4	

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

0	2	3	4	7	8

- B

1	1		3	4	4	4	6

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

0	2	3	4	7	7

- B

1	1	3	3	4	4	4	6

Leszámláló rendezés

- Minden i-re n ... 1 között $A[i]$-t betesszük B megfelelő pozíciójába - ezt a C-ből állapítjuk meg.
- A

3	6	4	1	3	4	1	4

- C

0	2	2	4	7	7

- B

1	1	3	3	4	4	4	6

Leszámláló rendezés

Az algoritmus pszeudokódja:

```
for i←1 to k do
    C[i]←0
for i←1 to hossz(A) do
    C[A[i]]←C[A[i]]+1
for i←2 to k do
    C[i]←C[i]+C[i-1]
for i←hossz(A) downto 1 do
    B[C[A[i]]]←A[i]
    C[A[i]]←C[A[i]]-1
```

C-ben az i-vel egyenlő elemek száma

C-ben az i-nél kisebb, vagy egyenlő elemek száma

Leszámláló rendezés

- Futási idő:
  - 1. for ciklus: $\Theta(k)$
  - 2. for ciklus: $\Theta(n)$
  - 3. for ciklus: $\Theta(k)$
  - 4. for ciklus: $\Theta(n)$
- Így a teljes időigény: $\Theta(k+n)$
  - Ha $k = \Theta(n)$, akkor a rendezés futási ideje $\Theta(n)$!
- Ez nem összehasonlító rendezés
  - A helyigénye viszont nagyobb ☹

Edényrendezés

Edényrendezések

- Tegyük fel, hogy tudjuk, hogy a bemenő elemek ($A[1..n]$ elemei) egy m elemű U halmazból kerülnek ki
  - Például $\forall\ A[i]$-re igaz, hogy $A[i] \in [1..m]$.
- Lefoglalunk egy U elemeivel indexelt B tömböt (m db ládát), először mind üres
- B segédtömb elemei lehetnek láncolt listák például

Edényrendezések

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

Edényrendezések

- 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

Edényrendezések

- Lépésszám:
  - B létrehozása: $\mathcal{O}(m)$,
  - első fázis $\mathcal{O}(n)$,
  - második fázis $\mathcal{O}(n+m)$, összesen $\mathcal{O}(n+m)$.
- Ez gyorsabb, mint az általános alsó korlát, ha:
  - $m \leq c * n$

Edényrendezések

- Általánosabban:
  - Legyen K az S sorozat elemeinek típusértékhalmaza, φ: $K \rightarrow [0 \ldots M-1]$ olyan függvény, amire igaz, hogy ha $\varphi(k_1) < \varphi(k_2)$, akkor $k_1 < k_2$.
  - Legyenek $E_0, E_1, \ldots E_{M-1}$ edények, melyek éppen olyan sorozatok, mint S.
  - Az egyes edényekben megmarad az S-beli elemek ottani relatív sorrendje.

Edényrendezések

- Edenyrendezo(S)

$E_0, E_1, \dots, E_{M-1} \leftarrow \varepsilon, \varepsilon, \dots, \varepsilon$
$S \neq \varepsilon$
$\text{Out}(S, x)$ $In(E_{\varphi(x)}, x)$
$S \leftarrow$ Összefűzés$(E_0, E_1, \dots E_{M-1})$

Edényrendezések

- Ahhoz, hogy Edenyrendezo(S) egyszeri szétrakással rendezzen, elégséges, ha:
  - minden edényben legfeljebb egy elem van, $(\forall i \in [0, M-1]: |E_i| \leq 1)$ vagy:
  - az egyes edényekben csak azonos elemek vannak, vagy:
  - az egyes edények rendezettek.
- Ha az előző feltételek valamelyike teljesül, akkor tökéletes az edényrendezés.
  - Ennek műveletigénye: $\Theta(n)$, ahol $\mathrm{n} = |S|$.
  - Példa: nagyon sok embert kell magasság szerint sorba rakni
    - megéri minden egyes testmagasság számára egy edényt létrehozni

Radix rendezés

- Tegyük fel, hogy a kulcsok összetettek, több komponensből állnak, $(t_1, \ldots t_k)$ alakú szavak, ahol a t_i komponens az L_i rendezett típusból való, a rendezés lexikografikus
- Példa: Legyen $(U; <)$ a huszadik századi dátumok összessége az időrendnek megfelelő rendezéssel:
  - $L_1 = \{1900; 1901; \ldots; 1999\}$ $\quad n_1 = 100$
  - $L_2 = \{\text{január}, \text{február}, \ldots, \text{december}\}$ $\quad n_2 = 12$
  - $L_3 = \{1; 2; \ldots; 31\}$ $\quad n_3 = 31$
  - A dátumok rendezése éppen az L_i típusokból származó lexikografikus rendezés lesz

Radix rendezés

1. Rendezzük a sorozatot az utolsó, a k-adik komponensek szerint edényrendezéssel!
2. A kapott eredményt rendezzük a $k-1$-edik komponensek szerint edényrendezéssel, stb.

- Fontos, hogy az edényrendezésnél 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 $\Rightarrow$ stabil rendezés.

Radix rendezés

- 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.
  - Stabil rendezés $\Rightarrow$ később már nem változik a sorrendjük

Radix rendezés

- Példa:

  1969. jan. 18. 1969. jan. 1. 1955. dec. 18. 1955. jan. 18. 1918. dec. 18.

  1. menet után:

  1969. jan. 1. 1969. jan. 18. 1955. dec. 18. 1955. jan. 18. 1918. dec. 18.

  2. menet után:

  1969. jan. 1. 1969. jan. 18. 1955. jan. 18. 1955. dec. 18. 1918. dec. 18.

  3. menet után:

  1918. dec. 18. 1955. jan. 18. 1955. dec. 18. 1969. jan. 1. 1969. jan. 18.

Radix rendezés

- Példa – ha fordítva lenne

  1969. jan. 18. 1969. jan. 1. 1955. dec. 18. 1955. jan. 18. 1918. dec. 18.

  1. menet után:

  1918. dec. 18. 1955. dec. 18. 1955. jan. 18. 1969. jan. 18. 1969. jan. 1.

  2. menet után:

  1955. jan. 18. 1969. jan. 18. 1969. jan. 1. 1918. dec. 18. 1955. dec. 18.

  3. menet után:

  1969. jan. 1. 1955. jan. 18. 1969. jan. 18. 1918. dec. 18. 1955. dec. 18.

- Helytelen a sorrend

Radix rendezés

- Általánosan
  - Az $e = e_d e_{d-1} \dots e_2 e_1$ számot jobbról balra, az alacsony helyiértékek felől indulva pozíciónként szétrakja edényekbe, majd összefűzi az edények tartalmát
  - Az i. pozíción a φ_i függvényt alkalmazzuk: $\varphi_i(e) = e_i$
  - Az i. pozíción végrehajtott szétrakás és összefűzés után S „i-rendezett" lesz.

Radix rendezés

- Definíció
  - S „i-rendezett" (jelölés: $x \leq_i y$), ha minden
    - $x = x_d x_{d-1} \dots x_2 x_1$ és $y = y_d y_{d-1} \dots y_2 y_1$–ra: $x <_0 y$
    - $x \leq_i y \Leftrightarrow x_i < y_i$ vagy $x_i = y_i$ és $x <_{i-1} y$ $(i > 0)$
  - Ekkor a „d-rendezés" a közönséges rendezés
- Hatékonyság:
  - $2 * d$-szer megyünk végig az S sorozaton, így

$$T(n) = \Theta(d * |S|)$$

Radix rendezés

- Radix(S)

$i \leftarrow 1$	
$i < d$	
	Edenyrendezo(S, φ_i) $i \leftarrow i + 1$

Radix rendezés

- Szokásos implementáció:
  - S fejelemes láncolt lista
  - Az edényeket egy „fej” és egy „vége” mutató ábrázolja.
  - A szétrakás és az összefűzés az elemek láncolásával megoldható.
  - Összefűzéskor nem kell az egyes edények részlistáit végigolvasni, hanem egy darabban lehet őket láncolni.

Külső rendezők
2-3 fák, B fák

Következő alkalommal