Sorting and Divide and Conquer

Diego Useche - dh.useche@uniandes.edu.co

Metodos Computacionales II

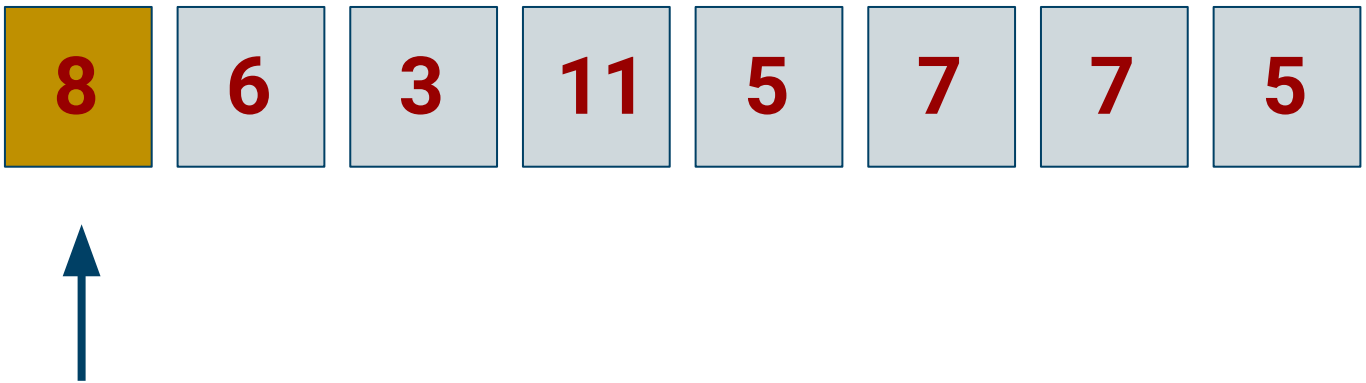
Physics Department, Universidad de los Andes, Bogotá

Sorting problem

- Given a set find an efficient algorithm that sorts the set.

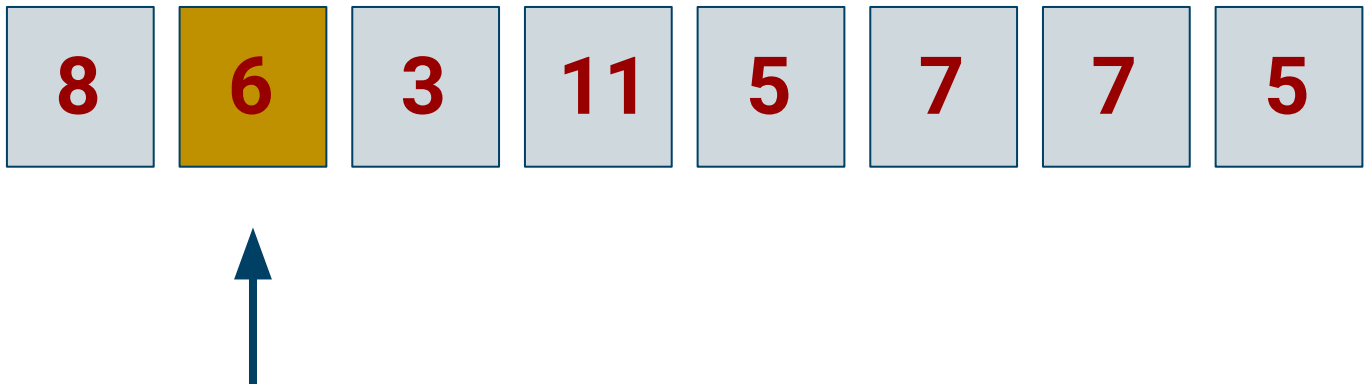
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.



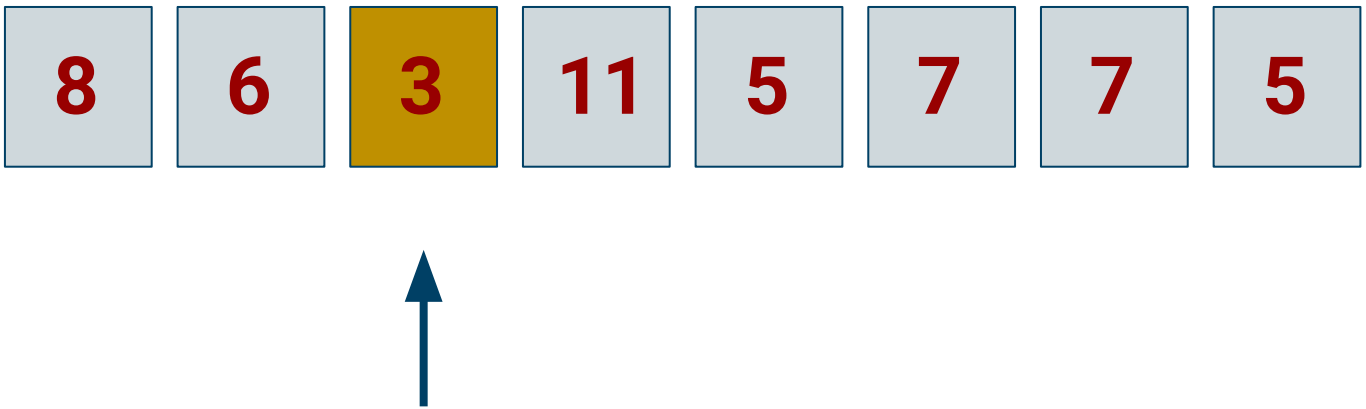
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.



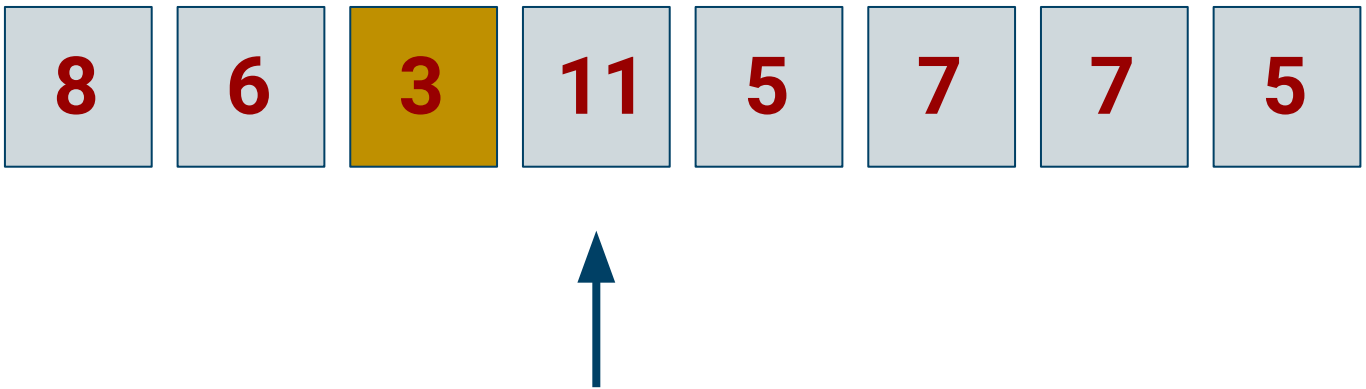
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.



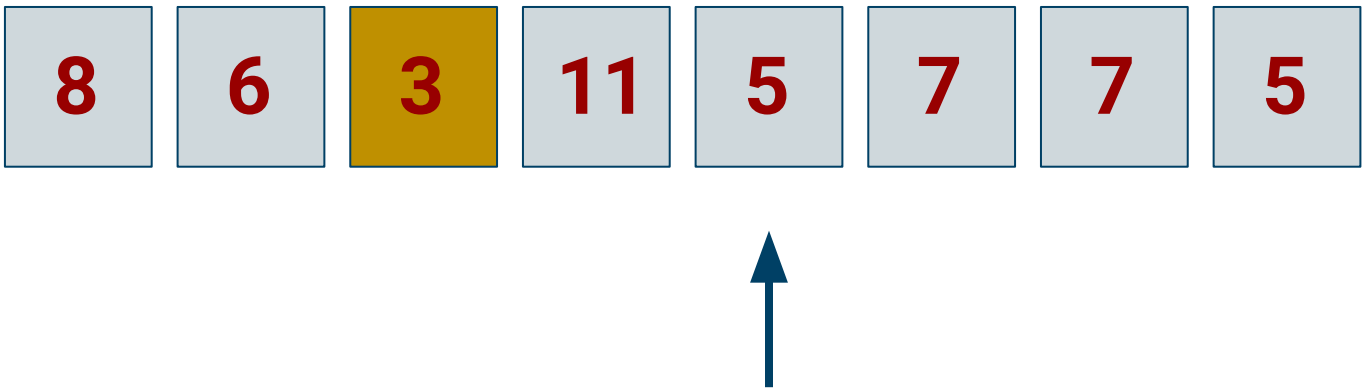
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.



Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.



Naive algorithm (Brute Force)

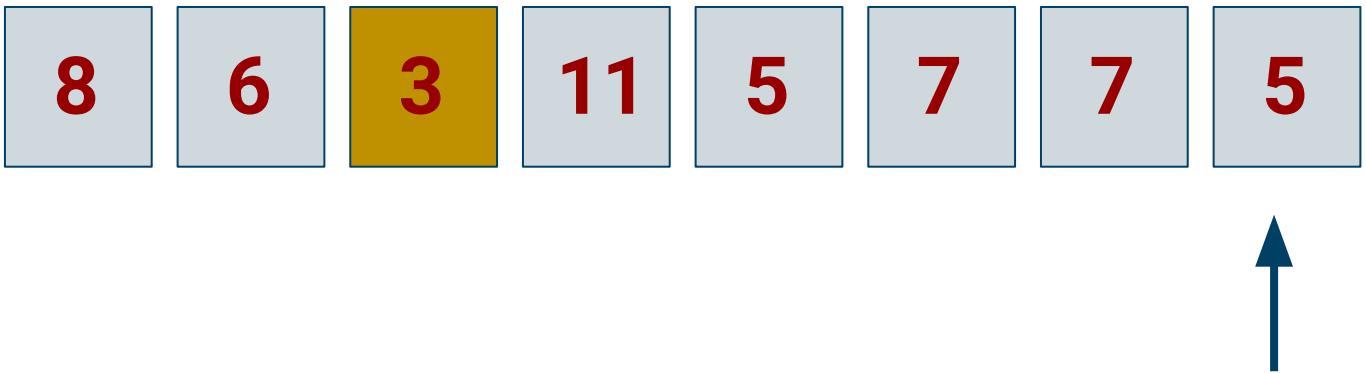
- Scroll through the whole list, find the minimum in the list move to the first position.

Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.

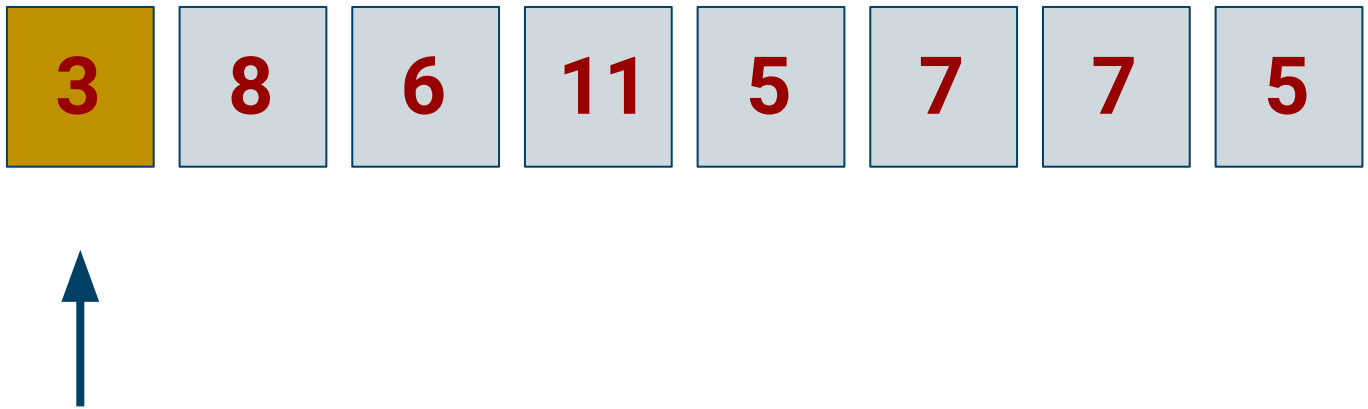
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move to the first position.



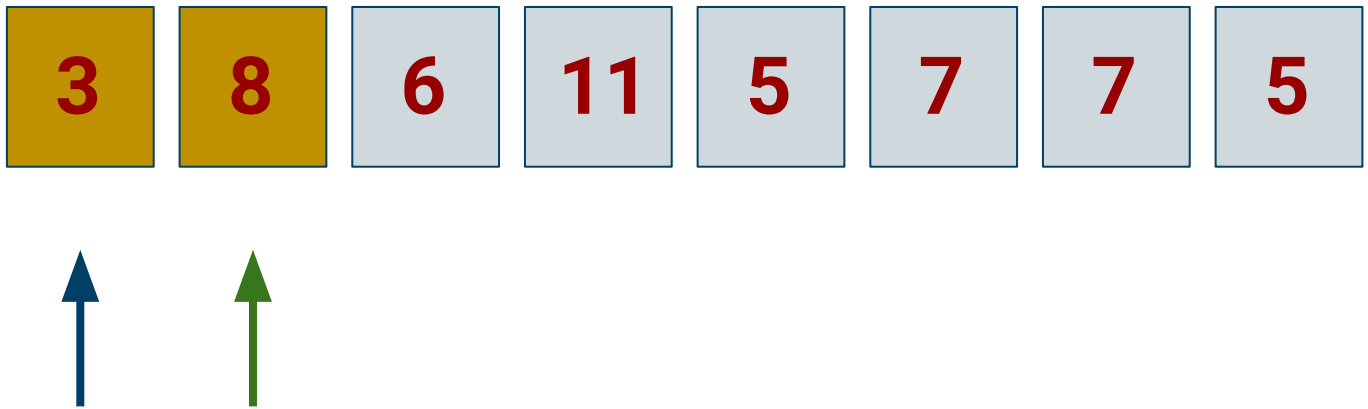
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



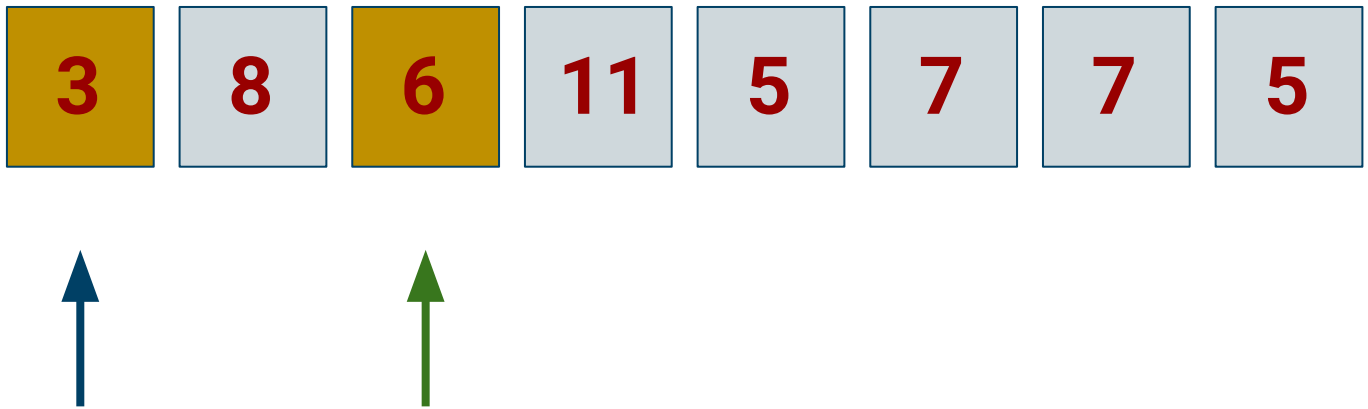
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



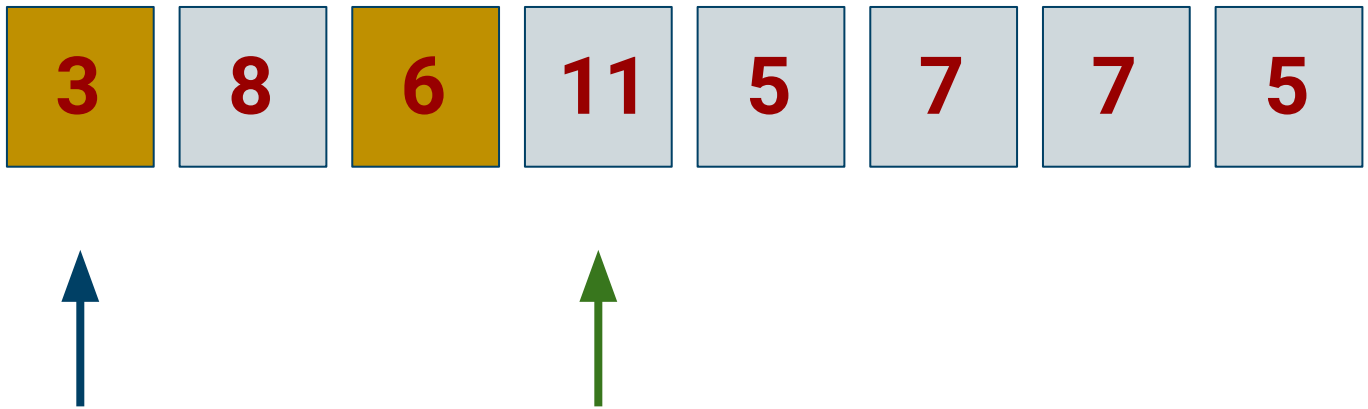
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



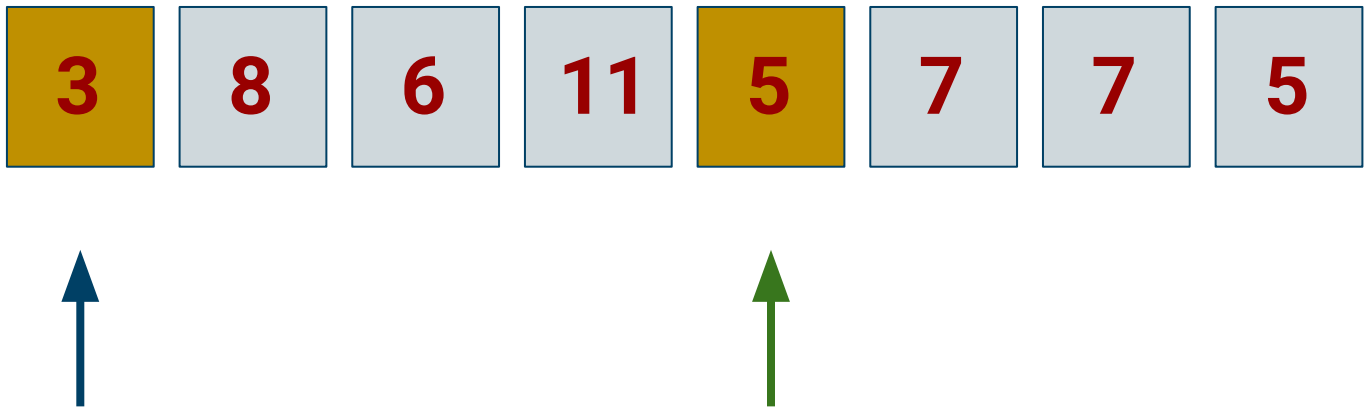
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



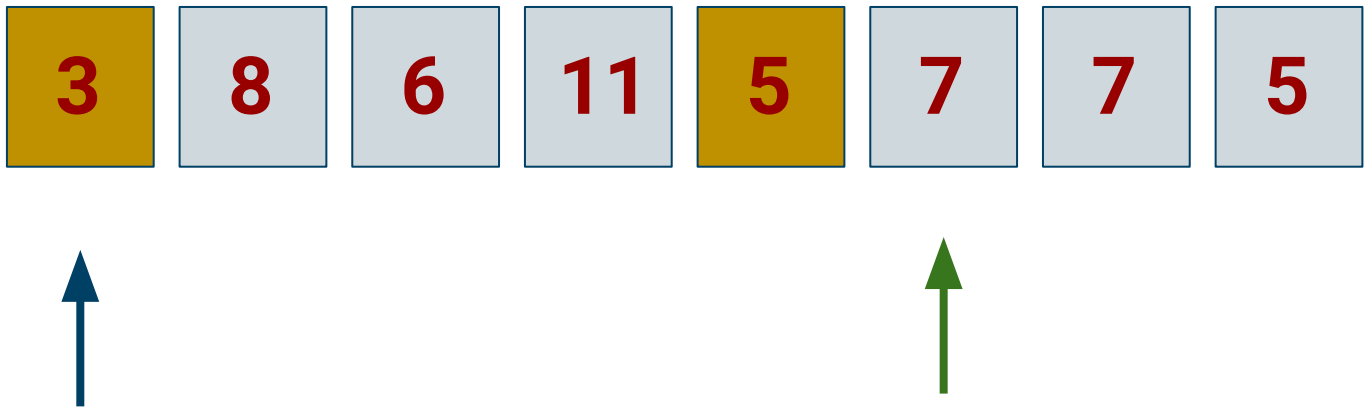
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



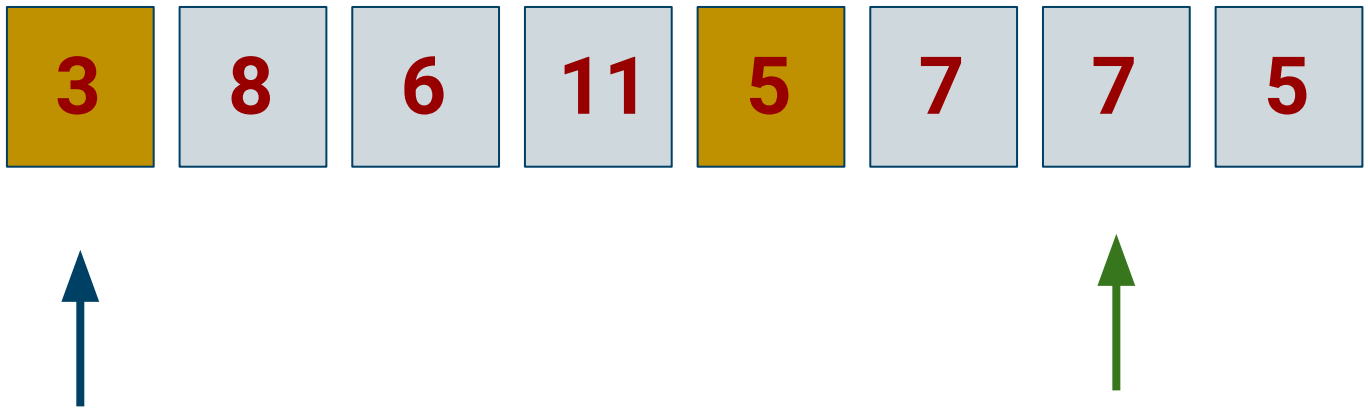
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



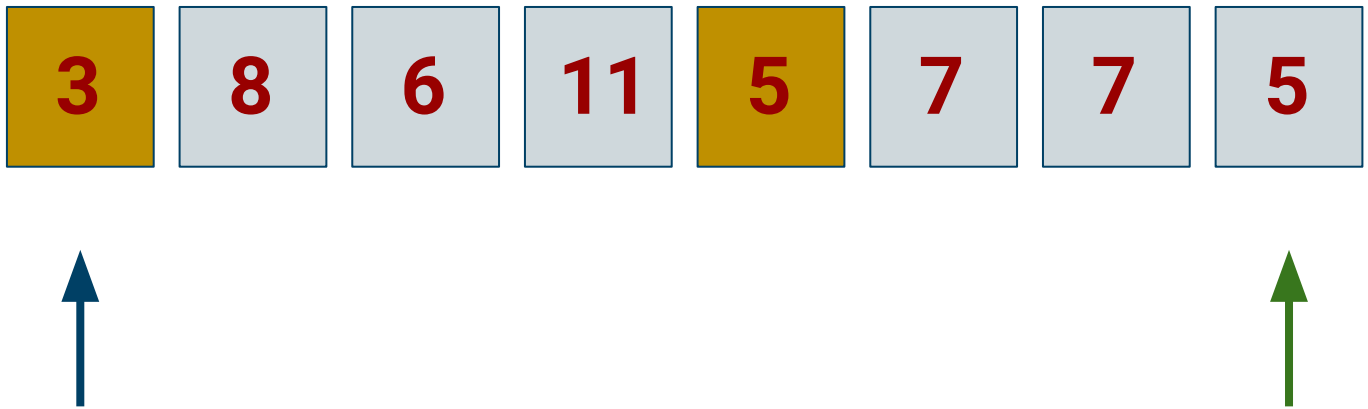
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



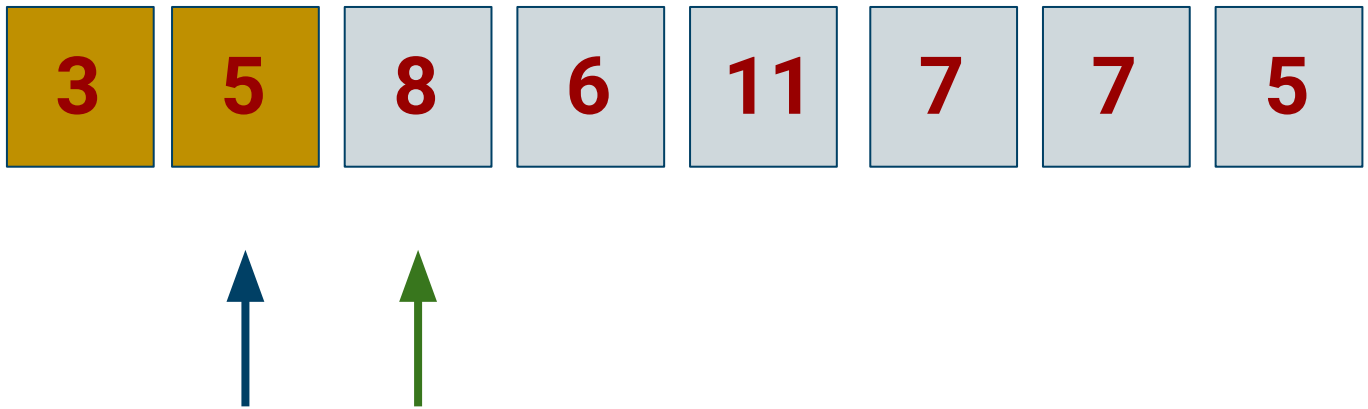
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



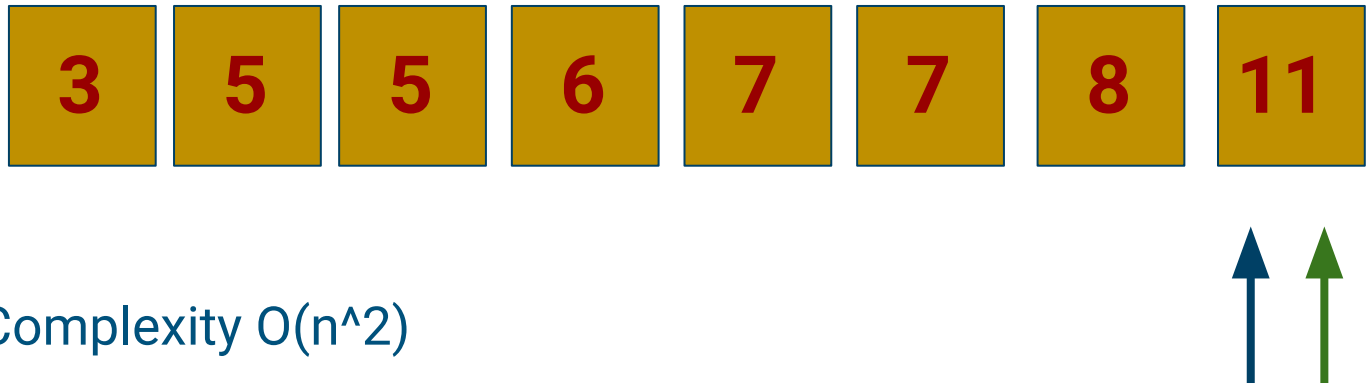
Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list
  move the minimum to the first position.



Naive algorithm (Brute Force)

- Scroll through the whole list, find the minimum in the list move the minimum to the first position.



- Complexity O(n^2)

Naive algorithm Pseudo Code

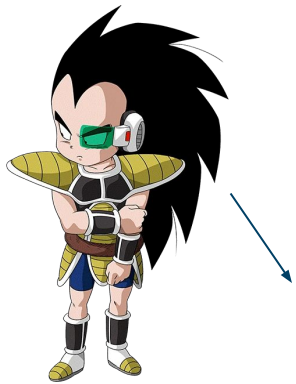
Input: An array $A[1..n]$ of n elements.
Output: $A[1..n]$ sorted in descending order

```
1. for i ← 1 to n - 1
2.   min ← i
3.   for j ← i + 1 to n   {Find the i th smallest element.}
4.       if A[j] < A[min] then
5.           min ← j
6.   end for
7.   if min ≠ i then interchange A[i] and A[min]
8. end for
```

Other algorithms for sorting

Method	Average Complexity
Bubble Sort	O(n^2)
Selection Sort	O(n^2)
Insertion Sort	O(n^2)
Heap Sort	O(n log (n))
Quick Sort	O(n log (n))
Radix Sort	O(n)
Merge Sort	O(n log (n))

Other algorithms for sorting



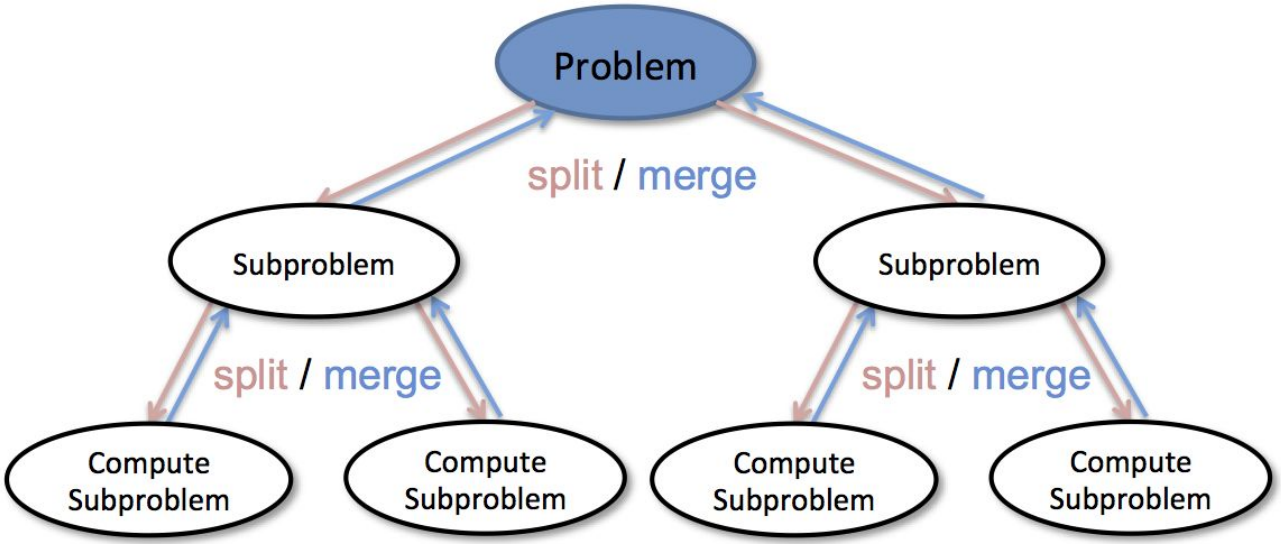
Method	Average Complexity
Bubble Sort	O(n^2)
Selection Sort	O(n^2)
Insertion Sort	O(n^2)
Heap Sort	O(n log (n))
Quick Sort	O(n log (n))
Radix Sort	O(n)
Merge Sort	O(n log (n))

Other algorithms for sorting

Method	Average Complexity
Bubble Sort	O(n^2)
Selection Sort	O(n^2)
Insertion Sort	O(n^2)
Heap Sort	O(n log (n))
Quick Sort	O(n log (n))
Radix Sort	O(n)
Merge Sort	O(n log (n))

Merge Sort (Divide and Conquer Algorithm)

- Divide the problems into subproblems, solve iteratively the subproblems.



Divide and Conquer Algorithm Applications

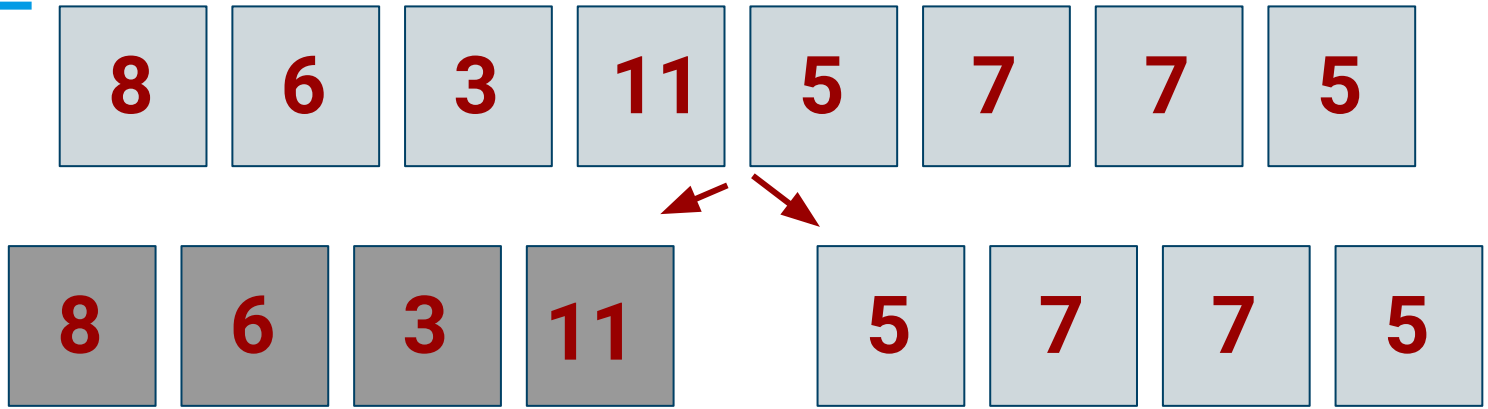
- Binary Search
- Merge Sort
- Quick Sort
- Closest Pair of Points
- Strassen’s Multiplication
- Karatsuba Algorithm
- Cooley-Tukey Algorithm
- Fast Fourier Transform

Sorting problem

- Given a set find an efficient algorithm that sorts the set.

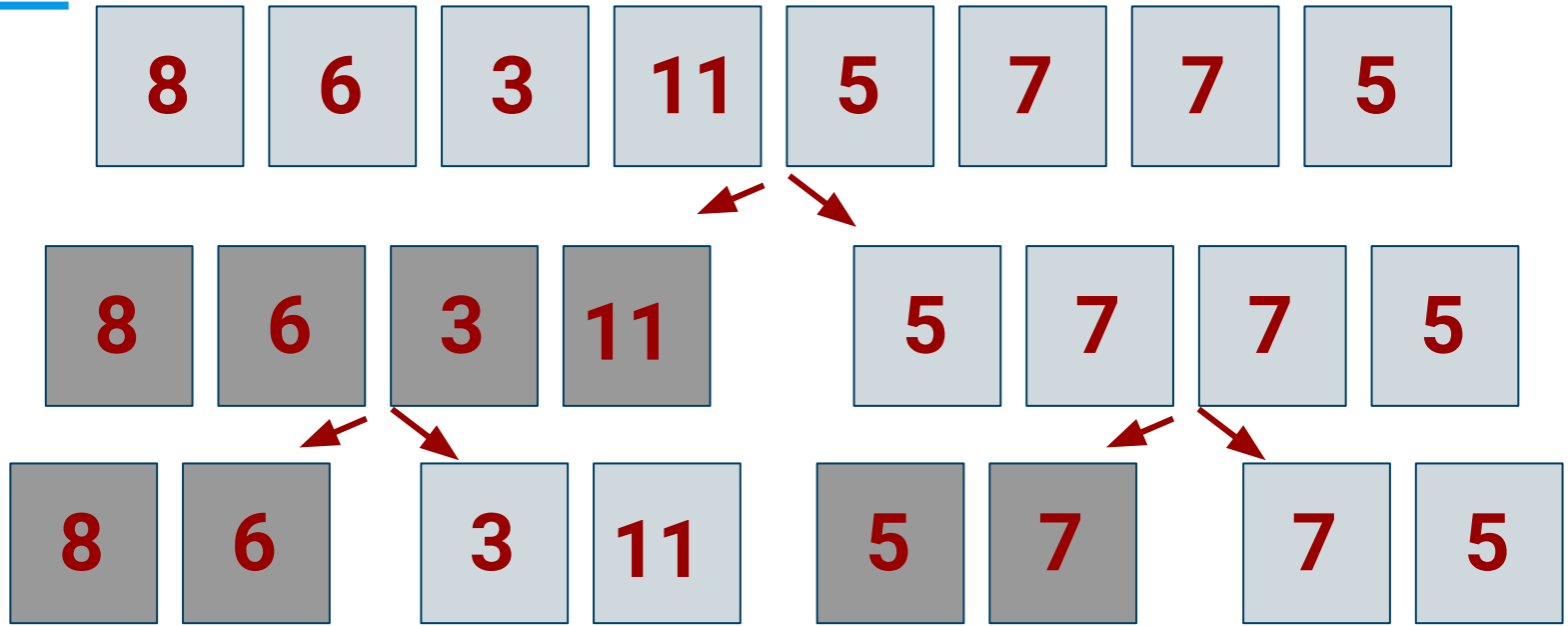
Merge Sort

- Divide the problem into subproblems



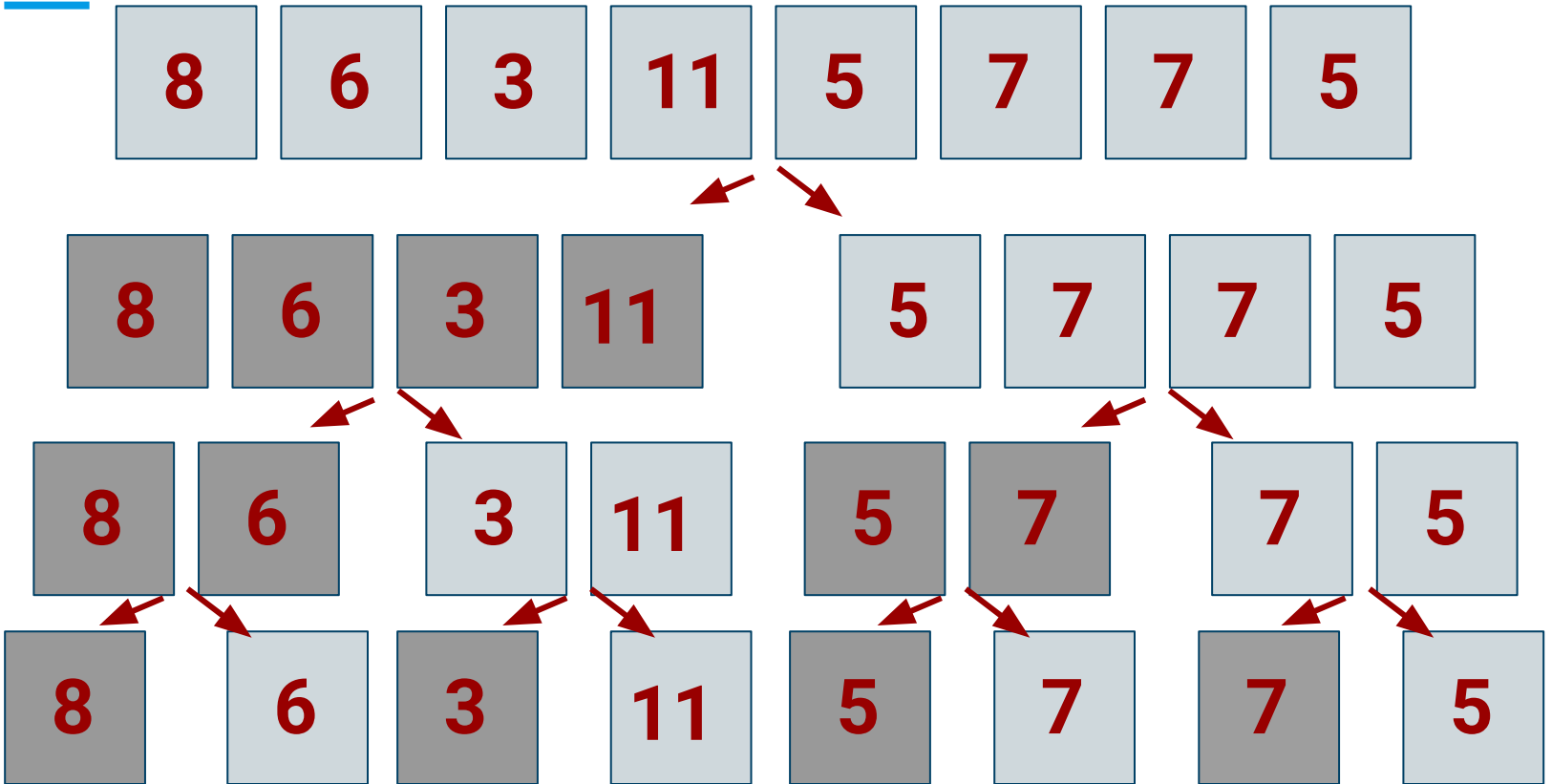
Merge Sort

- Divide the problem into subproblems



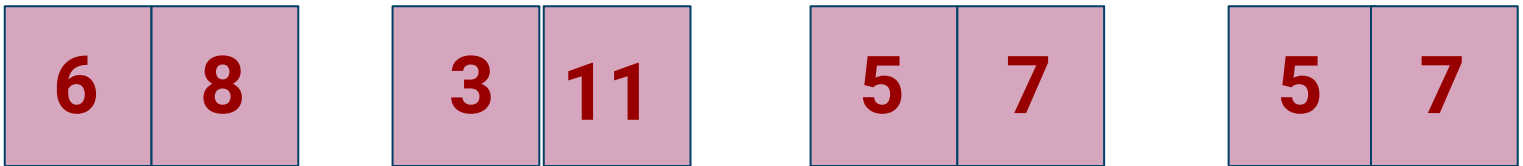
Merge Sort

- Divide the problem into subproblems



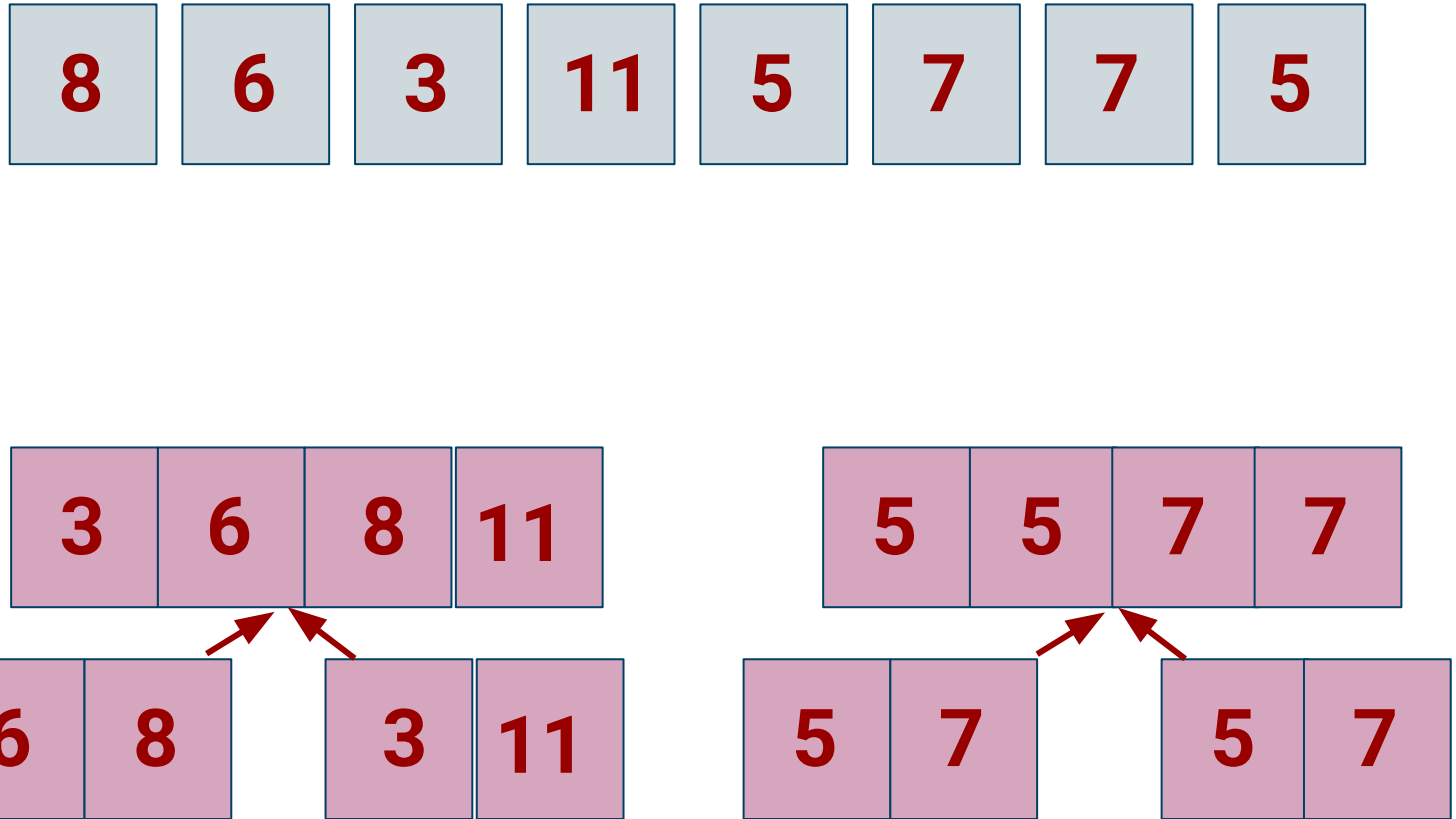
Merge Sort

- Divide the problem into subproblems



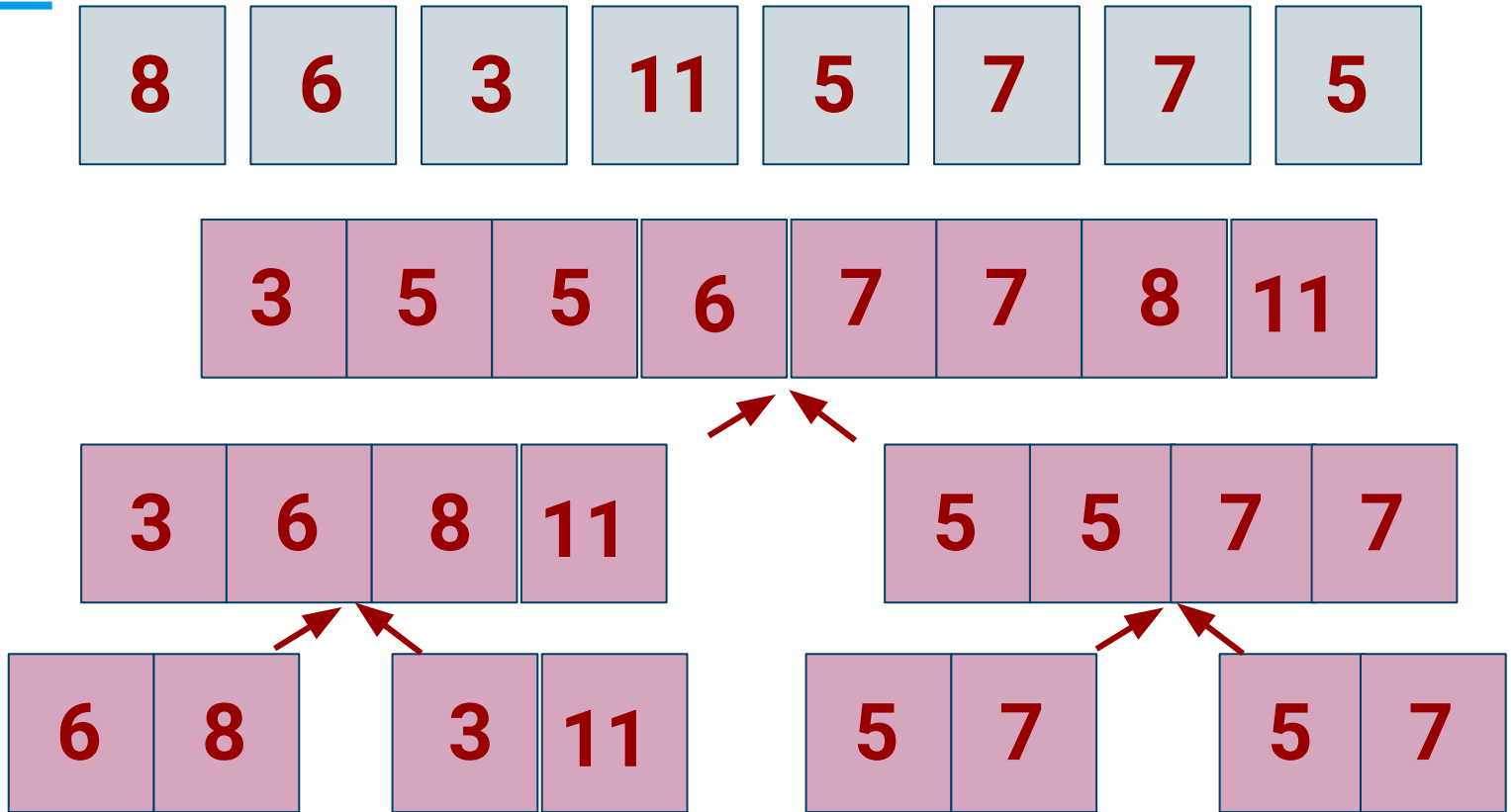
Merge Sort

- Divide the problem into subproblems



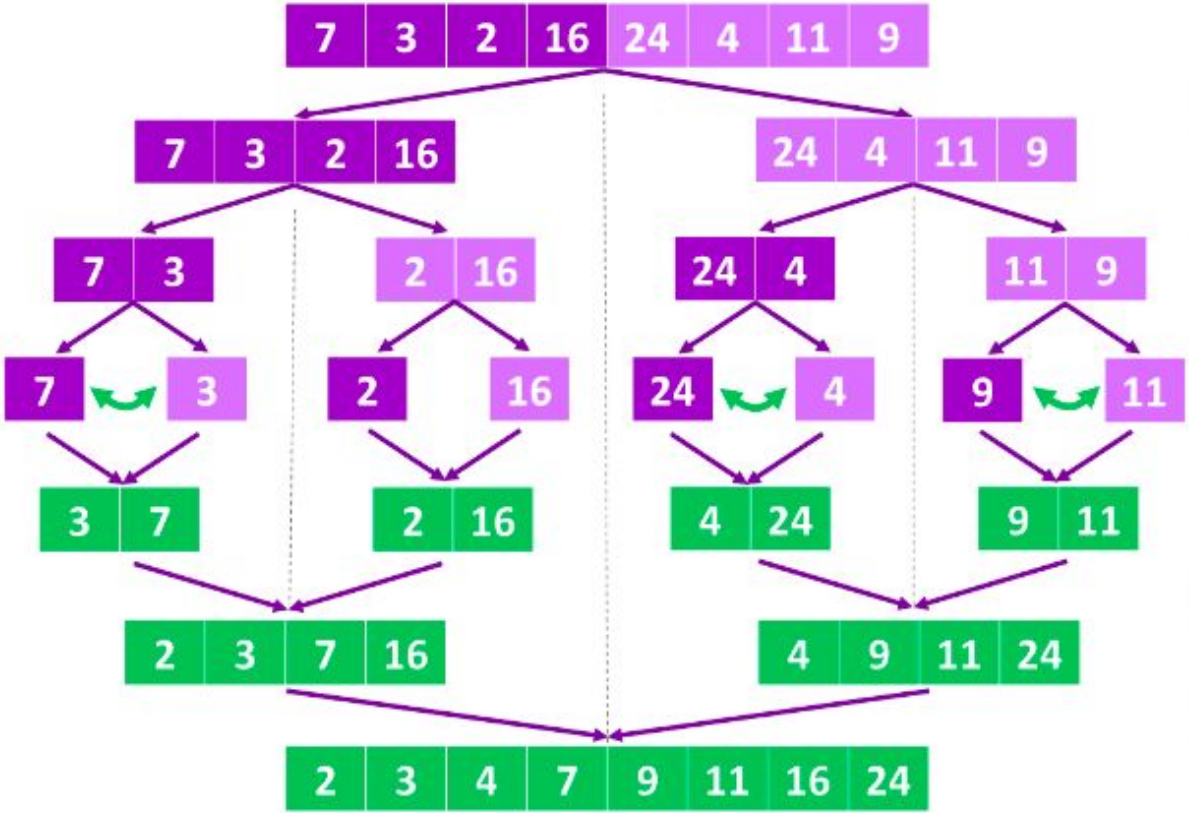
Merge Sort

- Divide the problem into subproblems



Merge Sort

O(n log(n))



Step 1:
Split sub-lists in two until you reach pair of values.

Step 3:
Sort/swap pair of values if needed.

Step 4:
Merge and sort sub-lists and repeat process till you merge to the full list.

https://www.101computing.net/merge-sort-algorithm/

Merge Sort Pseudo Code

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

References

https://www.includehelp.com/algorithms/divide-and-conquer-paradigm.aspx

https://dragonball.fandom.com/es/wiki/Raditz

https://www.101computing.net/merge-sort-algorithm/