A'LL.: $f(n) = a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \dots + a_0$

és $a_k > 0 \implies f(n) \in \Theta(n^k)$

"Műveletidő" = alprogram-hívások száma + ciklusiterációk száma

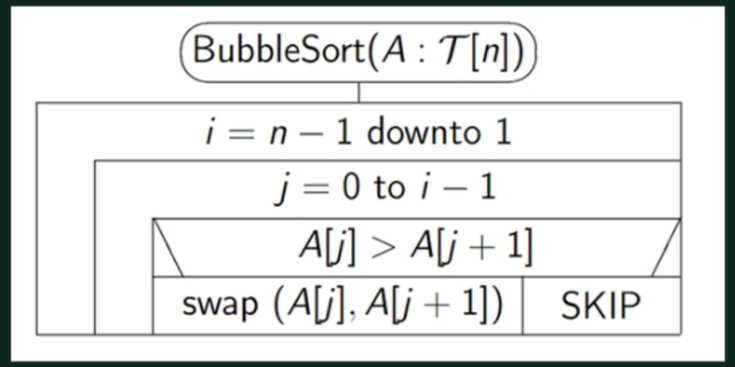
$T(n)$

$mT(n)$ min.

$MT(n)$ maximális

$AT(n)$ átlagos

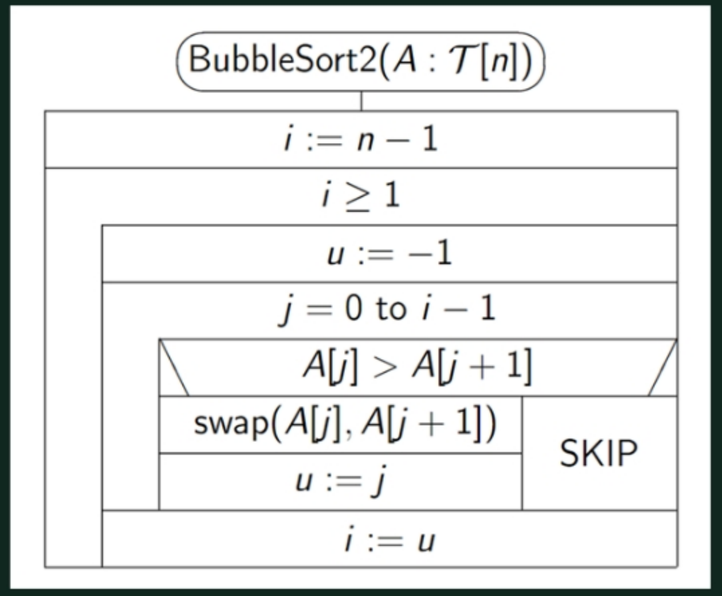
} műk. idő



$T(n) \in \Theta(n^2)$

$mT(n) = MT(n)$

$\therefore Oh(n) \in \Theta(n^2)$



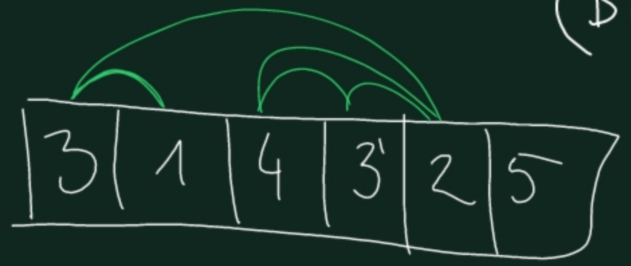
$mT(n) \in \Theta(n)$

$MT(n) \in \Theta(n^2)$

$\therefore Oh(n) \in O(n^2)$
$\in \Omega(n)$

Def.: Egy A tömbben 2 elem: A[i] és A[j] **inverzióban áll**, ha $i < j$ és $A[i] > A[j]$.

ÁLL.: Buborékban cserék száma = A-beli inverziók száma (Bub2-ben is!)



5 db

↳ gyakorlásként nézzék meg, hogy a Buborék tényleg 5 cserét hajt végre

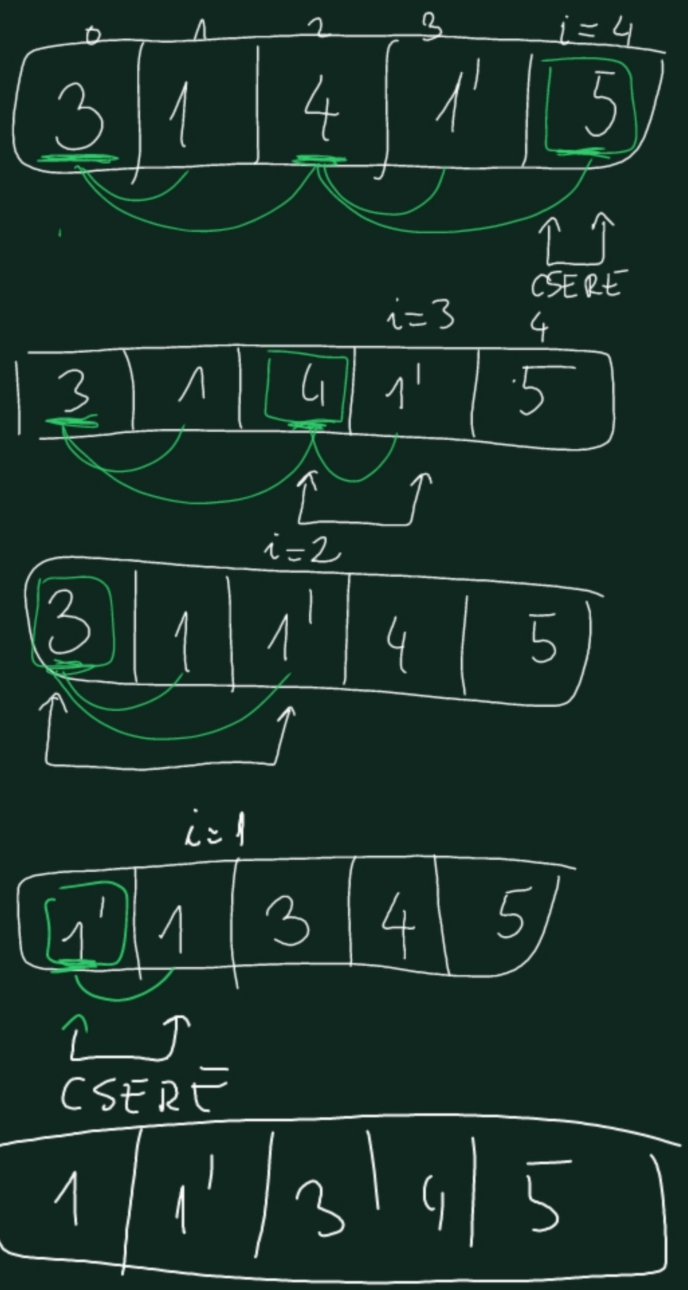
MaxSelectionSort($A : \mathcal{T}[n]$)

- $i = n - 1$ downto 1
  - $Maxind := 0$
  - $j = 1$ to i
    - $A[j] > A[Maxind]$
      - $Maxind := j$ | SKIP
  - swap($A[Maxind], A[i]$)

$T(n) \in \Theta(n^2)$

$\ddot{O}h(n) \in \Theta(n^2)$

$Csere(n) \in \Theta(n)$



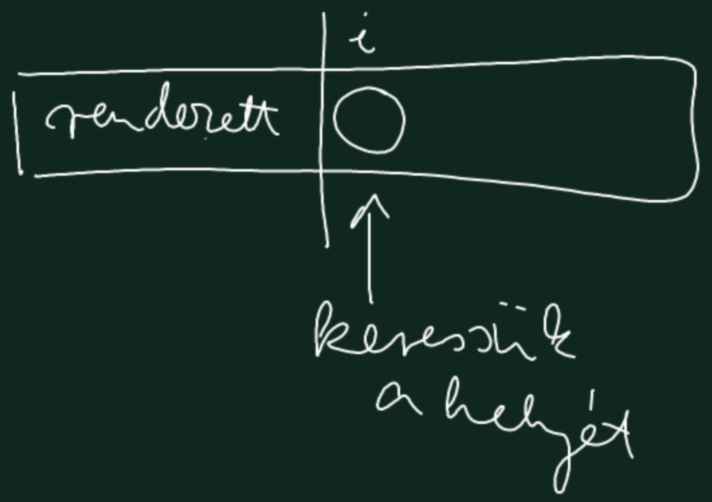
Öh	Csere
4 db	1 db
3	1
2	1
1	1

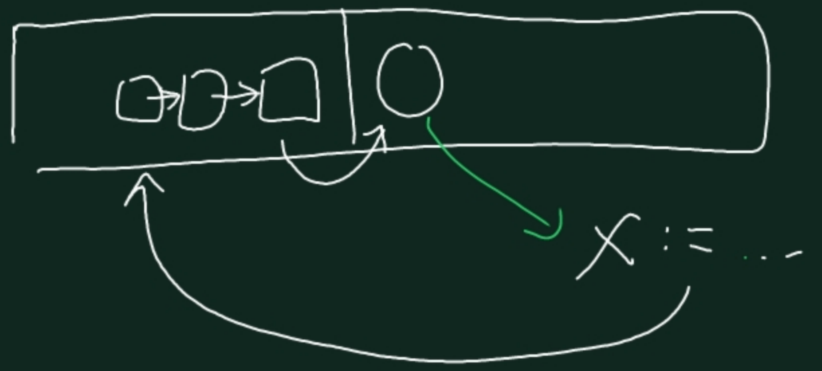
naiveInsertionSort($A : \mathcal{T}[n]$)

- $i := 1$ **to** $n - 1$
  - $j := i$
  - $j > 0 \wedge A[j-1] > A[j]$
    - swap($A[j-1], A[j]$)
    - $j := j - 1$

insertionSort($A : \mathcal{T}[n]$)

- $i := 1$ to $n - 1$
  - $A[i-1] > A[i]$
    - $x := A[i]$
    - $A[i] := A[i-1]$
    - $j := i - 2$
    - $j \geq 0 \wedge A[j] > x$
      - $A[j+1] := A[j]$
      - $j := j - 1$
    - $A[j+1] := x$
  - SKIP





insertionSort($A : \mathcal{T}[n]$)

$i := 1$ to $n-1$

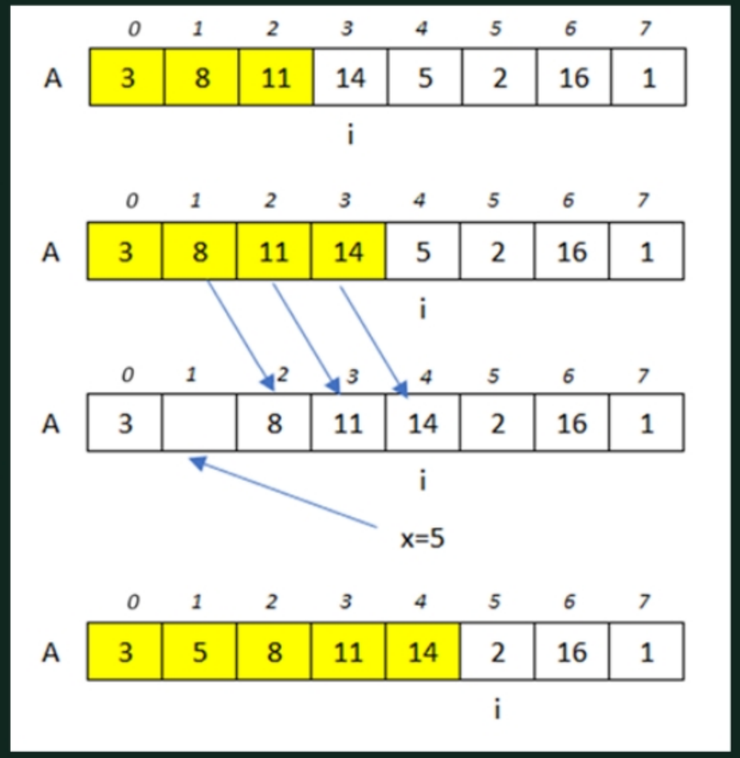
$A[i-1] > A[i]$

$x := A[i]$	
$A[i] := A[i-1]$	
$j := i-2$	
$j \geq 0 \wedge A[j] > x$	SKIP
$A[j+1] := A[j]$	
$j := j-1$	
$A[j+1] := x$	

$i = 3$

$i = 4$

$i = 5$



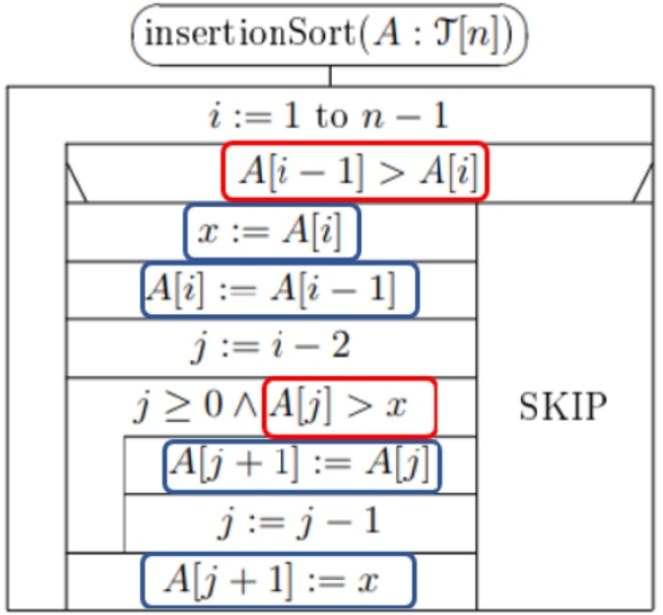
$x = 5$

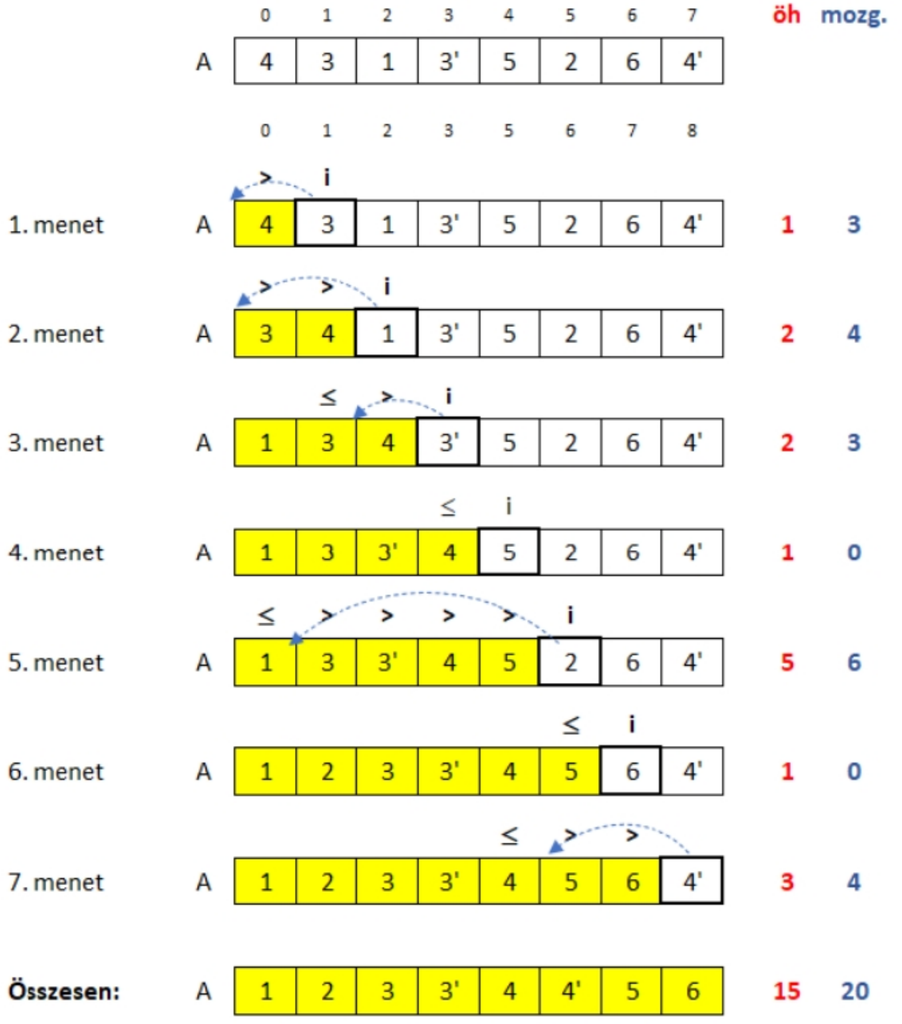
$mT(n) \in \Theta(n)$

$MT(n) \in \Theta(n^2)$

Piros keret jelzi a kulcs-összehasonlításokat, kék a kulcs-mozgatásokat.

Jelöljük, mely kulcsokkal történt összehasonlítás!





Összefésülő rendezés / Merge Sort

mergeSort$(A : \mathcal{T}[n])$

$B : \mathcal{T}[n]$; $B[0\,..\,n) := A[0\,..\,n)$
// Sort $B[0\,..\,n)$ into $A[0\,..\,n)$ non-decreasingly:
ms(B, A)

ms$(B, A : \mathcal{T}[n])$

// Initially $B[0\,..\,n) = A[0\,..\,n)$.	
// Sort $B[0\,..\,n)$ into $A[0\,..\,n)$ non-decreasingly:	
$n > 1$	
$m := \lfloor \frac{n}{2} \rfloor$	SKIP
ms$(A[0\,..\,m), B[0\,..\,m))$ // Sort $A[0\,..\,m)$ into $B[0\,..\,m)$	
ms$(A[m\,..\,n), B[m\,..\,n))$ // Sort $A[m\,..\,n)$ into $B[m\,..\,n)$	
merge$(B[0\,..\,m), B[m\,..\,n), A[0\,..\,n))$ // sorted merge	

merge$(A : \mathcal{T}[l]\ ;\ B : \mathcal{T}[m]\ ;\ C : \mathcal{T}[n])$

// sorted merge of A and B into C where $l + m = n$	
$k := 0$ // in loop, copy into $C[k]$	
$i := 0$; $j := 0$ // from $A[i]$ or $B[j]$	
$i < l \wedge j < m$	
$A[i] \leq B[j]$	
$C[k] := A[i]$	$C[k] := B[j]$
$i := i + 1$	$j := j + 1$
$k := k + 1$	
$i < l$	
$C[k\,..\,n) := A[i\,..\,l)$	$C[k\,..\,n) := B[j\,..\,m)$

merge($A : \mathcal{T}[l]$; $B : \mathcal{T}[m]$; $C : \mathcal{T}[n]$)

// sorted merge of A and B into C where $l + m = n$

$k := 0$ // in loop, copy into $C[k]$

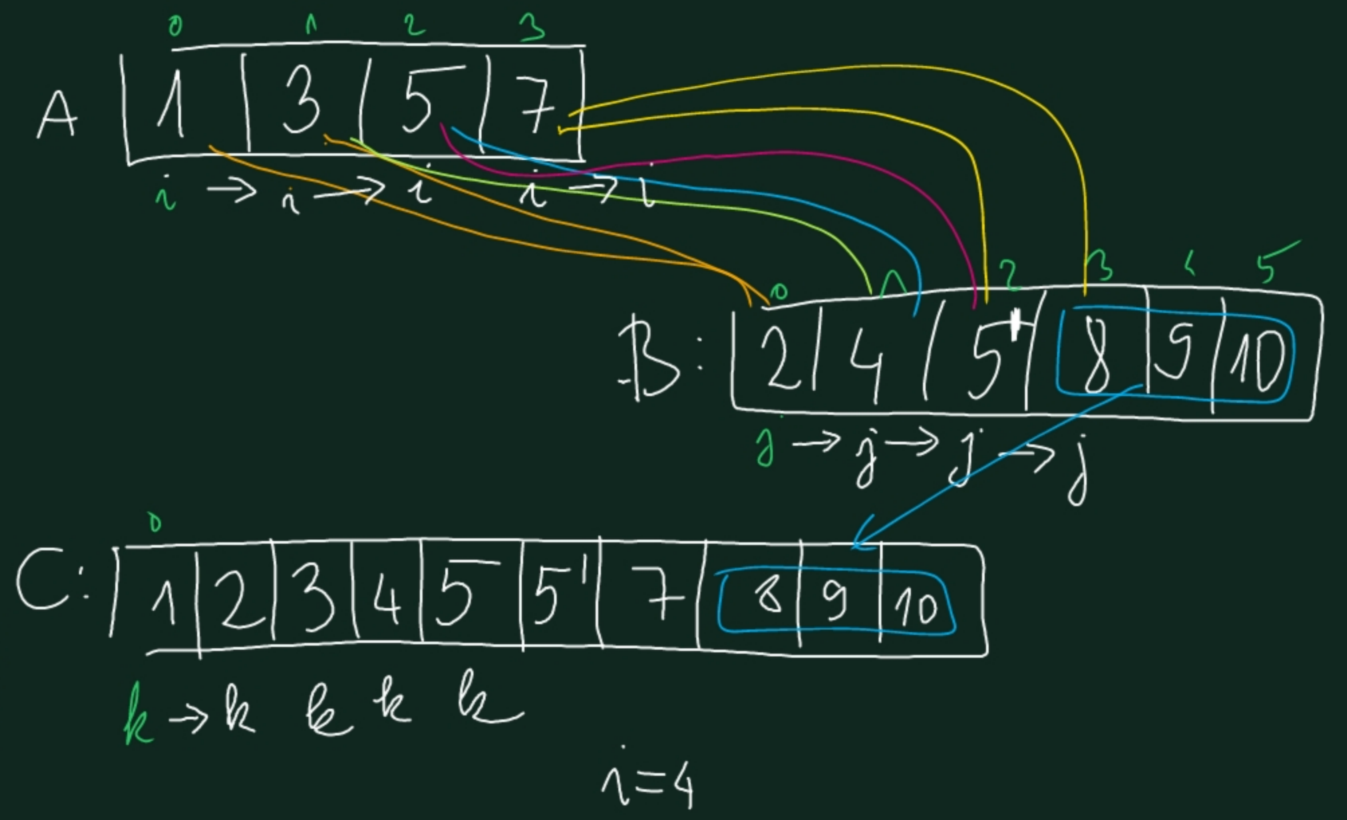
$i := 0$; $j := 0$ // from $A[i]$ or $B[j]$

$i < l \wedge j < m$

$A[i] \leq B[j]$	
$C[k] := A[i]$	$C[k] := B[j]$
$i := i + 1$	$j := j + 1$
$k := k + 1$	

$i < l$

$C[k\,..\,n) := A[i\,..\,l)$	$C[k\,..\,n) := B[j\,..\,m)$



Akl. it # = n

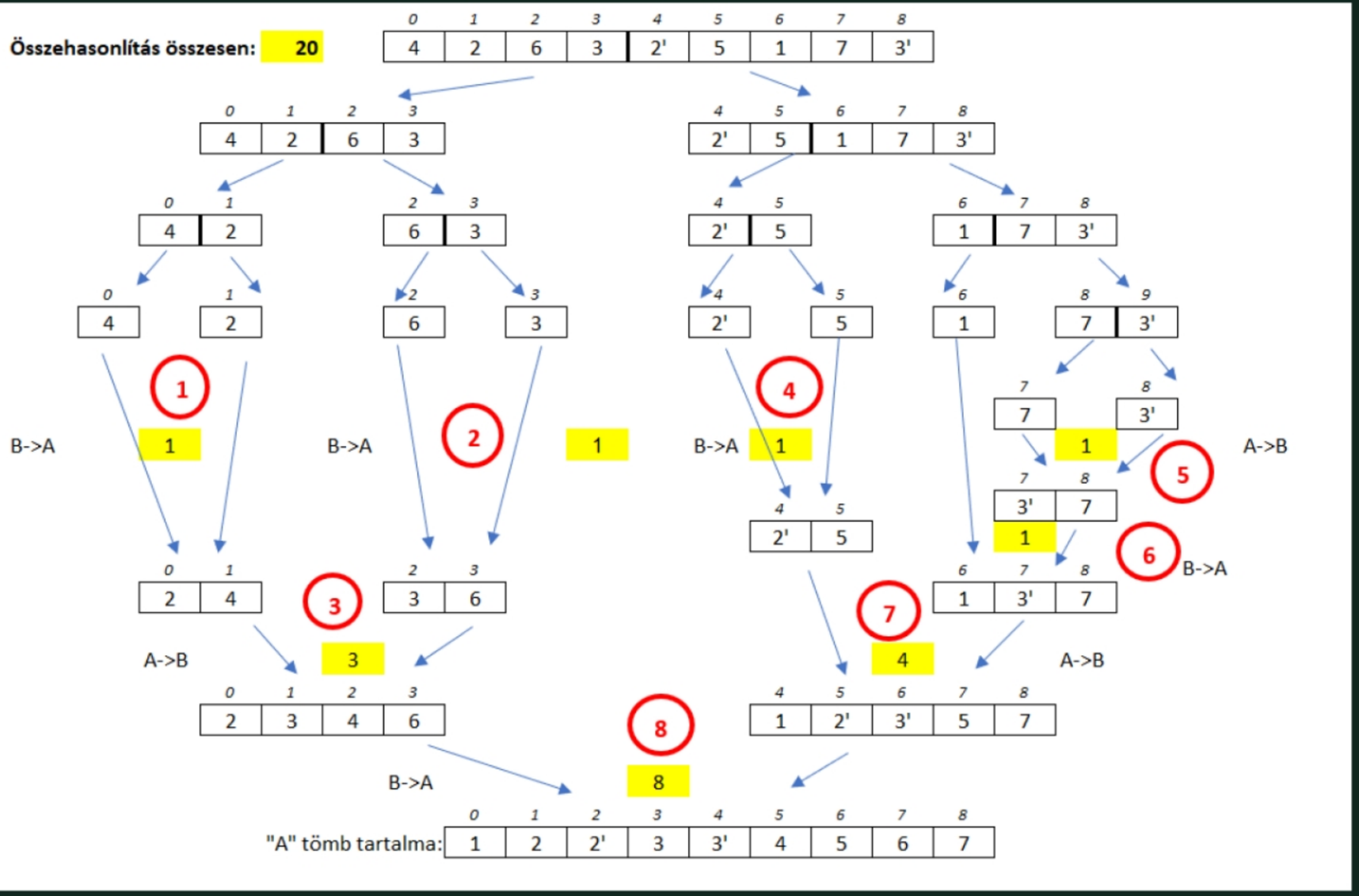
Oh = 7

mergeSort($A : \mathcal{T}[n]$)

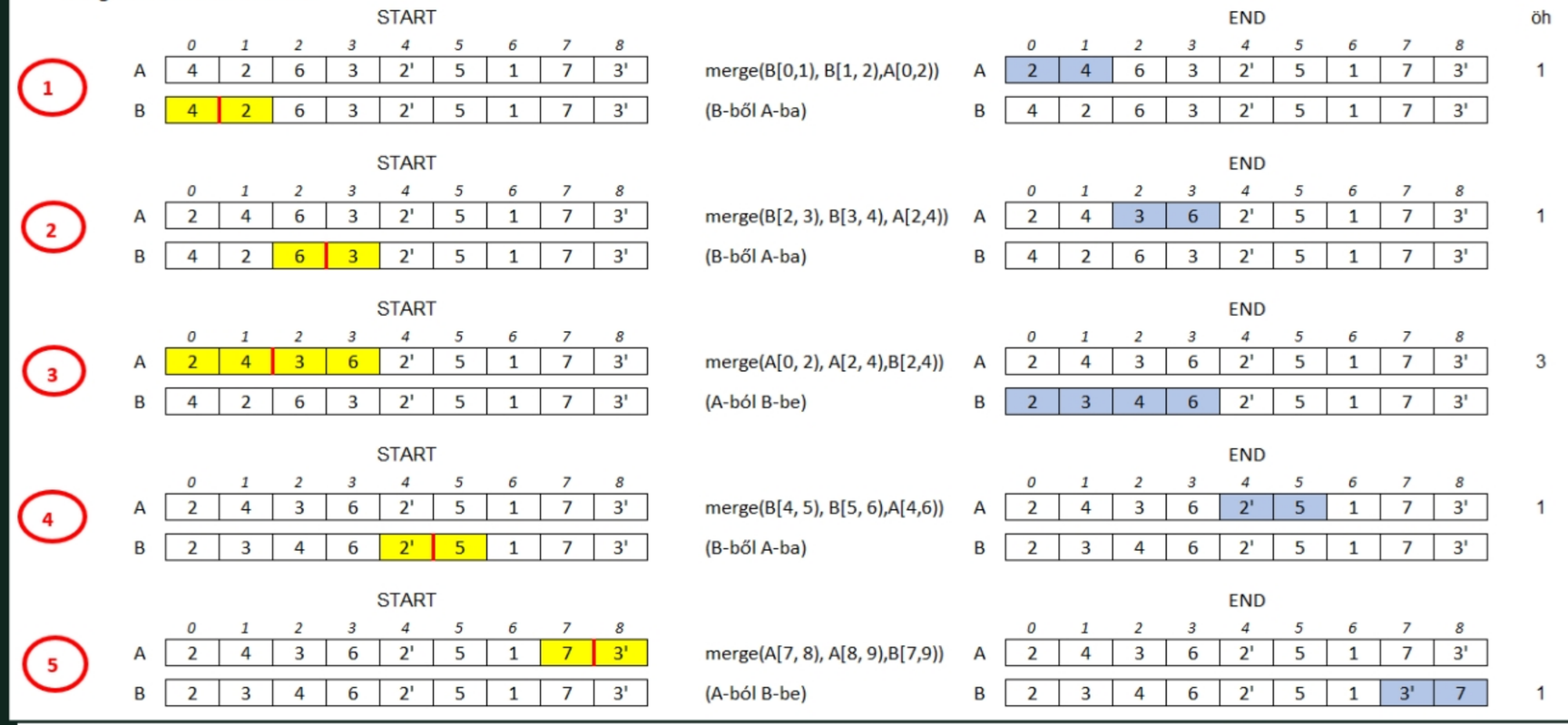
$B : \mathcal{T}[n]$; $B[0\,..\,n) := A[0\,..\,n)$
// Sort $B[0\,..\,n)$ into $A[0\,..\,n)$ non-decreasingly:
ms(B, A)

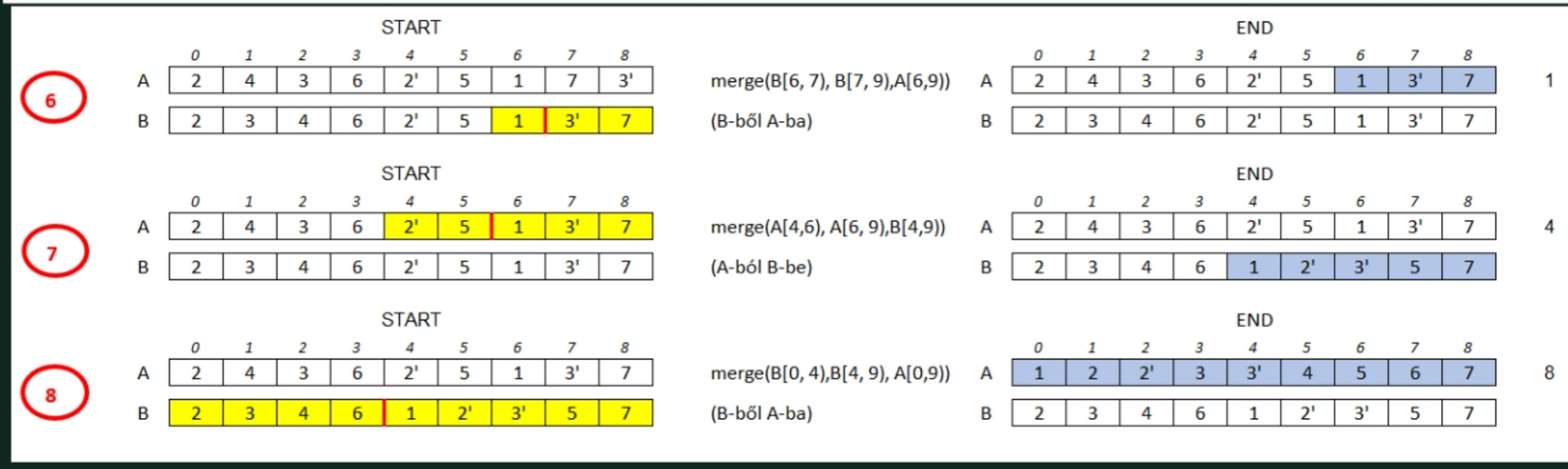
ms($B, A : \mathcal{T}[n]$)

// Initially $B[0\,..\,n) = A[0\,..\,n)$.	
// Sort $B[0\,..\,n)$ into $A[0\,..\,n)$ non-decreasingly:	
$n > 1$	
$m := \lfloor \frac{n}{2} \rfloor$	SKIP
ms($A[0\,..\,m), B[0\,..\,m)$) // Sort $A[0\,..\,m)$ into $B[0\,..\,m)$	
ms($A[m\,..\,n), B[m\,..\,n)$) // Sort $A[m\,..\,n)$ into $B[m\,..\,n)$	
merge($B[0\,..\,m), B[m\,..\,n), A[0\,..\,n)$) // sorted merge	



Merge futások szemléltetése:





Verem → feltöltött pdf