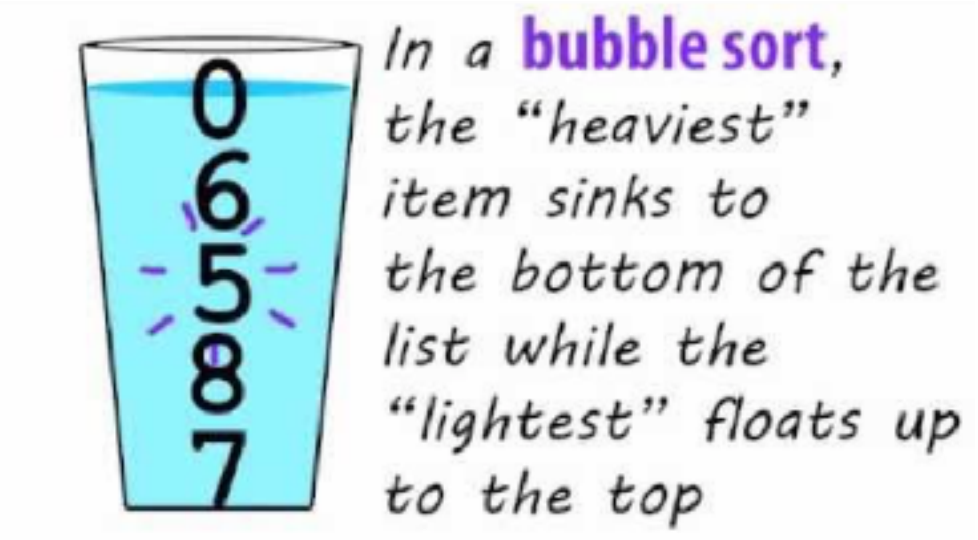
15-112
Fundamentals of Programming

Week 3 - Lecture 3:
Efficiency Continued + Sorting



June 7, 2017

The Plan

→ Measuring running time when the input is an int

> Sorting a given list

- Selection sort
- Bubble sort
- Merge sort

1. Algorithm
2. Running time
3. Code

Integer inputs

```
def isPrime(n):
    if (n < 2):
        return False
    for factor in range(2, n):
        if (n % factor == 0):
            return False
    return True
```

Simplifying assumption in 15-112:

Arithmetic operations take constant time.

Integer inputs

```
def isPrime(n):
    if (n < 2):
        return False
    for factor in range(2, n):
        if (n % factor == 0):
            return False
    return True
```

What is the input length?

= number of digits in n

$\sim \log_{10} n$

Integer Inputs

```
def isPrime(m):
    if (m < 2):
        return False
    for factor in range(2, m):
        if (m % factor == 0):
            return False
    return True
```

What is the input length?

= number of digits in m

~ $\log_{10} m$ (actually $\log_2 m$ because it is in binary)

So $N \sim \log_2 m$ i.e., $m \sim 2^N$

What is the running time? $O(m) = O(2^N)$

Integer Inputs

```
def fasterIsPrime(m):
    if (m < 2):
        return False
    maxFactor = int(round(m**0.5))
    for factor in range(3, maxFactor+1):
        if (m % factor == 0):
            return False
    return True
```

What is the running time? $O(2^{N/2})$

isPrime

Amazing result from 2002:

There is a polynomial-time algorithm for primality testing.

Agrawal, Kayal, Saxena

↓ ↓

undergraduate students at the time

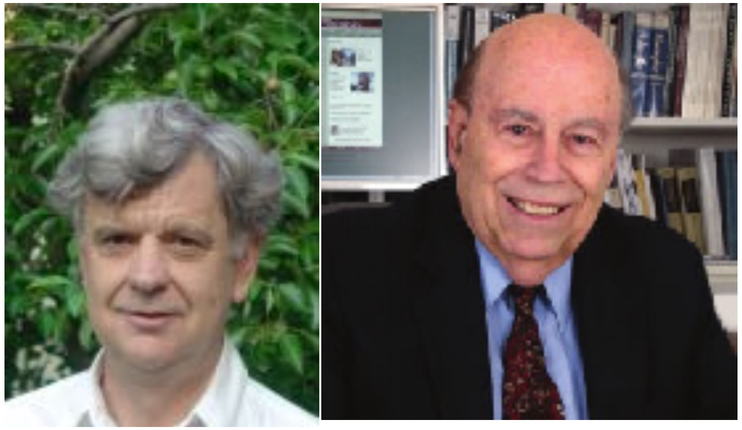
However, best known implementation is ~$O(N^6)$ time.

Not feasible when $N = 2048$.

isPrime

So that's not what we use in practice.

Everyone uses the Miller-Rabin algorithm (1975).



CMU
Professor

The running time is ~ $O(N^2)$.

It is a randomized algorithm with a tiny error probability.

(say $1/2^{300}$)

The Plan

> Measuring running time when the input is an int

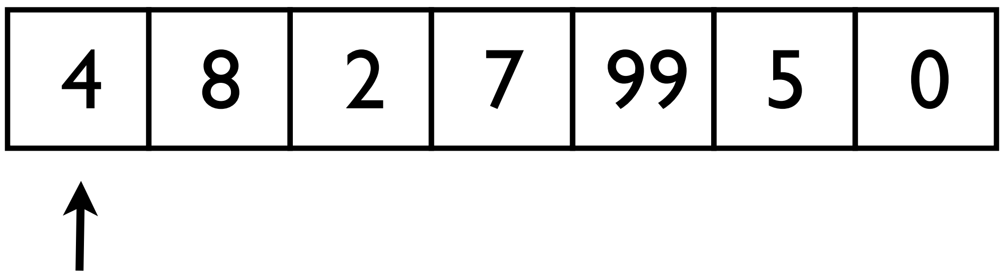
Sorting a given list

- Selection sort
- Bubble sort
- Merge sort

1. Algorithm
2. Running time
3. Code

Selection Sort: Algorithm

Sort a given list of integers (from small to large).



Selection Sort

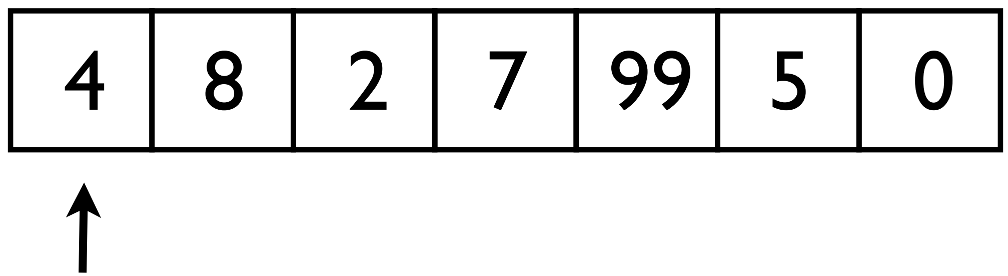
Find the minimum element.

Put it on the left.

Repeat process on the remaining n-1 elements.

Selection Sort: Algorithm

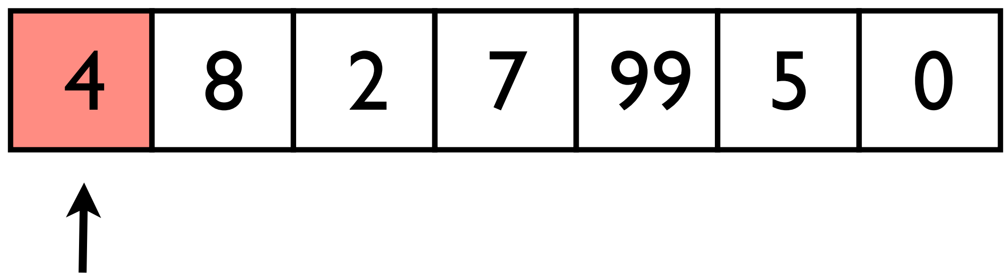
Sort a given list of integers (from small to large).



Selection Sort

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

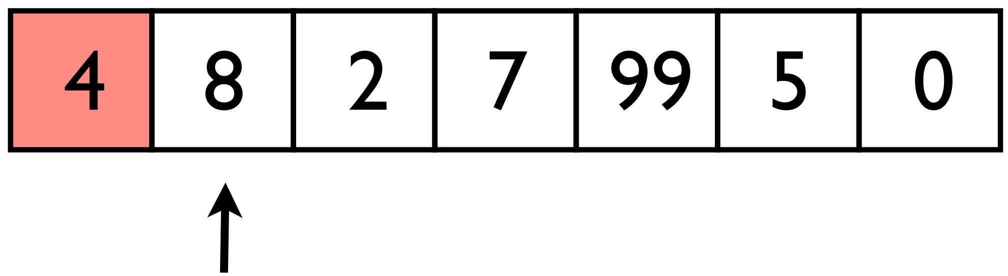


Selection Sort

Current min: 4

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

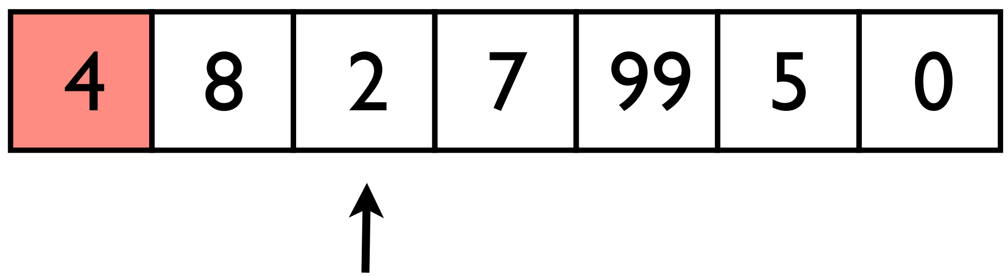


Selection Sort

Current min: 4

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

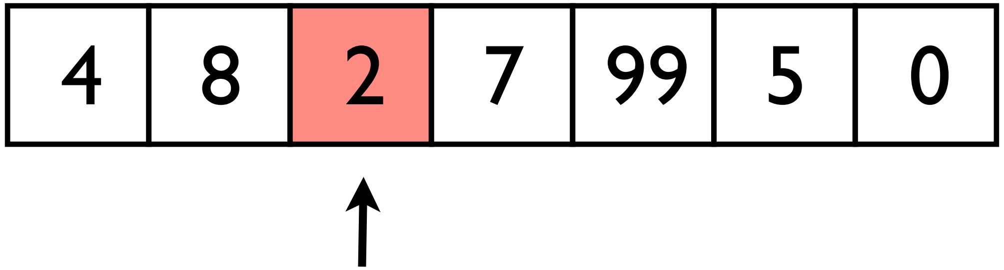


Selection Sort

Current min: 4

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

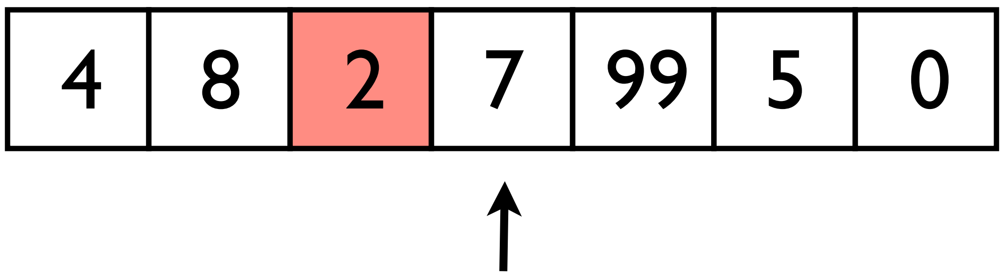


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

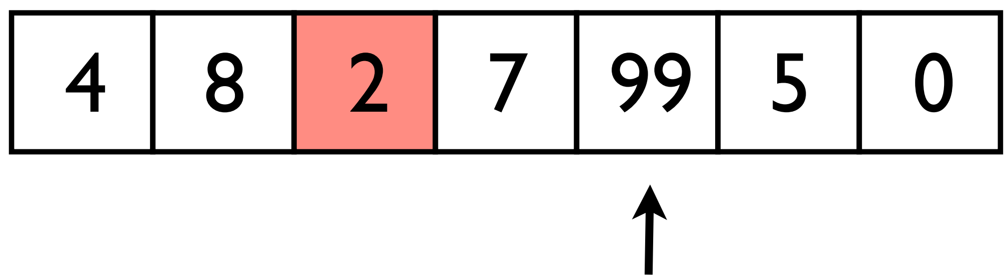


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

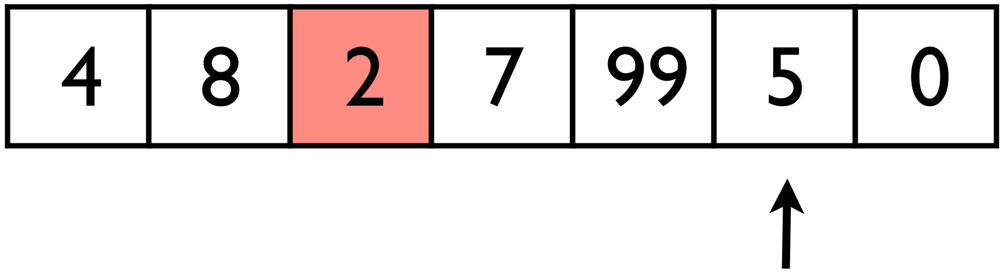


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

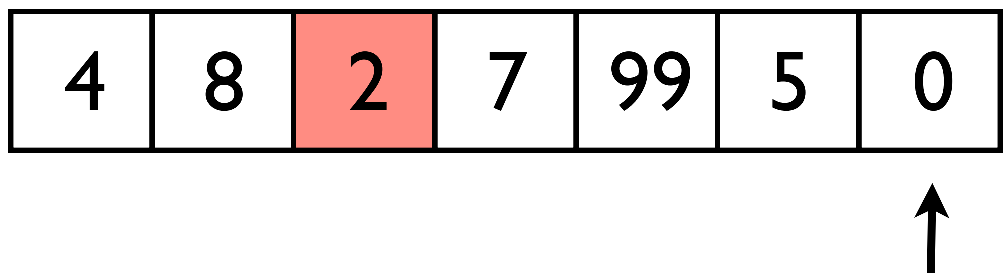


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

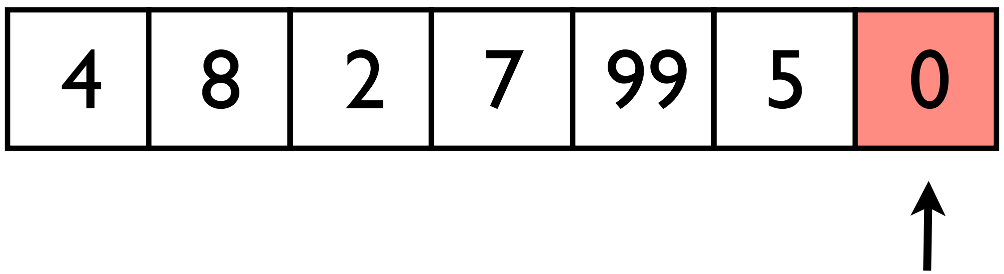


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

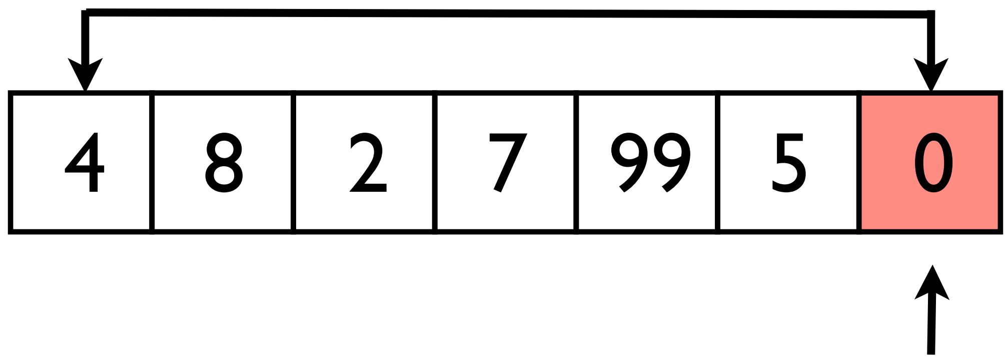


Selection Sort

Current min: 0

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

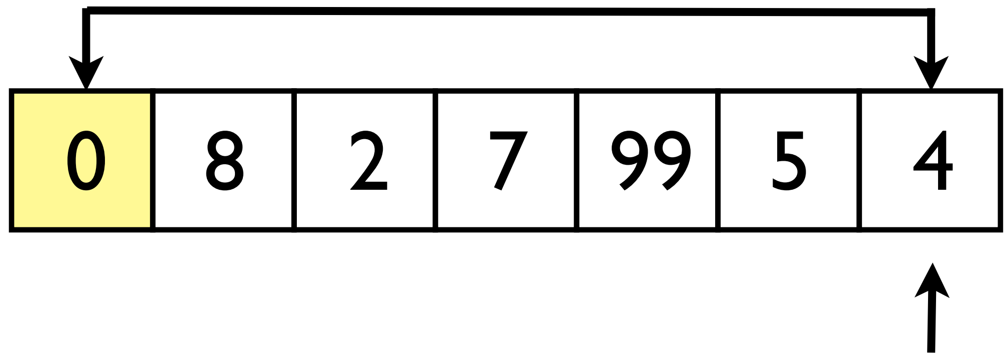


Selection Sort

Swap current min with first element of the array

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

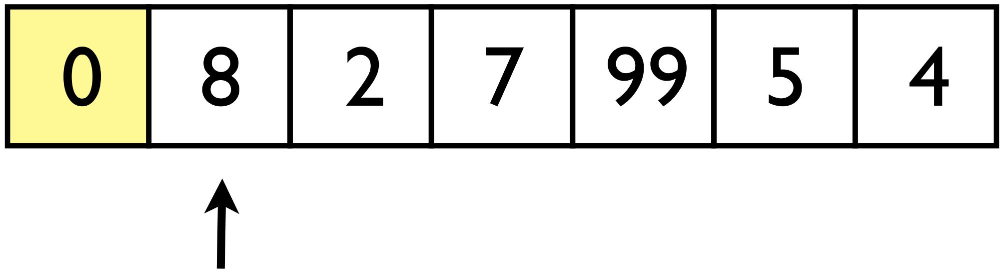


Selection Sort

Swap current min with first element of the array

Selection Sort: Algorithm

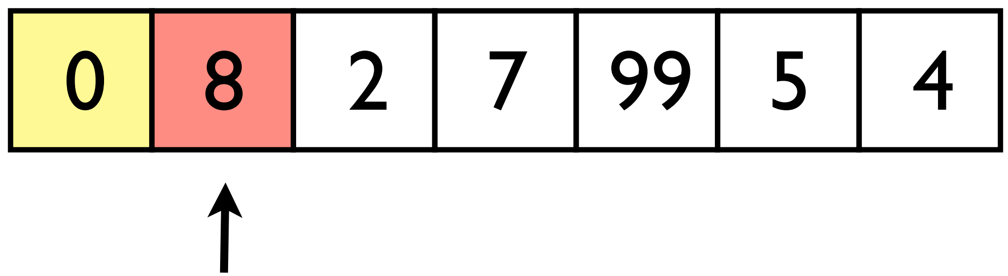
Sort a given list of integers (from small to large).



Selection Sort

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

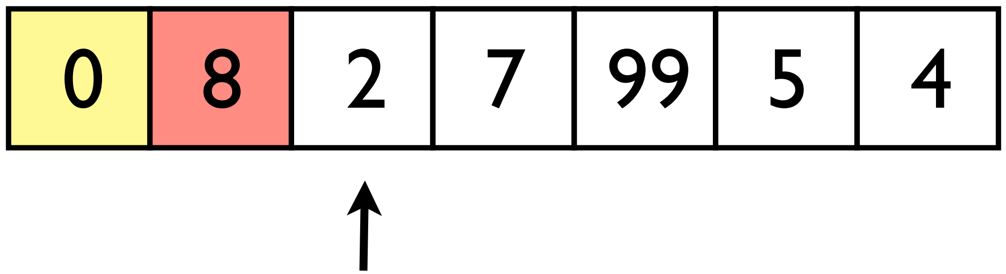


Selection Sort

Current min: 8

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

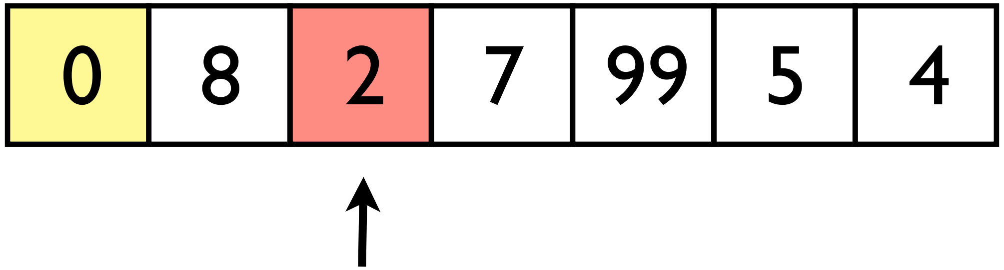


Selection Sort

Current min: 8

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

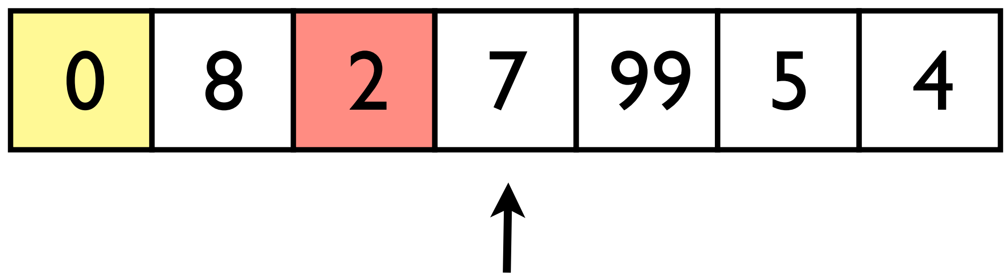


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

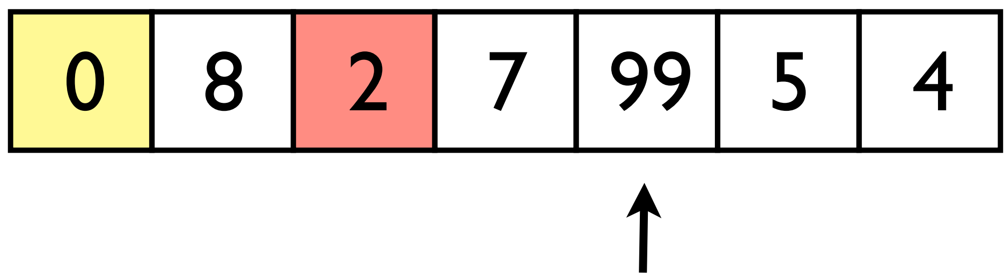


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

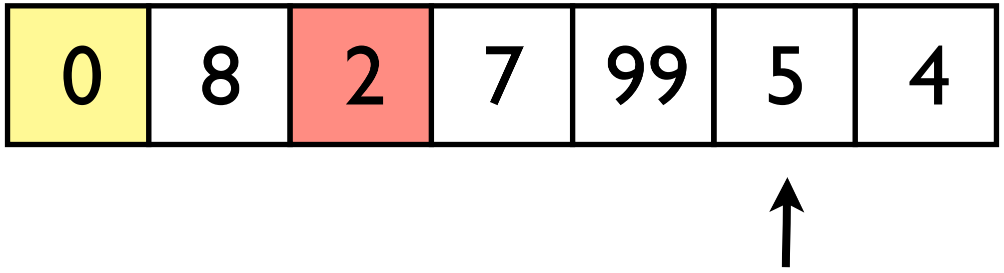


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

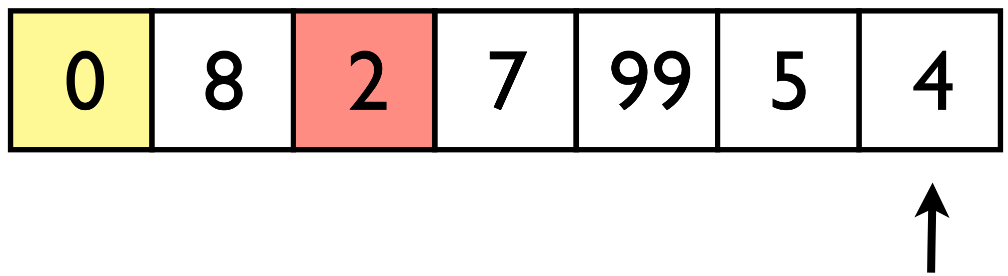


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

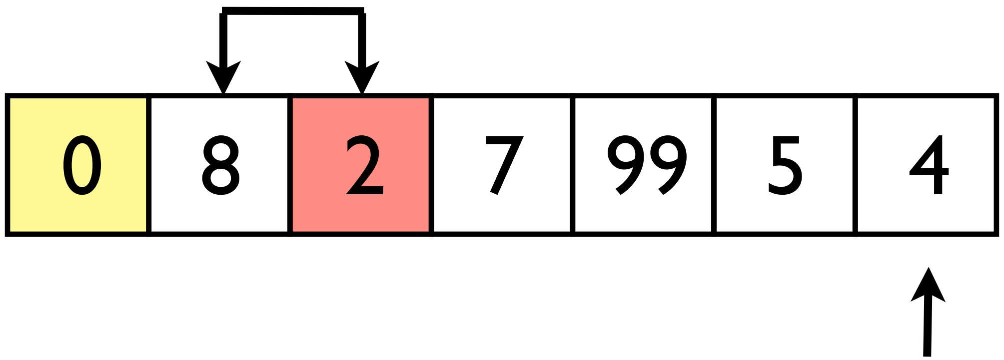


Selection Sort

Current min: 2

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

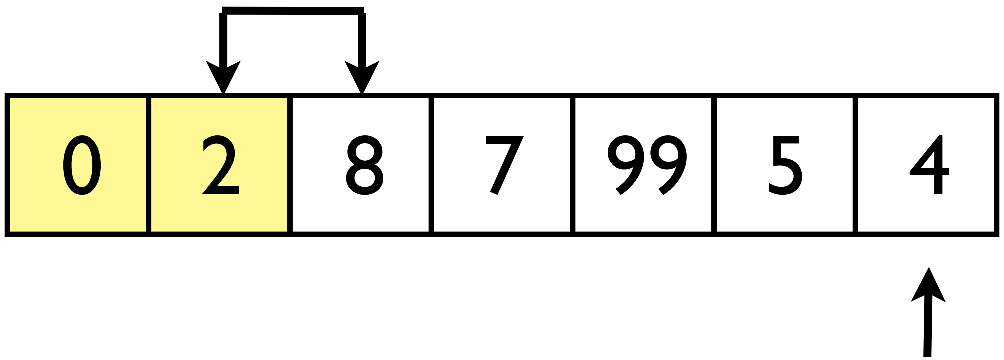


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

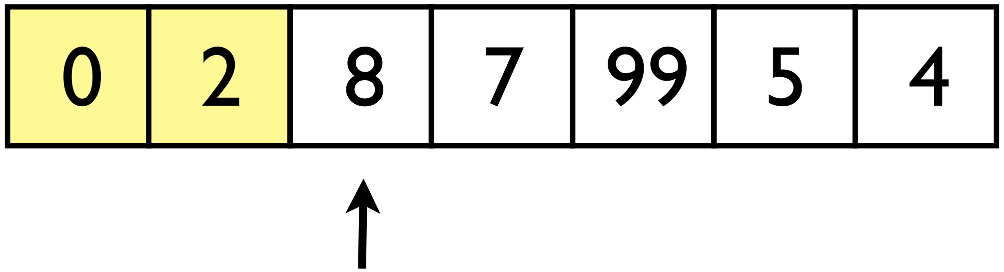


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

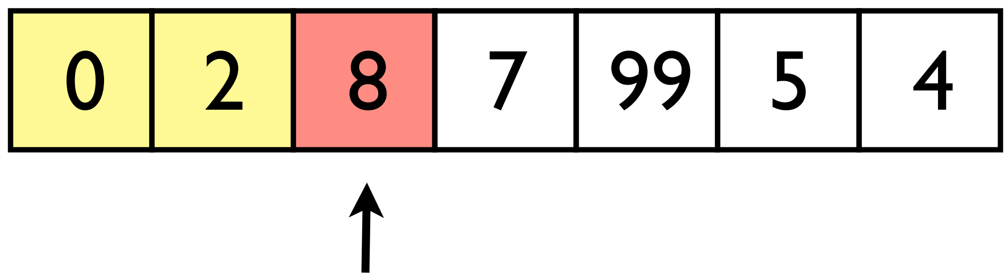
Sort a given list of integers (from small to large).



Selection Sort

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

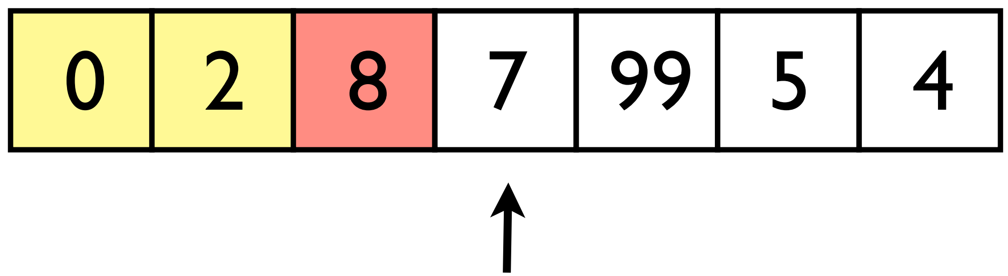


Selection Sort

Current min: 8

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

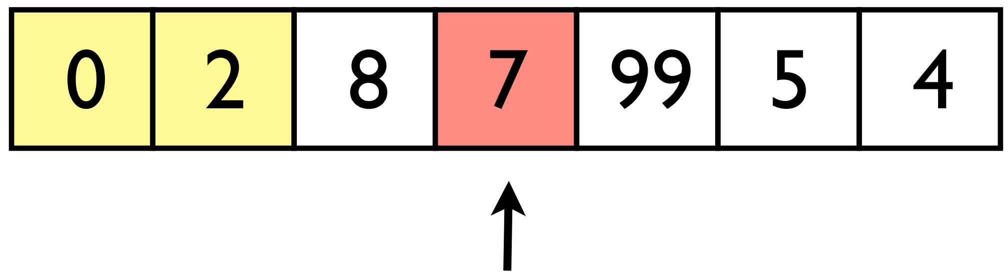


Selection Sort

Current min: 8

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

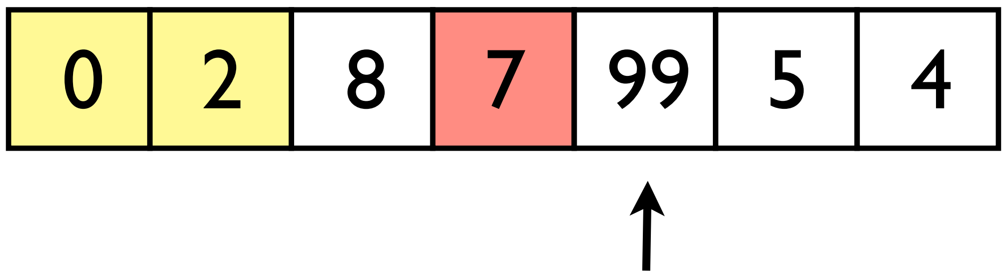


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

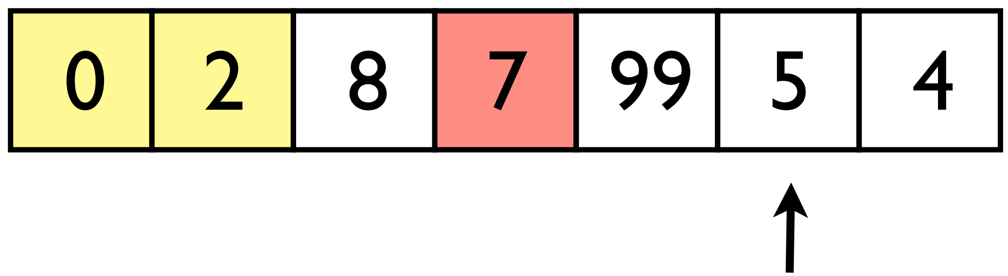


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

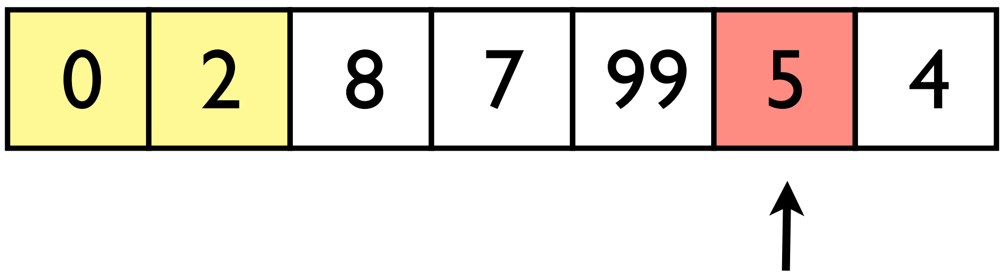


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

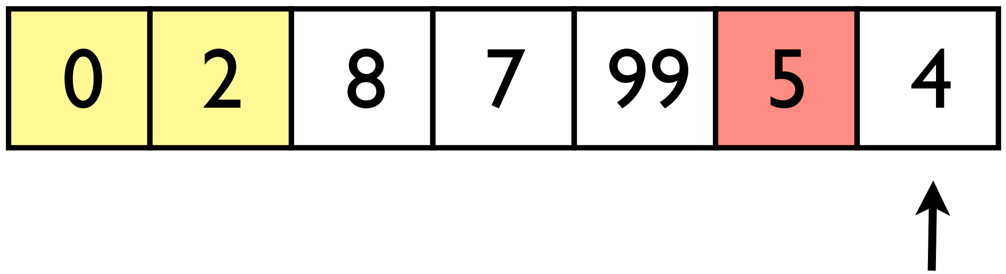


Selection Sort

Current min: 5

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

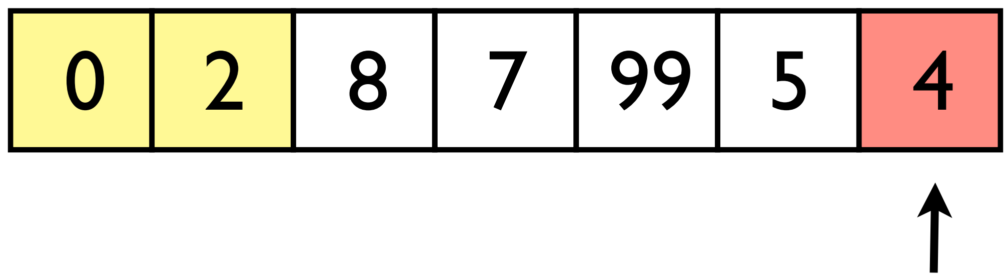


Selection Sort

Current min: 5

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

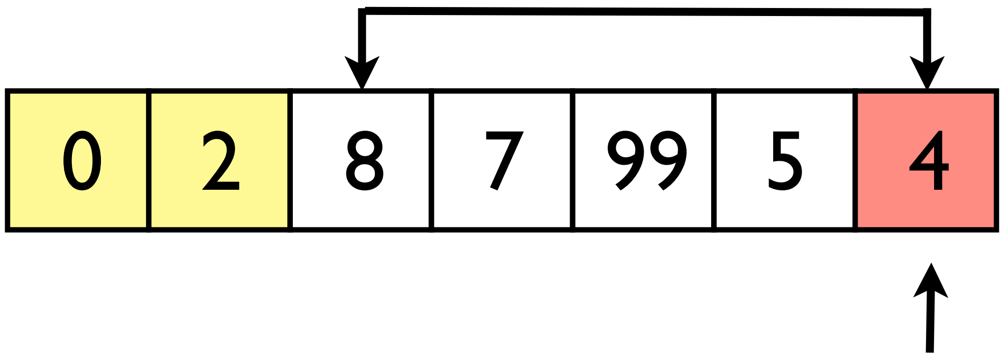


Selection Sort

Current min: 4

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

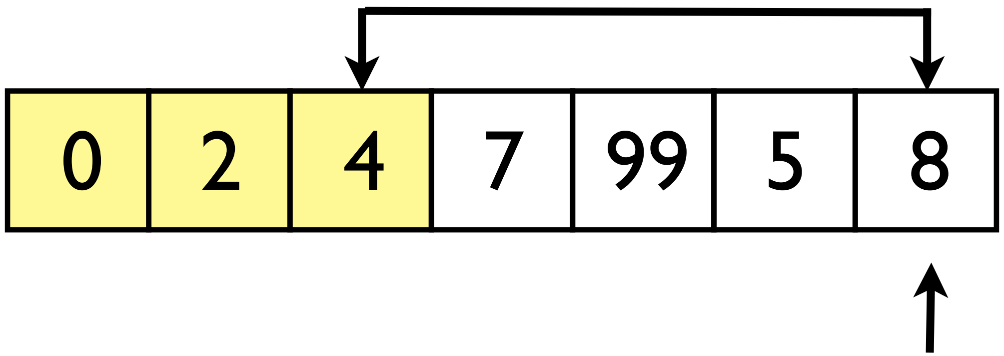


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

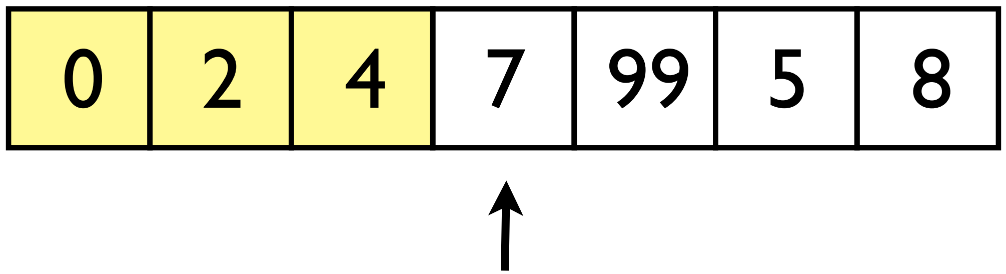


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

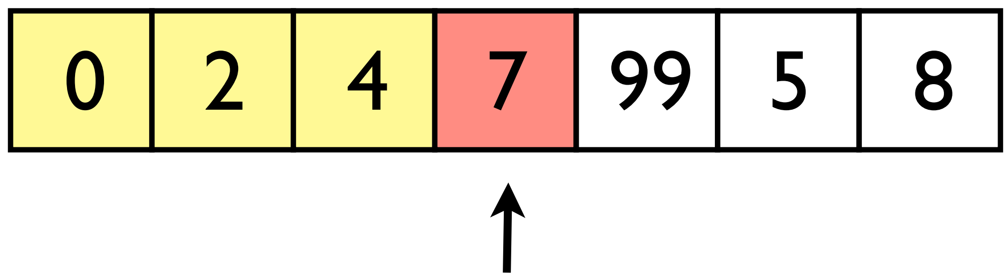
Sort a given list of integers (from small to large).



Selection Sort

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

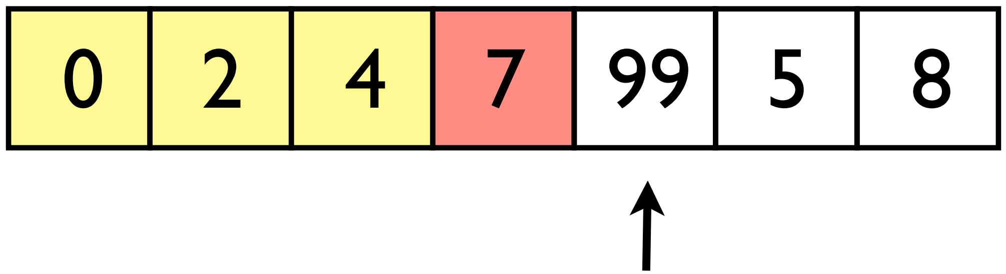


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

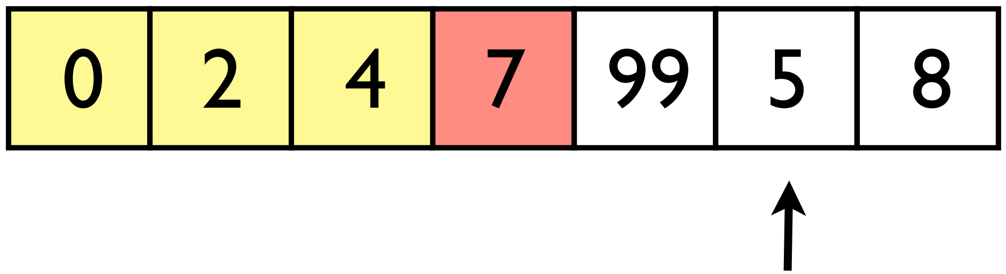


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

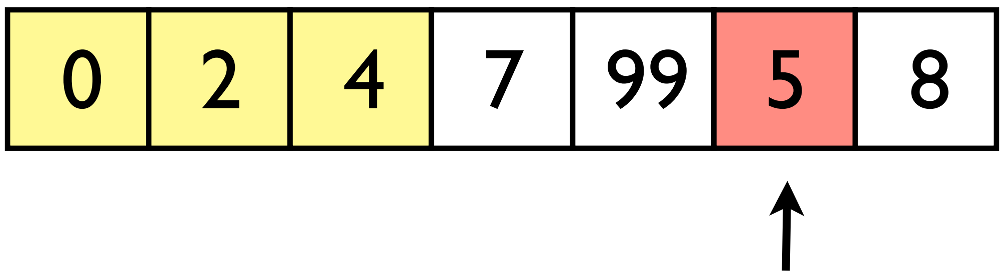


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

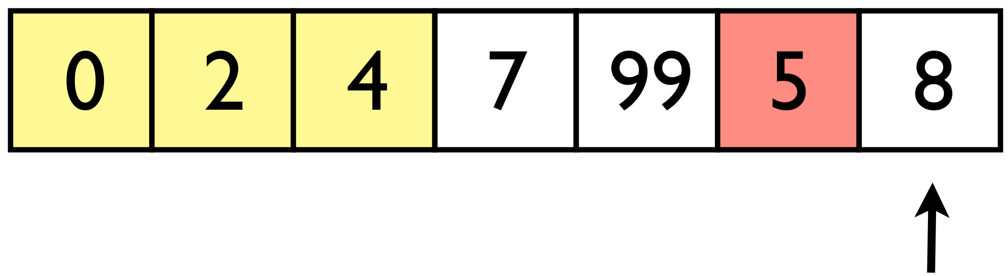


Selection Sort

Current min: 5

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

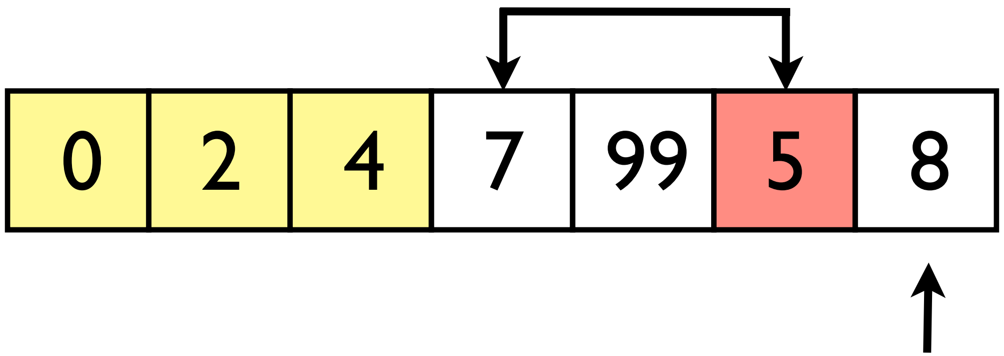


Selection Sort

Current min: 5

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

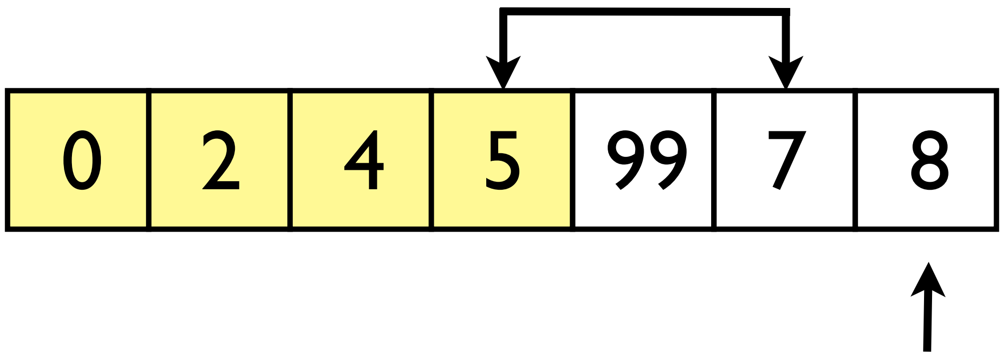


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

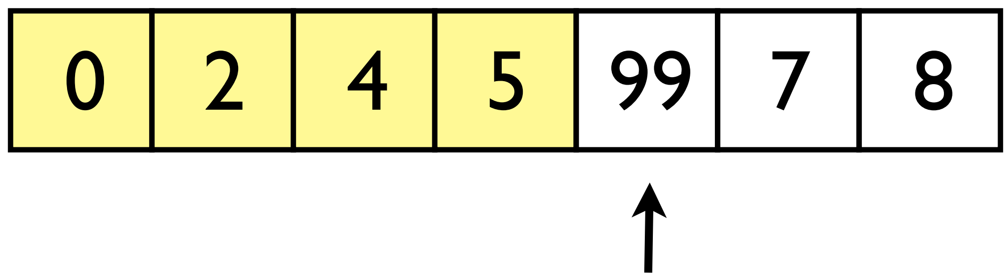


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

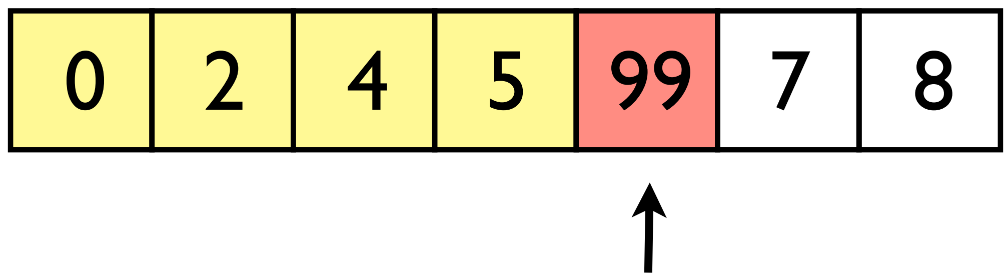
Sort a given list of integers (from small to large).



Selection Sort

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

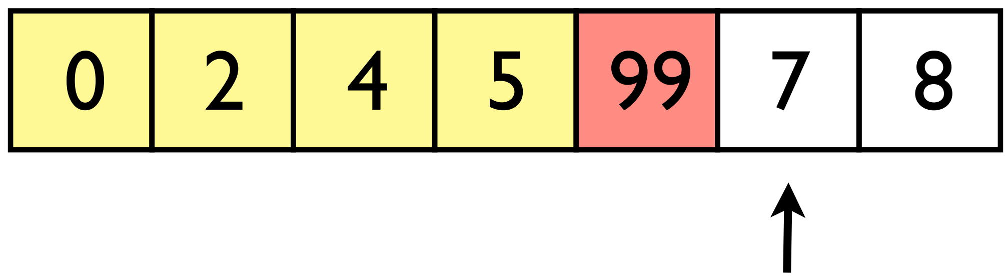


Selection Sort

Current min: 99

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

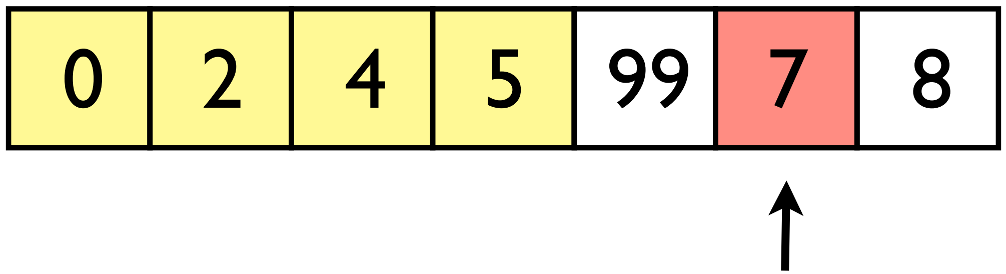


Selection Sort

Current min: 99

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

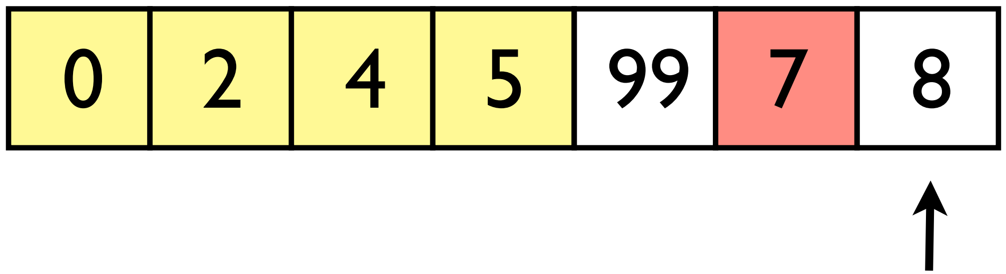


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

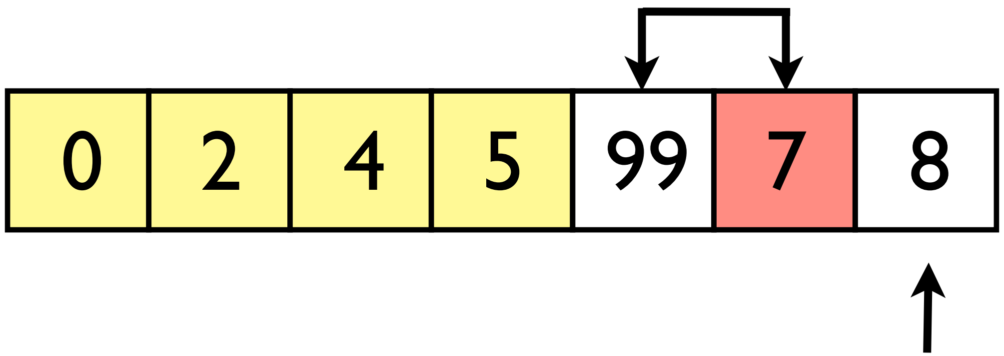


Selection Sort

Current min: 7

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

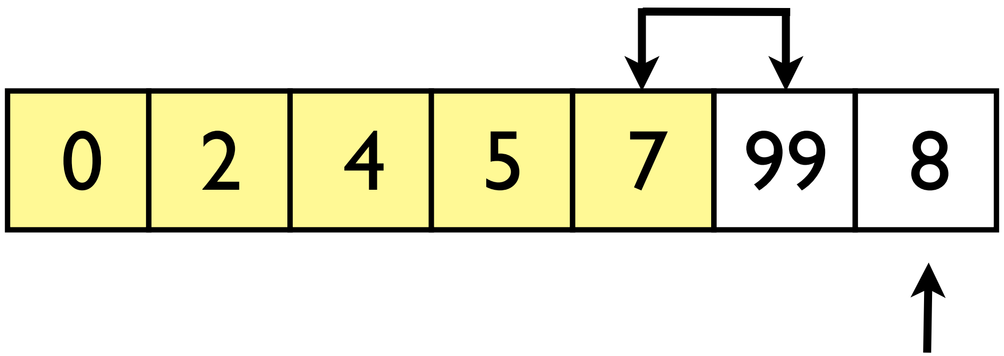


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

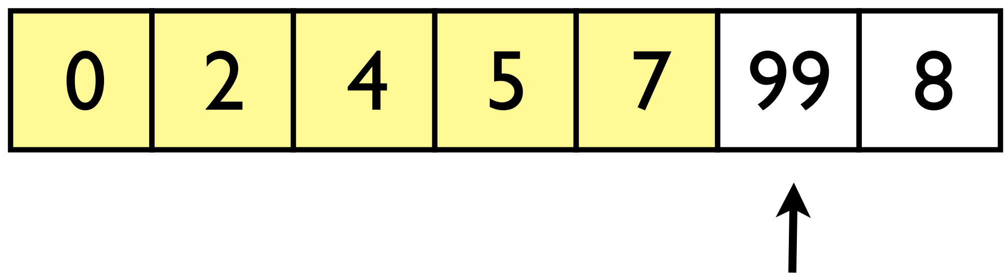


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

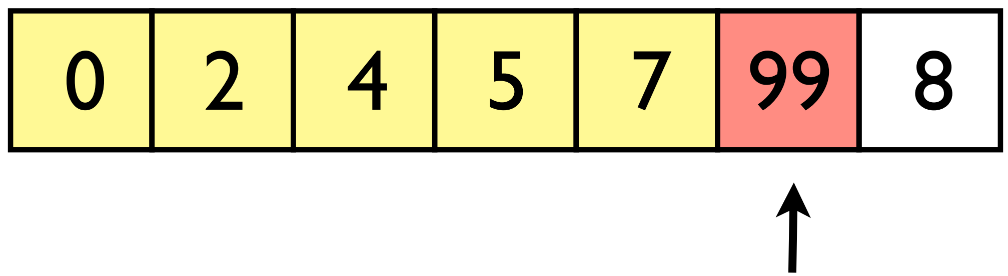
Sort a given list of integers (from small to large).



Selection Sort

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

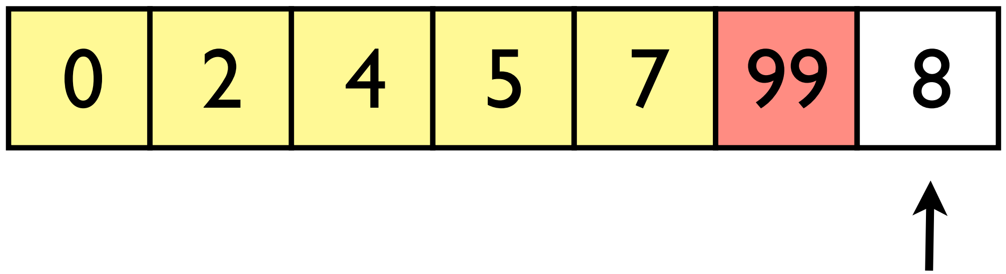


Selection Sort

Current min: 99

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

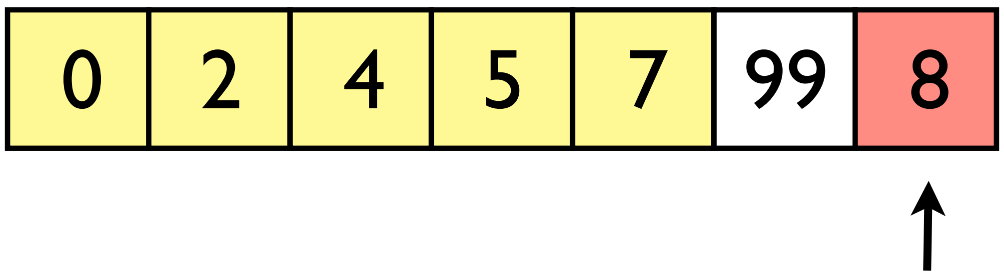


Selection Sort

Current min: 99

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

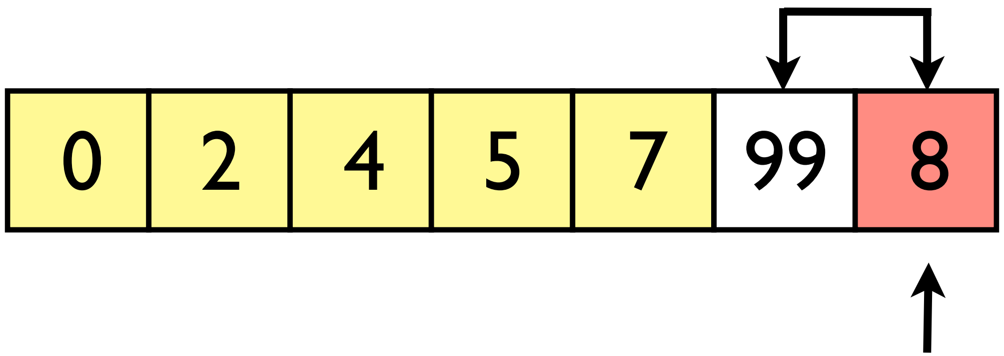


Selection Sort

Current min: 8

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

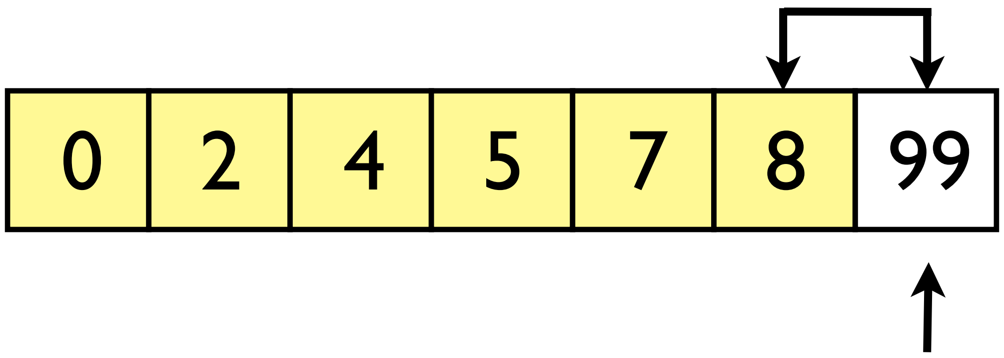


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

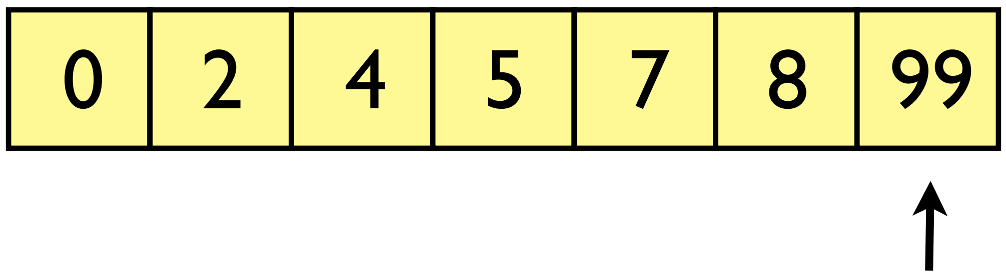


Selection Sort

Swap current min with first element of unsorted part

Selection Sort: Algorithm

Sort a given list of integers (from small to large).

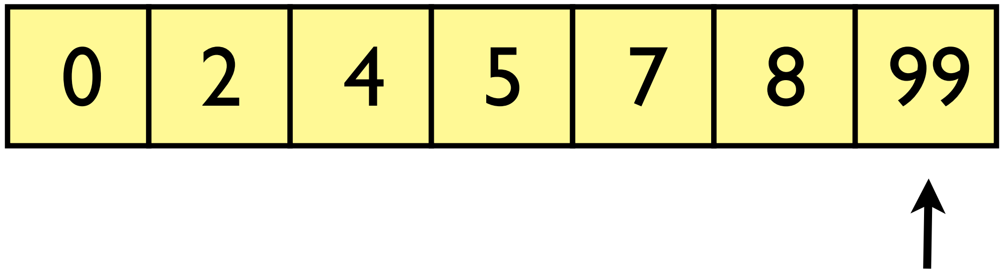


Selection Sort

Done!

Selection Sort: Running Time

Sort a given list of integers (from small to large).



Selection Sort

How many steps does this take (in the worst case)?

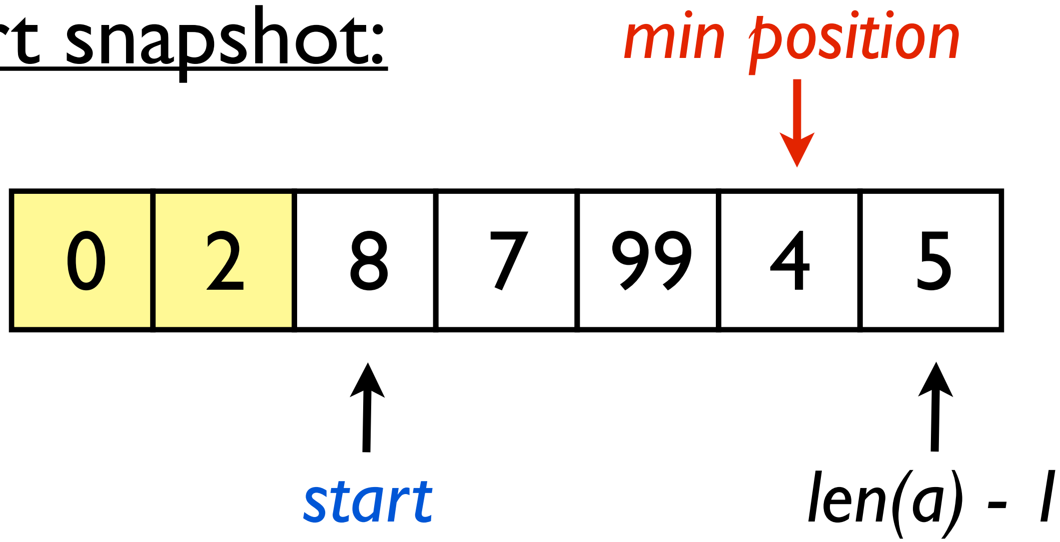
$$\sim N + (N-1) + (N-2) + \cdots + 1 \quad = \frac{N^2}{2} + \frac{N}{2}$$

(As N increases, small terms lose significance.)

Running time is $O(N^2)$.

Selection Sort: Code

<u>Selection sort snapshot:</u>



Find the *min position* from *start* to *len(a) - 1*

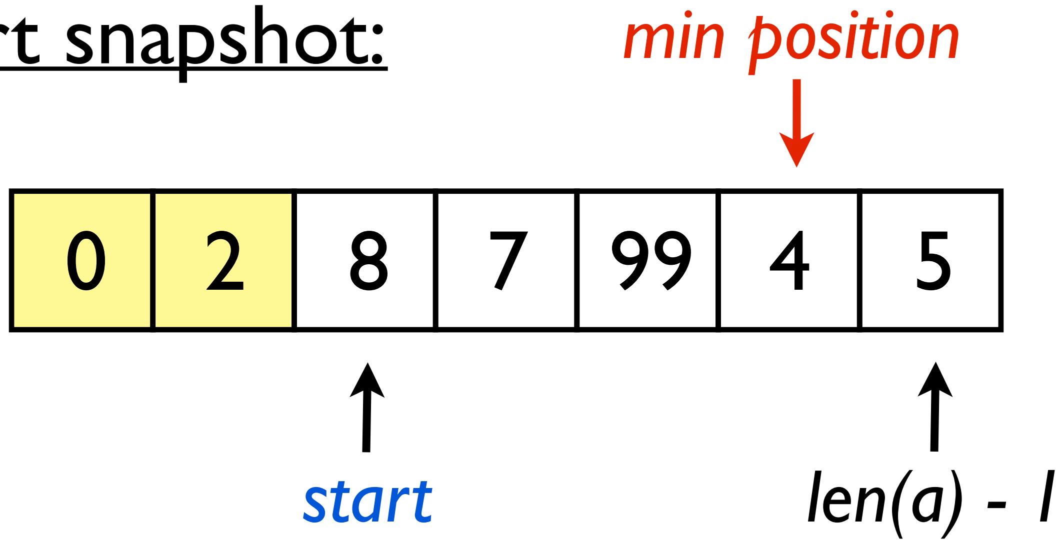
Swap elements in *min position* and *start*

Increment *start*

Repeat

Selection Sort: Code

Selection sort snapshot:



for *start* = 0 to *len(a)-1*:

Find the *min position* from *start* to *len(a) - 1*

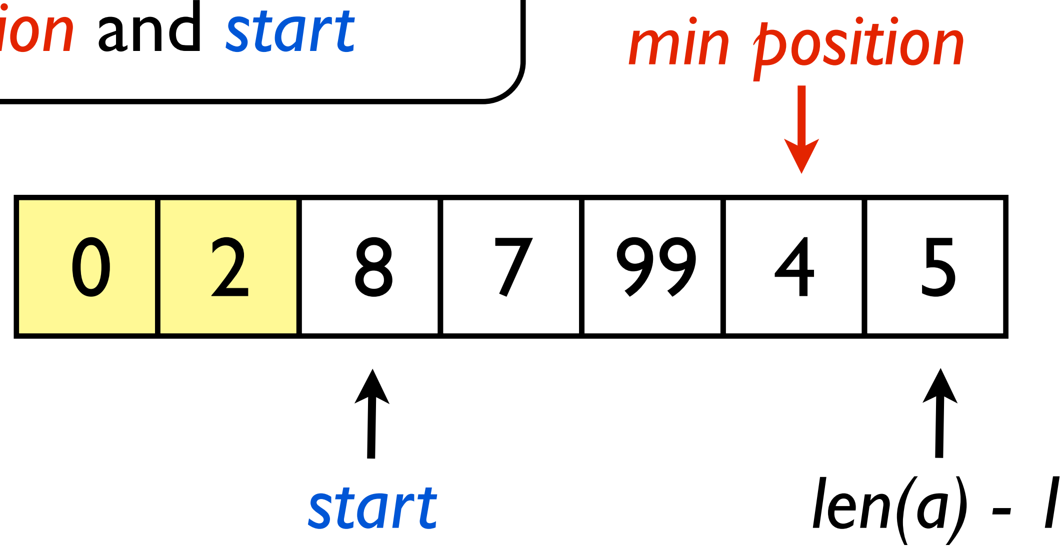
Swap elements in *min position* and *start*

Selection Sort: Code

for *start* = 0 to *len(a)-1*:

  Find the *min position* from *start* to *len(a) - 1*

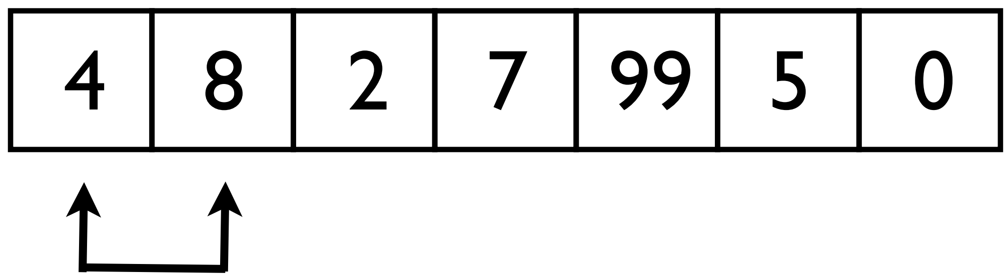
  Swap elements in *min position* and *start*




```
def selectionSort(a):
  for start in range(len(a)):
    currentMinIndex = start
    for i in range(start, len(a)):
      if(a[i] < a[currentMinIndex]):
        currentMinIndex = i
    (a[currentMinIndex], a[start]) = (a[start], a[currentMinIndex])
```

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

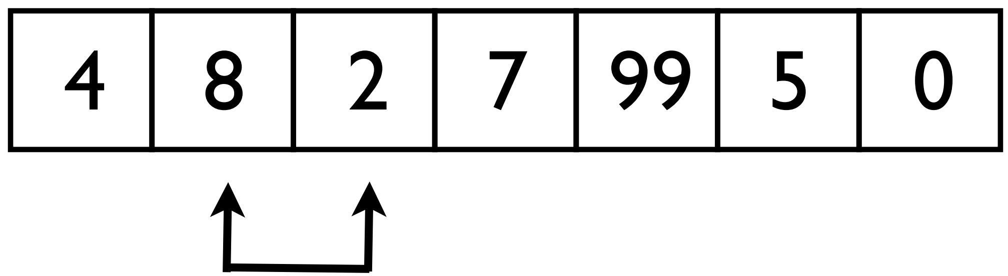
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

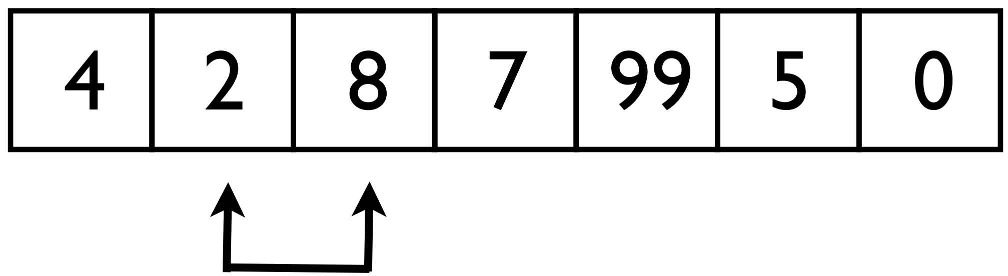
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

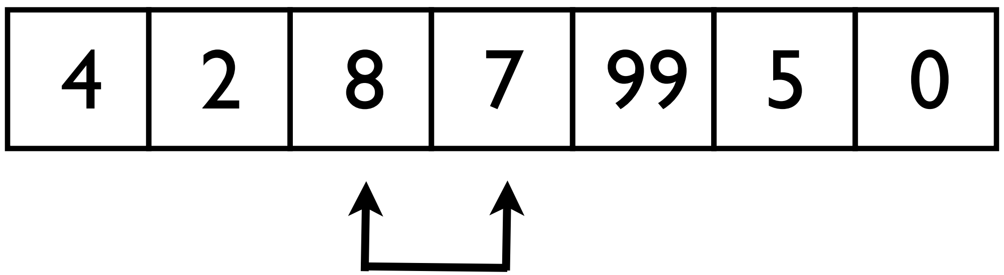
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

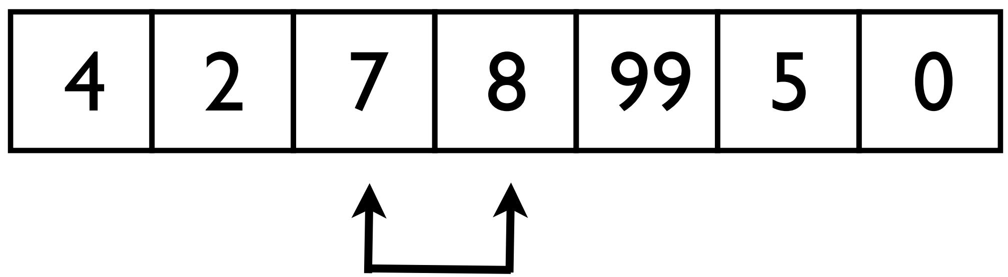
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

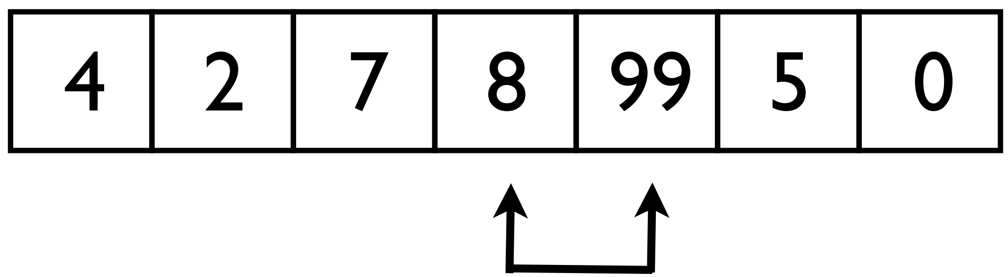
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

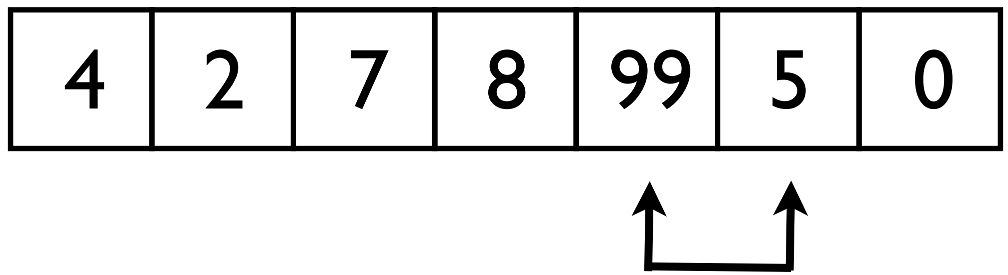
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

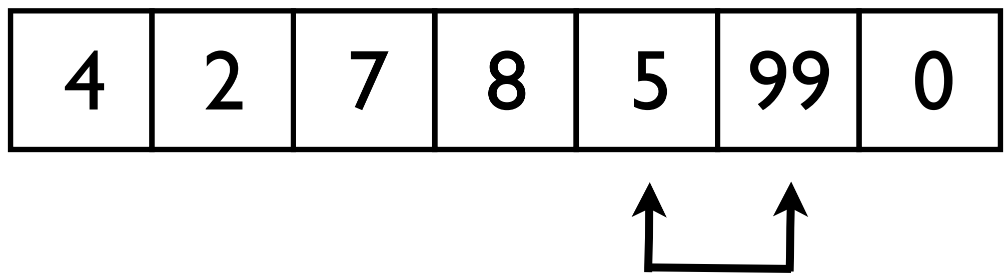
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

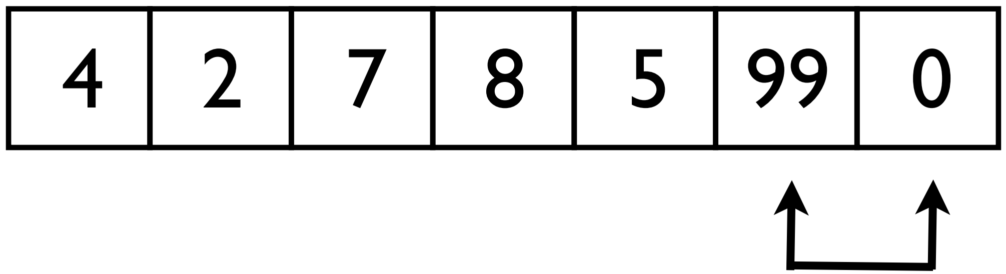
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

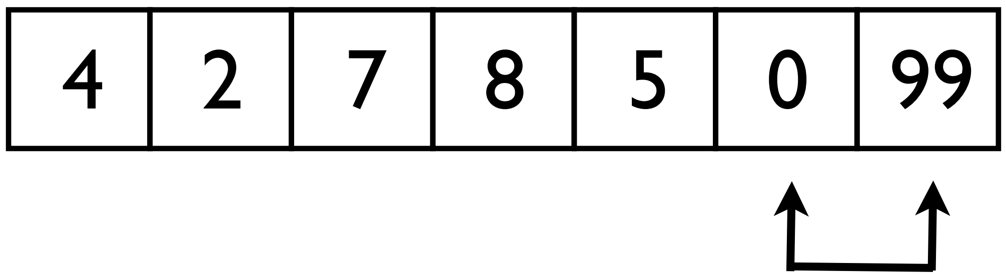
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

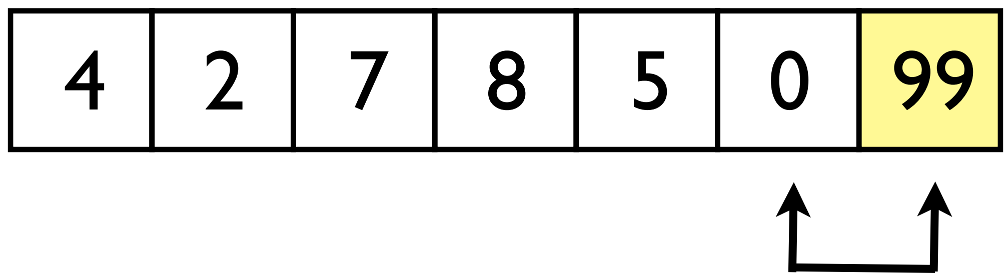
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

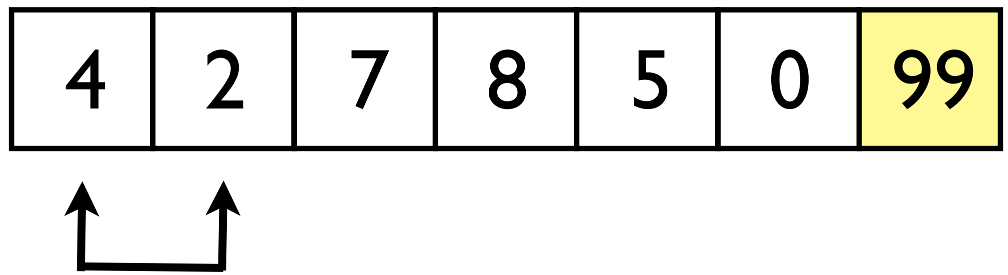
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

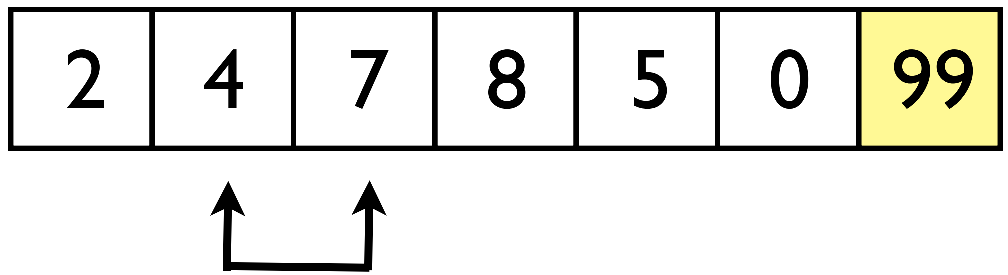
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

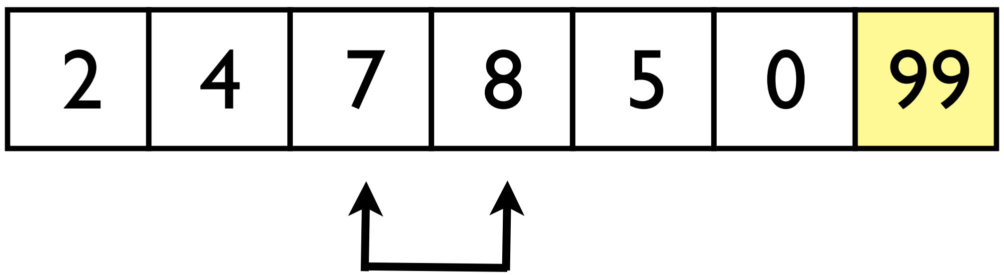
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

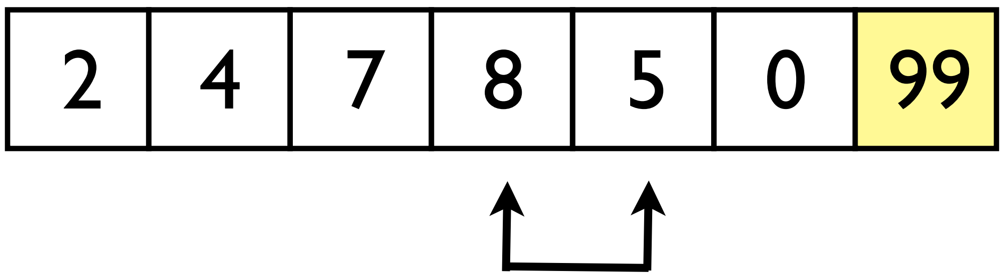
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

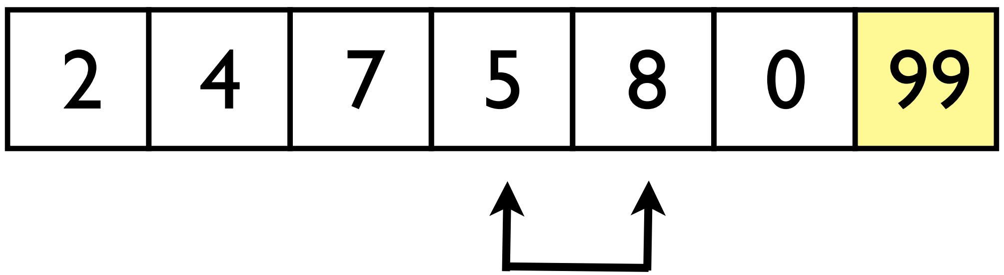
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

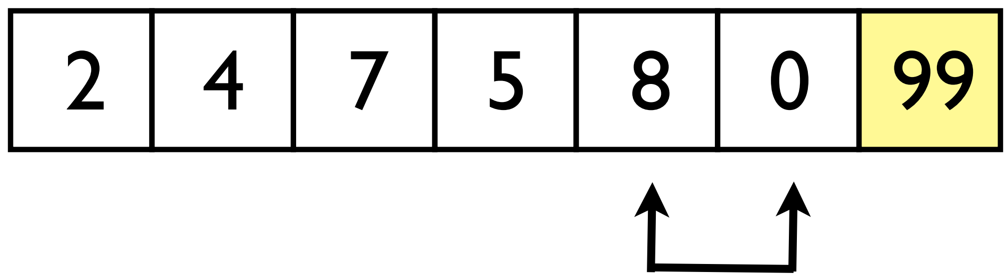
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

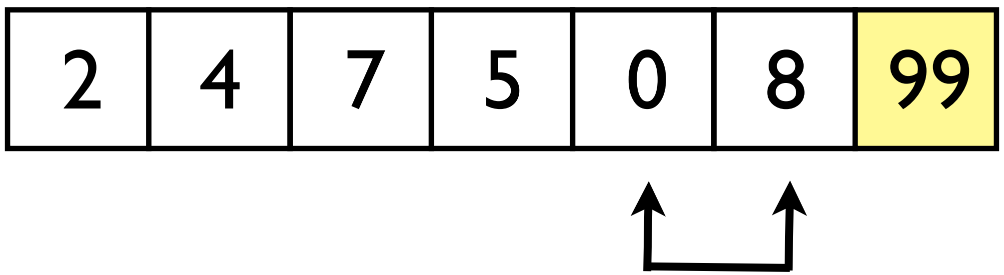
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

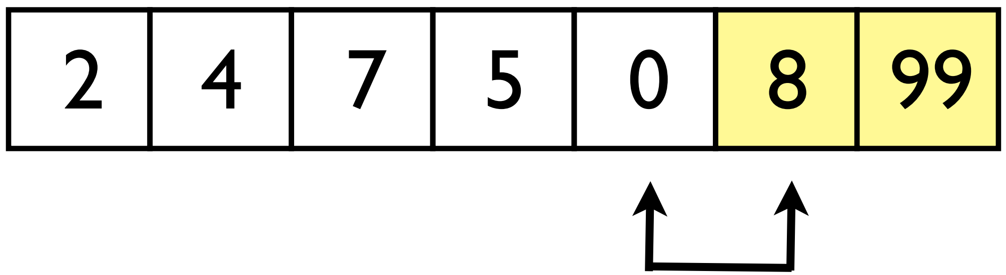
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

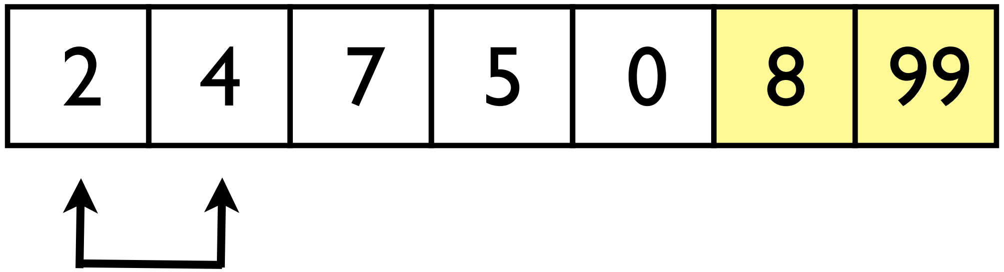
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

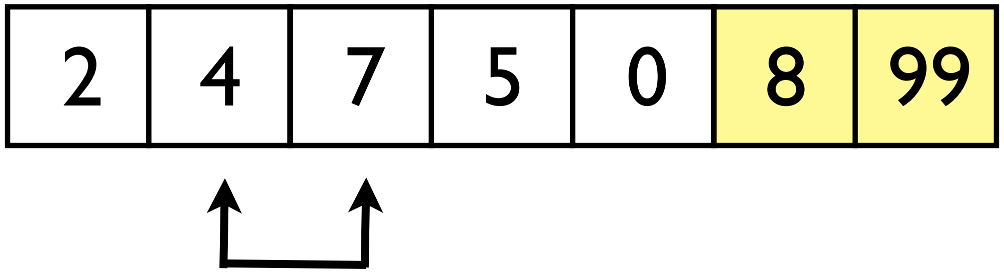
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

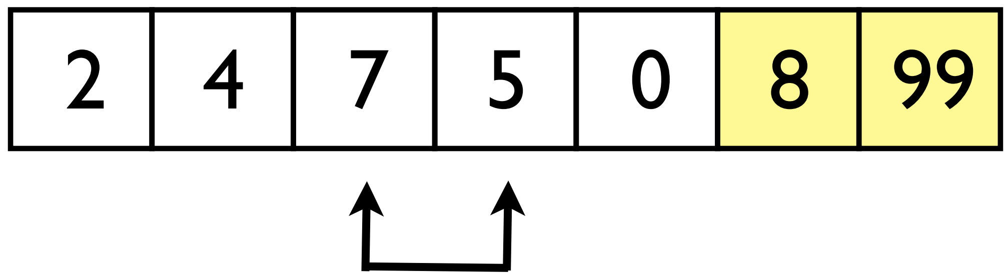
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

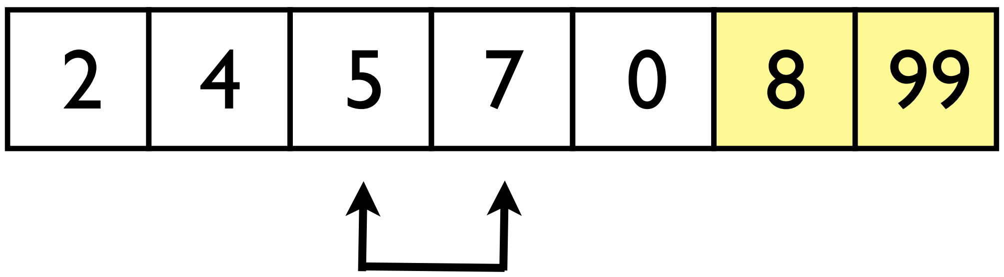
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

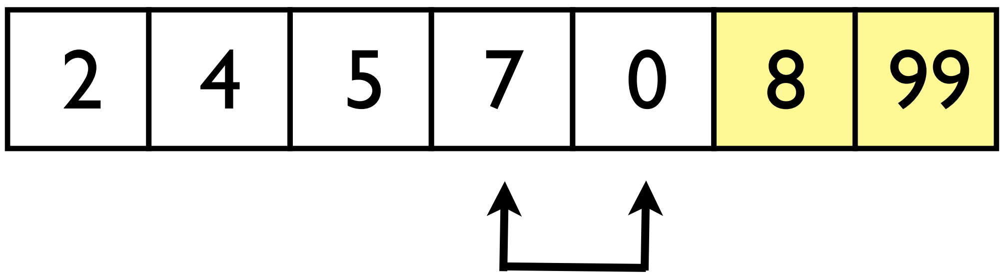
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

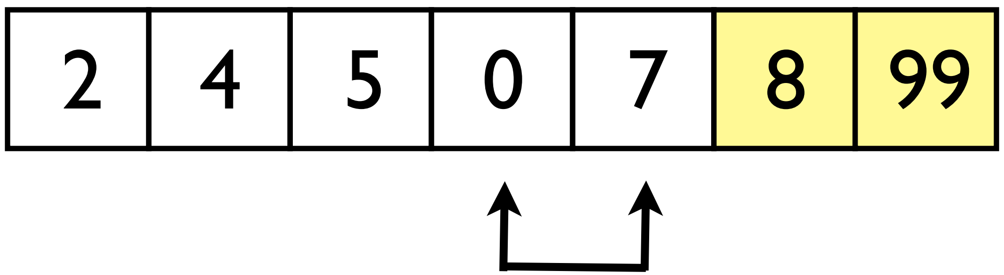
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

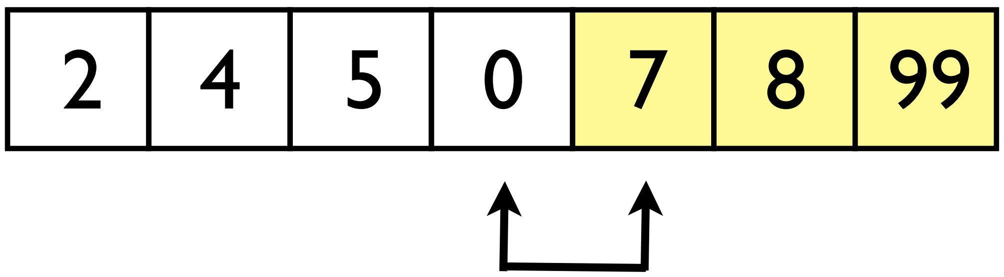
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

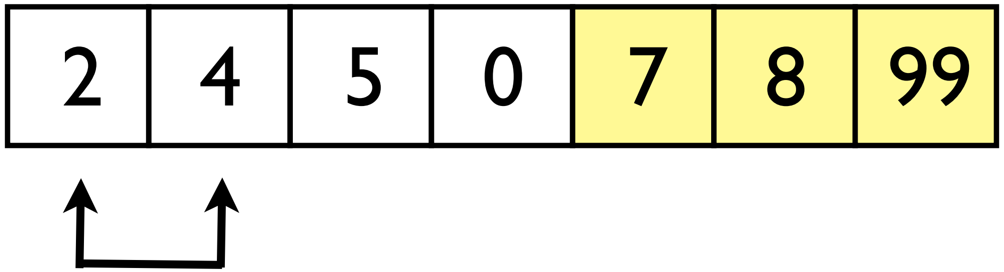
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

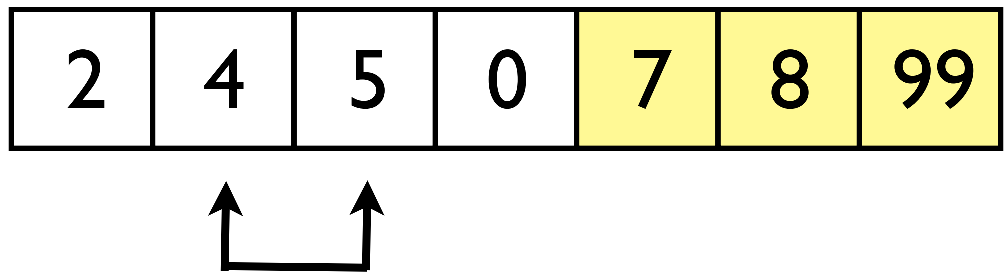
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

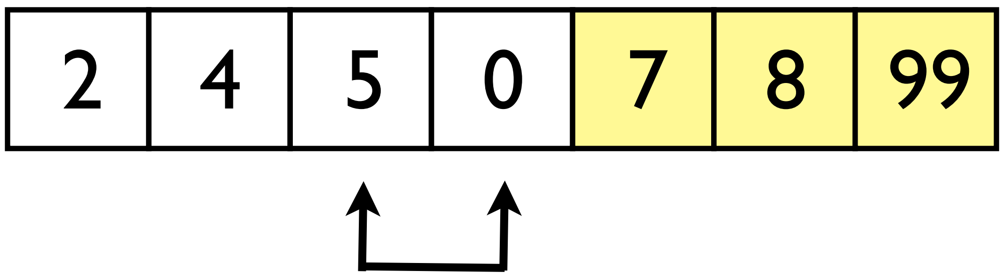
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

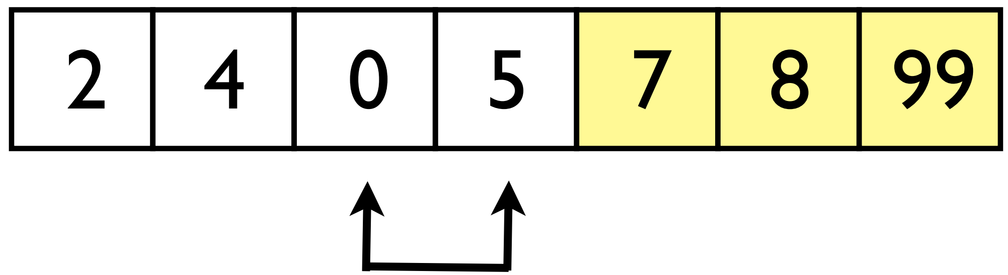
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

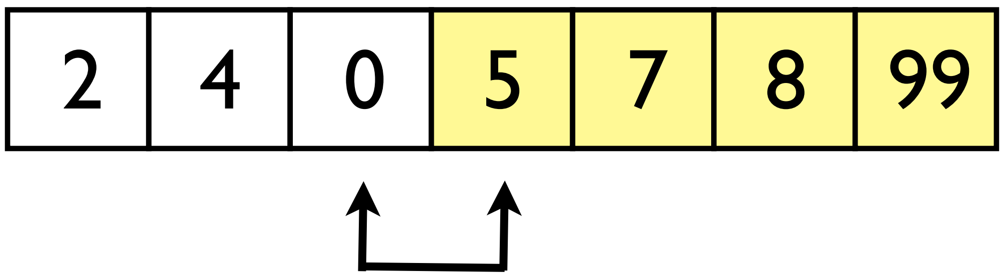
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

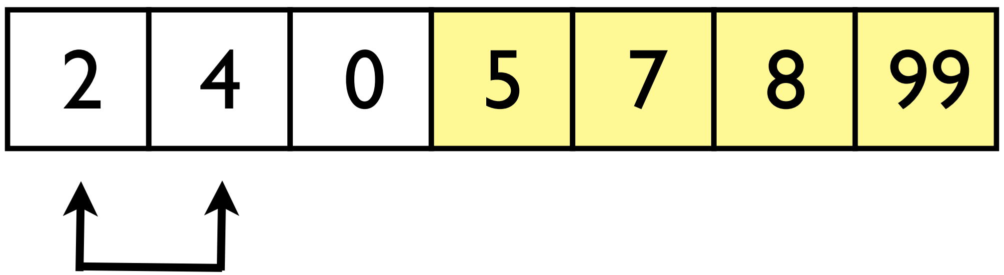
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

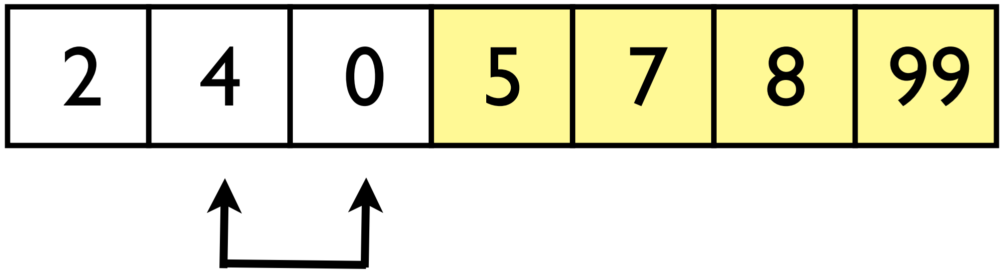
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

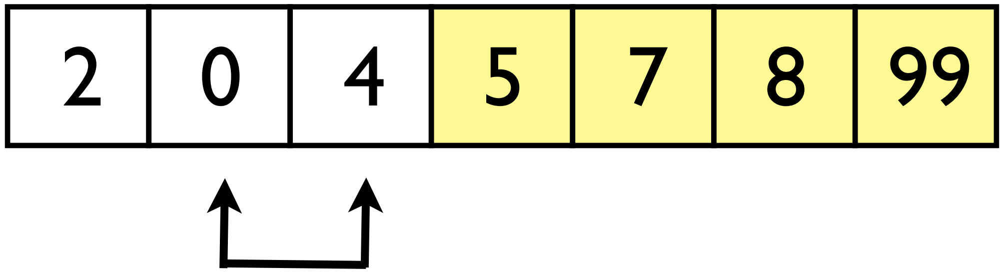
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

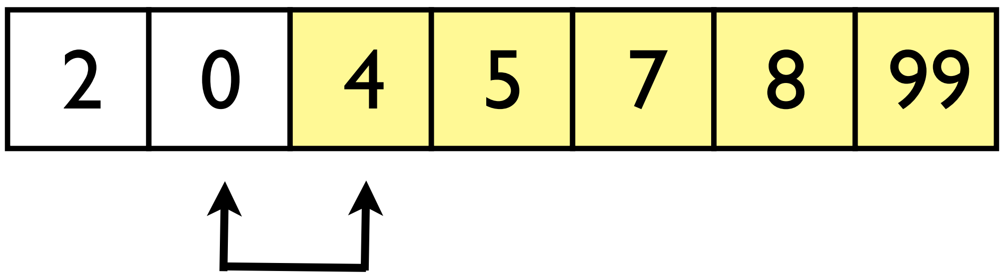
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

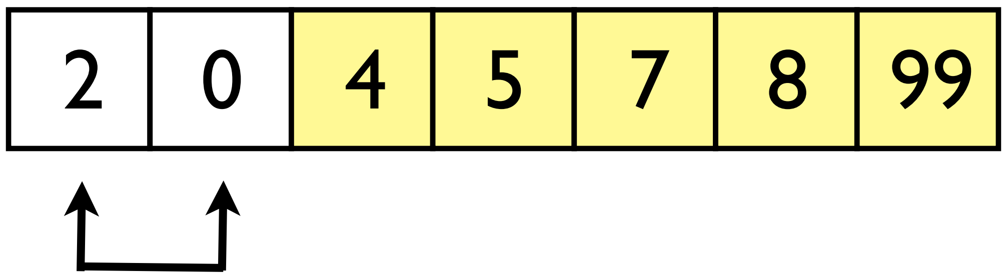
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

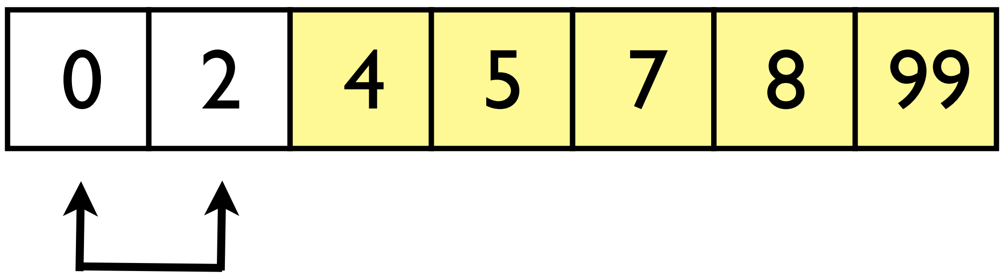
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

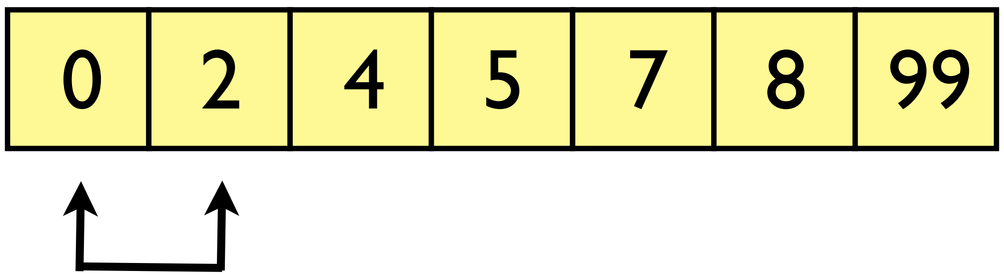
Compare each pair of adjacent items (left to right).

Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Bubble Sort: Algorithm

Sort a given list of integers (from small to large).



Bubble Sort

Compare each pair of adjacent items (left to right).

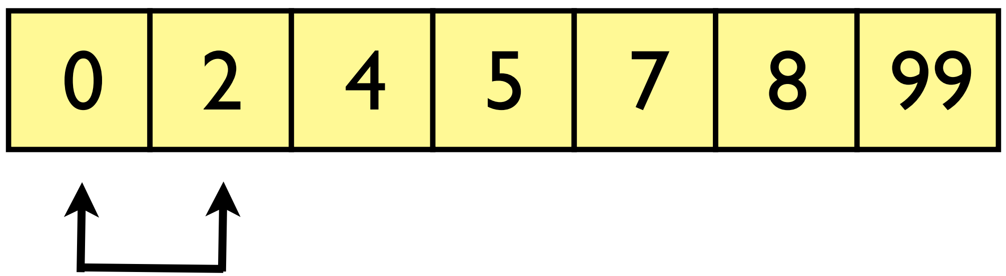
Swap them if they are in the wrong order.

Repeat until no more swaps are needed.

Large elements “bubble up”

Bubble Sort: Running Time

Sort a given list of integers (from small to large).



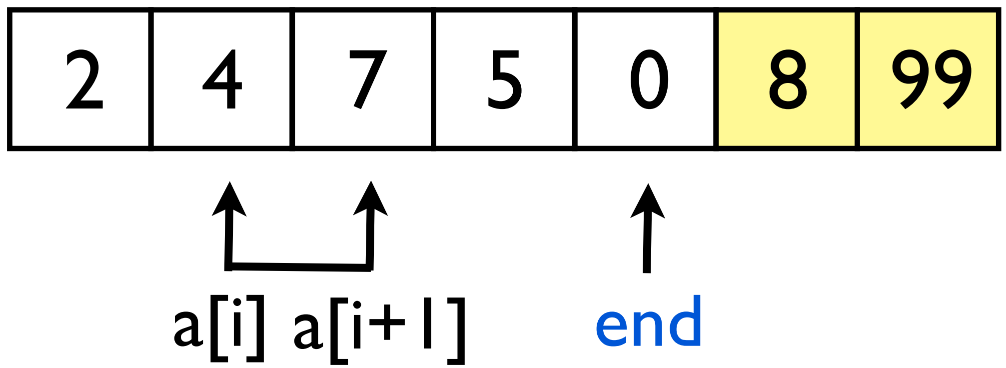
Bubble Sort

How many steps does this take (in the worst case)?

$$O(N^2)$$

Bubble Sort: Code

<u>Bubble sort snapshot</u>



repeat until no more swaps:

for i = 0 to end:

if a[i] > a[i+1], swap a[i] and a[i+1]

decrement end

Bubble Sort: Code

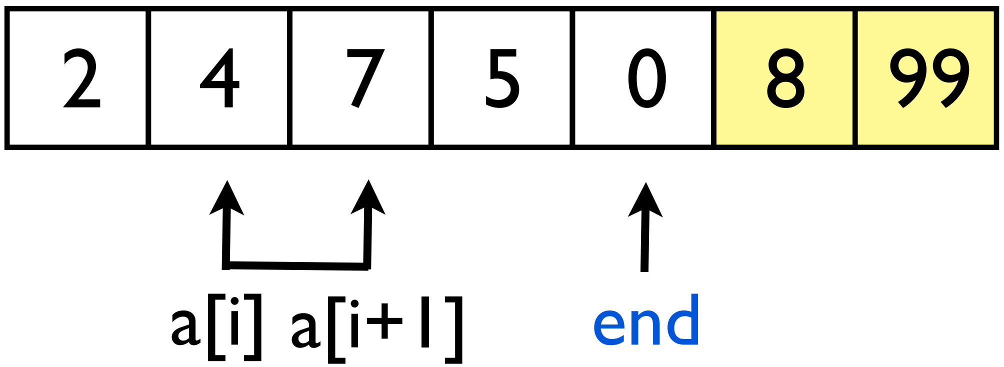
repeat until no more swaps:

for i = 0 to end:

if a[i] > a[i+1], swap a[i] and a[i+1]

decrement end

```
def bubbleSort(a):
    swapped = True
    end = len(a)-1
    while(swapped):
        swapped = False
        for i in range(end):
            if(a[i] > a[i+1]):
                (a[i], a[i+1]) = (a[i+1], a[i])
                swapped = True
        end -= 1
```



Merge Sort: Merge

Merge

The key subroutine/helper function:

merge(a, b)

<u>Input</u>: two sorted lists a and b

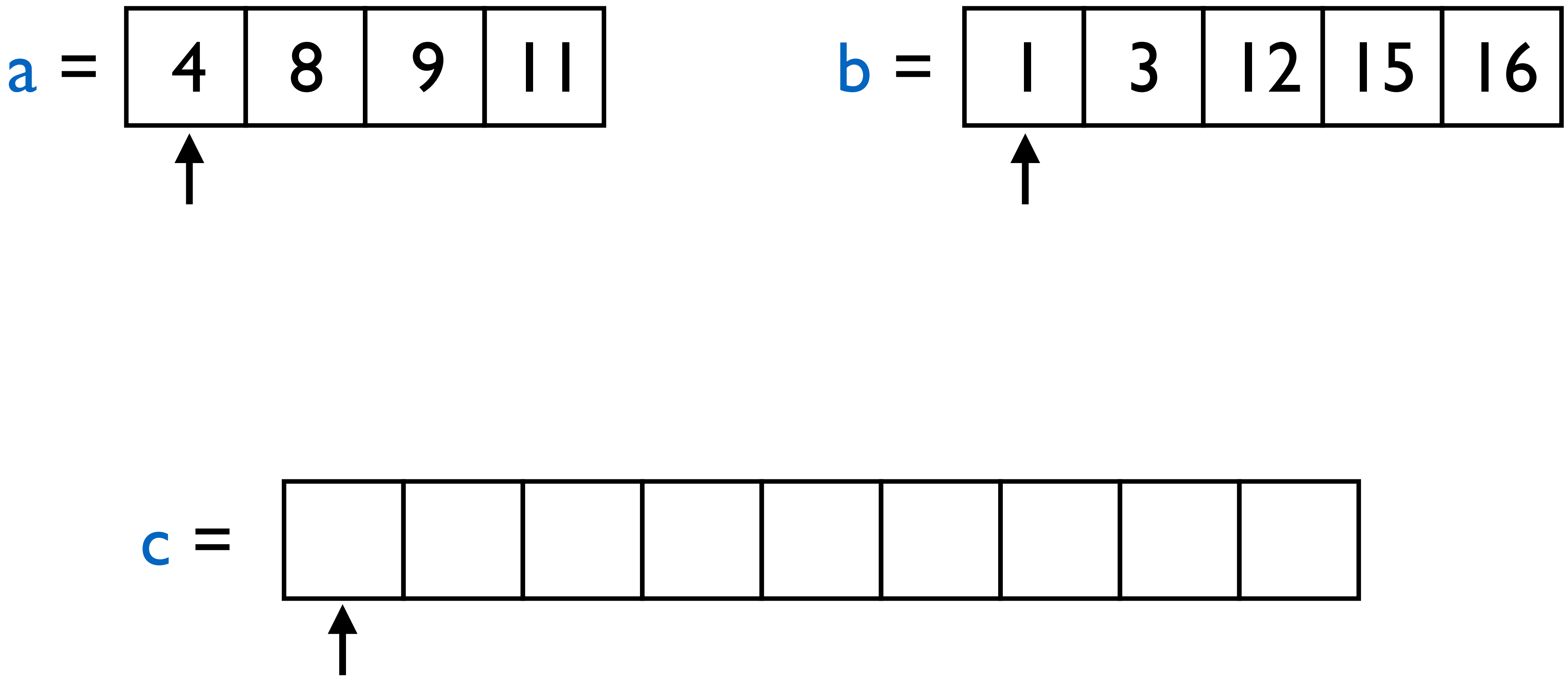
<u>Output</u>: a and b merged into a single list, all sorted.

Turns out we can do this pretty efficiently.

And that turns out to be quite useful!

Merge Sort: Merge Algorithm

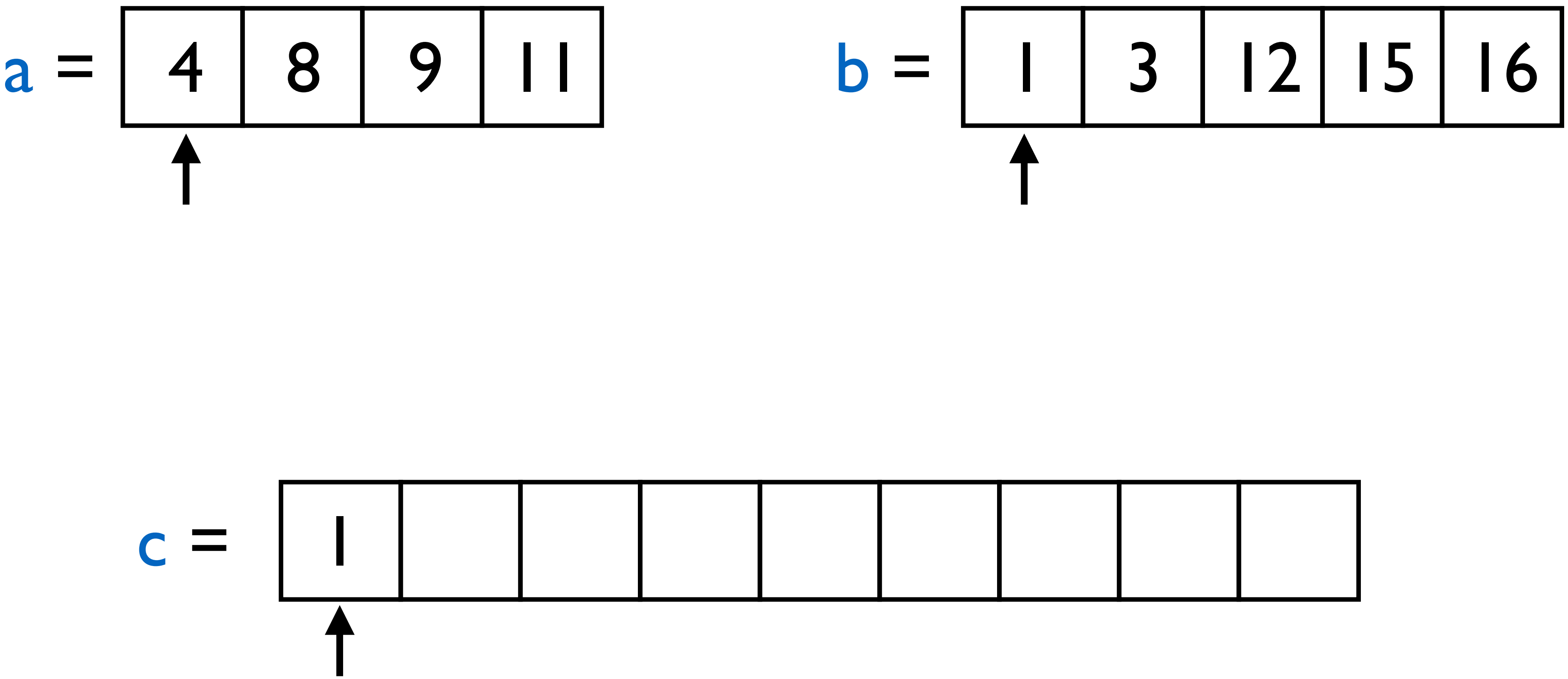
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

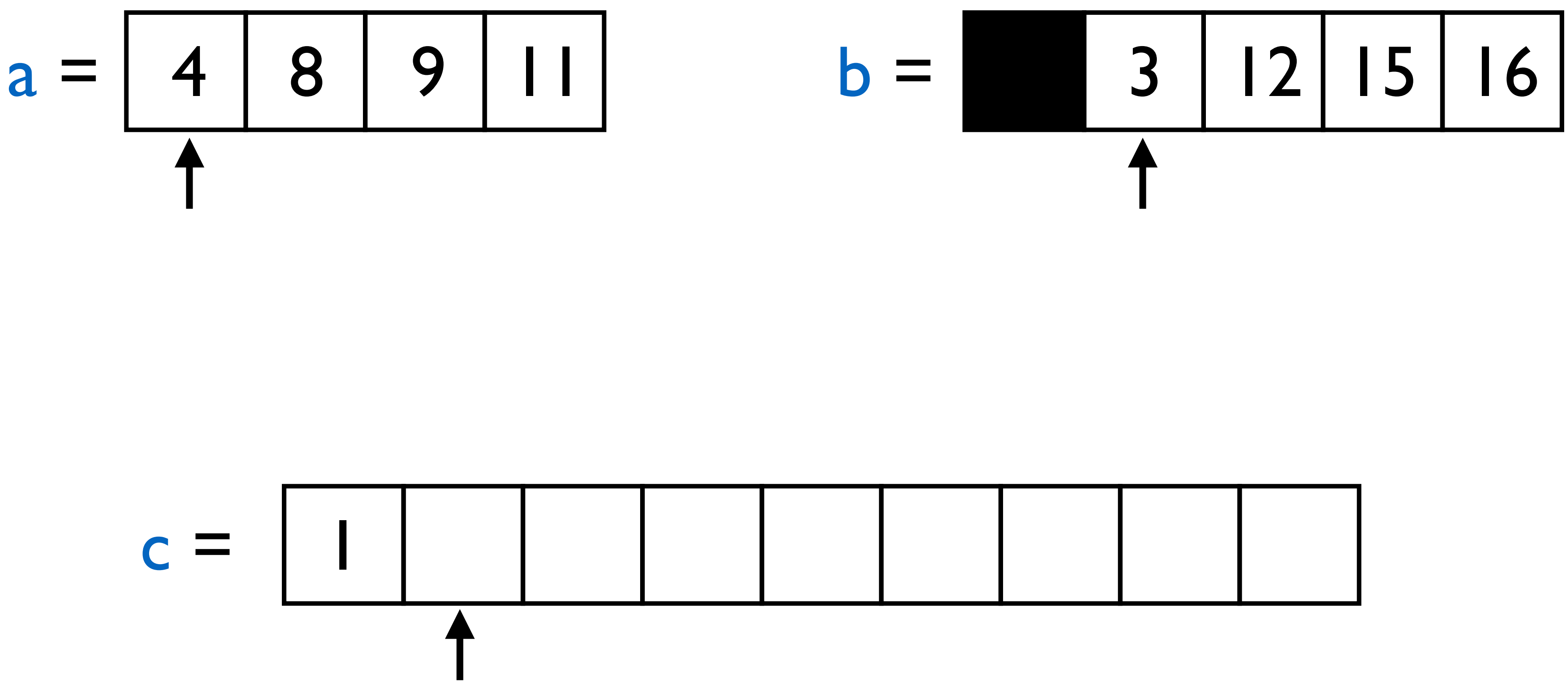
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

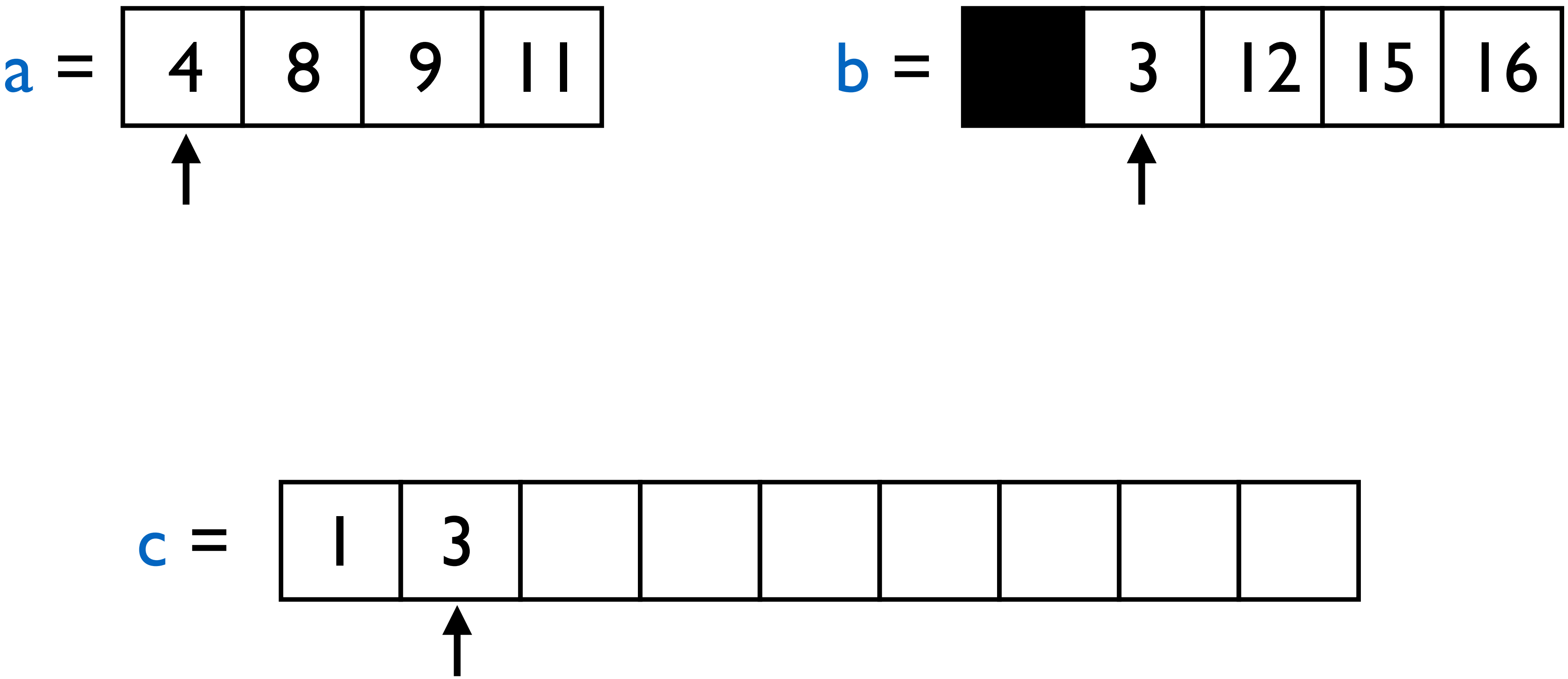
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

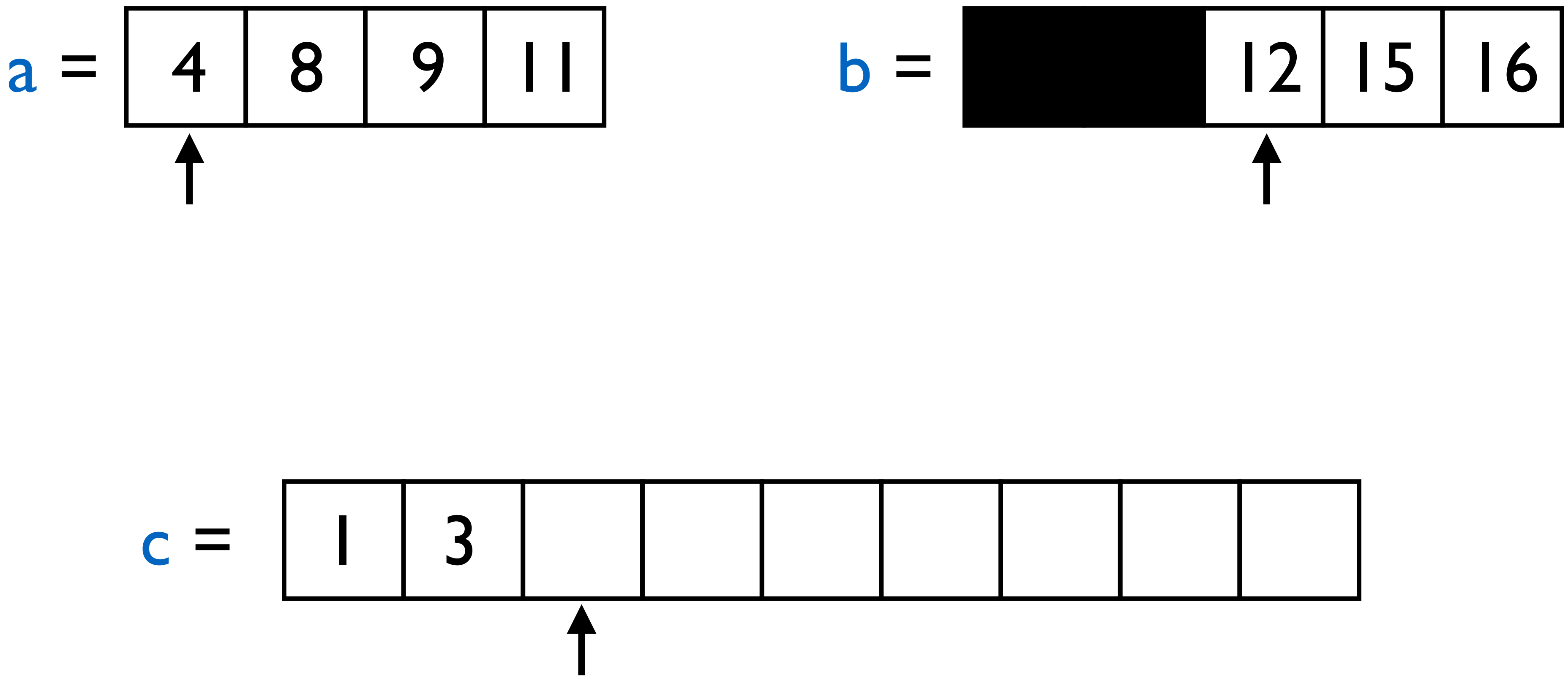
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

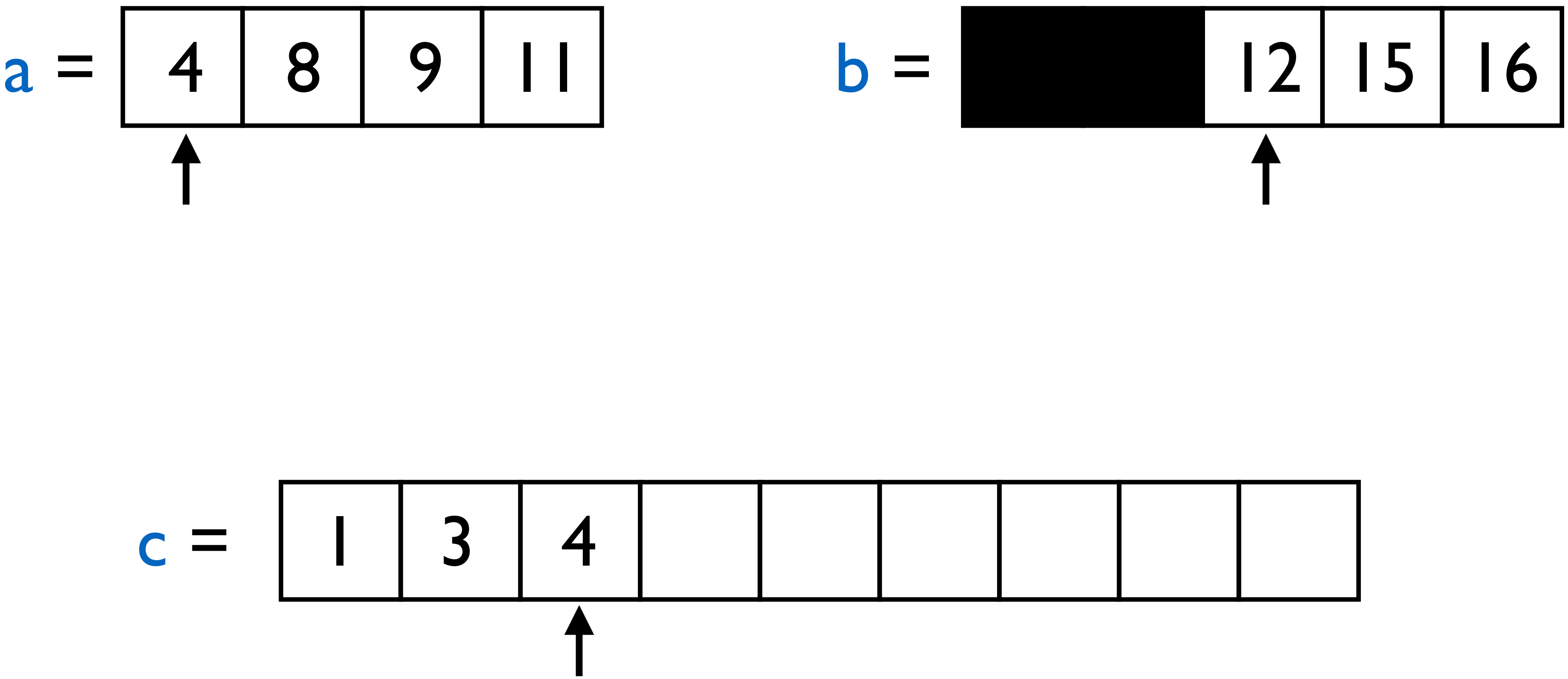
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

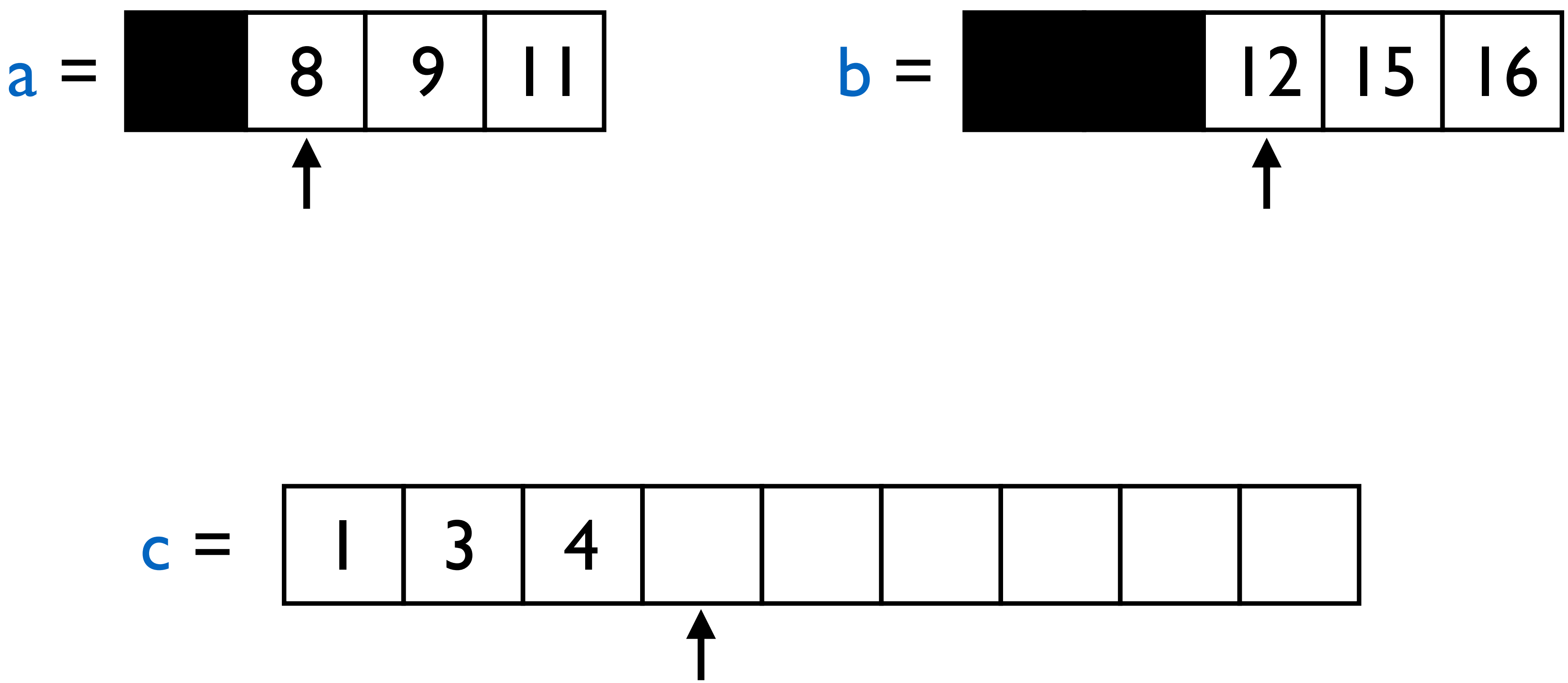
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

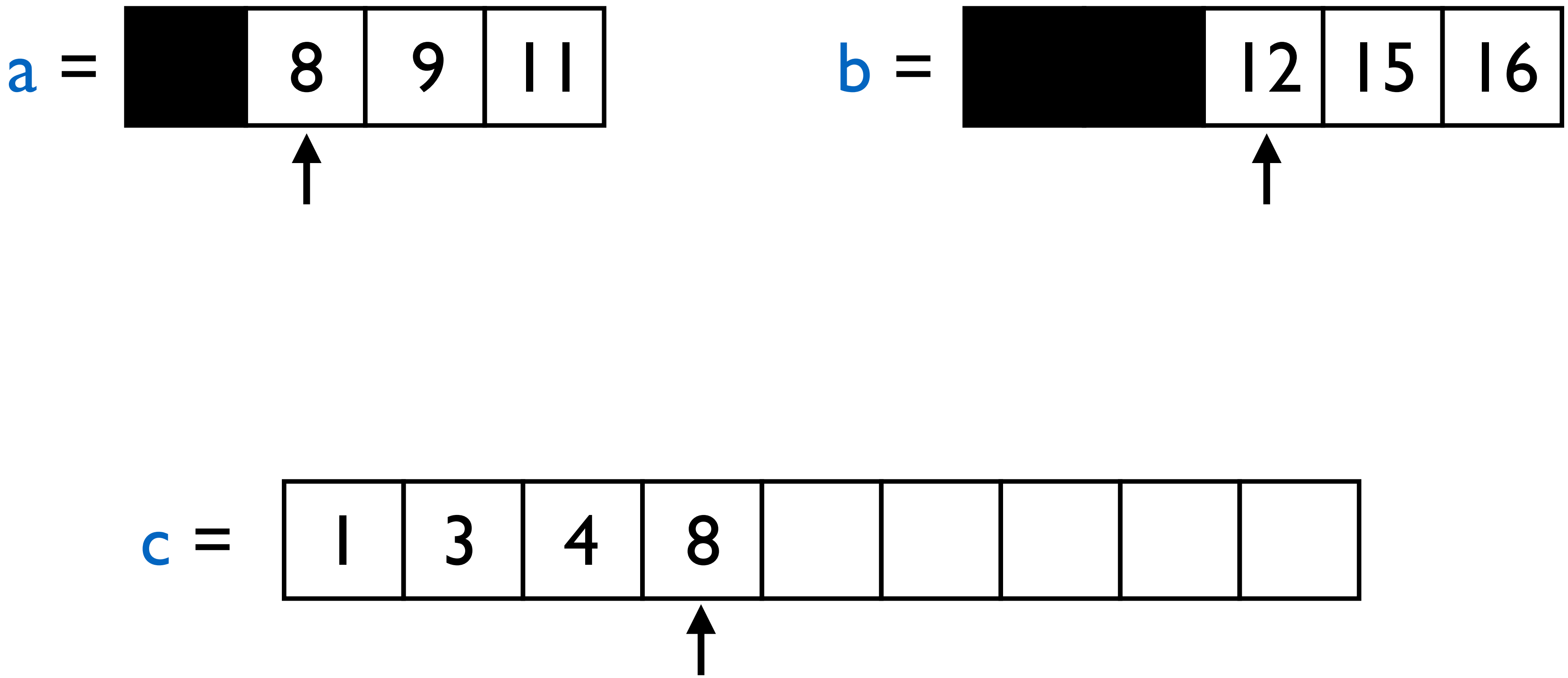
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

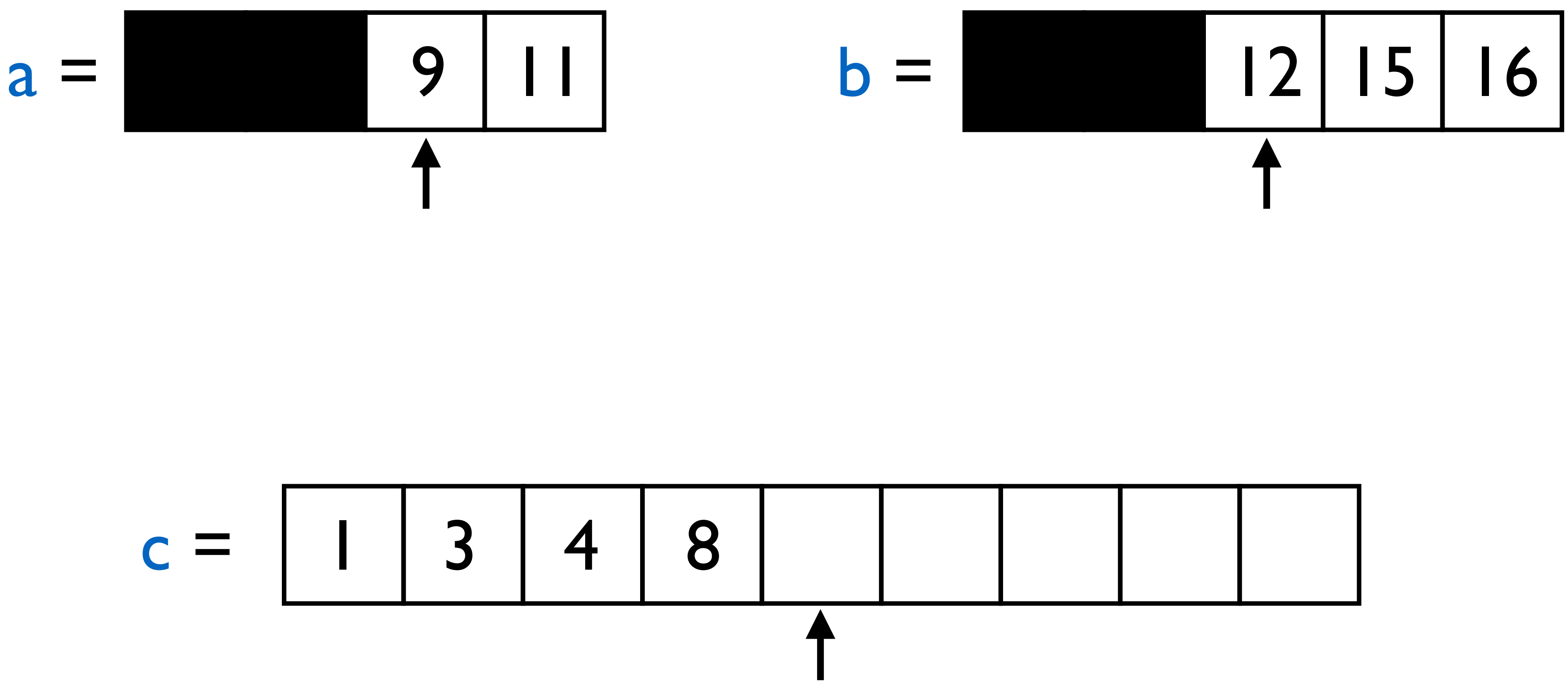
Merge



Main idea: $\min(c) = \min(\min(a), \min(b))$

Merge Sort: Merge Algorithm

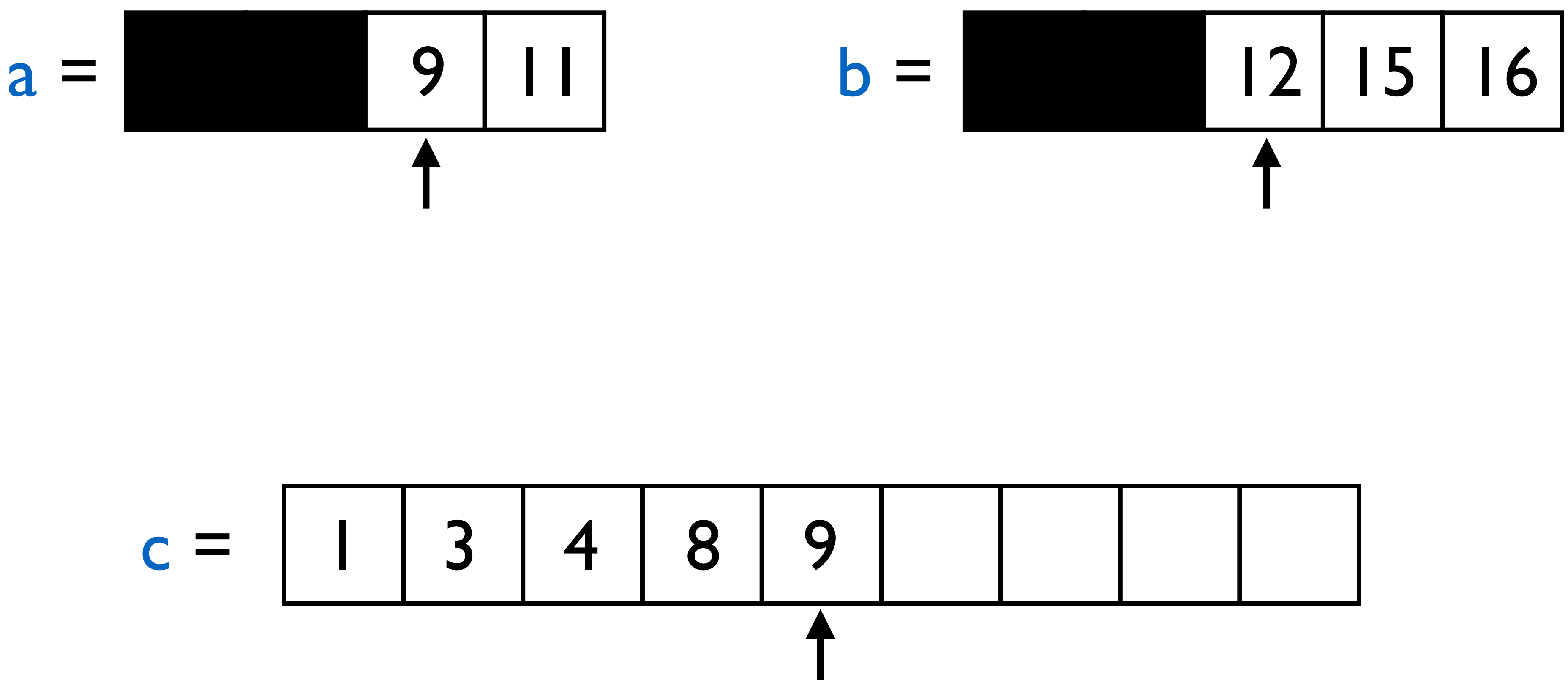
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

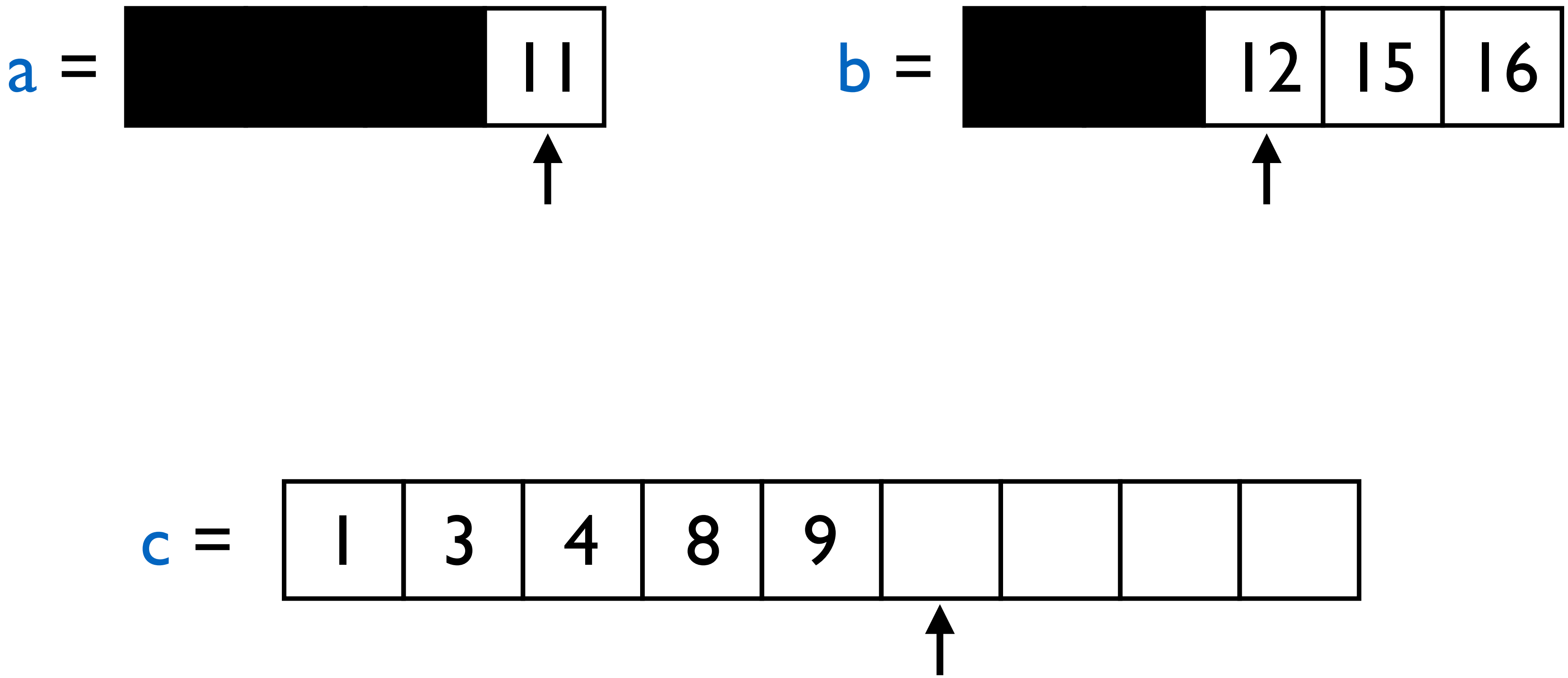
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

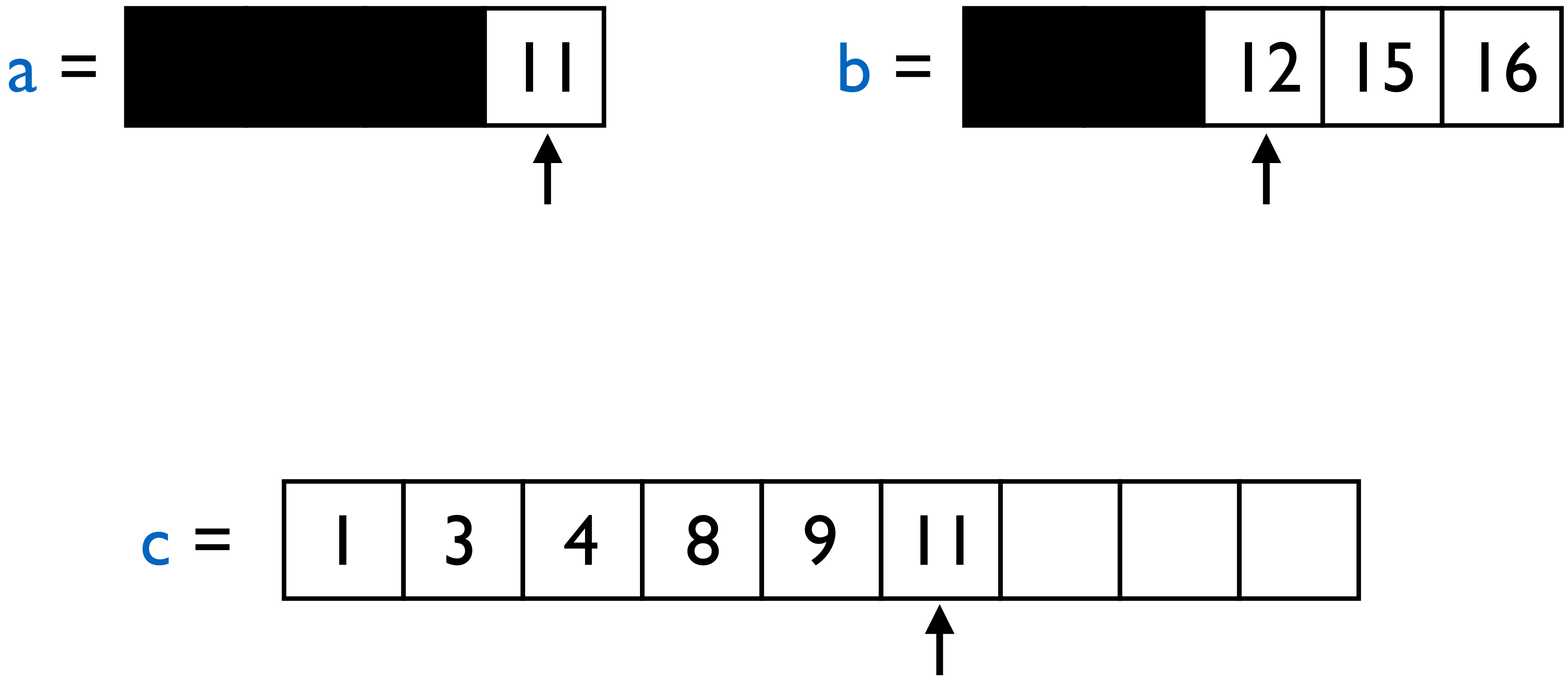
Merge



Main idea: min(c) = min(min(a), min(b))

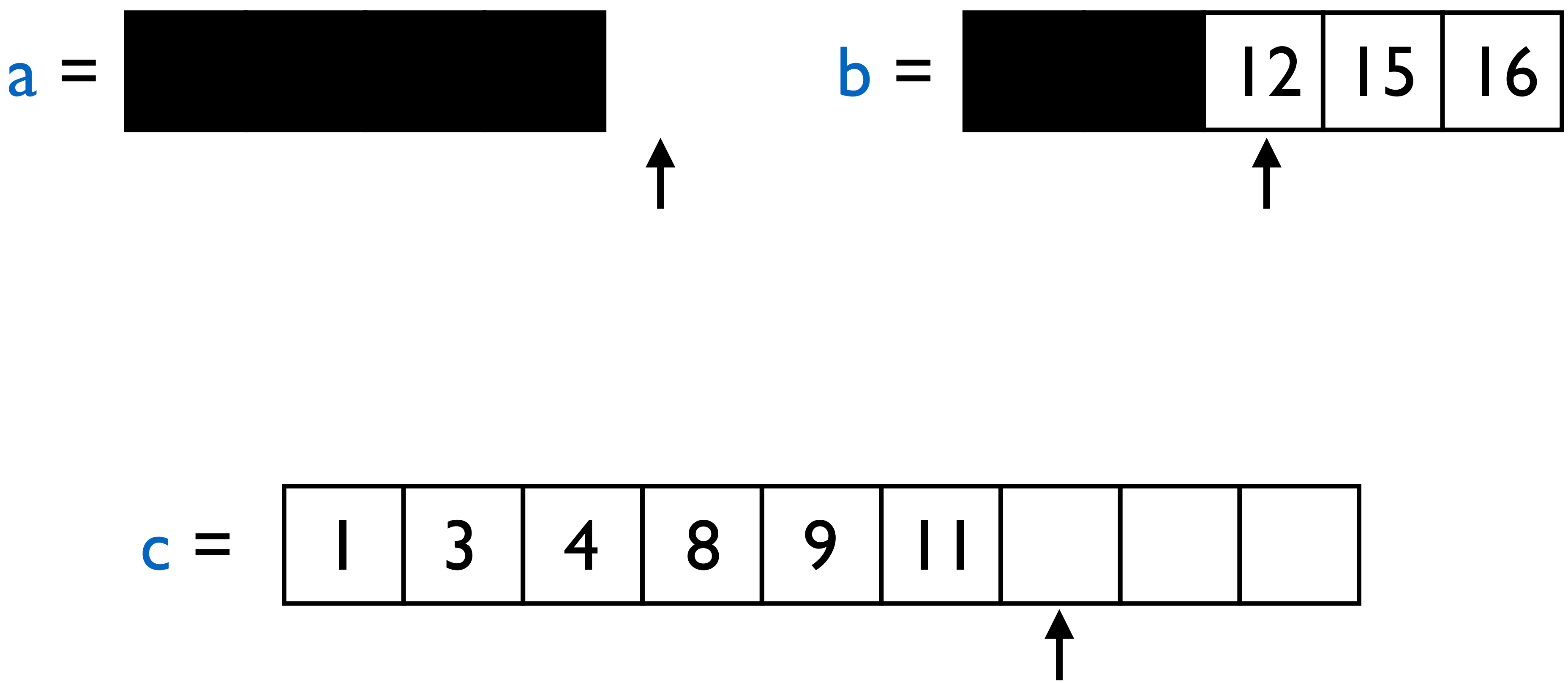
Merge Sort: Merge Algorithm

Merge



Main idea: min(c) = min(min(a), min(b))

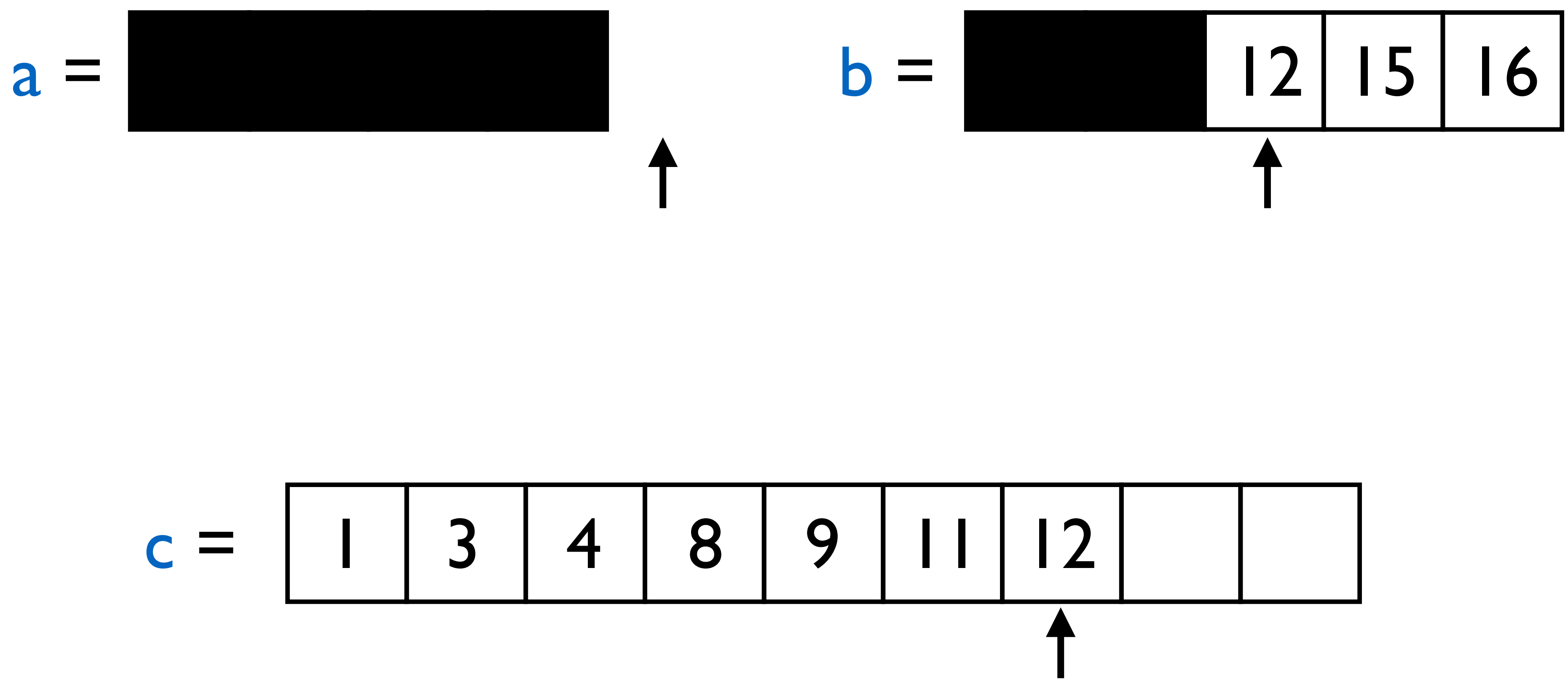
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

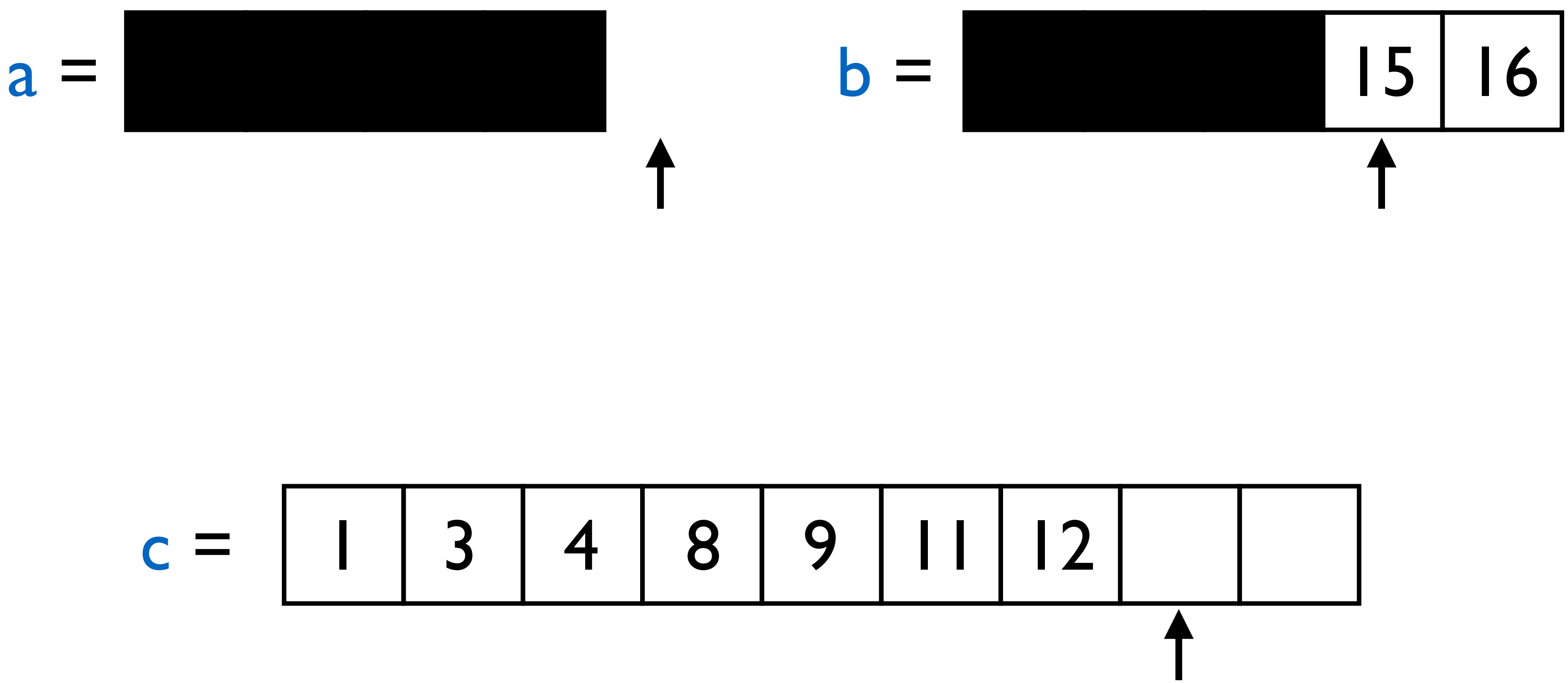
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

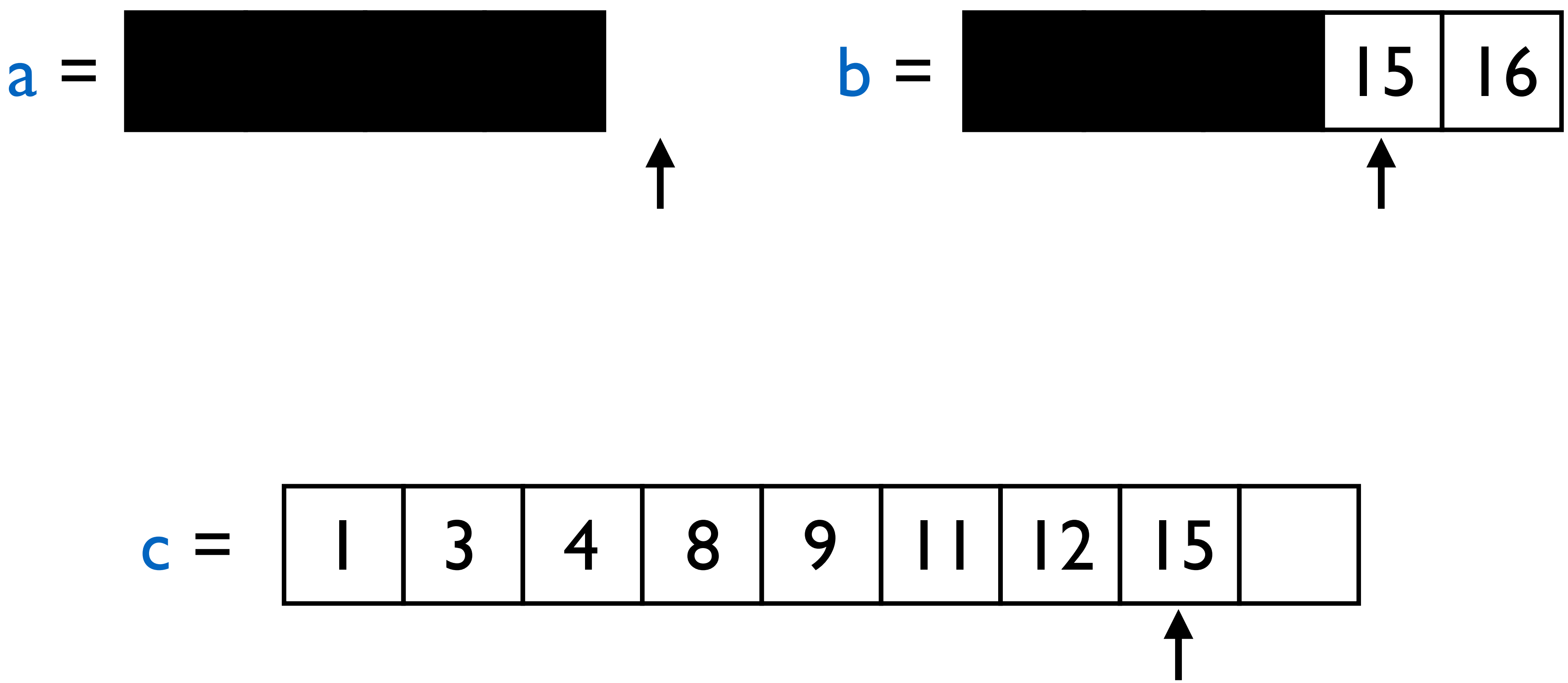
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

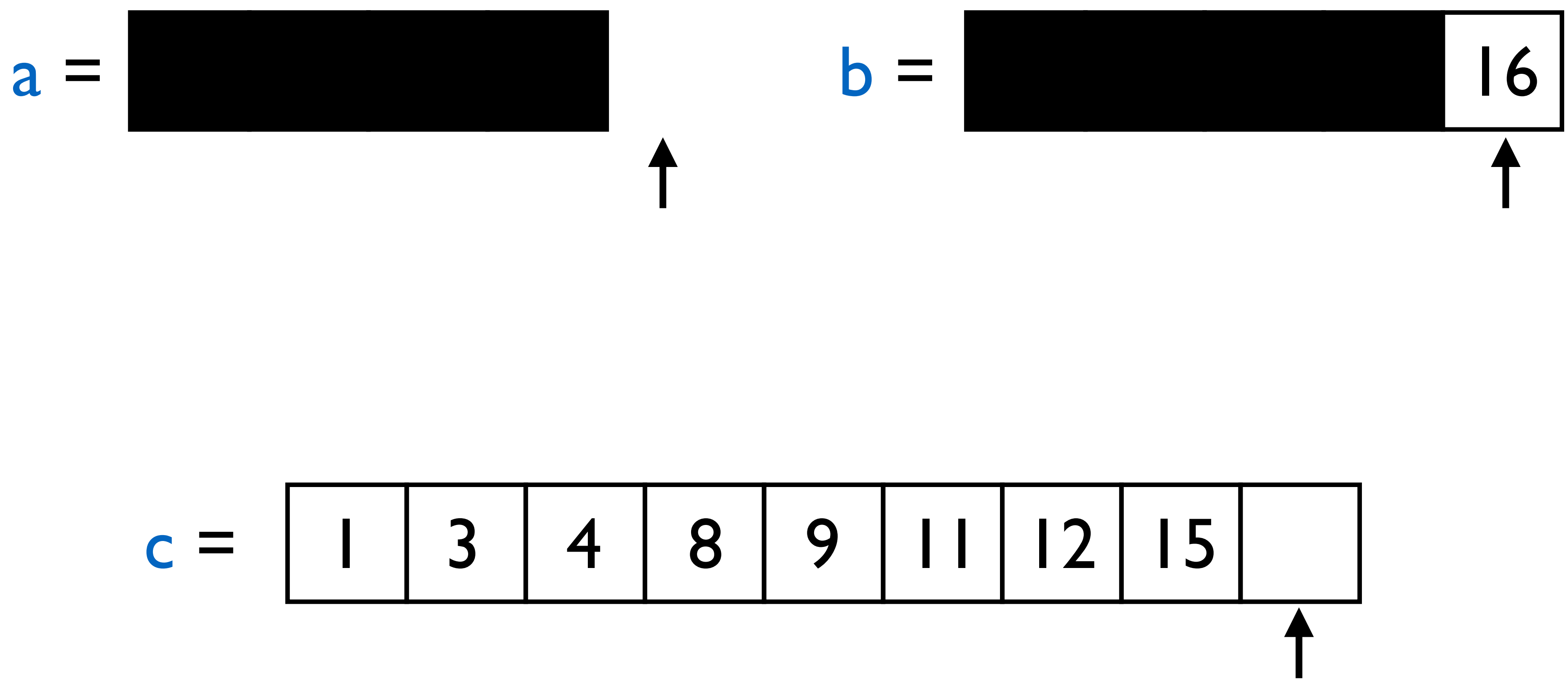
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

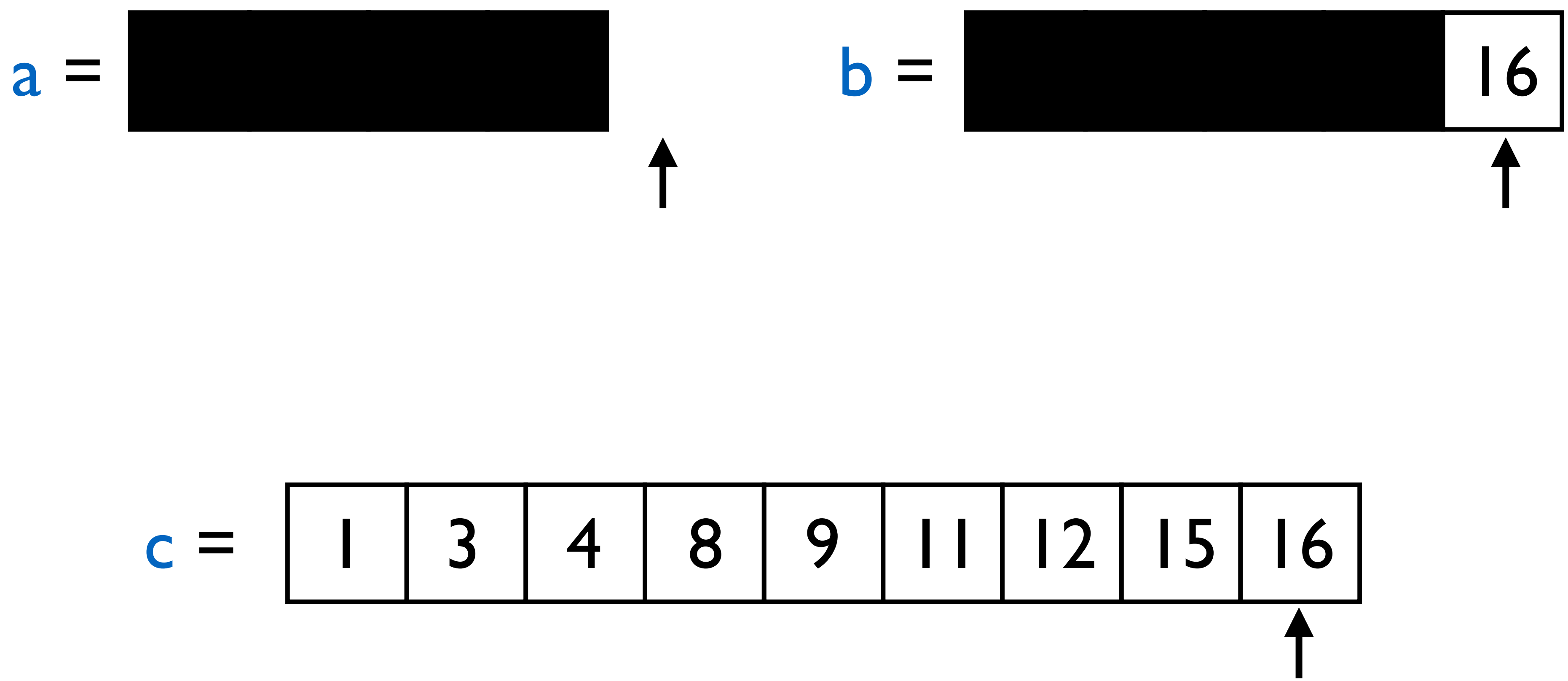
Merge



Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Algorithm

Merge



Main idea: min(c) = min(min(a), min(b))

Merge

a = [redacted] ↑

b = [redacted] ↑

c =

1	3	4	8	9	11	12	15	16

↑

Main idea: min(c) = min(min(a), min(b))

Merge Sort: Merge Running Time

Merge

a = [redacted]

b = [redacted]

c =

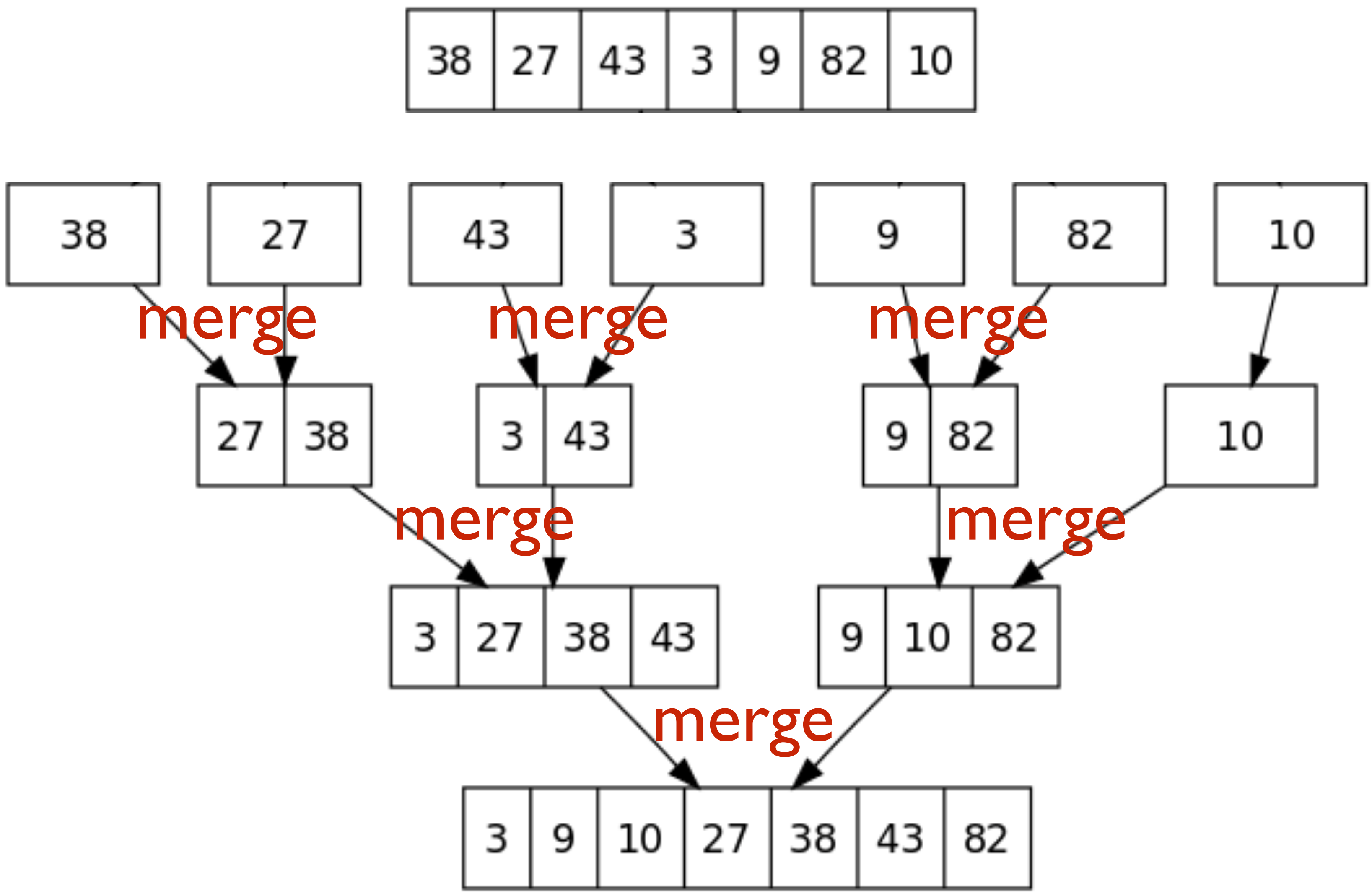
1	3	4	8	9	11	12	15	16

Running time? $N = \text{len}(a) + \text{len}(b)$

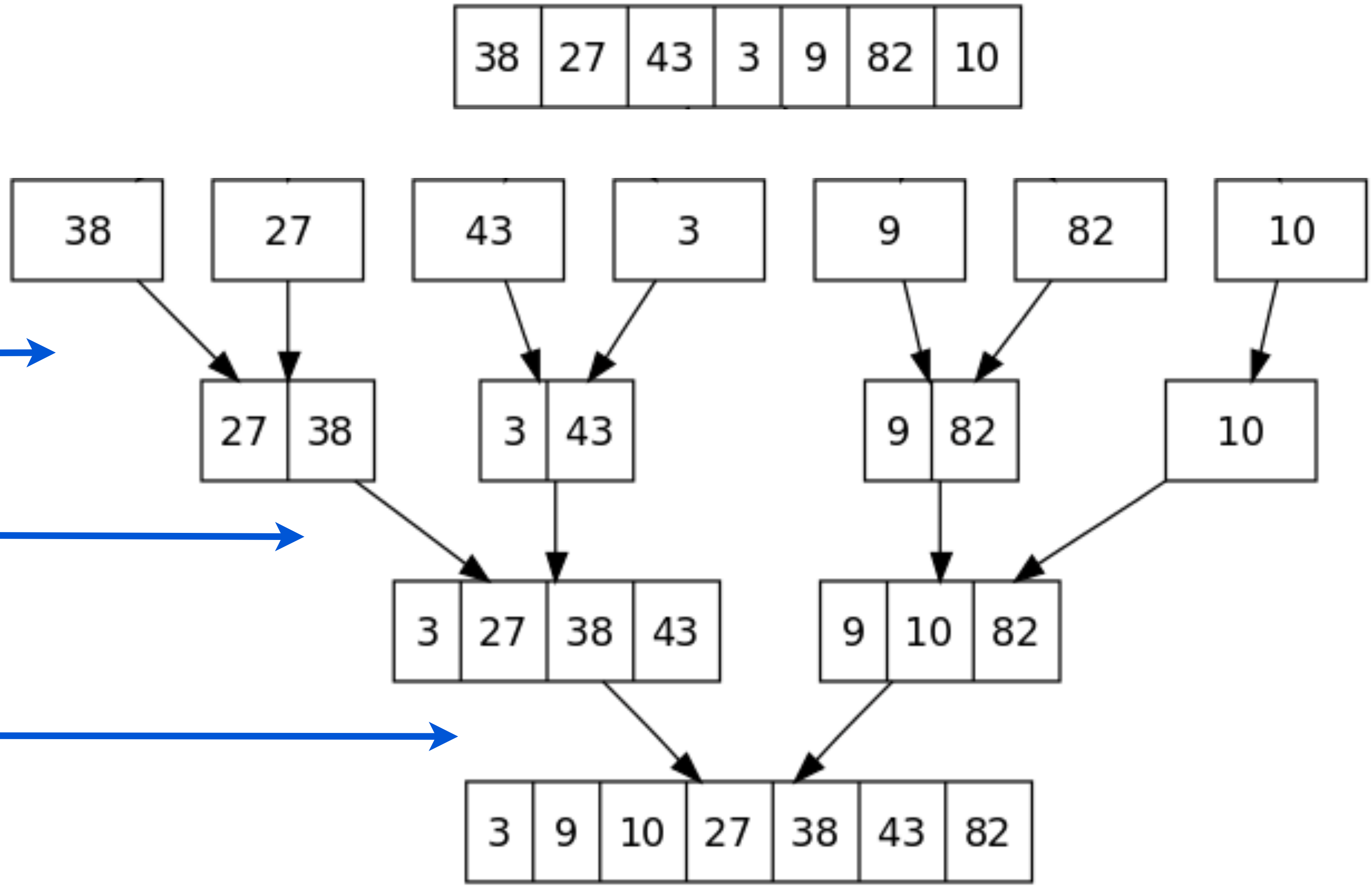
\# steps: $O(N)$

Merge Sort: Algorithm

Merge Sort



Merge Sort: Running Time



$O(N)$

$O(N)$

$O(N)$

$O(\log N)$ levels

Total: $O(N \log N)$