Sortering

Denne uge: Insertion sort, merge sort

Senere i kurset: Heap sort, counting sort, radix sort, bucket sort

Mikkel Abrahamsen

Sortering

Givet array A af n tal, byt om på rækkefølgen sådan at

$$A[0] \leq A[1] \leq \ldots \leq A[n-1].$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Sortering

Givet array A af n tal, byt om på rækkefølgen sådan at

$$A[0] \leq A[1] \leq \ldots \leq A[n-1].$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Ønsket resultat:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	4	7	12	16	18	25	28	31	33	36	42	45	47	50

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	18	25	28	33	45	7	12	36	1	47	42	50	16	31

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	18	25	28	33	45	7	12	36	1	47	42	50	16	31

$$key = 7$$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	18	25	28	33	45	7 **45**	12	36	1	47	42	50	16	31

$key = 7$

Insertion sort

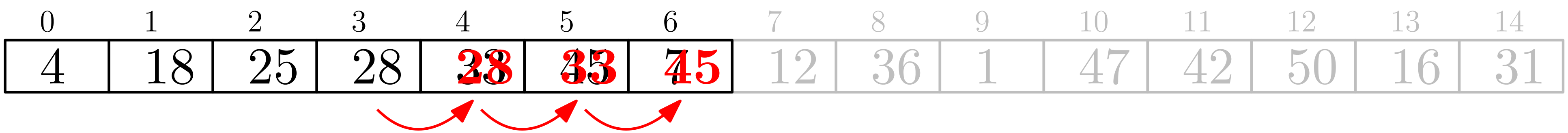
Antag alle tal pånær det sidste er i sorteret rækkefølge.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	18	25	28	33	45 **33**	7 **45**	12	36	1	47	42	50	16	31

$$key = 7$$

Insertion sort

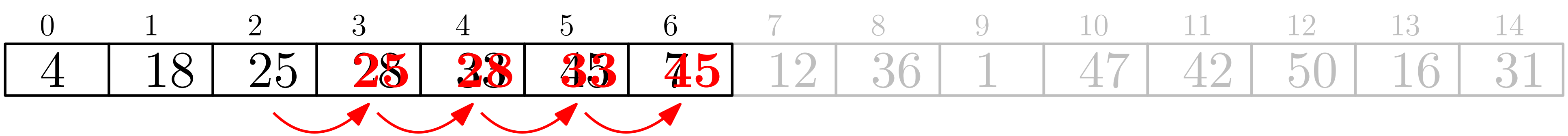
Antag alle tal pånær det sidste er i sorteret rækkefølge.



$$key = 7$$

Insertion sort

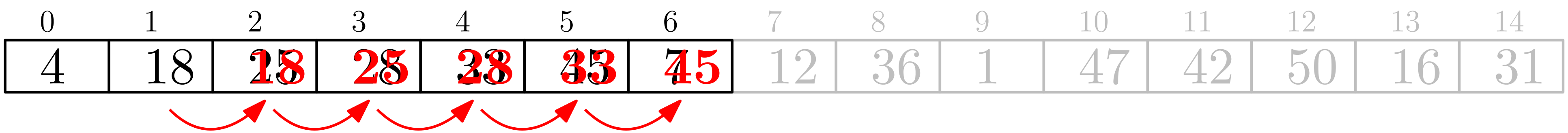
Antag alle tal pånær det sidste er i sorteret rækkefølge.



$key = 7$

Insertion sort

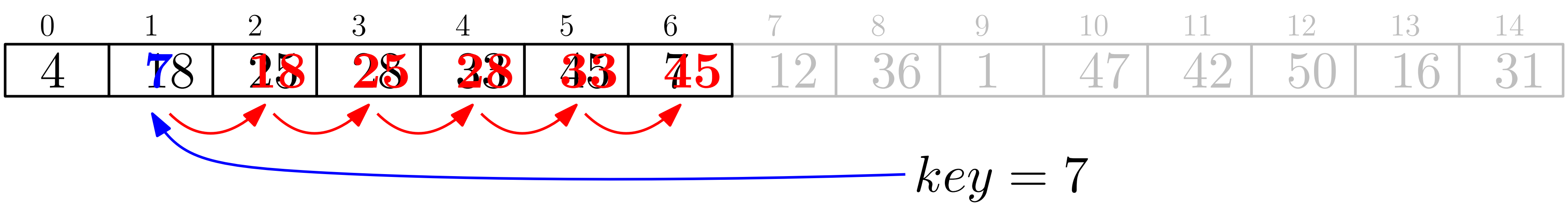
Antag alle tal pånær det sidste er i sorteret rækkefølge.



$$key = 7$$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.



Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

					i	j								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	18	25	28	33	45	7	12	36	1	47	42	50	16	31

$$key = 7$$

$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
    $A[i+1] = A[i]$
    $i = i - 1$
$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

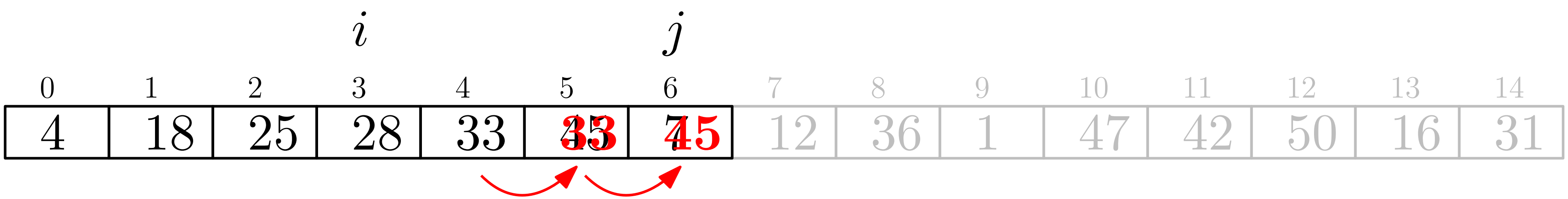
				i		j								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	18	25	28	33	45	~~7~~ **45**	12	36	1	47	42	50	16	31

$key = 7$

$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
  $A[i+1] = A[i]$
  $i = i - 1$
$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.



$key = 7$

$key = A[j]$

$i = j - 1$

while $i \geq 0$ and $A[i] > key$

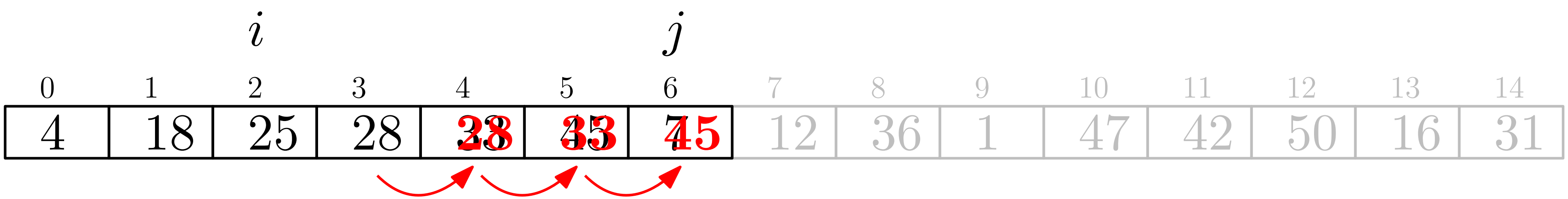
  $A[i+1] = A[i]$

  $i = i - 1$

$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

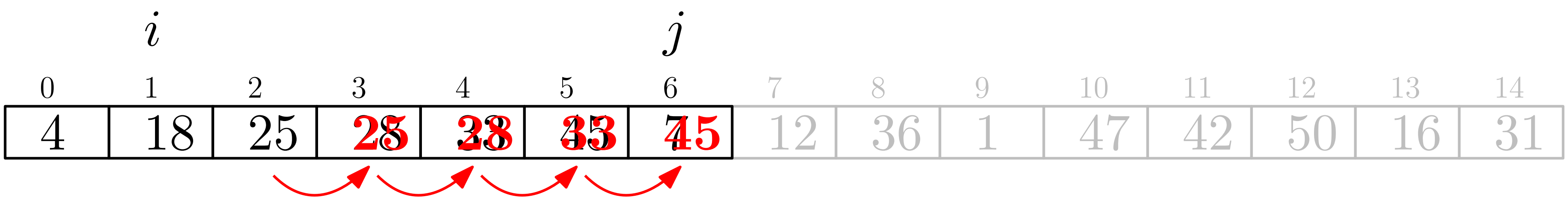


$key = 7$

$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
$\quad A[i+1] = A[i]$
$\quad i = i - 1$
$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

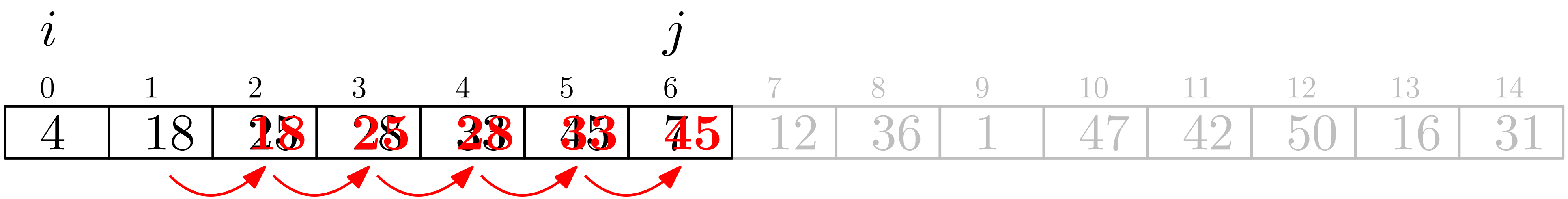


$key = 7$

$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
  $A[i+1] = A[i]$
  $i = i - 1$
$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

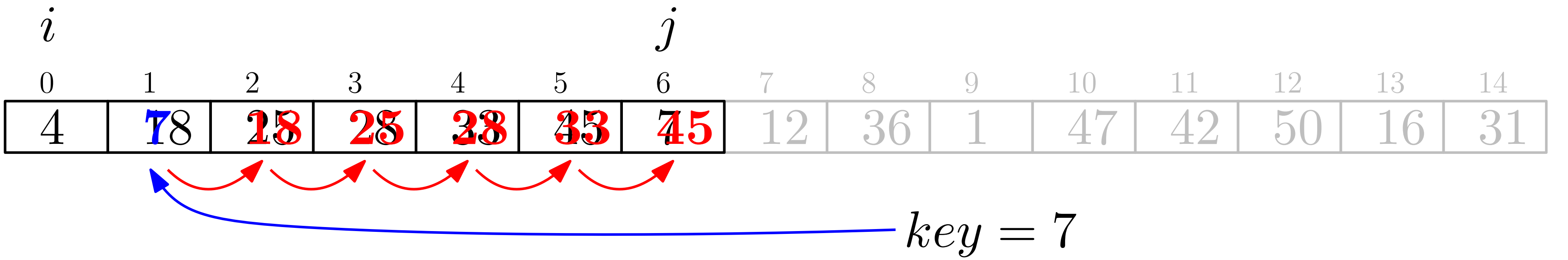


$key = 7$

$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
$\quad A[i+1] = A[i]$
$\quad i = i - 1$
$A[i+1] = key$

Insertion sort

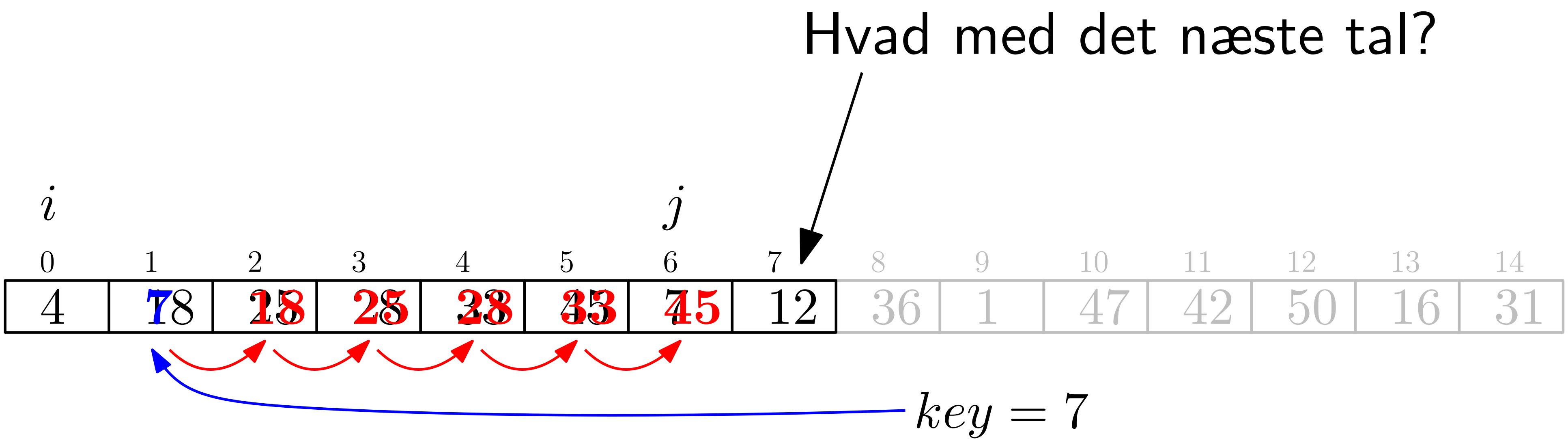
Antag alle tal pånær det sidste er i sorteret rækkefølge.



$$key = A[j]$$
$$i = j - 1$$
while $i \geq 0$ and $A[i] > key$
$\quad A[i+1] = A[i]$
$\quad i = i - 1$
$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.



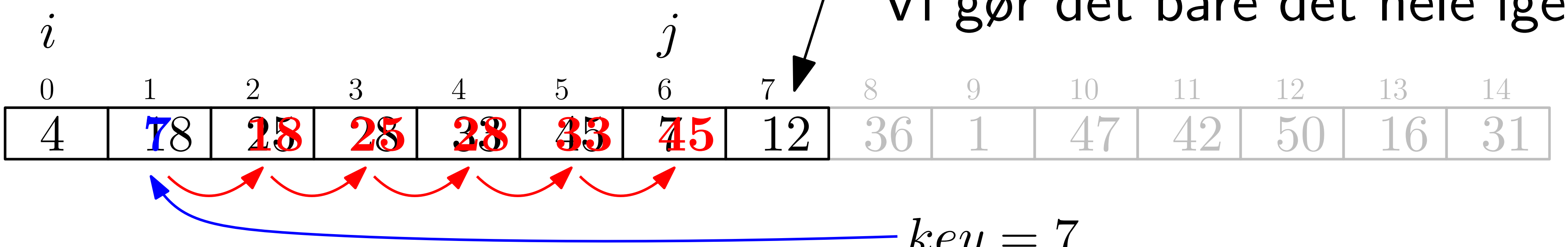
$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
  $A[i+1] = A[i]$
  $i = i - 1$
$A[i+1] = key$

Insertion sort

Antag alle tal pånær det sidste er i sorteret rækkefølge.

Hvad med det næste tal?

Vi gør det bare det hele igen!



$key = A[j]$

$i = j - 1$

while $i \geq 0$ and $A[i] > key$

  $A[i+1] = A[i]$

  $i = i - 1$

$A[i+1] = key$

Sortering

Vi begyndte med:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

$key = A[j]$
$i = j - 1$
while $i \geq 0$ and $A[i] > key$
    $A[i+1] = A[i]$
    $i = i - 1$
$A[i+1] = key$

Sortering

Vi begyndte med:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

Sortering

Vi begyndte med:

$key = 4$

i	j													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Insertion-Sort(A, n)
  for $j = 1$ to $n - 1$
    $key = A[j]$
    $i = j - 1$
    while $i \geq 0$ and $A[i] > key$
      $A[i+1] = A[i]$
      $i = i - 1$
    $A[i+1] = key$

Sortering

Vi begyndte med:

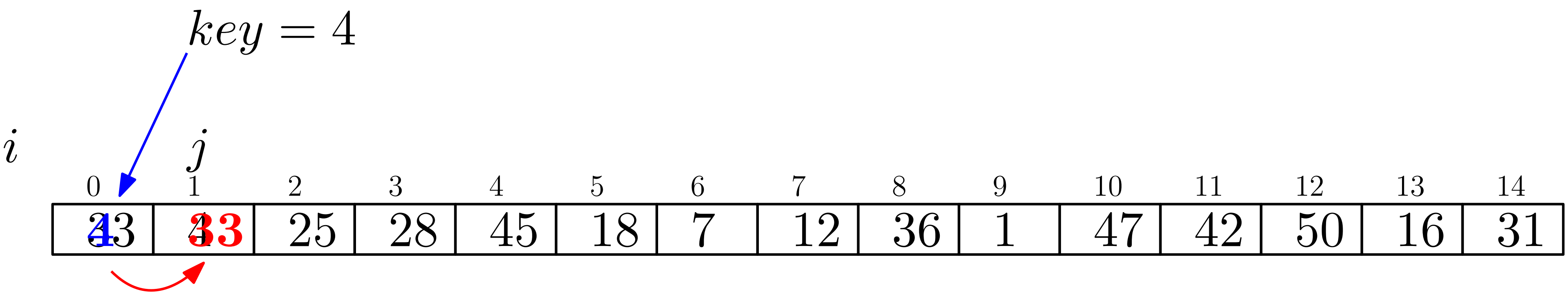
$key = 4$

i		j													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	33	**33**	25	28	45	18	7	12	36	1	47	42	50	16	31

Insertion-Sort(A, n)
  for $j = 1$ to $n - 1$
    $key = A[j]$
    $i = j - 1$
    while $i \geq 0$ and $A[i] > key$
      $A[i + 1] = A[i]$
      $i = i - 1$
    $A[i + 1] = key$

Sortering

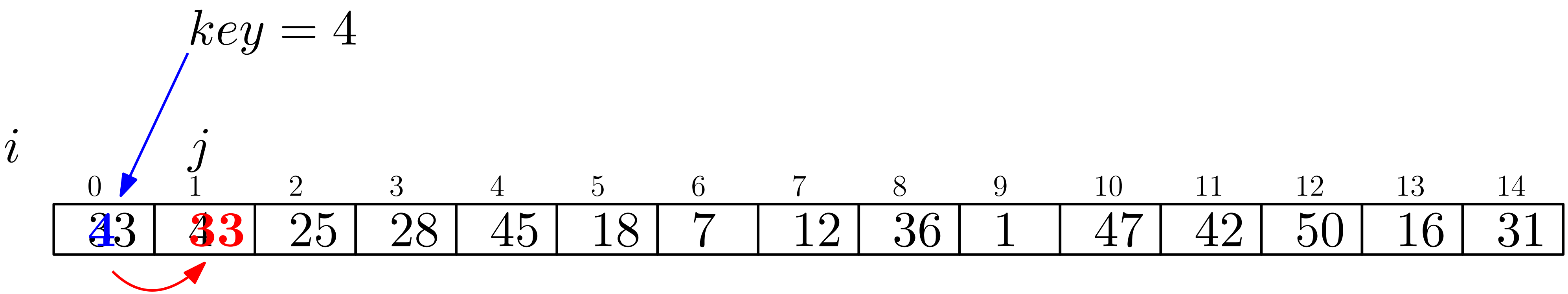
Vi begyndte med:



Insertion-Sort(A, n)
  for $j = 1$ to $n - 1$
    $key = A[j]$
    $i = j - 1$
    while $i \geq 0$ and $A[i] > key$
      $A[i+1] = A[i]$
      $i = i - 1$
    $A[i+1] = key$

Sortering

Vi begyndte med:



Insertion-Sort(A, n)
    for $j = 1$ to $n - 1$
        $key = A[j]$
        $i = j - 1$
        while $i \geq 0$ and $A[i] > key$
            $A[i+1] = A[i]$
            $i = i - 1$
        $A[i+1] = key$

OBS: I CLRS bruger denne algoritme 1-indeksering. Derfor en lille forskel.

Køretidsanalyse

Insertion-Sort(A, n)
  for $j = 1$ to $n - 1$
    $key = A[j]$
    $i = j - 1$
    while $i \geq 0$ and $A[i] > key$
      $A[i+1] = A[i]$
      $i = i - 1$
    $A[i+1] = key$

skridt	max. gange
c_1	n
c_2	$1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2}$
c_3	n

Køretidsanalyse

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

skridt	max. gange
c_1	n
c_2	$1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2}$
c_3	n

Køretid: $T(n) = c_1 n + c_3 n + c_2 \cdot \frac{n \cdot (n+1)}{2}$

$= c_2/2 \cdot n^2 + (c_1 + c_3 + c_2/2) \cdot n$

Køretidsanalyse

Insertion-Sort(A, n)	skridt	max. gange
for $j = 1$ to $n - 1$		
$key = A[j]$	c_1	n
$i = j - 1$		
while $i \geq 0$ and $A[i] > key$		
$A[i+1] = A[i]$	c_2	$1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2}$
$i = i - 1$		
$A[i+1] = key$	c_3	n

Køretid: $T(n) = c_1 n + c_3 n + c_2 \cdot \frac{n \cdot (n+1)}{2}$
$= c_2/2 \cdot n^2 + (c_1 + c_3 + c_2/2) \cdot n$

$= \boxed{\Theta(n^2)}$

Asymptotisk notation: Vi udelader langsomt voksende led og konstanter.
Interesseret i hvordan køretiden vokser som funktion af n.

Køretidsanalyse

Insertion-Sort(A, n)	skridt	max. gange
for $j = 1$ to $n-1$		
$key = A[j]$	c_1	n
$i = j - 1$		
while $i \geq 0$ and $A[i] > key$		
$A[i+1] = A[i]$	c_2	$1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2}$
$i = i - 1$		
$A[i+1] = key$	c_3	n

Køretid: $T(n) = c_1 n + c_3 n + c_2 \cdot \frac{n \cdot (n+1)}{2}$
$= c_2/2 \cdot n^2 + (c_1 + c_3 + c_2/2) \cdot n$

$= \boxed{\Theta(n^2)}$

Asymptotisk notation: Vi udelader langsomt voksende led og konstanter.
Interesseret i hvordan køretiden vokser som funktion af n.

Udfordring: Kan vi sortere hurtigere?

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	i								j						
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50
B															

Nyt array B.

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

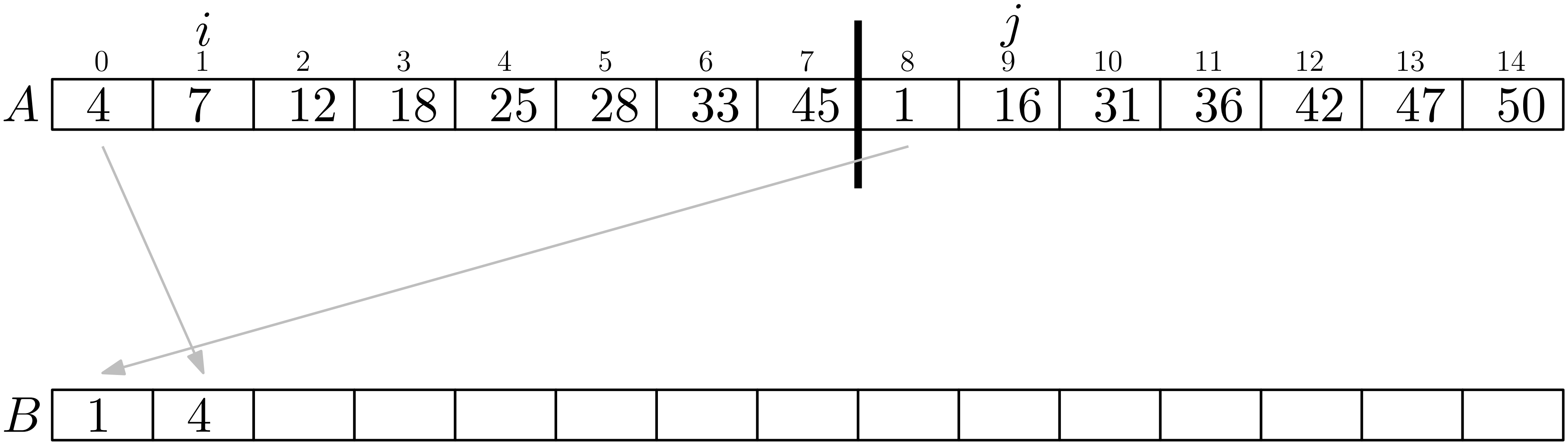
	i									j					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1														

Nyt array B.

Merge sort

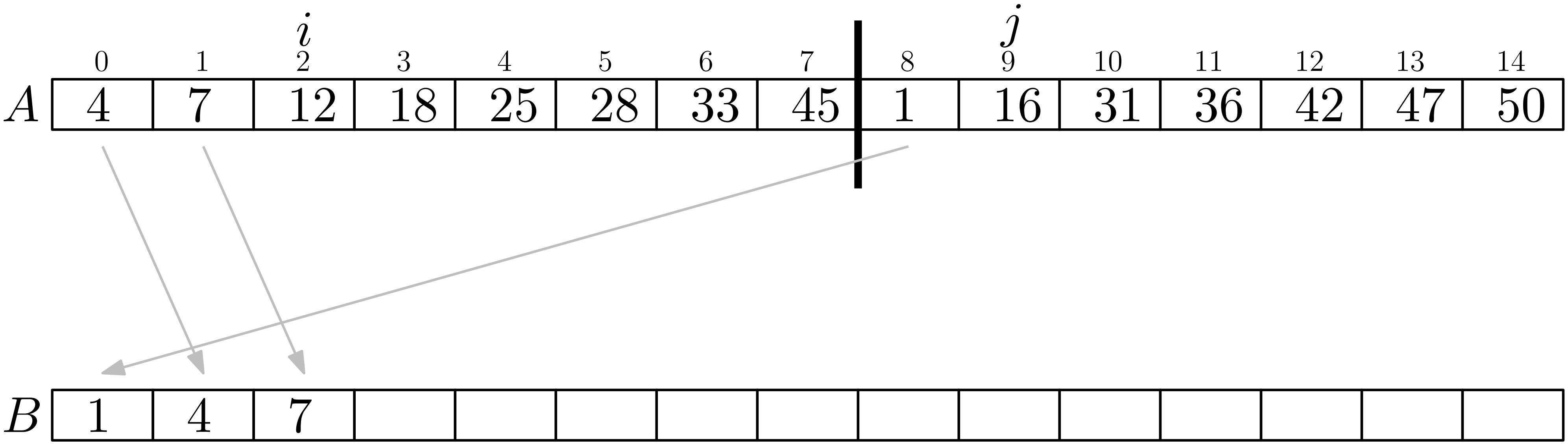
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

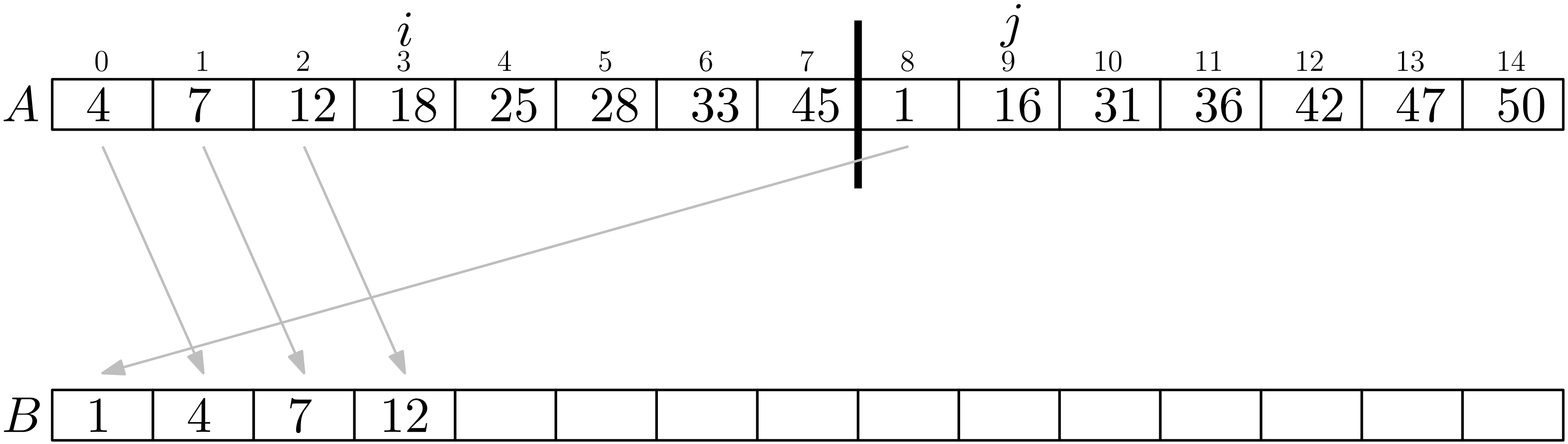
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

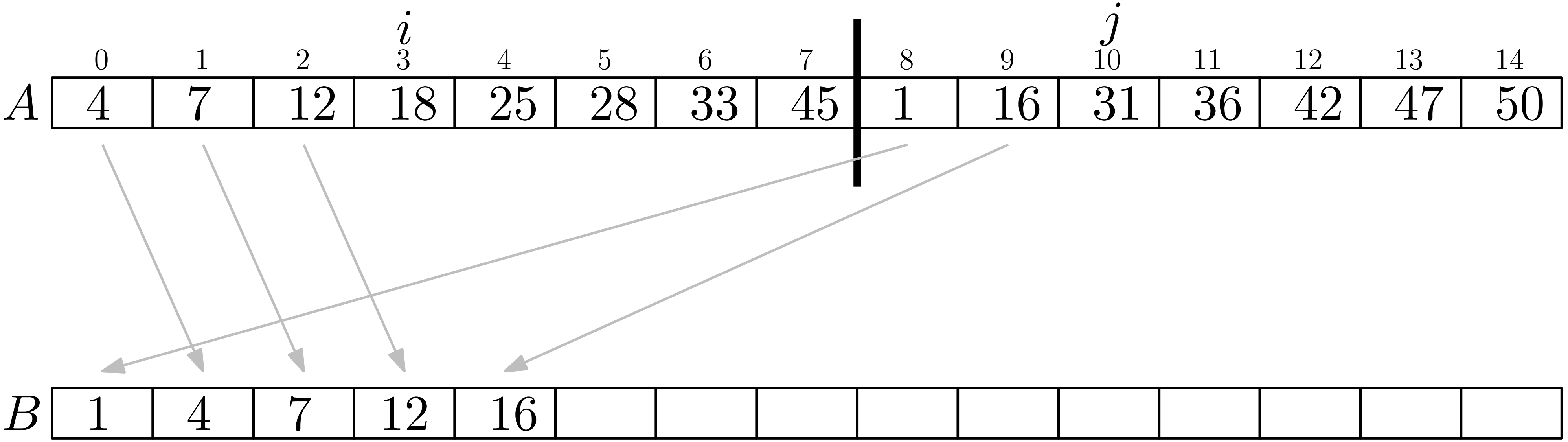
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

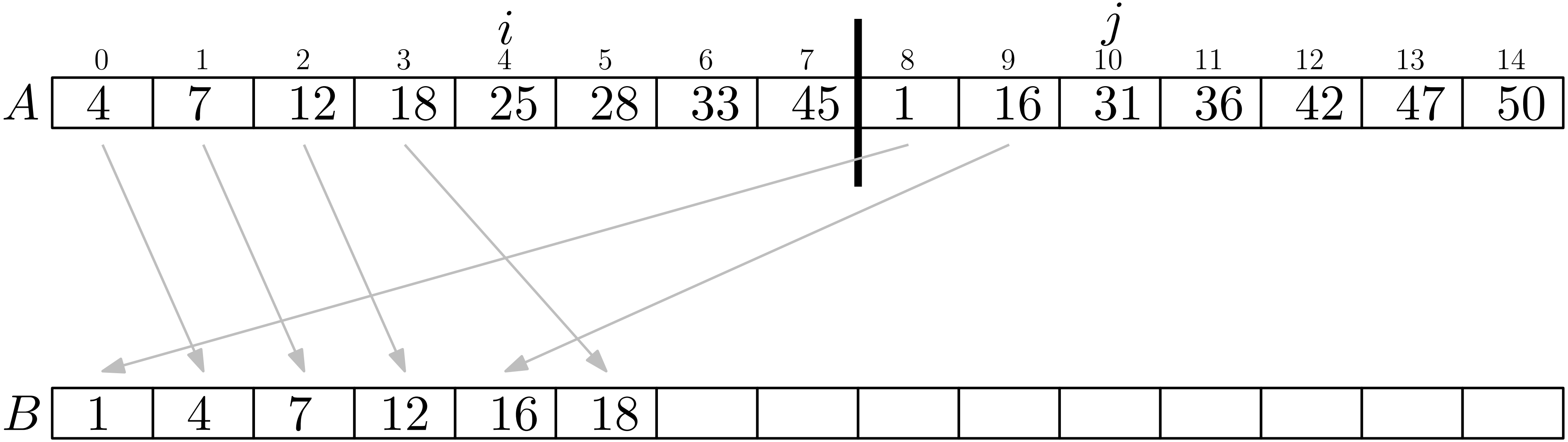
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

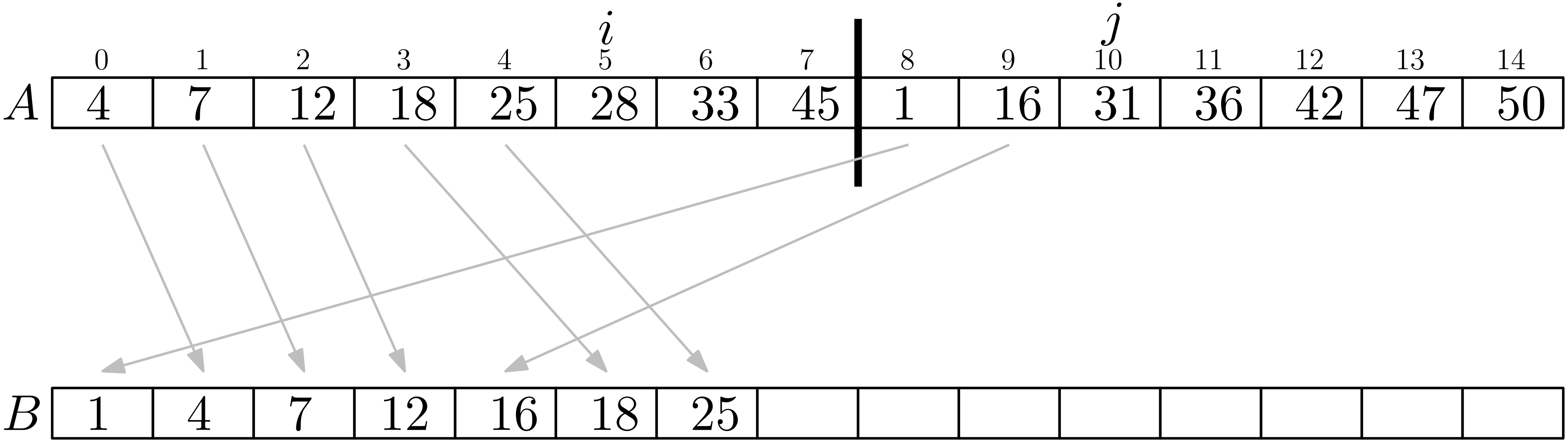
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

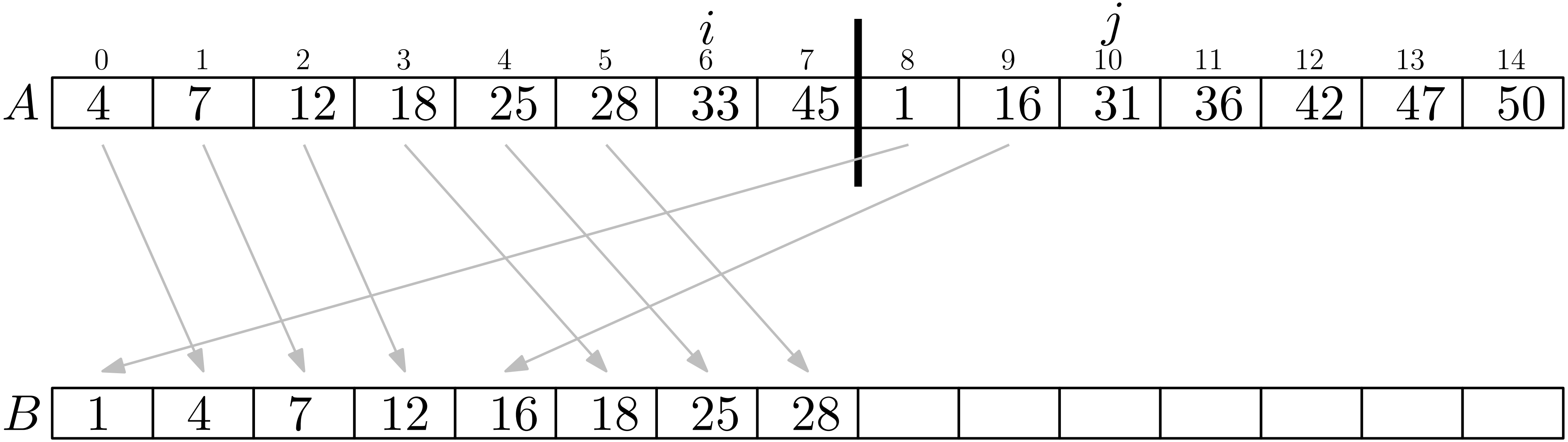
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

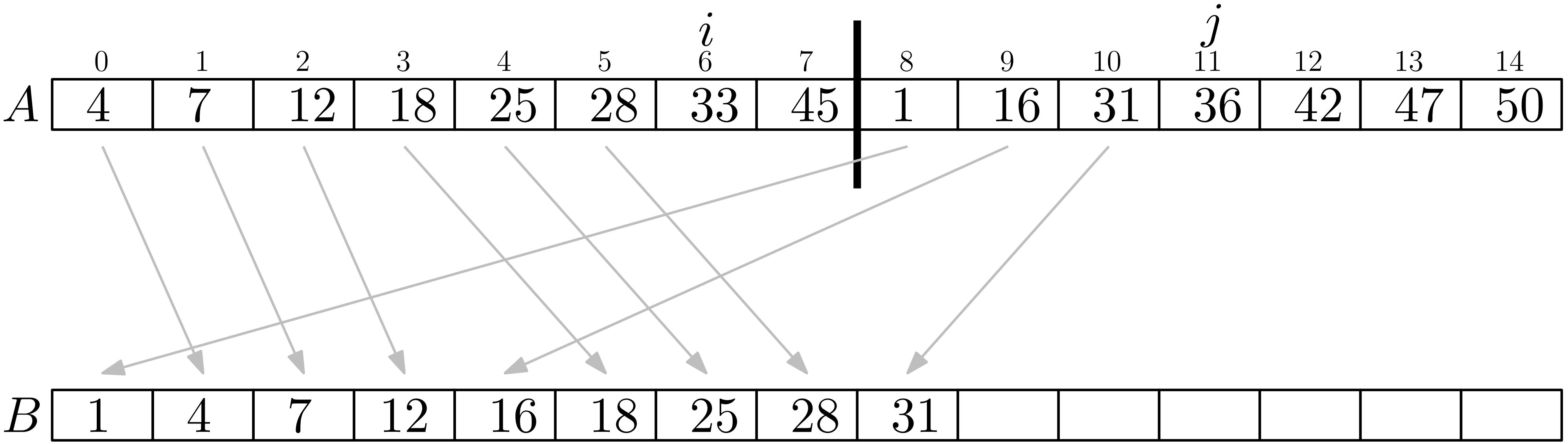
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

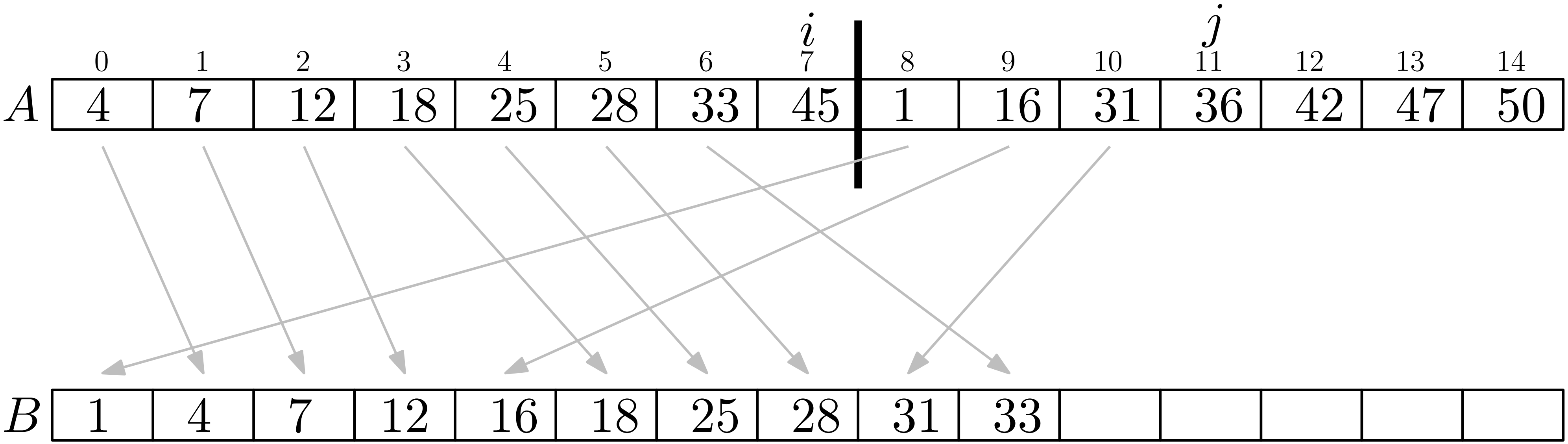
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

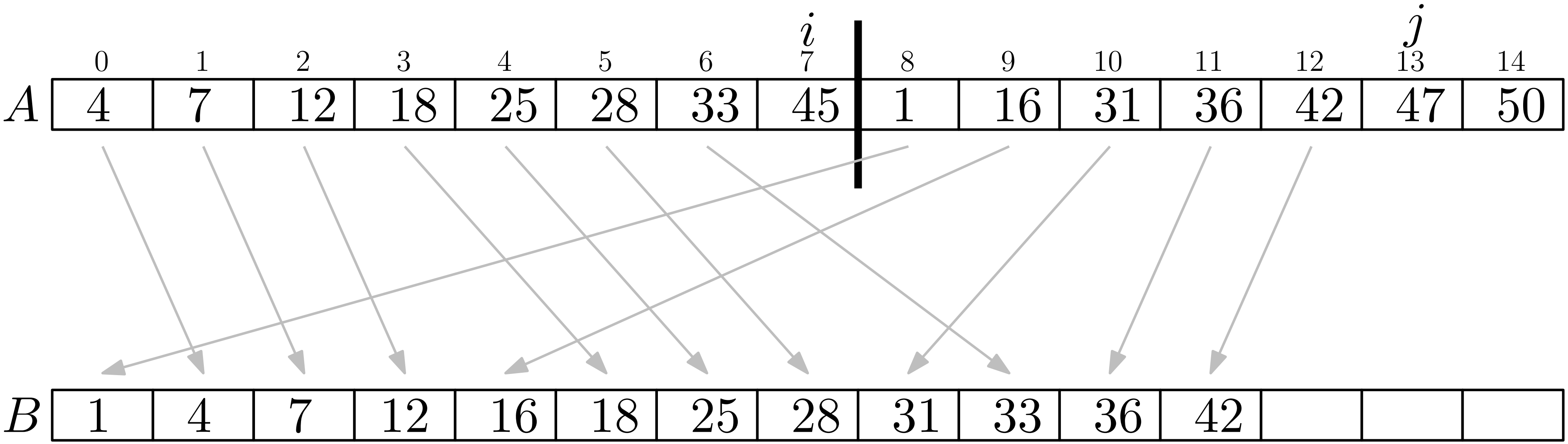
Antag hver halvdel af A er sorteret. Vi merger:

								i					j		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50
B	1	4	7	12	16	18	25	28	31	33	36				

Nyt array B.

Merge sort

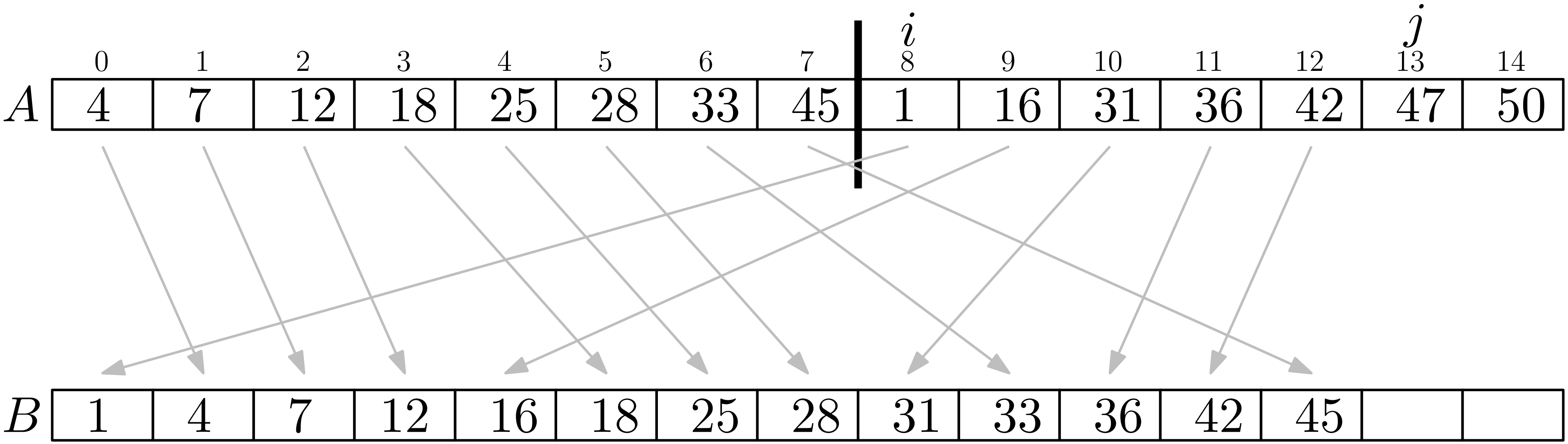
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

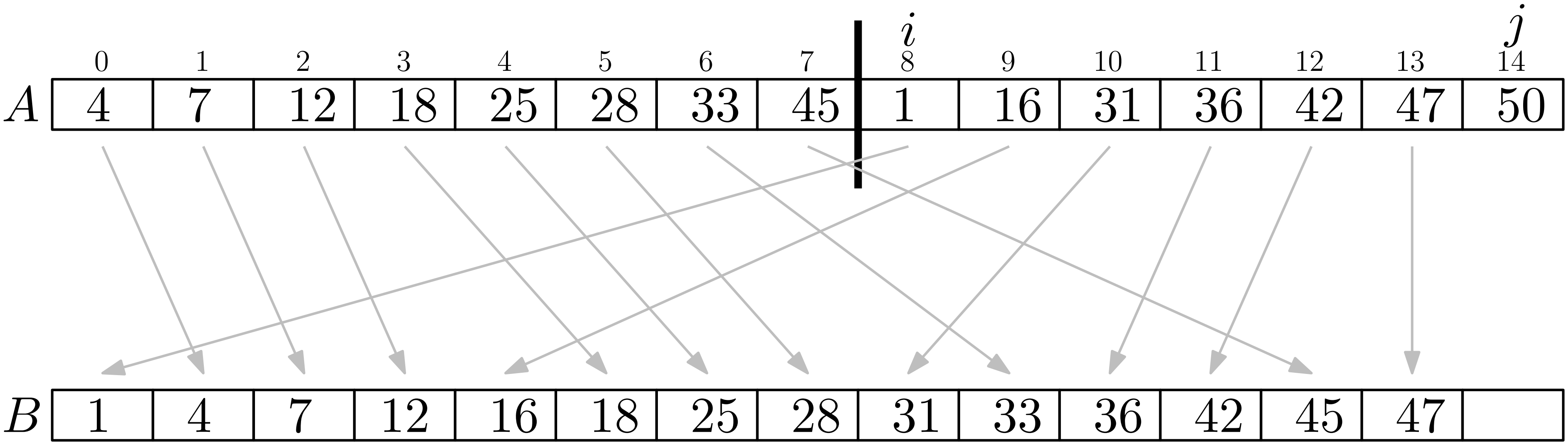
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

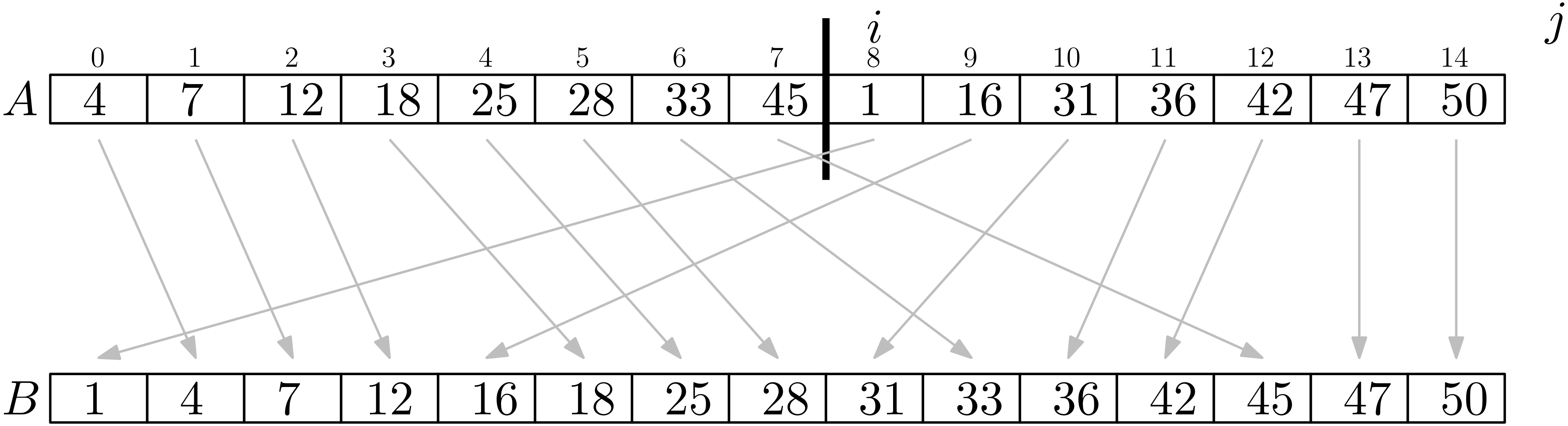
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

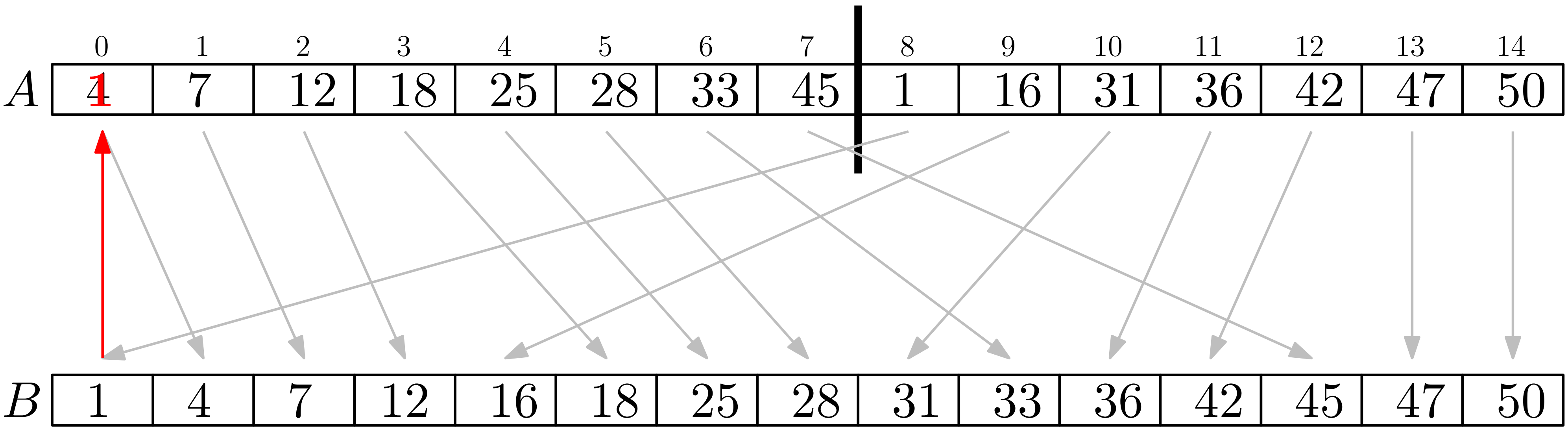
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

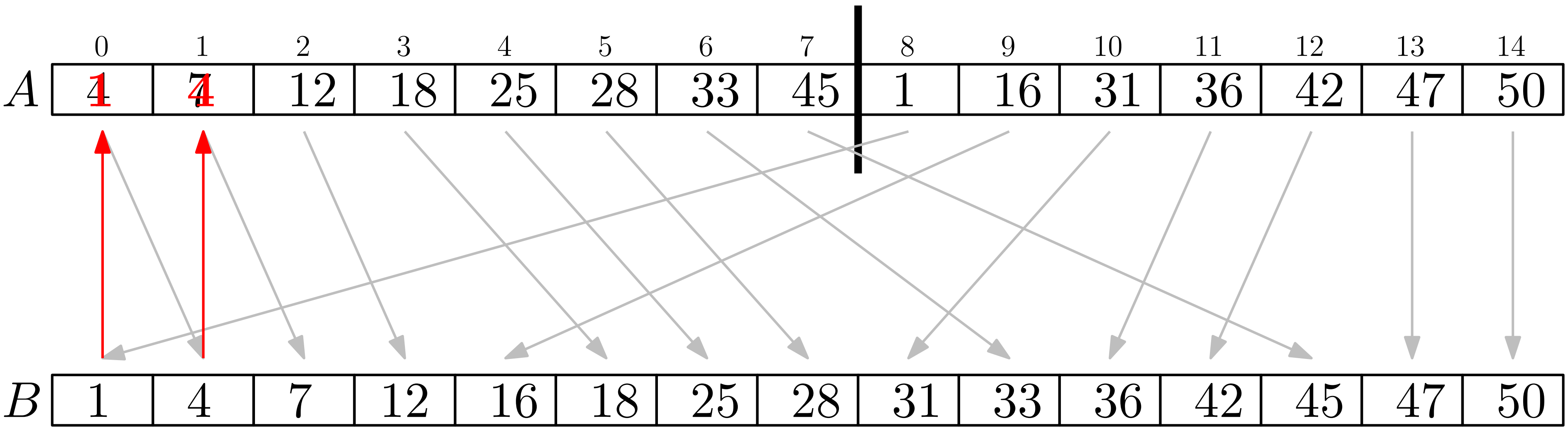
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

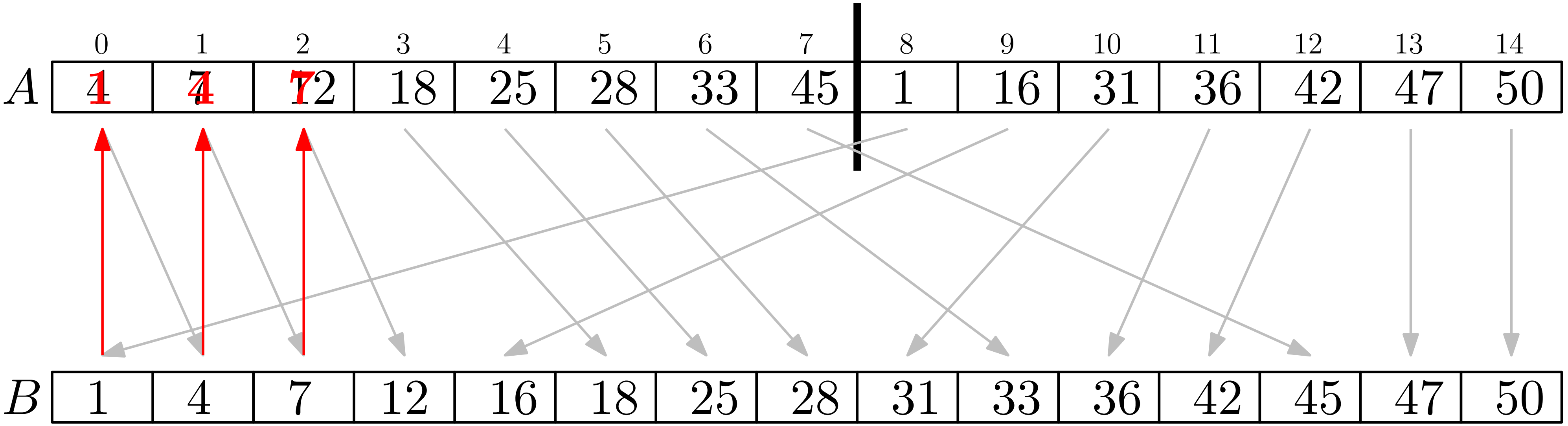
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

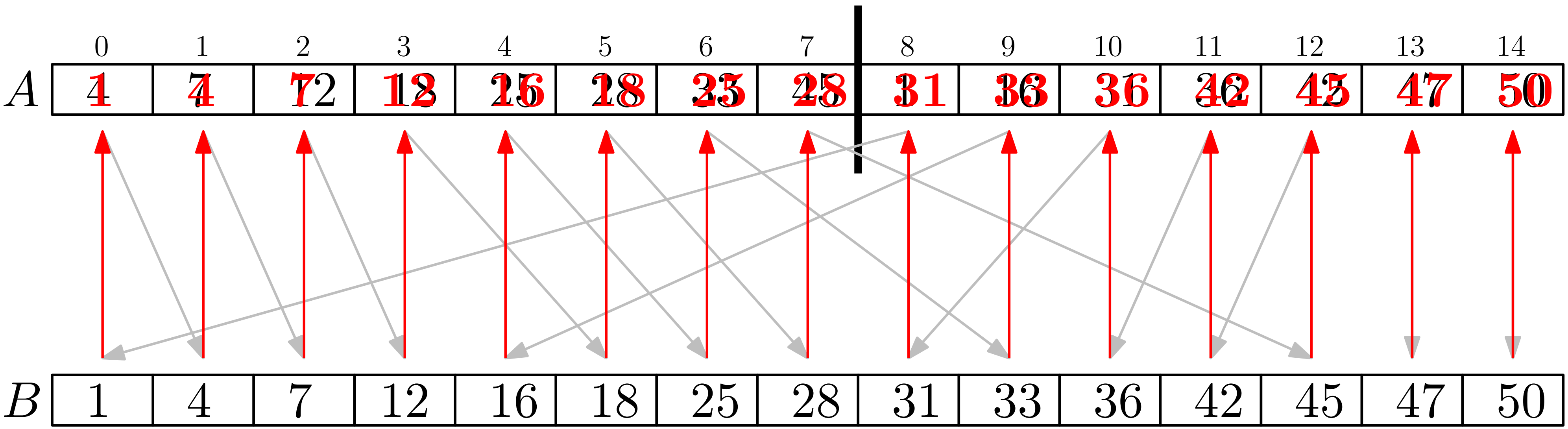
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

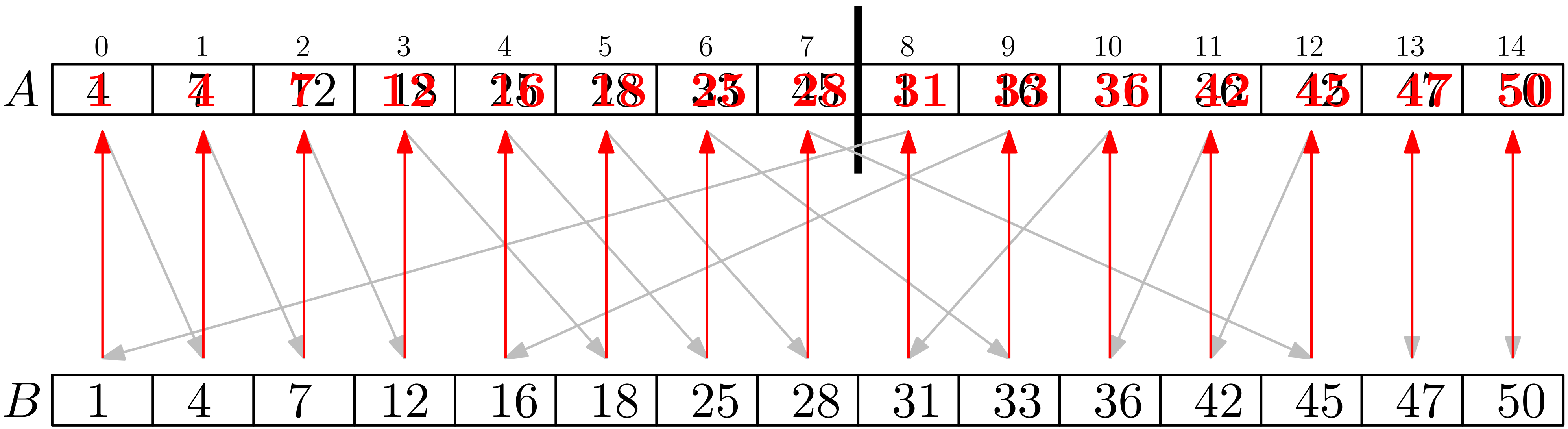
Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

NB: Lidt anderledes i CLRS: kopiér først halvdelene over i to andre arrays og merge tilbage i A.

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	i								j						
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B															

Nyt array B.

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or $(i \leq q$ and $A[i] \leq A[j])$
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$
Højre del: $A[q + 1 \ldots r]$
Ovenstående kald:
Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	i									j					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1														

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p . . . r]
```

Venstre del: $A[p \ldots q]$

Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	i								j					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

A															
	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B															
	1	4													

Nyt array B.

Merge(A, p, q, r)
- let B be an array of size $r - p + 1$
- $i = p$
- $j = q + 1$
- for $k = 0$ to $r - p$
  - if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
    - $B[k] = A[i]$
    - $i = i + 1$
  - else
    - $B[k] = A[j]$
    - $j = j + 1$
- copy B to $A[p \dots r]$

Venstre del: $A[p \dots q]$

Højre del: $A[q + 1 \dots r]$

Ovenstående kald:

Merge($A, 0, 7, 14$)

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2 (i)	3	4	5	6	7	8	9 (j)	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1	4	7												

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p ... r]
```

Venstre del: $A[p \ldots q]$

Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

				i						j					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50
B	1	4	7	12											

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p ... r]
```

Venstre del: $A[p \ldots q]$

Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

				i							j				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1	4	7	12	16										

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p...r]
```

Venstre del: $A[p \ldots q]$

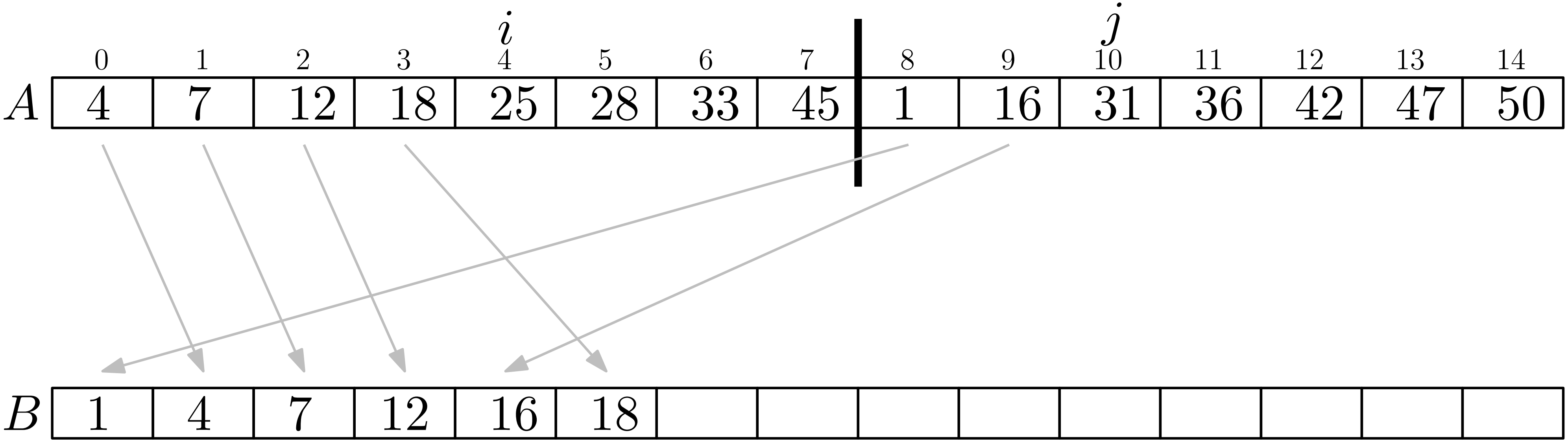
Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2	3	4	5 i	6	7	8	9	10 j	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1	4	7	12	16	18	25								

Nyt array B.

Merge(A, p, q, r)
    let B be an array of size $r - p + 1$
    $i = p$
    $j = q + 1$
    for $k = 0$ to $r - p$
        if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
            $B[k] = A[i]$
            $i = i + 1$
        else
            $B[k] = A[j]$
            $j = j + 1$
    copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

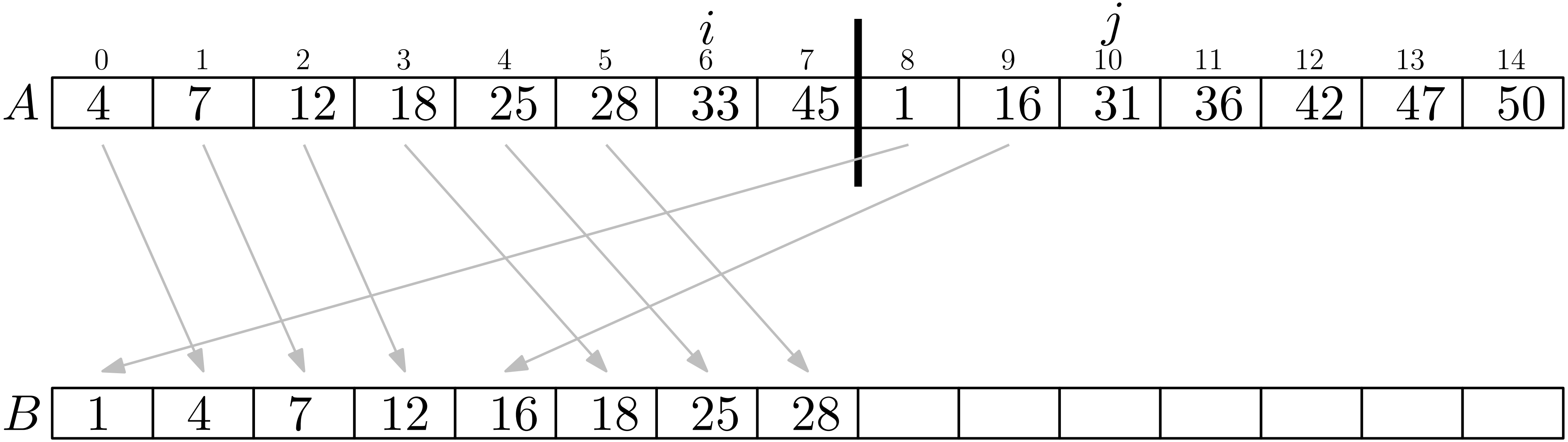
Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p...r]
```

Venstre del: $A[p \dots q]$

Højre del: $A[q+1 \dots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2	3	4	5	6 (i)	7	8	9	10	11 (j)	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50
B	1	4	7	12	16	18	25	28	31						

Nyt array B.

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2	3	4	5	6	7 (i)	8	9	10	11 (j)	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1	4	7	12	16	18	25	28	31	33					

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p ... r]
```

Venstre del: $A[p \dots q]$

Højre del: $A[q+1 \dots r]$

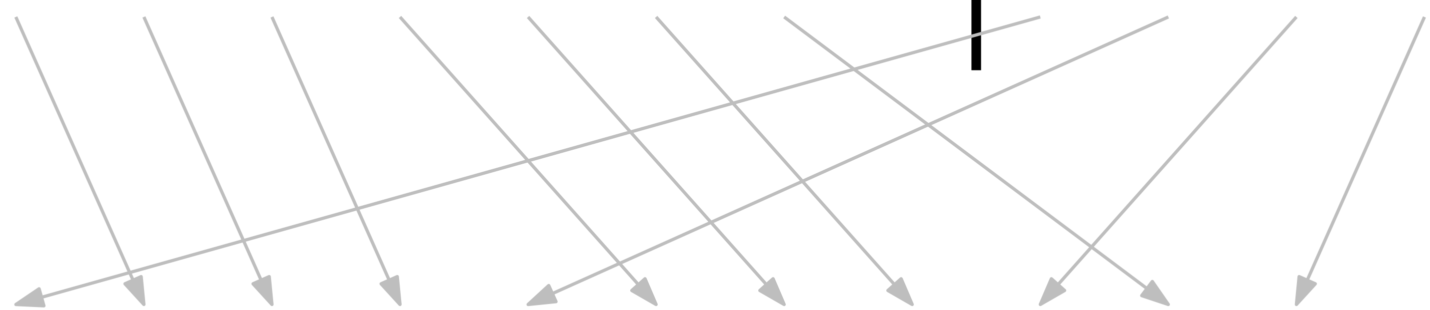
Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
								i					j		
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50



B	1	4	7	12	16	18	25	28	31	33	36				

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p ... r]
```

Venstre del: $A[p \ldots q]$

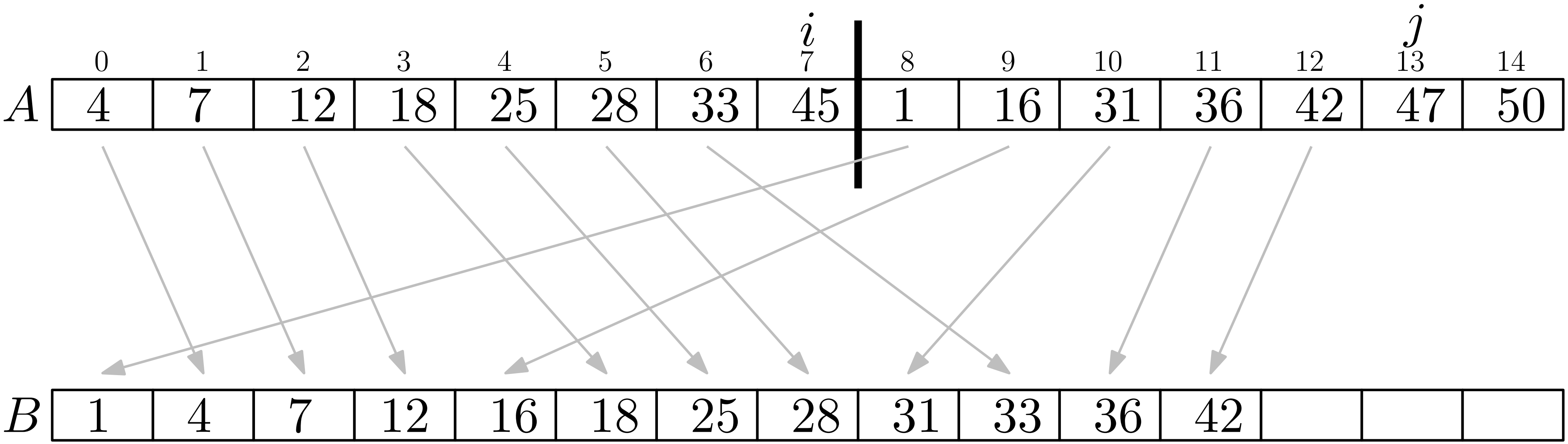
Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or $(i \leq q$ and $A[i] \leq A[j])$
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

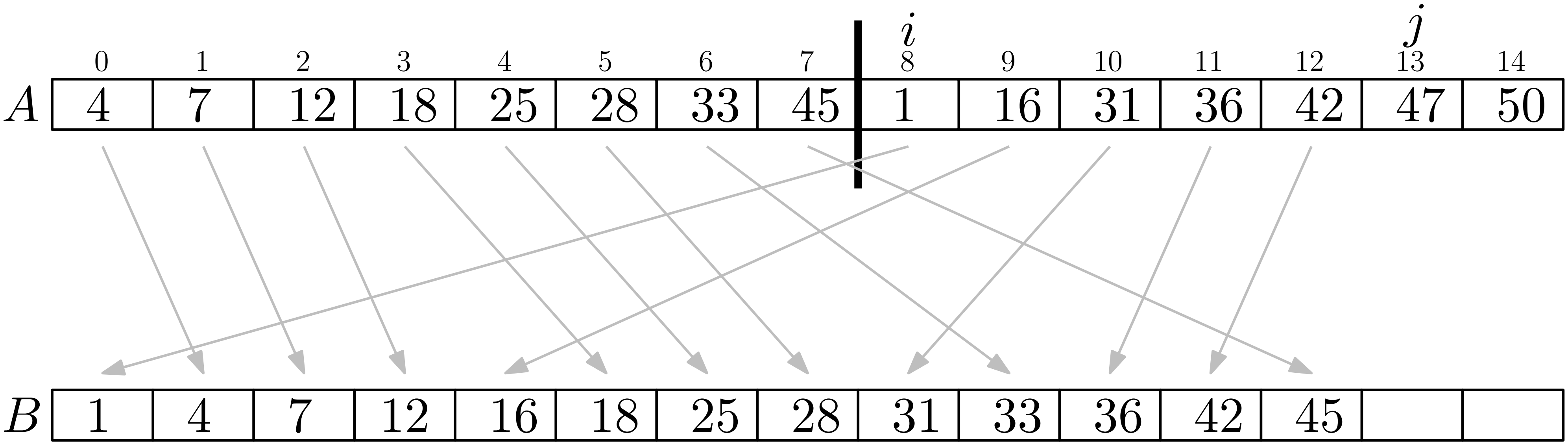
Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge(A, p, q, r)
 let B be an array of size $r - p + 1$
 $i = p$
 $j = q + 1$
 for $k = 0$ to $r - p$
  if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
   $B[k] = A[i]$
   $i = i + 1$
  else
   $B[k] = A[j]$
   $j = j + 1$
 copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

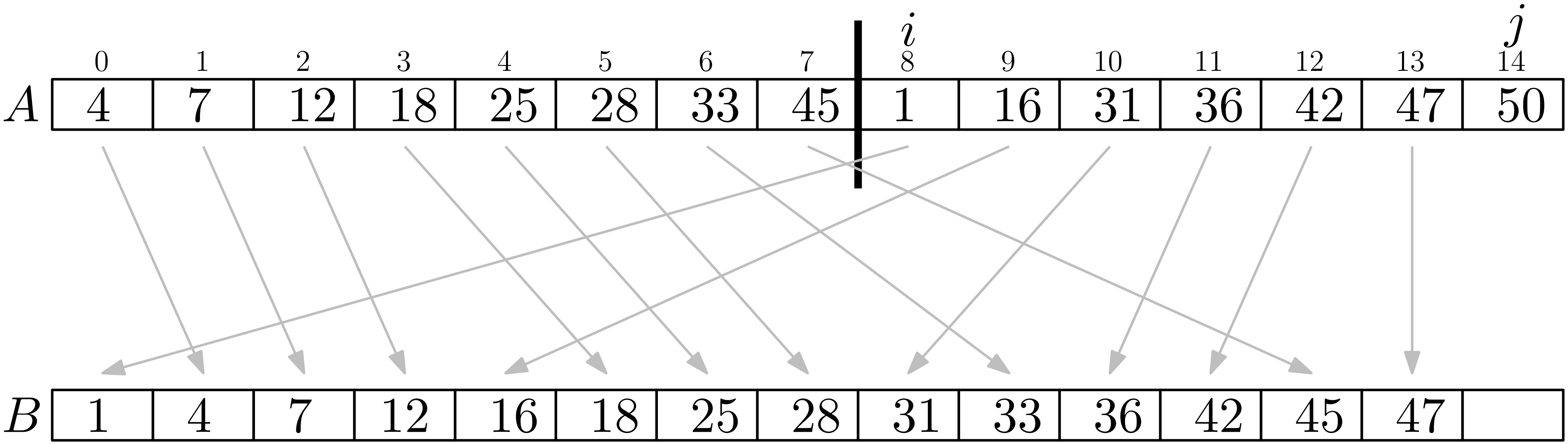
Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge($A, 0, 7, 14$)

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or $(i \leq q$ and $A[i] \leq A[j])$
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

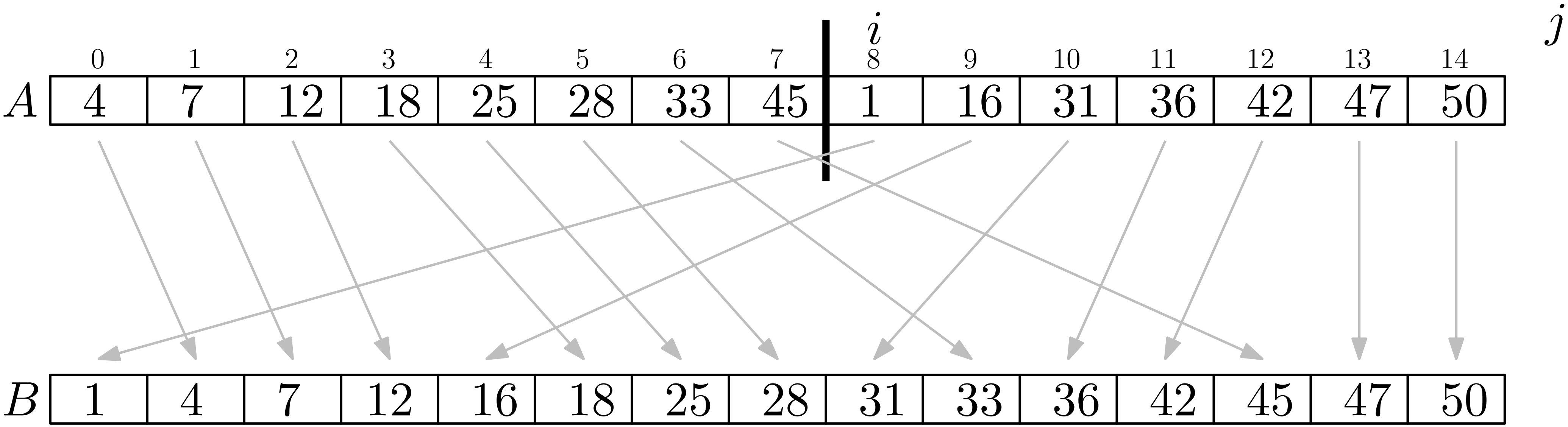
Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p . . . r]
```

Venstre del: $A[p \ldots q]$

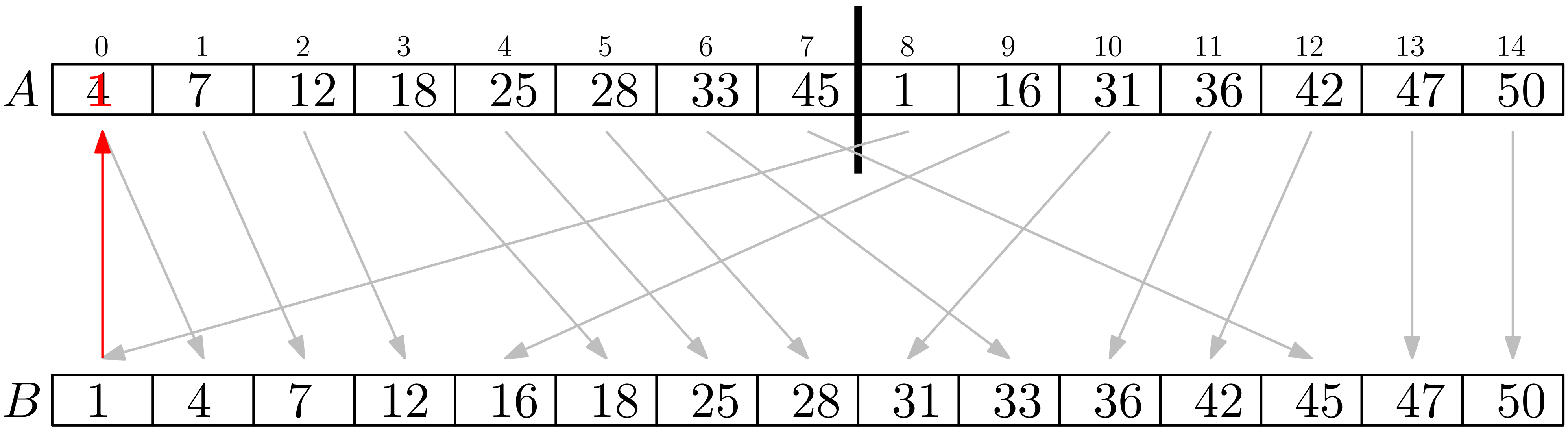
Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p . . . r]
```

Venstre del: $A[p \dots q]$

Højre del: $A[q+1 \dots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4 1	7 4	12	18	25	28	33	45	1	16	31	36	42	47	50

B	1	4	7	12	16	18	25	28	31	33	36	42	45	47	50

Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p...r]
```

Venstre del: $A[p \ldots q]$

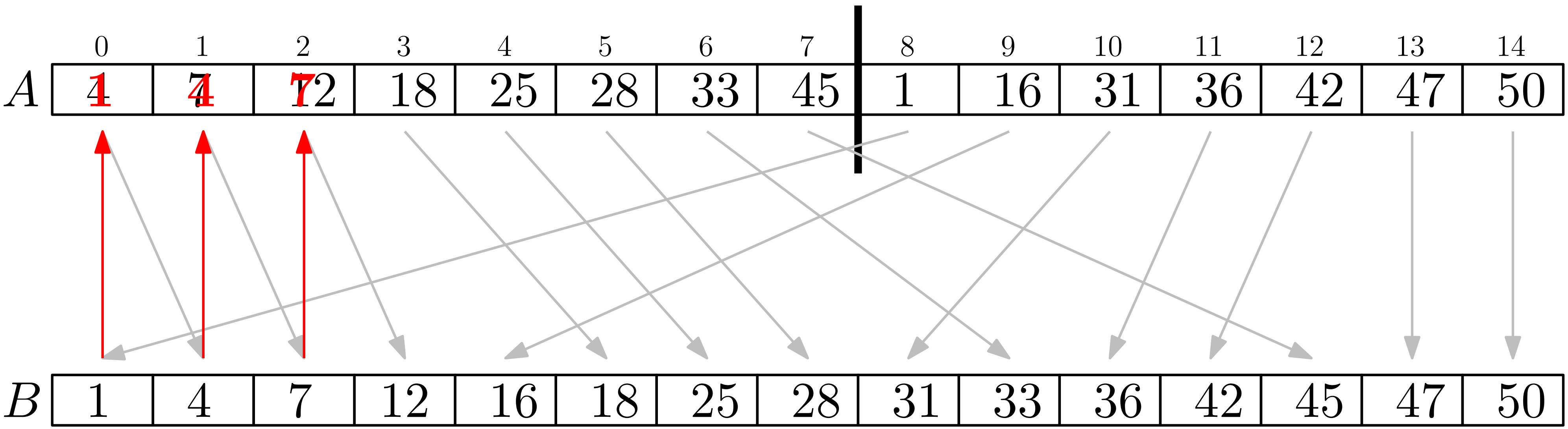
Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p...r]
```

Venstre del: $A[p \ldots q]$

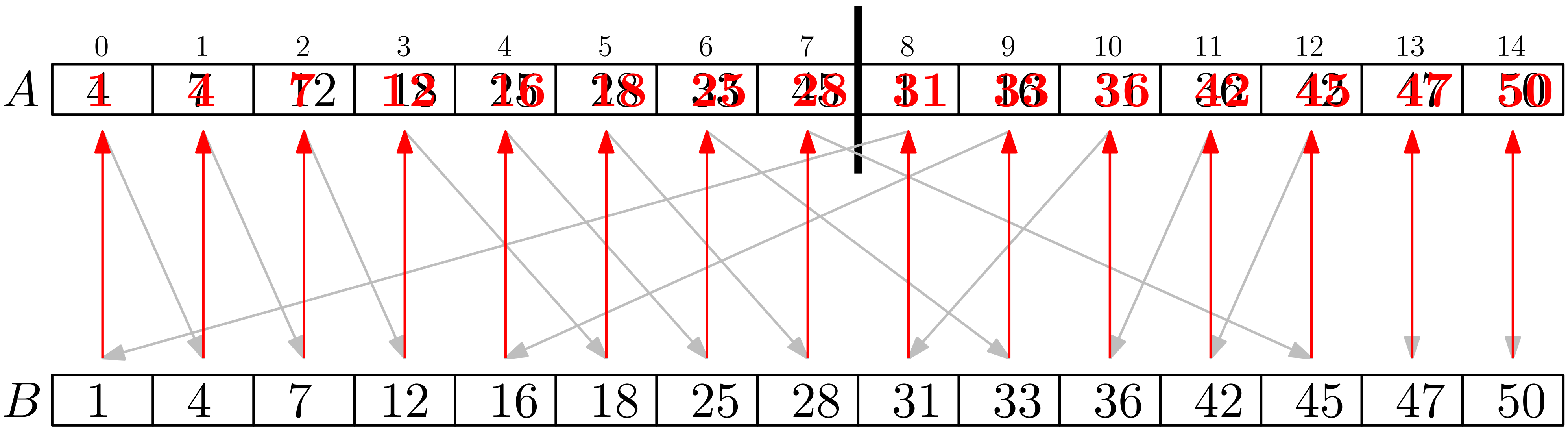
Højre del: $A[q+1 \ldots r]$

Ovenstående kald:

Merge$(A, 0, 7, 14)$

Merge sort

Antag hver halvdel af A er sorteret. Vi merger:



Nyt array B.

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$

Venstre del: $A[p \ldots q]$

Højre del: $A[q + 1 \ldots r]$

Ovenstående kald:

Merge($A, 0, 7, 14$)

Køretid

```
Merge(A, p, q, r)
   let B be an array of size r − p + 1
   i = p
   j = q + 1
   for k = 0 to r − p
      if j > r or (i ≤ q and A[i] ≤ A[j])
         B[k] = A[i]
         i = i + 1
      else
         B[k] = A[j]
         j = j + 1
   copy B to A[p . . . r]
```

Arbejde ved kald Merge($A, 0, \lfloor \frac{n}{2} \rfloor, n-1$):
n iterationer af for-løkke, hver konstant tid
n gange kopiering fra B til A
I alt: $\Theta(n)$ tid.

Køretid

```
Merge(A, p, q, r)
  let B be an array of size r − p + 1
  i = p
  j = q + 1
  for k = 0 to r − p
    if j > r or (i ≤ q and A[i] ≤ A[j])
      B[k] = A[i]
      i = i + 1
    else
      B[k] = A[j]
      j = j + 1
  copy B to A[p...r]
```

Arbejde ved kald Merge$(A, 0, \lfloor \frac{n}{2} \rfloor, n-1)$:

n iterationer af for-løkke, hver konstant tid

n gange kopiering fra B til A

I alt: $\Theta(n)$ tid.

Arbejde ved kald Merge(A, p, q, r):

$\Theta(n')$ tid, hvor $n' = r - p + 1$.

Merge Sort idé

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Merge Sort idé

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Merge Sort idé

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Sortér halvdele rekursivt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

Merge Sort idé

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Sortér halvdele rekursivt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

Merge

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	1	4	7	12	16	18	25	28	31	33	36	42	45	47	50

Merge Sort idé

Merge-Sort(A, p, r)
  if $p < r$
    $q = \lfloor \frac{p+r}{2} \rfloor$
    Merge-Sort(A, p, q)
    Merge-Sort($A, q+1, r$)
    Merge(A, p, q, r)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Sortér halvdele rekursivt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

Merge

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	1	4	7	12	16	18	25	28	31	33	36	42	45	47	50

Merge Sort idé

Merge-Sort(A, p, r)
  if $p < r$
    $q = \lfloor \frac{p+r}{2} \rfloor$
    Merge-Sort(A, p, q)
    Merge-Sort$(A, q+1, r)$
    Merge(A, p, q, r)

Divide and conquer (del og hersk): Del problemet i to dele, løs dem, og kombinér løsningerne.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Sortér halvdele rekursivt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

Merge

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	1	4	7	12	16	18	25	28	31	33	36	42	45	47	50

Merge Sort idé

```
Merge-Sort(A, p, r)
  if p < r
    q = ⌊(p+r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q+1, r)
    Merge(A, p, q, r)
```

Divide and conquer (del og hersk): Del problemet i to dele, løs dem, og kombinér løsningerne.

Tilfældet $p == r$: Vi kigger på et enkelt tal. Allerede sorteret!

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	33	4	25	28	45	18	7	12	36	1	47	42	50	16	31

Sortér halvdele rekursivt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	4	7	12	18	25	28	33	45	1	16	31	36	42	47	50

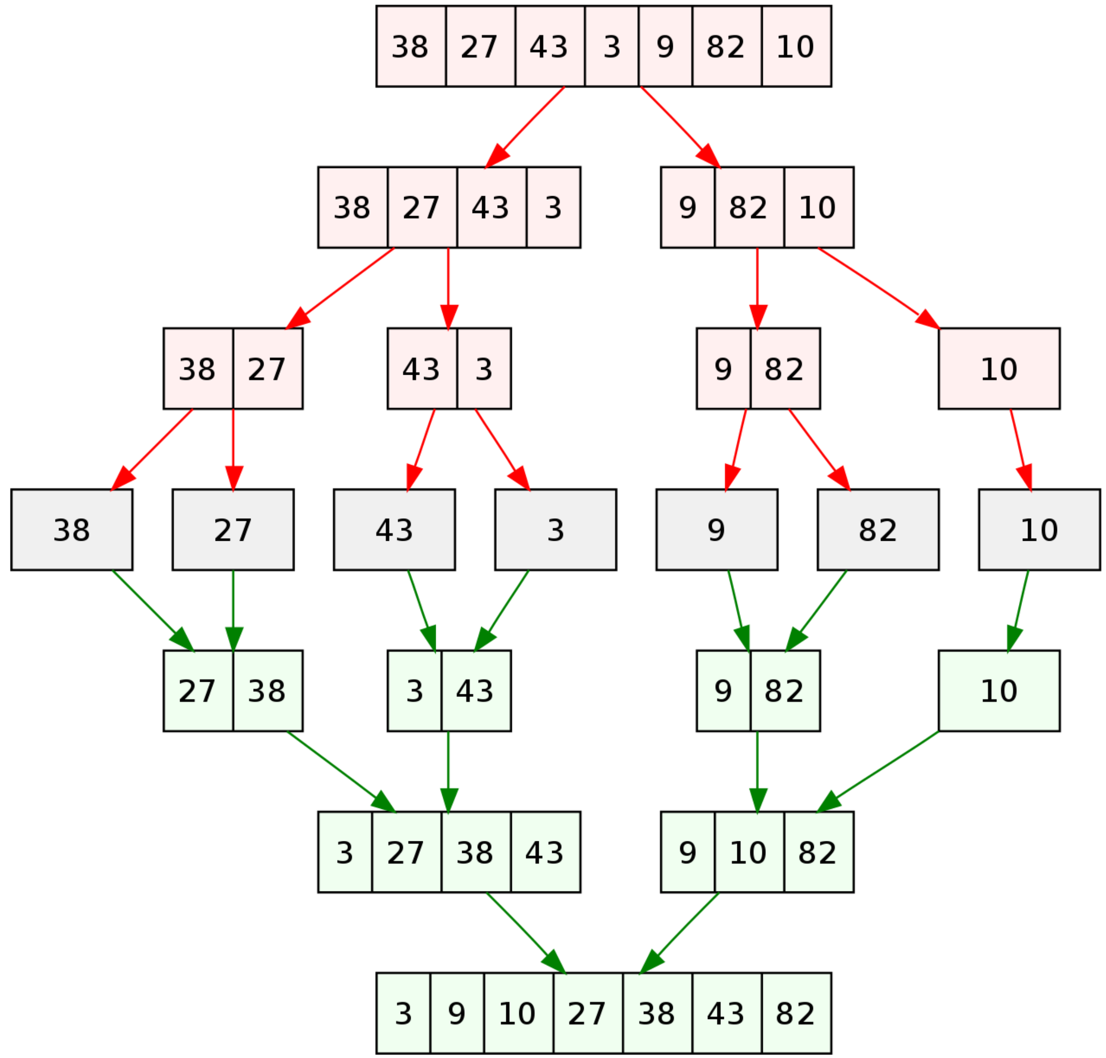
Merge

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	1	4	7	12	16	18	25	28	31	33	36	42	45	47	50

Eksempel

Merge-Sort(A, p, r)
  if $p < r$
    $q = \lfloor \frac{p+r}{2} \rfloor$
    Merge-Sort(A, p, q)
    Merge-Sort$(A, q+1, r)$
    Merge(A, p, q, r)

Merge(A, p, q, r)
  let B be an array of size $r - p + 1$
  $i = p$
  $j = q + 1$
  for $k = 0$ to $r - p$
    if $j > r$ or ($i \leq q$ and $A[i] \leq A[j]$)
      $B[k] = A[i]$
      $i = i + 1$
    else
      $B[k] = A[j]$
      $j = j + 1$
  copy B to $A[p \ldots r]$



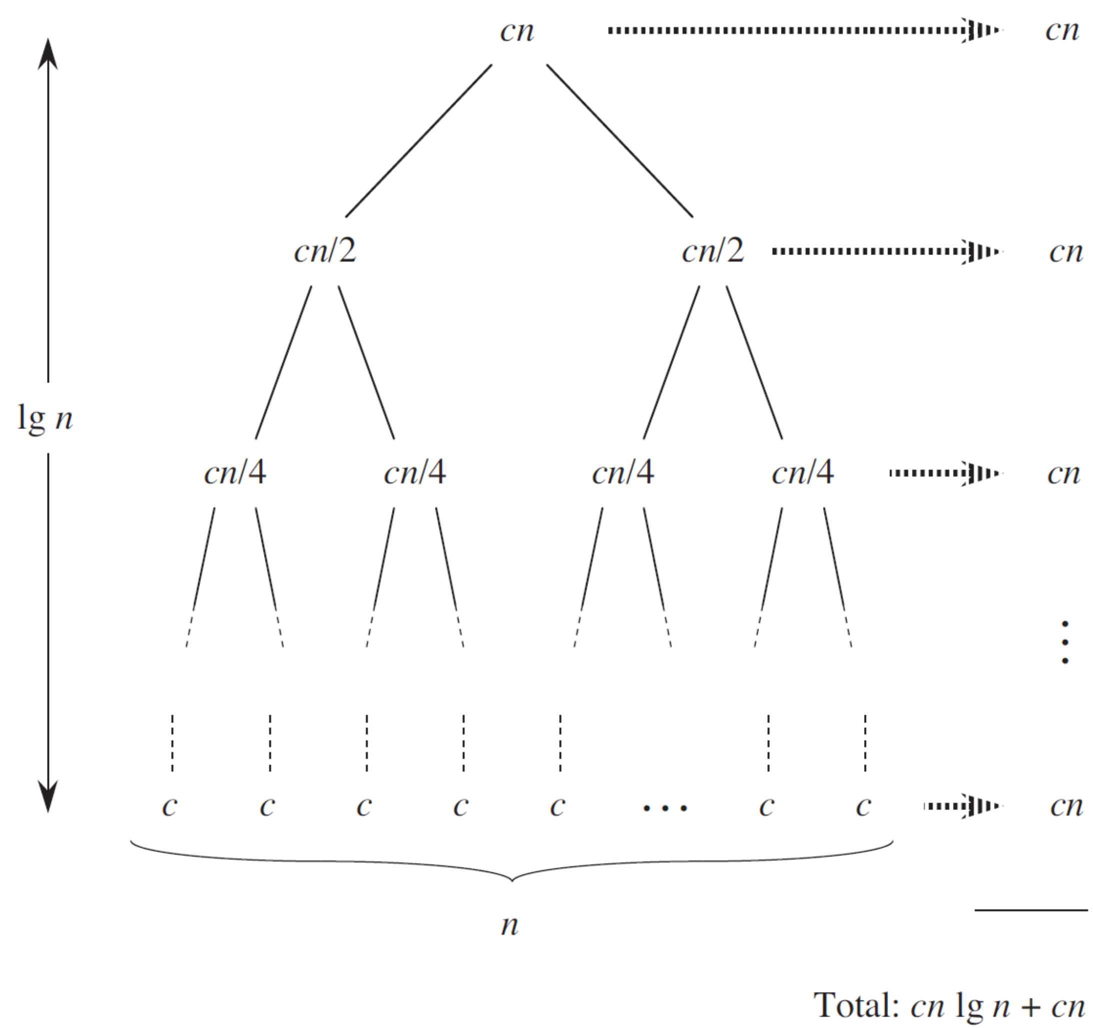
Første kald:
Merge-Sort$(A, 0, n-1)$

Køretid

Merge-Sort(A, p, r)
  if $p < r$
    $q = \lfloor \frac{p+r}{2} \rfloor$
    Merge-Sort(A, p, q)
    Merge-Sort$(A, q+1, r)$
    Merge(A, p, q, r)

$$T(n) = \begin{cases} c, & n = 1 \\ 2T(n/2) + cn, & n > 1 \end{cases}$$

antager $n = 2^k$, $k \in \mathbb{N}$



I alt: $T(n) = cn \lg n + cn$
$= \Theta(n \log n)$

Opsummering om sortering

Insertion sort: $\Theta(n^2)$ tid, kræver ikke ekstra plads.
Merge sort: $\Theta(n \log n)$ tid, kræver $\Theta(n)$ ekstra plads.

I DMA uge 4: Heap sort, bruger $\Theta(n \log n)$ tid, kræver ikke ekstra array (*in place*-algoritme).

I DMA uge 6: Ikke muligt at komme under $\Theta(n \log n)$ tid hvis man kun må deducere vha. sammenligninger. Vi skal se på andre sorteringsalgoritmer som kommer under vha. smarte "snydetricks".

Hvorfor er asymptotisk køretid så praktisk?

Antag S1(A) og S2(A) begge sorterer array A af længde n.
S1 har køretid $T_1(n) = n^2$ og S2 har køretid $T_2(n) = 100 \cdot n \log_2 n$.
$T_1(n) = \Theta(n^2)$ og $T_2(n) = \Theta(n \log n)$.

Hvorfor er asymptotisk køretid så praktisk?

Antag S1(A) og S2(A) begge sorterer array A af længde n.
S1 har køretid $T_1(n) = n^2$ og S2 har køretid $T_2(n) = 100 \cdot n \log_2 n$.
$T_1(n) = \Theta(n^2)$ og $T_2(n) = \Theta(n \log n)$.

$n = 20$: $T_1(20) = 400$ og $T_2(20) \approx 8600$, så

$$\frac{T_1(20)}{T_2(20)} \approx 0.05.$$

Hvorfor er asymptotisk køretid så praktisk?

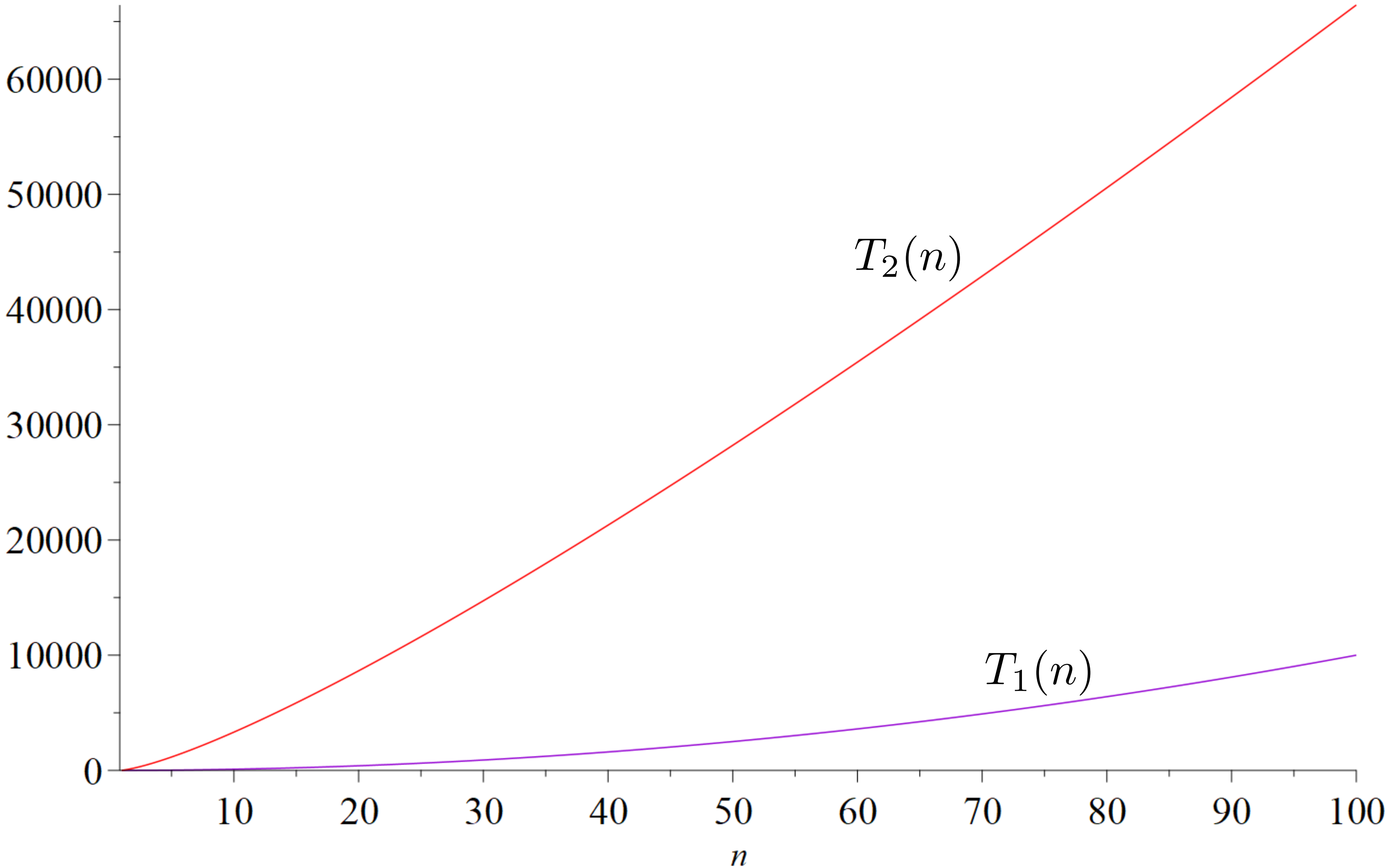
Antag S1(A) og S2(A) begge sorterer array A af længde n.
S1 har køretid $T_1(n) = n^2$ og S2 har køretid $T_2(n) = 100 \cdot n \log_2 n$.
$T_1(n) = \Theta(n^2)$ og $T_2(n) = \Theta(n \log n)$.

$n = 20$: $T_1(20) = 400$ og $T_2(20) \approx 8600$, så

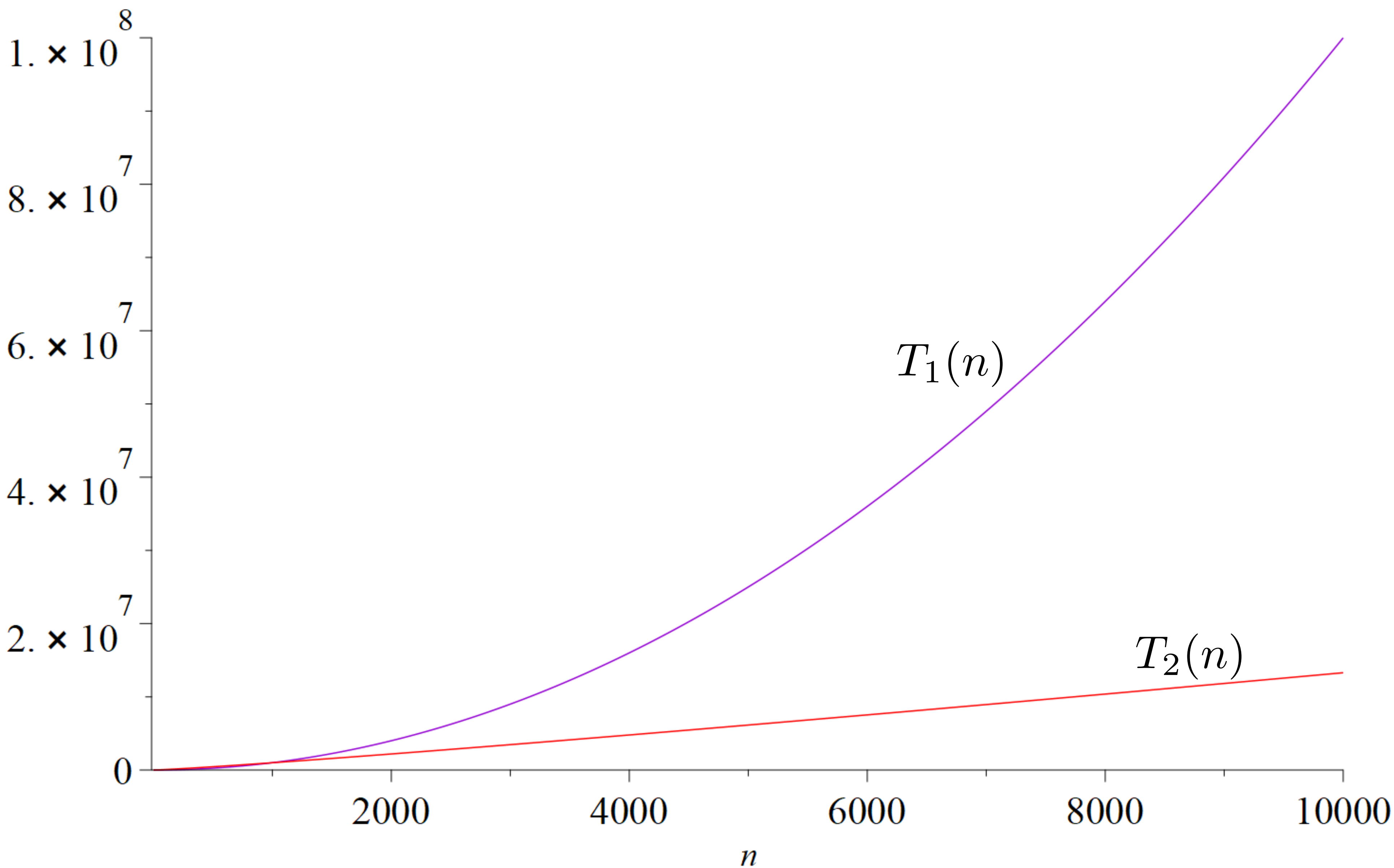
$$\frac{T_1(20)}{T_2(20)} \approx 0.05.$$

$n = 5.8 \cdot 10^6$: $T_1(5.8 \cdot 10^6) \approx 3.4 \cdot 10^{13}$ og $T_2(5.8 \cdot 10^6) \approx 1.3 \cdot 10^{10}$, så

$$\frac{T_1(5.8 \cdot 10^6)}{T_2(5.8 \cdot 10^6)} \approx 2600.$$



60000
50000
40000
30000
20000
10000
0
10
20
30
40
50
60
70
80
90
100
n
$T_2(n)$
$T_1(n)$



$1. \times 10^8$
$8. \times 10^7$
$6. \times 10^7$
$4. \times 10^7$
$2. \times 10^7$
0
2000
4000
6000
8000
10000
n
$T_1(n)$
$T_2(n)$

Hvorfor er vi ligeglade med konstanter?

Hvis køretiden er $T(n) = 100n \log n$ skriver vi $T(n) = \Theta(n \log n)$.
Vi igorerer konstanten 100 fordi:

- Konstanten afhænger af præcis hvordan vi tæller skridt.
- I praksis er det forskelligt hvor lang tid de basale skridt tager.
- Præcist antal skridt afhænger af programmeringssprog.
- Når n bliver stor er det vigtigste den asymptotiske opførsel.

Derfor ignorerer vi også langsomt voksende led.

Hvorfor er vi ligeglade med konstanter?

Hvis køretiden er $T(n) = 100n \log n$ skriver vi $T(n) = \Theta(n \log n)$.
Vi igorerer konstanten 100 fordi:

- Konstanten afhænger af præcis hvordan vi tæller skridt.
- I praksis er det forskelligt hvor lang tid de basale skridt tager.
- Præcist antal skridt afhænger af programmeringssprog.
- Når n bliver stor er det vigtigste den asymptotiske opførsel.

Derfor ignorerer vi også langsomt voksende led.

P.S:

- I praksis kan vi ikke ignorere astronomiske konstanter.
- En asymptotisk langsommere algoritme kan foretrækkes hvis
  - n aldrig bliver meget stor, eller
  - den langsommere algoritme er meget simplere og hurtig nok.
- To algoritmer med samme asymp. køretid behøver ikke være lige gode.