Univerzitet Sinergija

Strukture podataka i algoritmi

leto 2011/12

Analiza rekurzivnih algoritama

- Analiza iterativnih algoritama: prebrojavanje osnovnih instrukcija
- Analiza rekurzivnih algoritama: rešavanje rekurentnih jednačina
- Rekurentnom jednačinom se definiše vrednost funkcije za neki argument pomoću vrednosti iste funkcije za manje argumente

Rekurzivni algoritmi

- Primer: najveći element niza

```
// Ulaz:  niz a, njegov broj elemenata n
// Izlaz: indeks najvećeg elementa niza a
algorithm max(a, n)

   if (n == 1) then // bazni slučaj
      return 1;
   else                // opšti slučaj
      i = max(a, n-1);
      if (a[i] < a[n]) then
         return n;
      else
         return i;
```

Analiza rekurzivnih algoritama

- Vreme izvršavanja $T(n)$ za `max(a,n)`:

$$T(n) = \begin{cases} 1, & \text{ako je } n = 1 \\ T(n-1) + 1, & \text{ako je } n > 1 \end{cases}$$

Analiza rekurzivnih algoritama

- Rešenje za rekurentnu jednačinu $T(n)$:

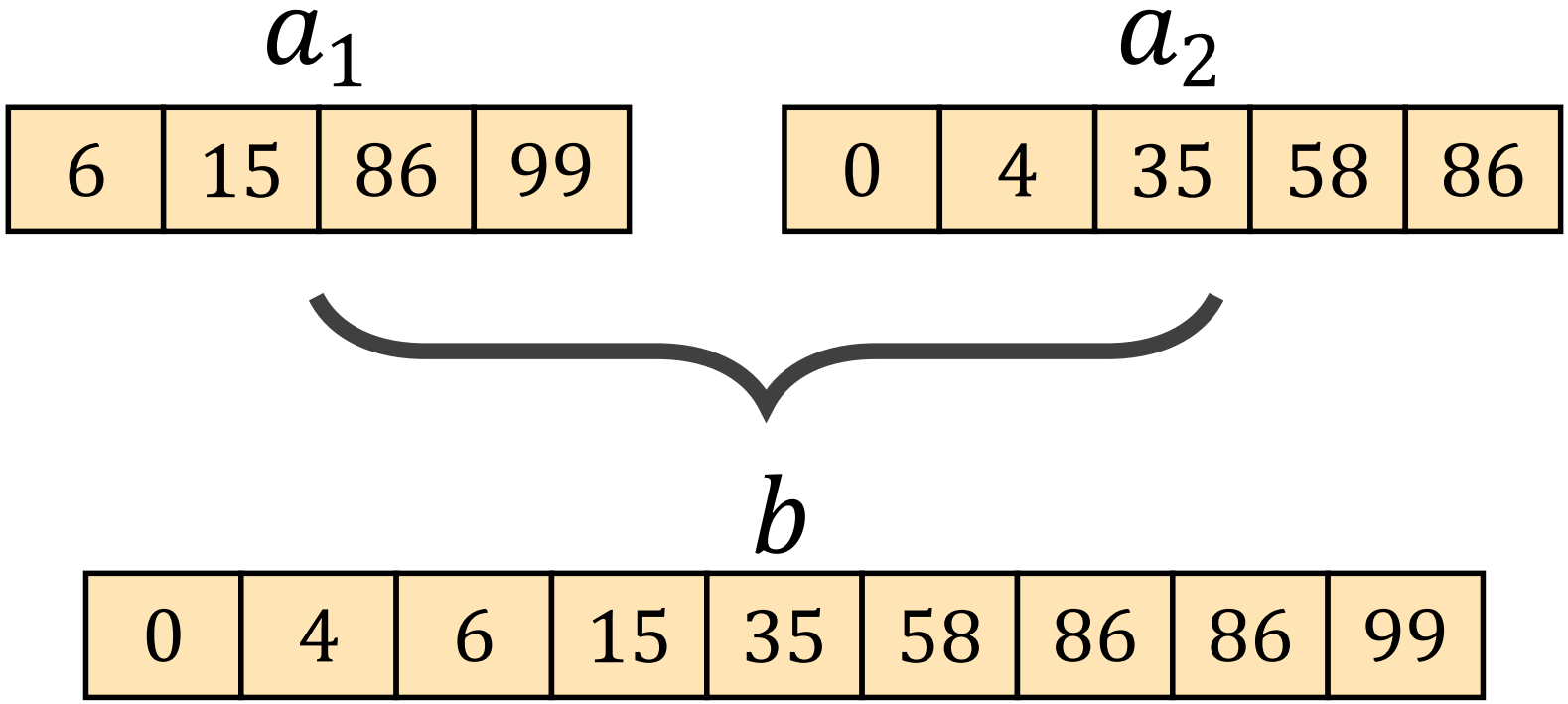
$$
\begin{aligned}
T(n) &= T(n-1) + 1 \\
&= \big(T(n-2) + 1\big) + 1 = T(n-2) + 2 \\
&= \big(T(n-3) + 1\big) + 2 = T(n-3) + 3 \\
&\vdots \\
&= \big(T(1) + 1\big) + n - 2 = T(1) + n - 1 \\
&= 1 + n - 1 \\
&= n
\end{aligned}
$$

Sortiranje objedinjavanjem (*merge-sort*)

- Zasniva se na objedinjavanju (*merge*) dva sortirana niza u jedan sortiran niz

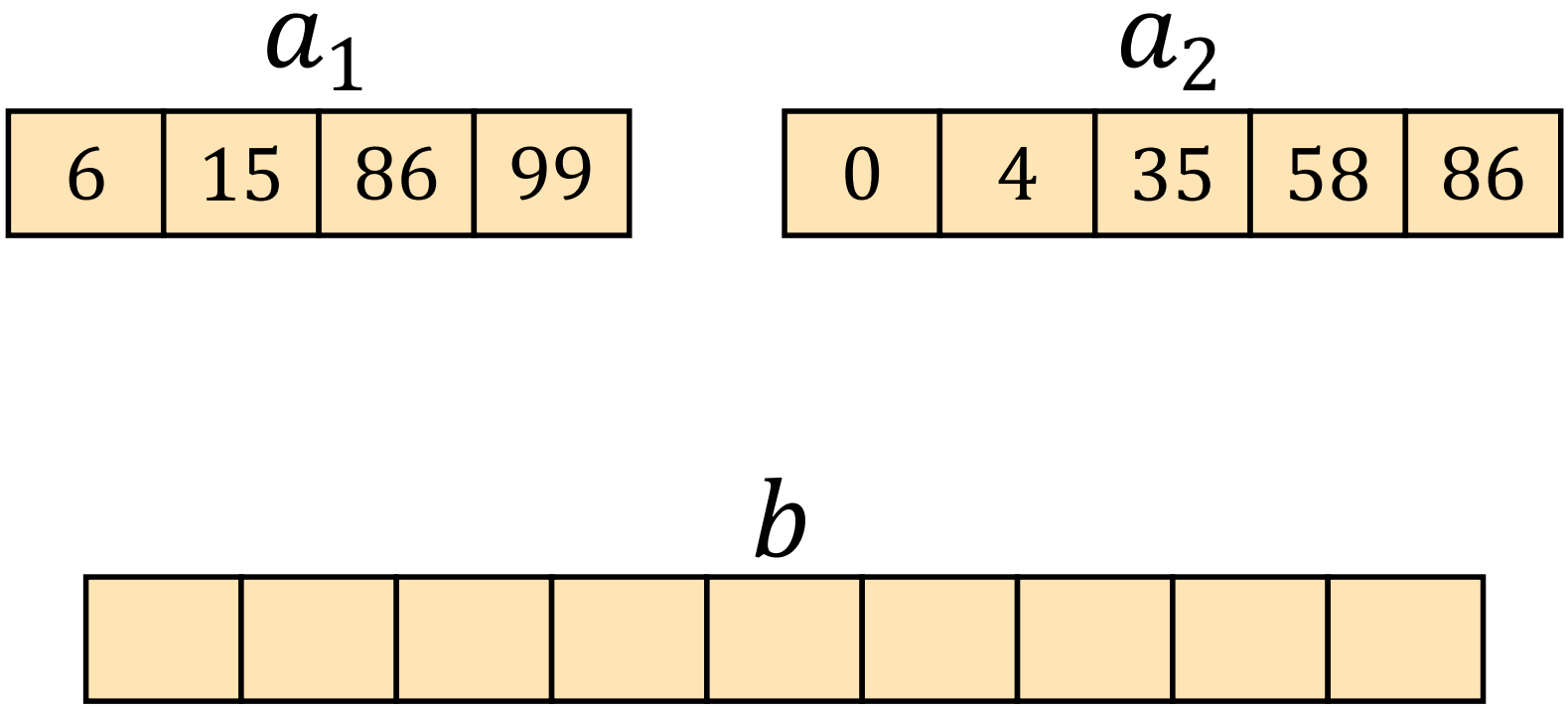
Sortiranje objedinjavanjem (*merge-sort*)

- Zasniva se na objedinjavanju (*merge*) dva sortirana niza u jedan sortiran niz
- Primer:



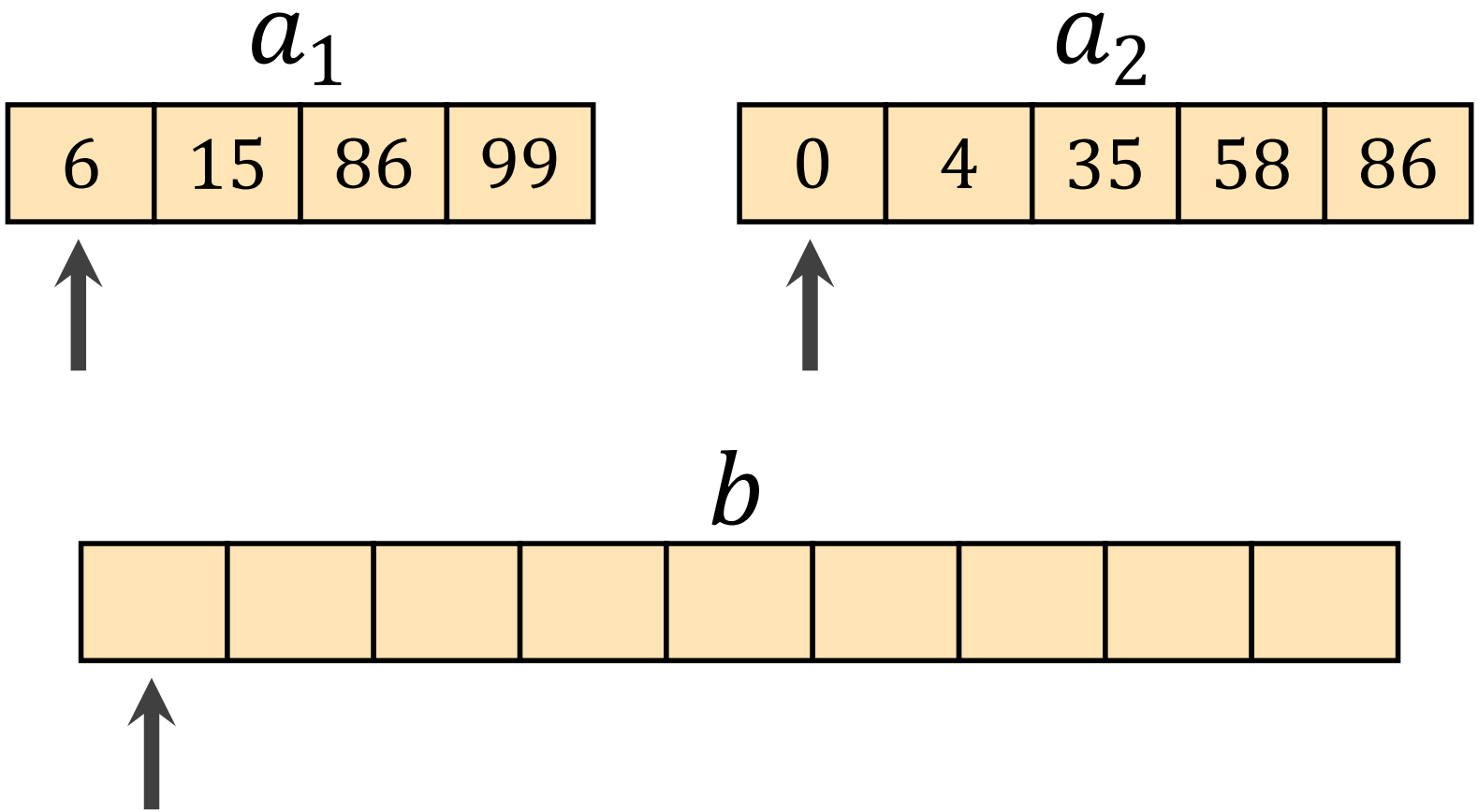
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



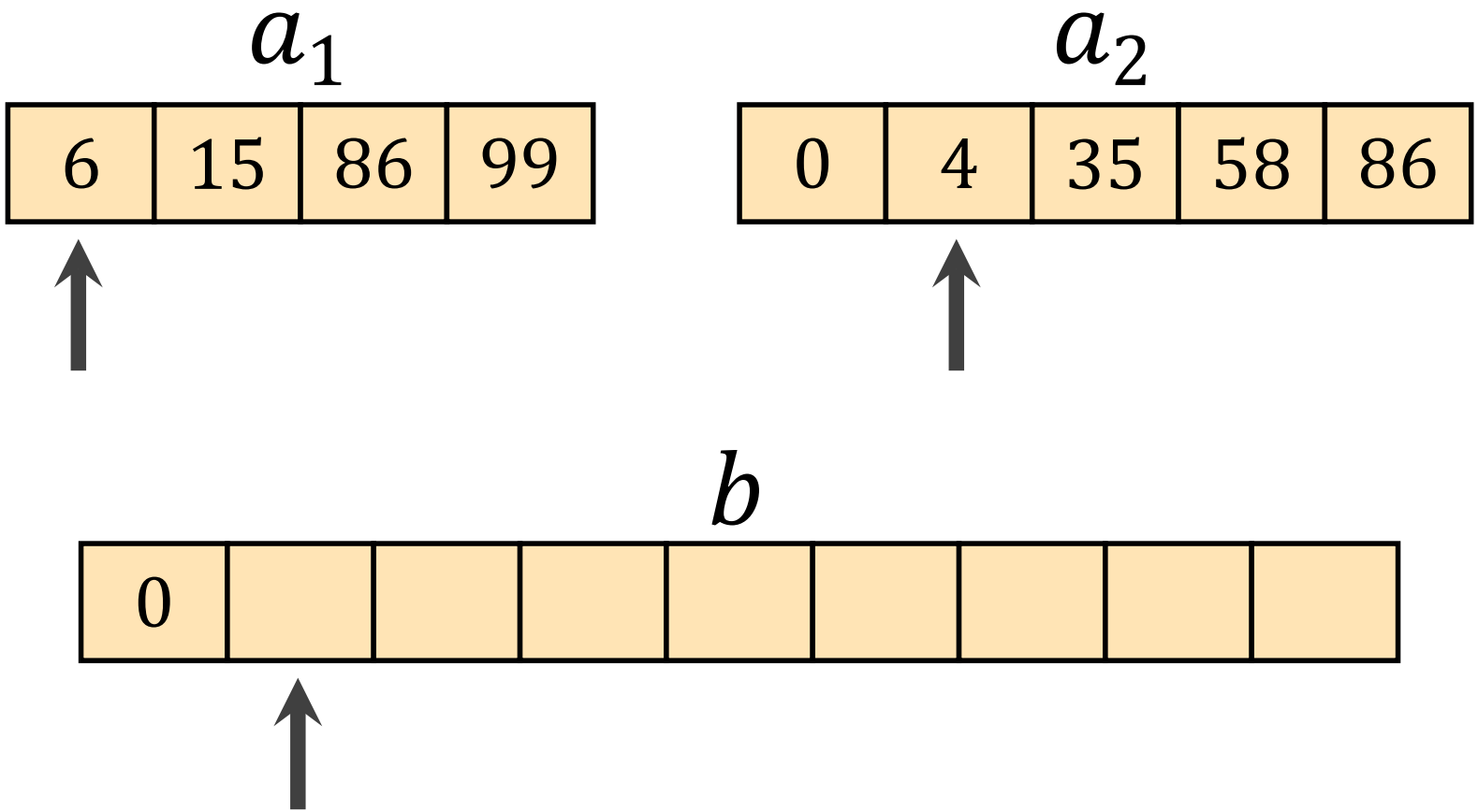
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



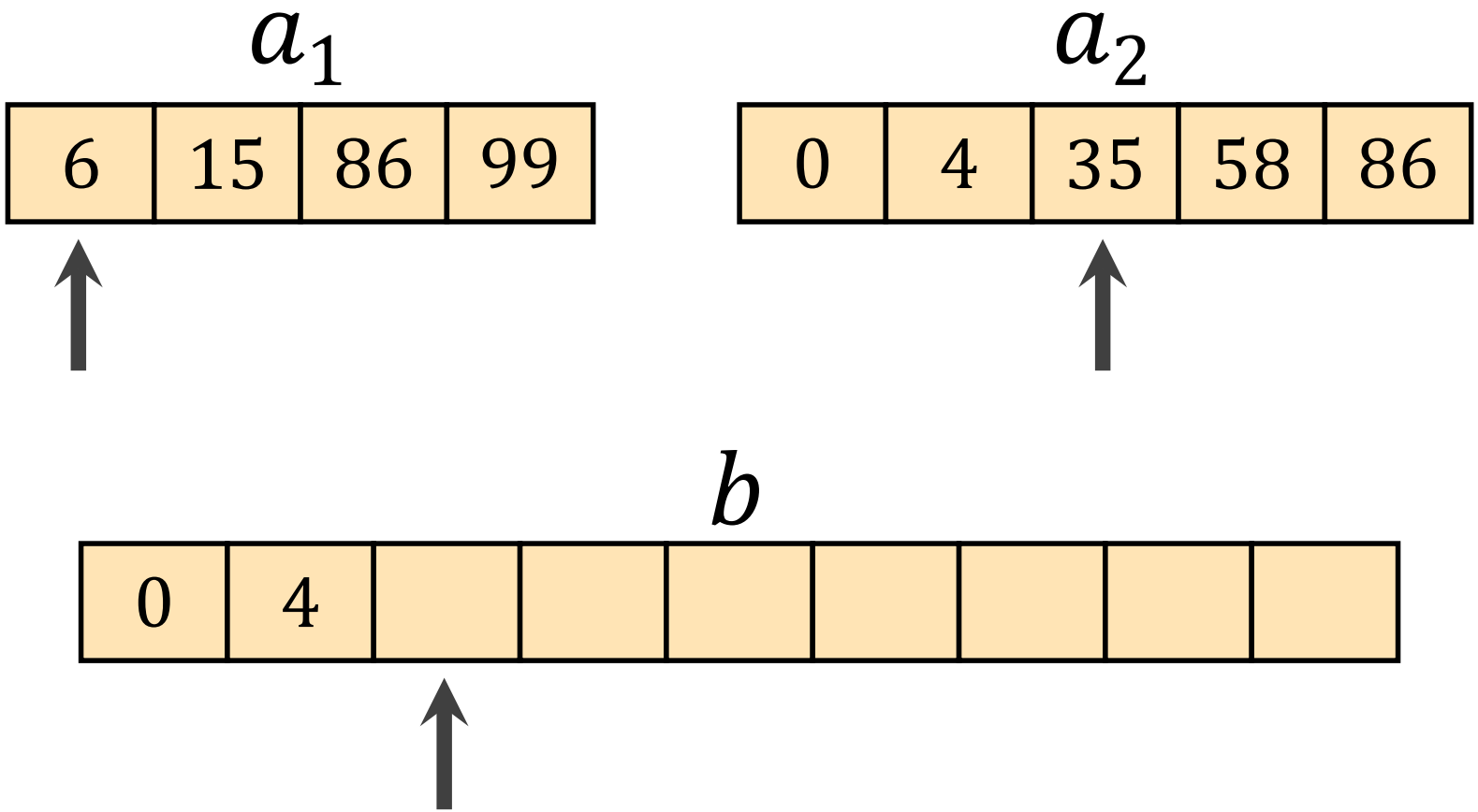
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



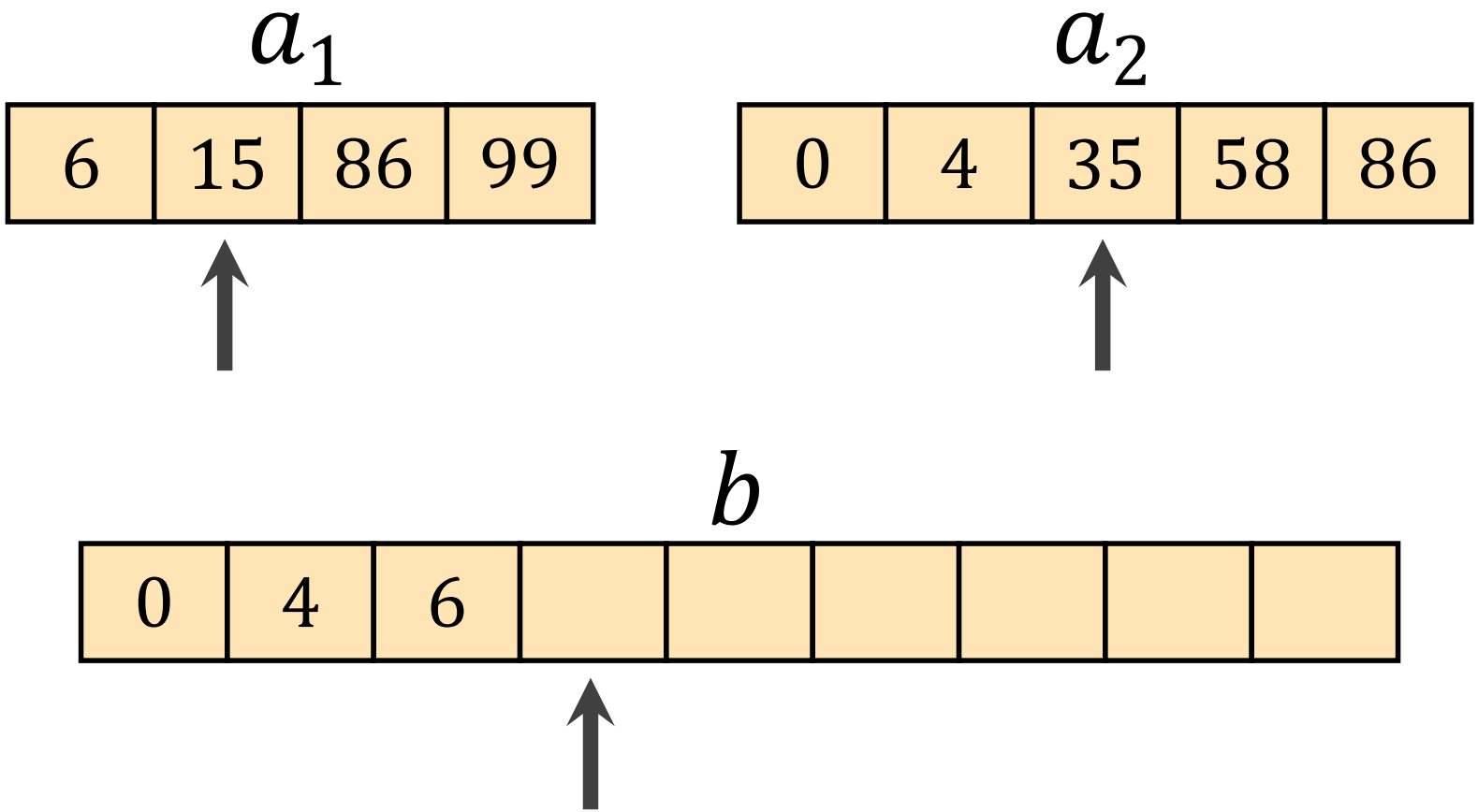
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



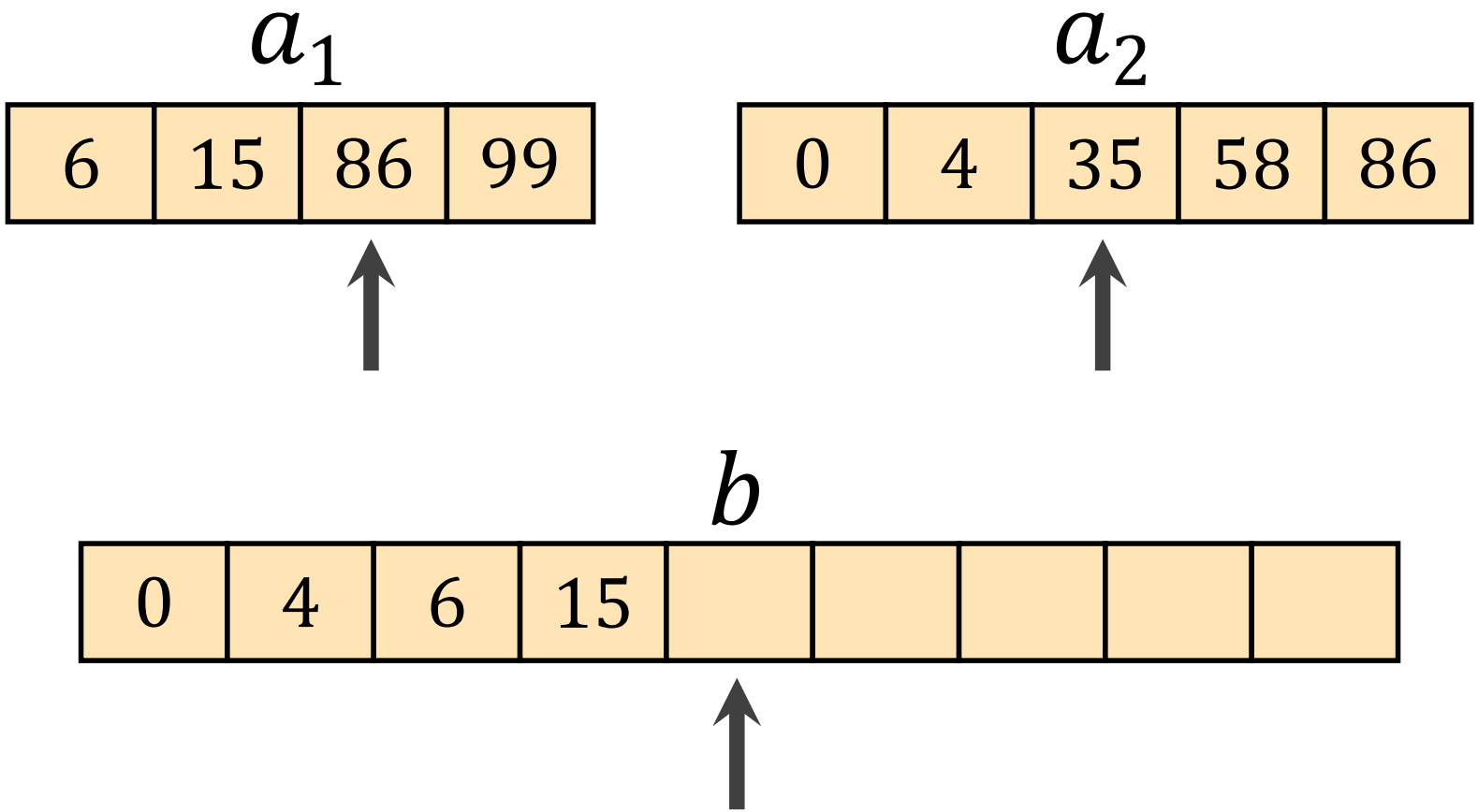
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



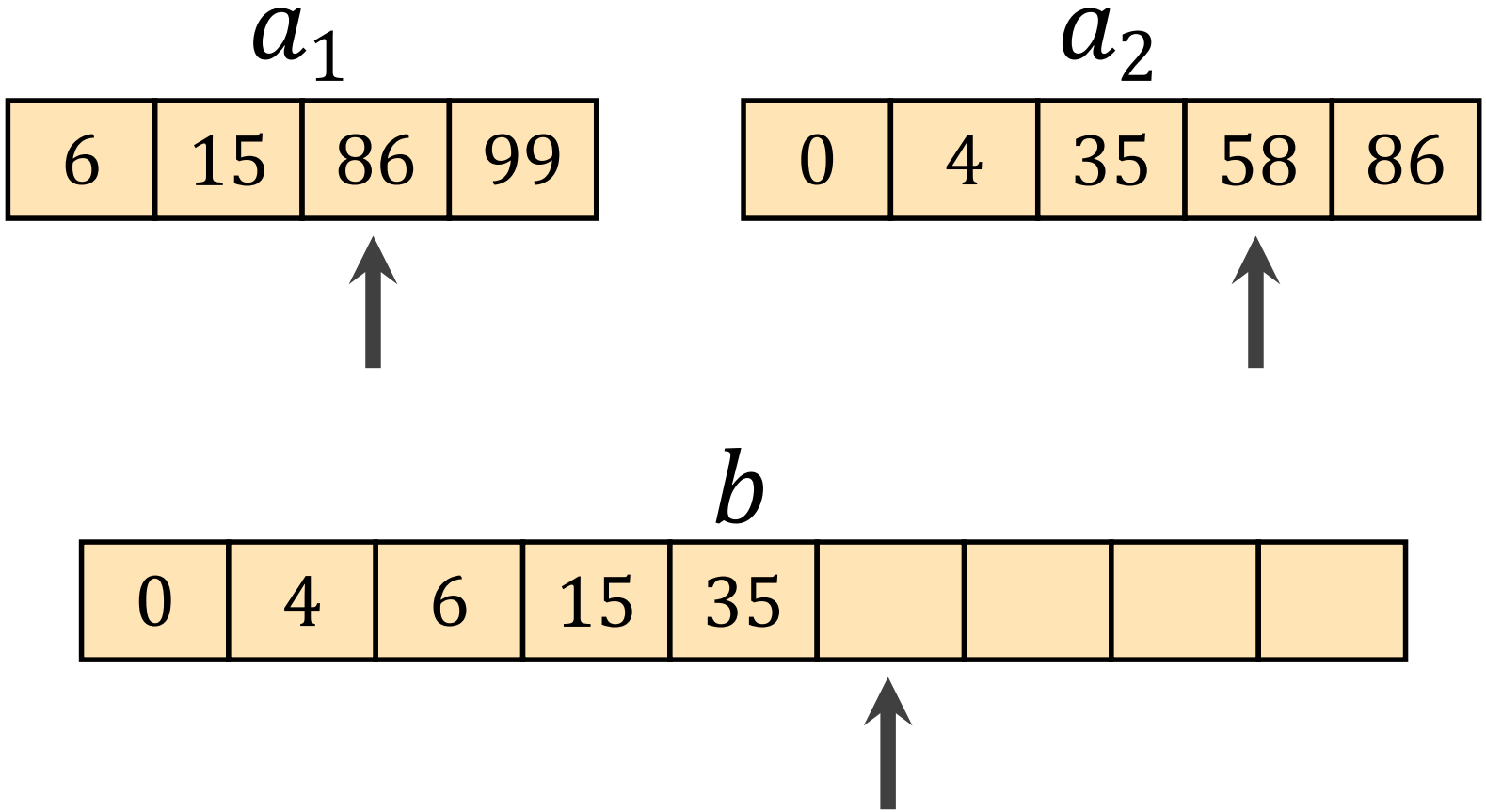
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



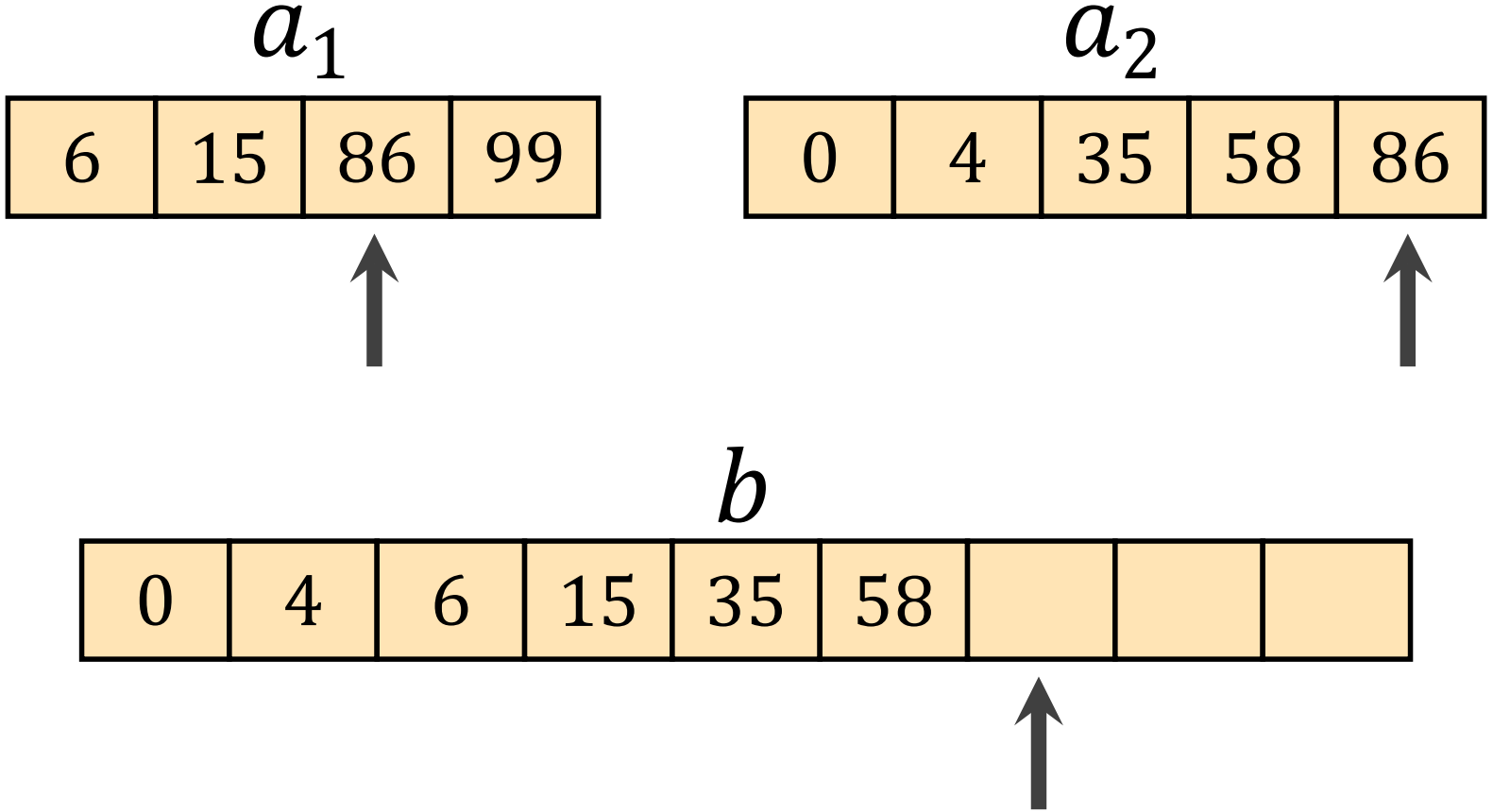
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



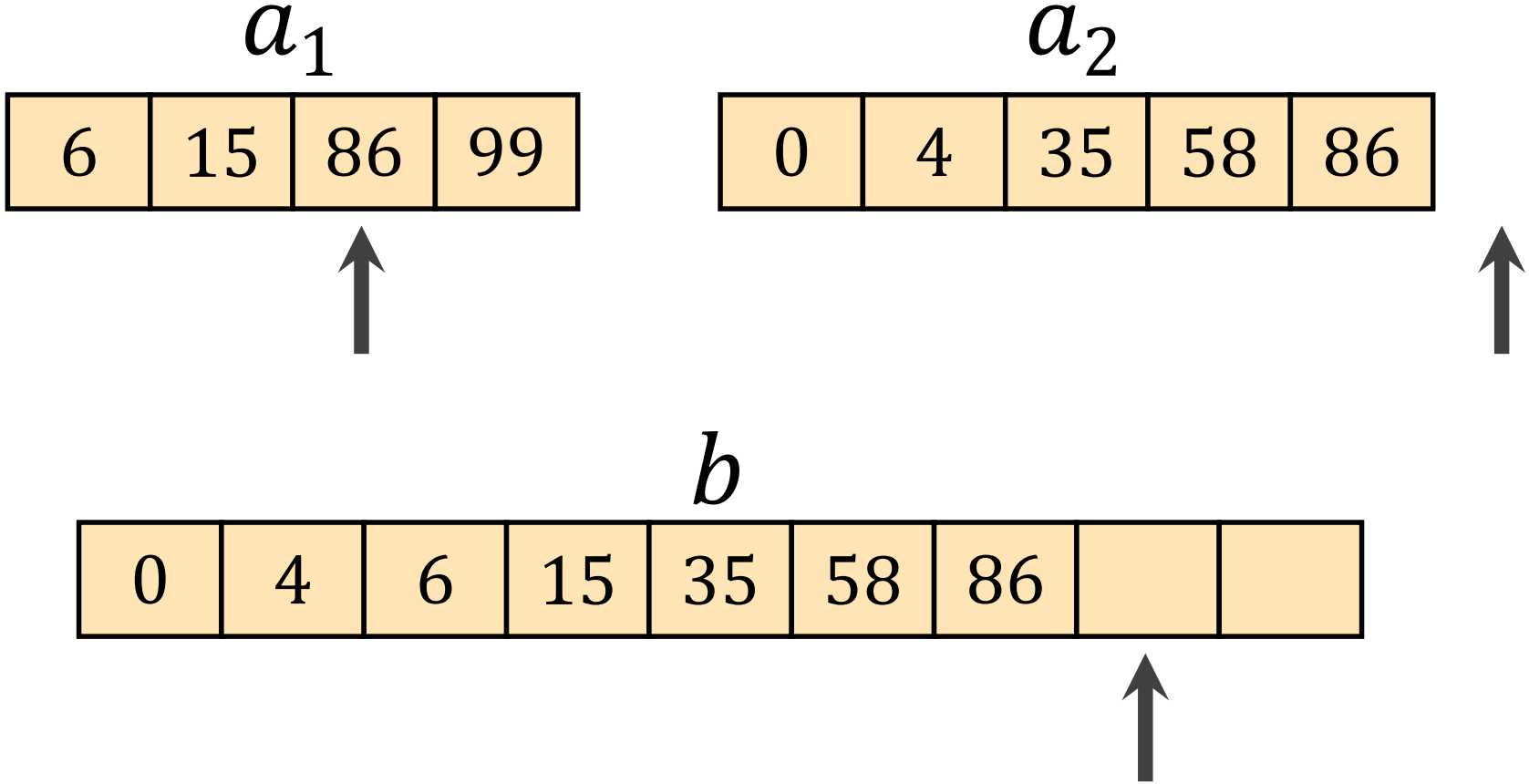
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



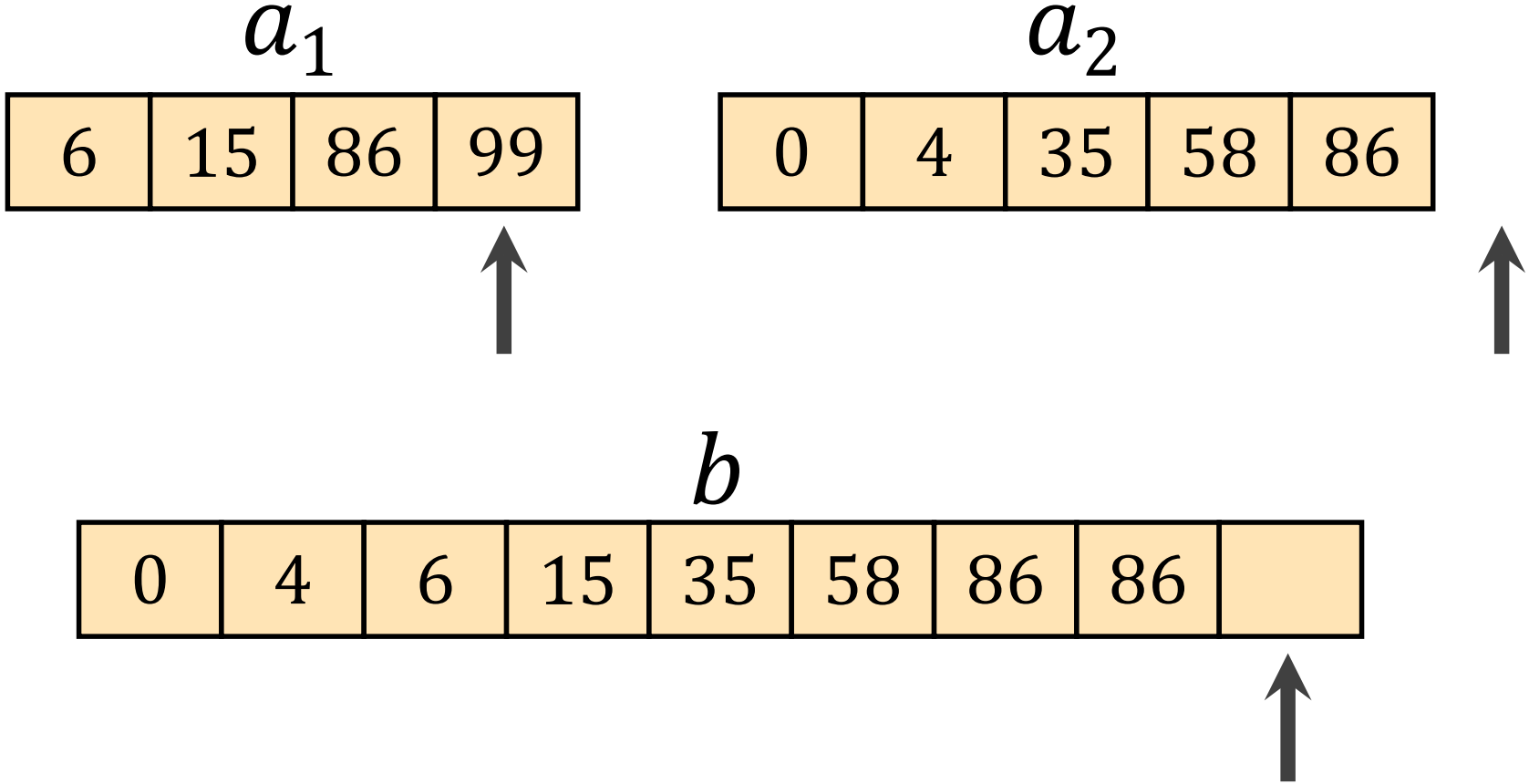
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



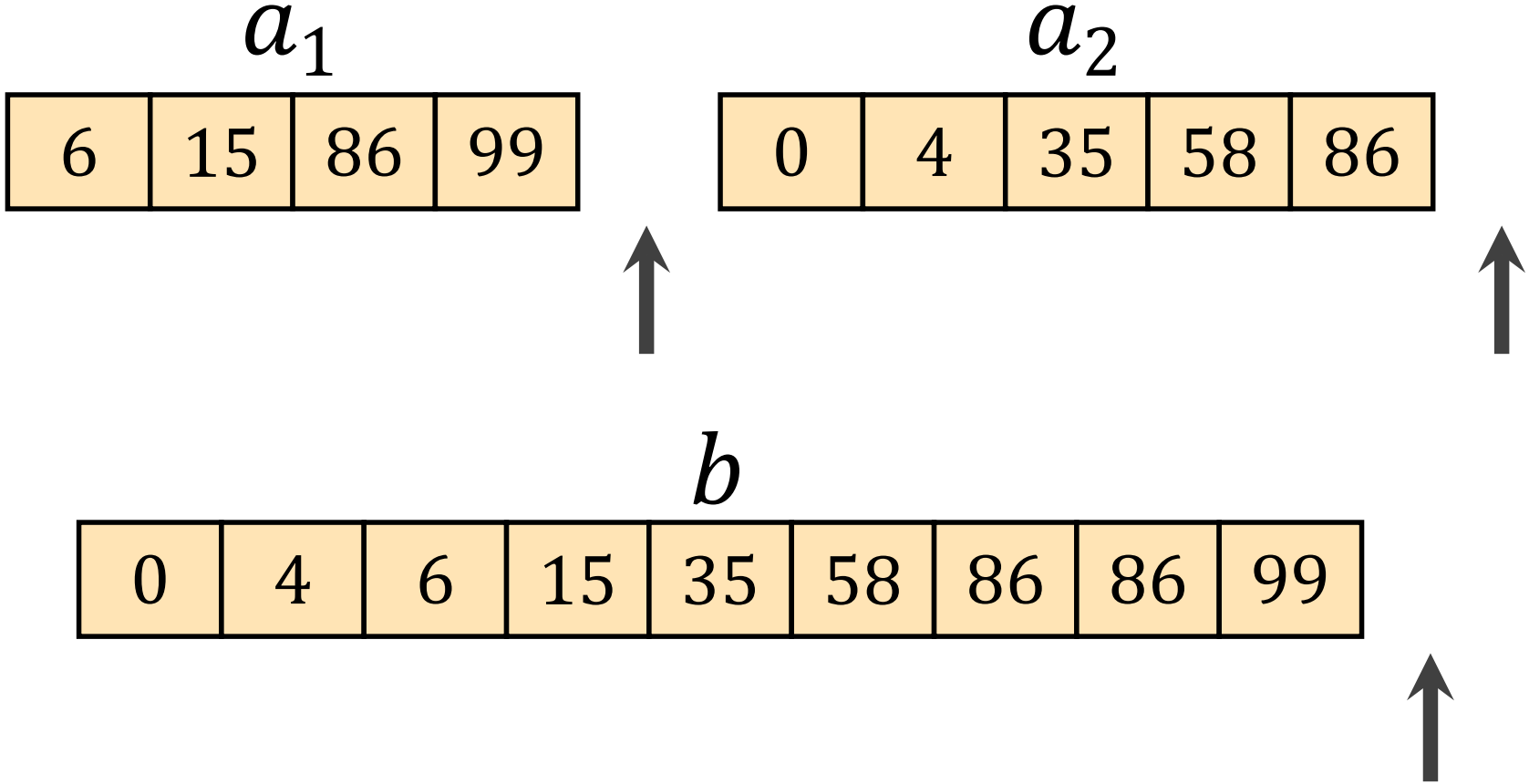
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



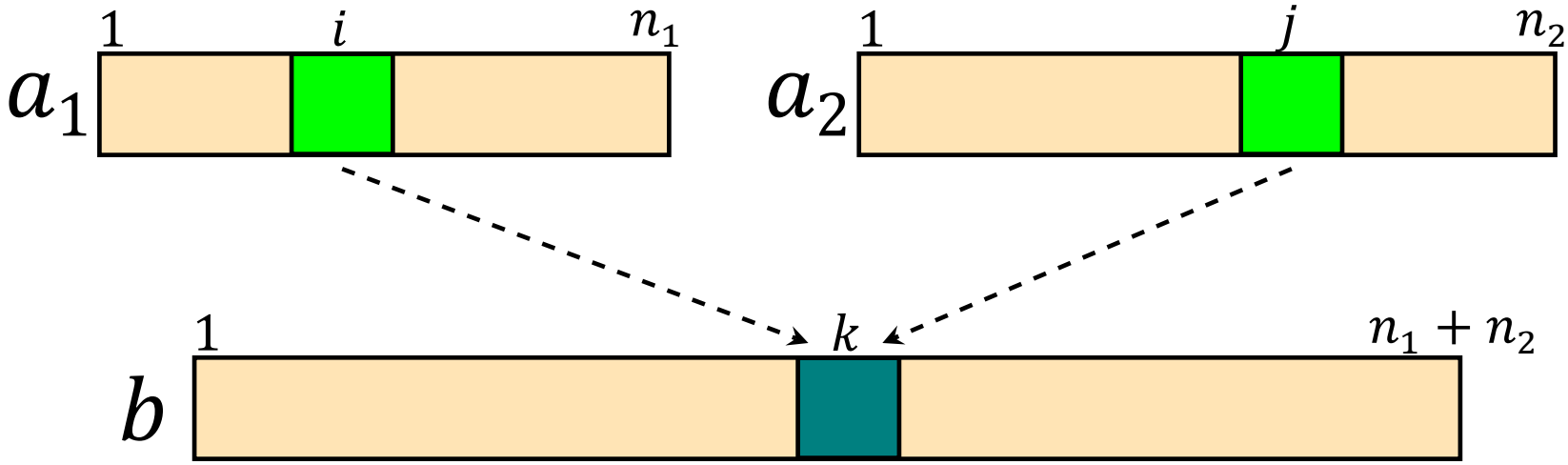
Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



Sortiranje objedinjavanjem (*merge-sort*)

- Operacija objedinjavanja (*merge*) dva sortirana niza u jedan sortiran niz:



- $a_1[i] < a_2[j] \Rightarrow b[k] = a_1[i],\ i = i + 1$
- $a_1[i] \geq a_2[j] \Rightarrow b[k] = a_2[j],\ j = j + 1$
- $k = k + 1$

Sortiranje objedinjavanjem (*merge-sort*)

```
// Ulaz:  dva sortirana niza a1 i a2 dužine n1 i n2
// Izlaz: objedinjeni nizovi a1 i a2 u sortiran niz b
algorithm merge(a1, n1, a2, n2)
   i = 1; j = 1; k = 1; // indeksi nizova a1, a2 i b
   while ((i <= n1) && (j <= n2)) do
      if (a1[i] < a2[j]) then
         b[k] = a1[i]; i = i + 1;
      else
         b[k] = a2[j]; j = j + 1;
      k = k + 1;
   while (i <= n1) do
      b[k] = a1[i]; i = i + 1; k = k + 1;
   while (j <= n2) do
      b[k] = a2[j]; j = j + 1; k = k + 1;

   return b;
```

Sortiranje objedinjavanjem (*merge-sort*)

- Vreme izvršavanja za `merge(a1,n1,a2,n2)`?

Sortiranje objedinjavanjem (*merge-sort*)

- Vreme izvršavanja za `merge(a1,n1,a2,n2)`?
- $O(n_1 + n_2)$

Sortiranje objedinjavanjem (*merge-sort*)

- Algoritam (rekurzivni) *merge-sort* za sortiranje niza
- Primer: $a = [99, 6, 86, 15, 58, 35, 86, 4, 0]$

Sortiranje objedinjavanjem (*merge-sort*)

- Algoritam (rekurzivni) *merge-sort* za sortiranje niza
- Primer: $a = [99, 6, 86, 15, 58, 35, 86, 4, 0]$

1. Podeliti niz u dve polovine (levu i desnu)
   $a_1 = [99, 6, 86, 15], \quad a_2 = [58, 35, 86, 4, 0]$

Sortiranje objedinjavanjem (*merge-sort*)

- Algoritam (rekurzivni) *merge-sort* za sortiranje niza
- Primer: $a = [99, 6, 86, 15, 58, 35, 86, 4, 0]$

1. Podeliti niz u dve polovine (levu i desnu)
   $a_1 = [99, 6, 86, 15], \quad a_2 = [58, 35, 86, 4, 0]$
2. Sortirati (rekurzivno) obe polovine nezavisno
   $a_1 = [6, 15, 86, 99], \quad a_2 = [0, 4, 35, 58, 86]$

Sortiranje objedinjavanjem (*merge-sort*)

- Algoritam (rekurzivni) *merge-sort* za sortiranje niza
- Primer: $a = [99, 6, 86, 15, 58, 35, 86, 4, 0]$

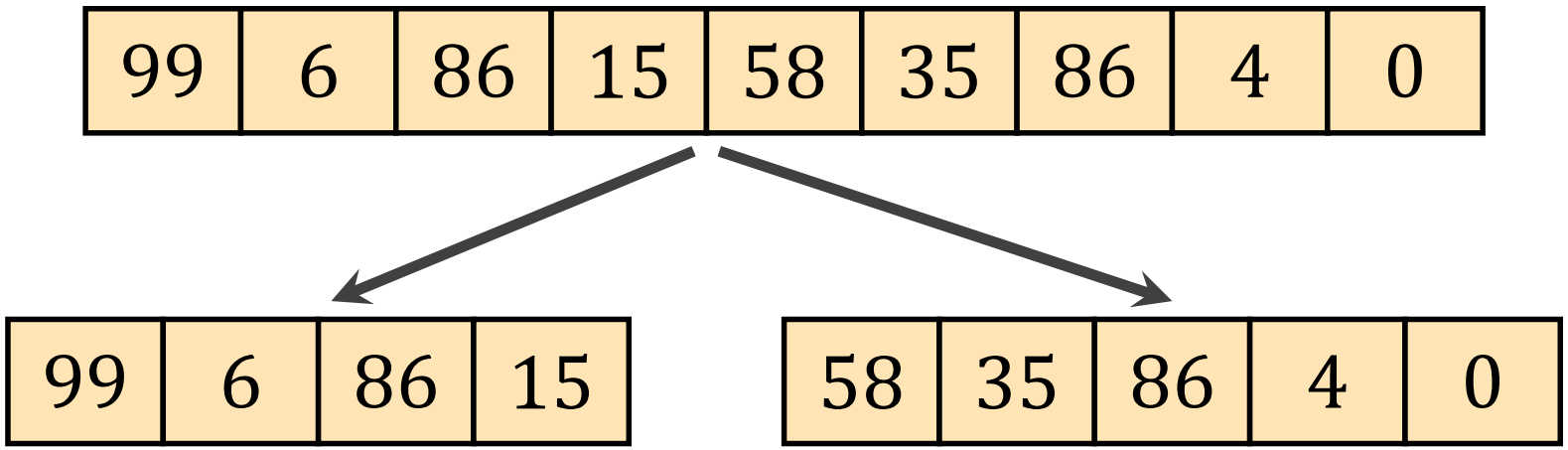
1. Podeliti niz u dve polovine (levu i desnu)
   $a_1 = [99, 6, 86, 15], \quad a_2 = [58, 35, 86, 4, 0]$
2. Sortirati (rekurzivno) obe polovine nezavisno
   $a_1 = [6, 15, 86, 99], \quad a_2 = [0, 4, 35, 58, 86]$
3. Objediniti sortirane polovine u ceo sortiran niz
   $a = [0, 4, 6, 15, 35, 58, 86, 86, 99]$

Sortiranje objedinjavanjem (*merge-sort*)

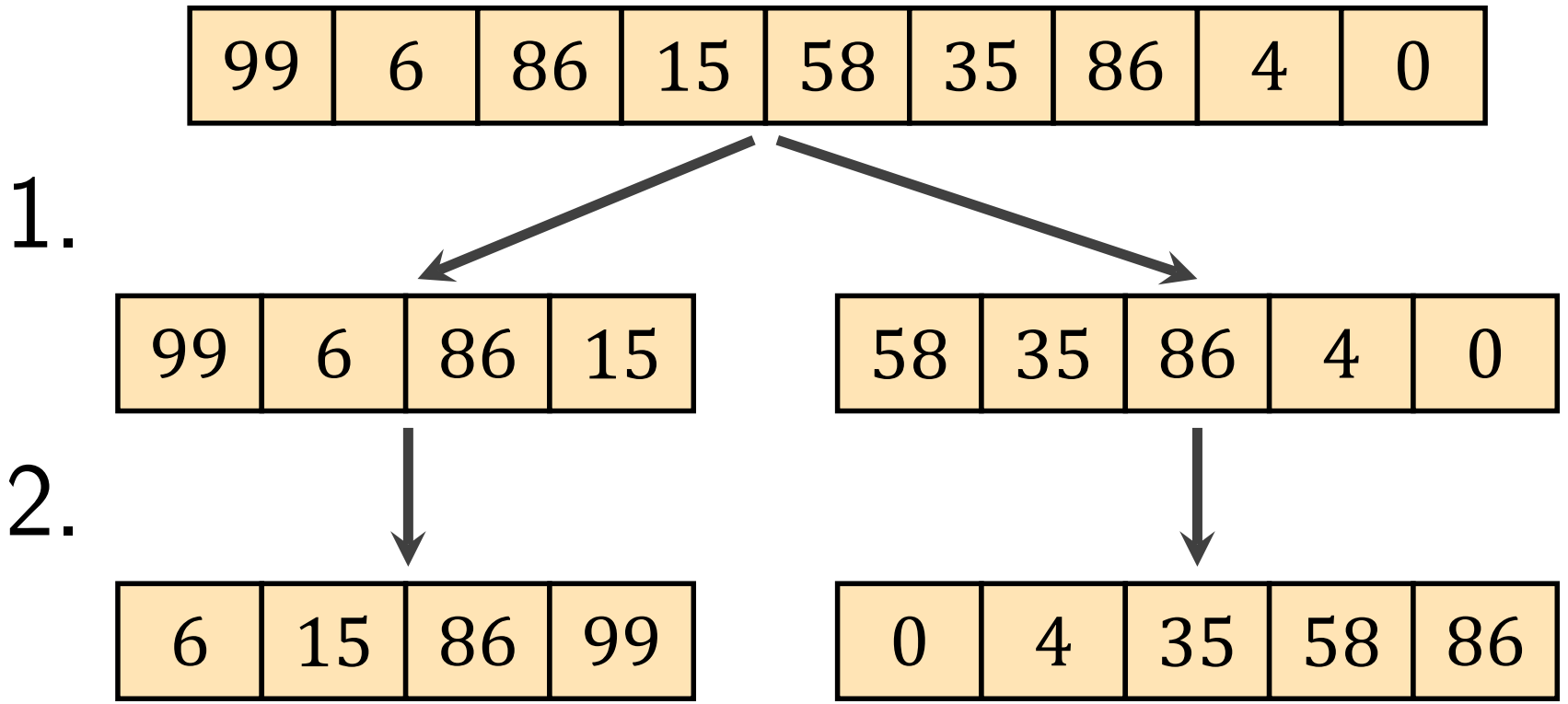
99	6	86	15	58	35	86	4	0

Sortiranje objedinjavanjem (*merge-sort*)

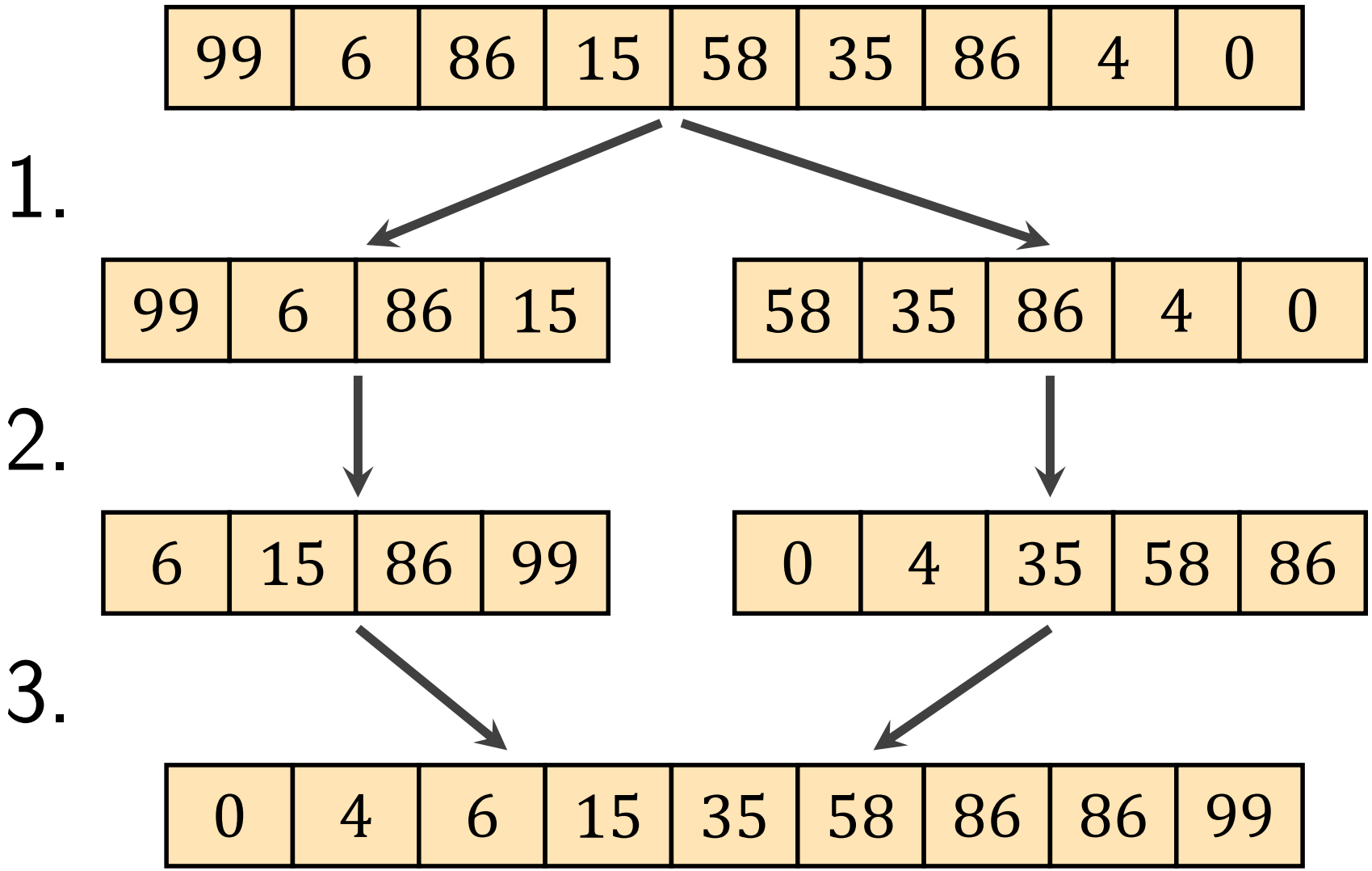
1.



Sortiranje objedinjavanjem (*merge-sort*)



Sortiranje objedinjavanjem (*merge-sort*)



Sortiranje objedinjavanjem (*merge-sort*)

```
// Ulaz:  niz a, njegov broj elemenata n
// Izlaz: sortiran niz a
algorithm merge-sort(a, n)

   if (n == 1) then  // bazni slučaj
      return a;
   else                 // opšti slučaj
      for i = 1 to n/2 do a1[i] = a[i];
      for i = 1 to n/2 do a2[i] = a[n/2+i];

      a1 = merge-sort(a1,n/2);
      a2 = merge-sort(a2,n/2);

      a = merge(a1,n/2,a2,n/2);

      return a;
```

Sortiranje objedinjavanjem (*merge-sort*)

- Vreme izvršavanja za `merge-sort(a,n)`:

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

Sortiranje objedinjavanjem (*merge-sort*)

- Vreme izvršavanja za `merge-sort(a,n)`:

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

- Rešenje: $T(n) = O(n \log n)$

Sortiranje objedinjavanjem (*merge-sort*)

- Zašto $T(n) = O(n \log n)$?

Sortiranje objedinjavanjem (*merge-sort*)

- Zašto $T(n) = O(n \log n)$?

$$
\begin{aligned}
T(n) &= 2T(n/2) + cn \\
&= 2\big(2T(n/4) + cn/2\big) + cn = 4T(n/4) + 2cn \\
&= 4\big(2T(n/8) + cn/4\big) + 2cn = 8T(n/8) + 3cn \\
&\vdots \\
&= n \cdot T(1) + \log n \cdot cn \\
&= cn + \log n \cdot cn \\
&= O(n \log n)
\end{aligned}
$$