NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 4, Lecture 3

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Merge Sort: Shortcomings

- Merging `A` and `B` creates a new array `C`
  - No obvious way to efficiently merge in place
- Extra storage can be costly
- Inherently recursive
  - Recursive call and return are expensive

Alternative approach

- Extra space is required to merge
- Merging happens because elements in left half must move right and vice versa
- Can we divide so that everything to the left is smaller than everything to the right?
  - No need to merge!

Divide and conquer without merging

- Suppose the median value in A is m
- Move all values ≤ m to left half of A
    - Right half has values > m
    - This shifting can be done in place, in time O(n)
- Recursively sort left and right halves
- A is now sorted! No need to merge
    - $T(n) = 2T(n/2) + n = O(n \log n)$

Divide and conquer without merging

- How do we find the median?
  - Sort and pick up middle element
  - But our aim is to sort!
- Instead, pick up some value in `A` — pivot
  - Split `A` with respect to this pivot element

Quicksort

- Choose a pivot element
  - Typically the first value in the array
- Partition `A` into lower and upper parts with respect to pivot
- Move pivot between lower and upper partition
- Recursively sort the two partitions

Quicksort

- High level view

43	32	22	78	63	57	91	13

Quicksort

- High level view

43	32	22	78	63	57	91	13

Quicksort

- High level view

43	32	22	78	63	57	91	13

Quicksort

- High level view

13	32	22	43	63	57	91	78

Quicksort

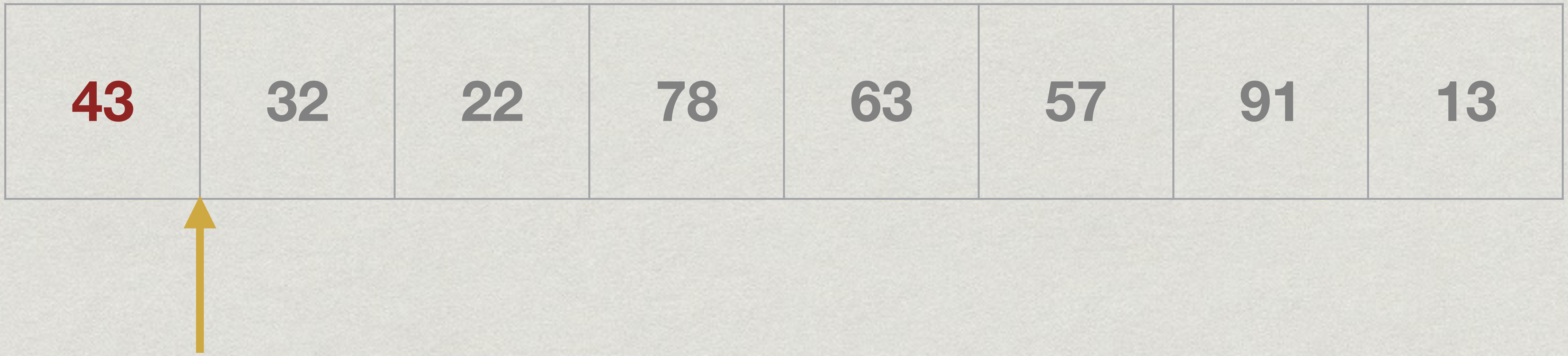
- High level view

13	22	32	43	57	63	78	91

Quicksort: Partitioning

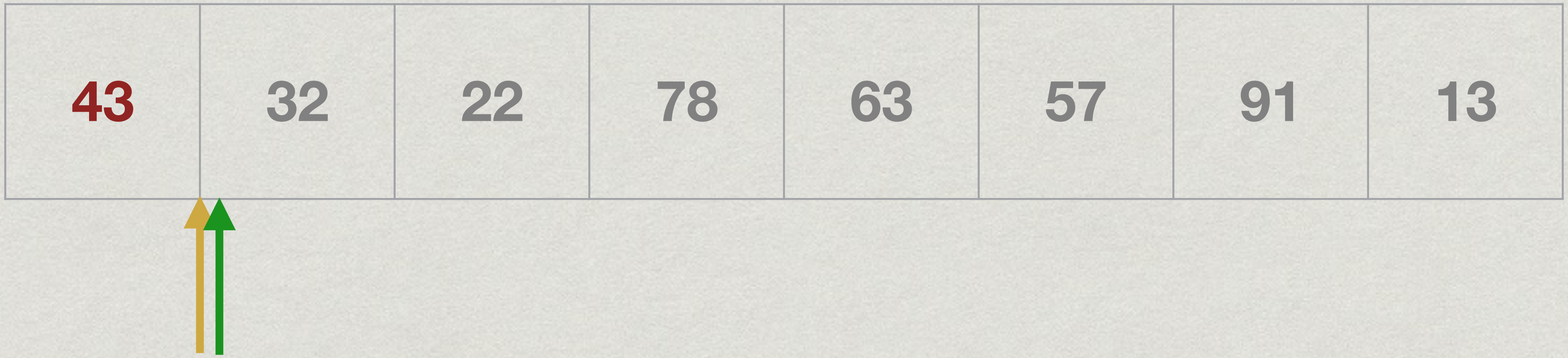
43	32	22	78	63	57	91	13

Quicksort: Partitioning



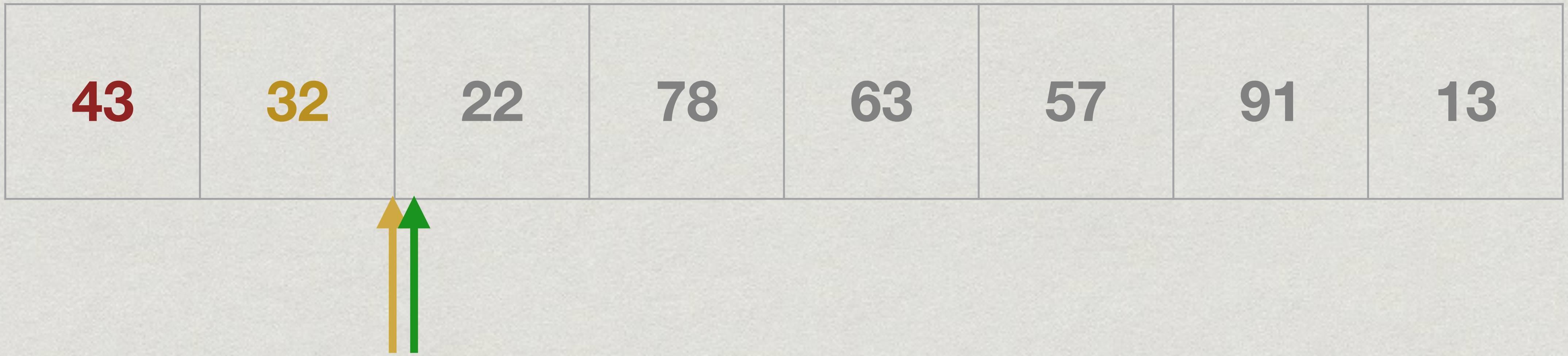
43	32	22	78	63	57	91	13

Quicksort: Partitioning



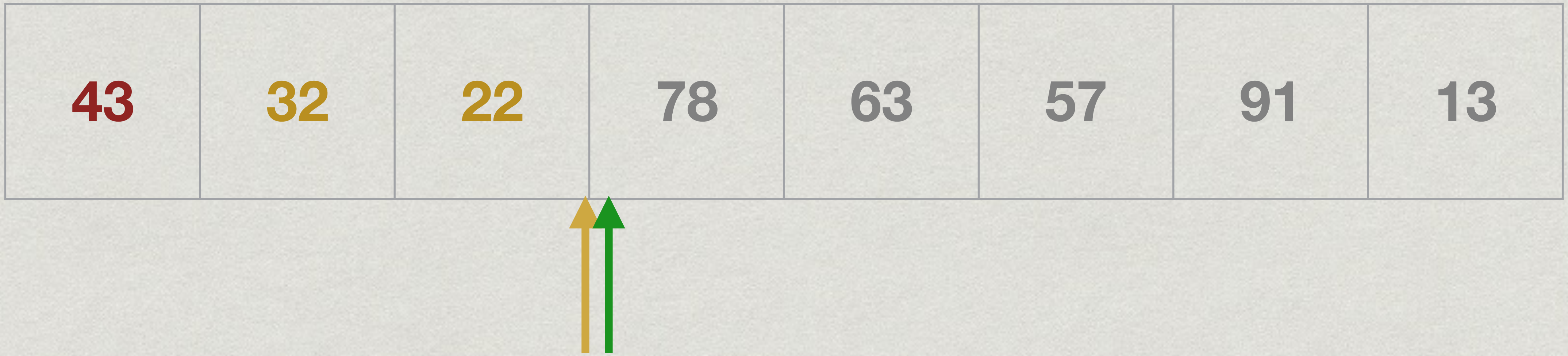
43	32	22	78	63	57	91	13

Quicksort: Partitioning



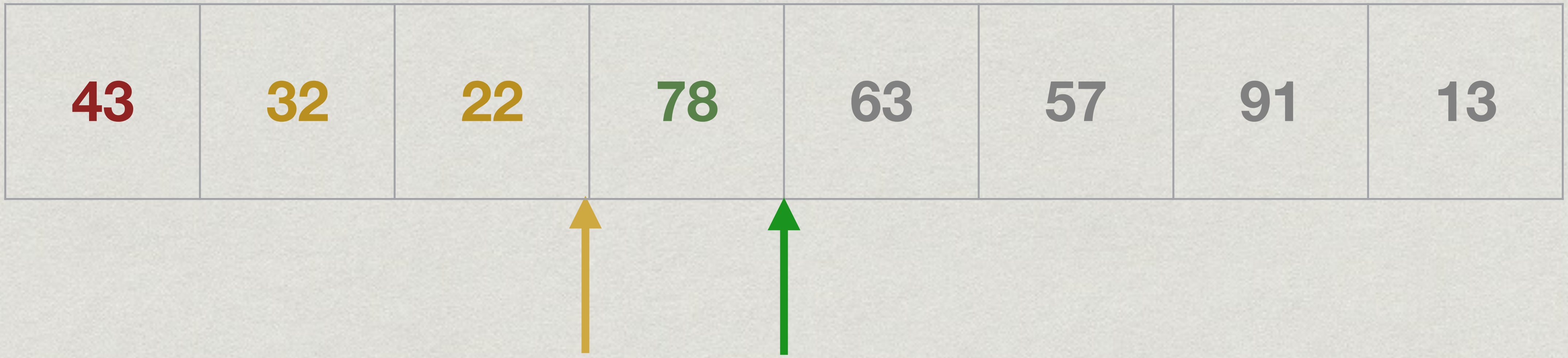
43	32	22	78	63	57	91	13

Quicksort: Partitioning



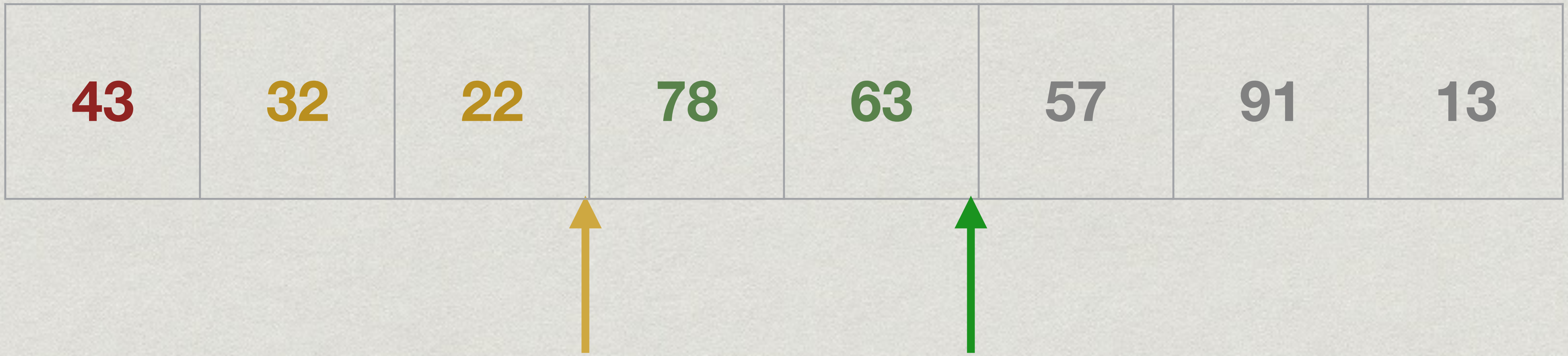
43	32	22	78	63	57	91	13

Quicksort: Partitioning



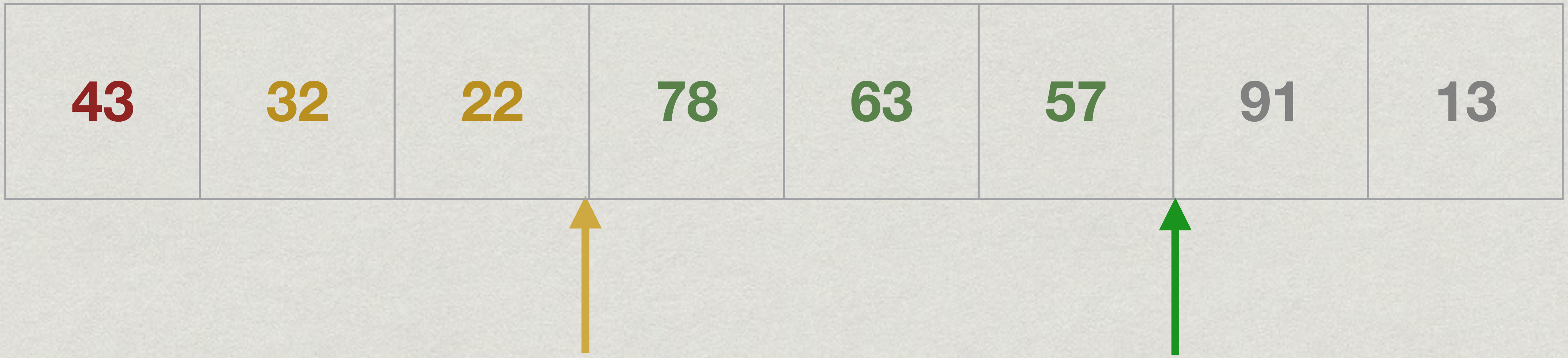
43	32	22	78	63	57	91	13

Quicksort: Partitioning



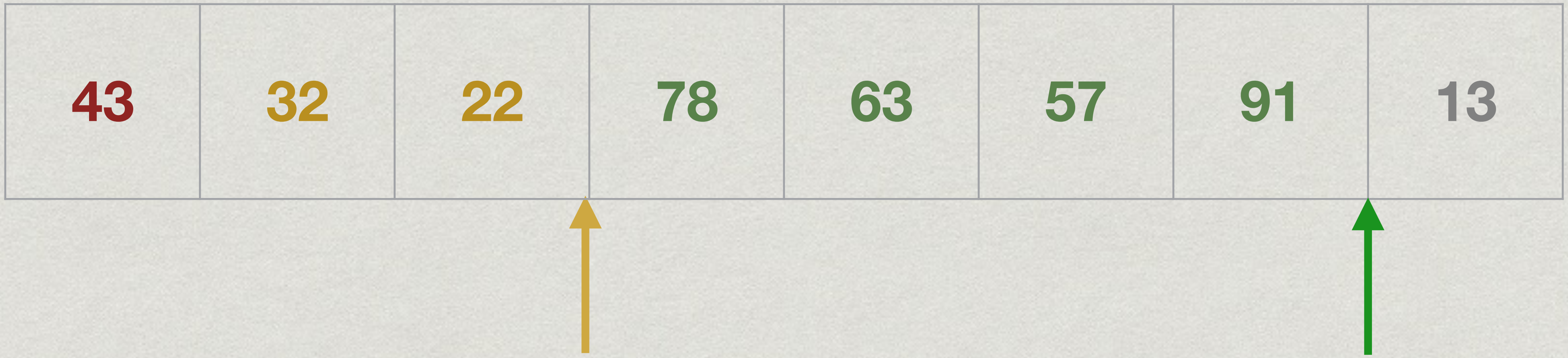
43	32	22	78	63	57	91	13

Quicksort: Partitioning



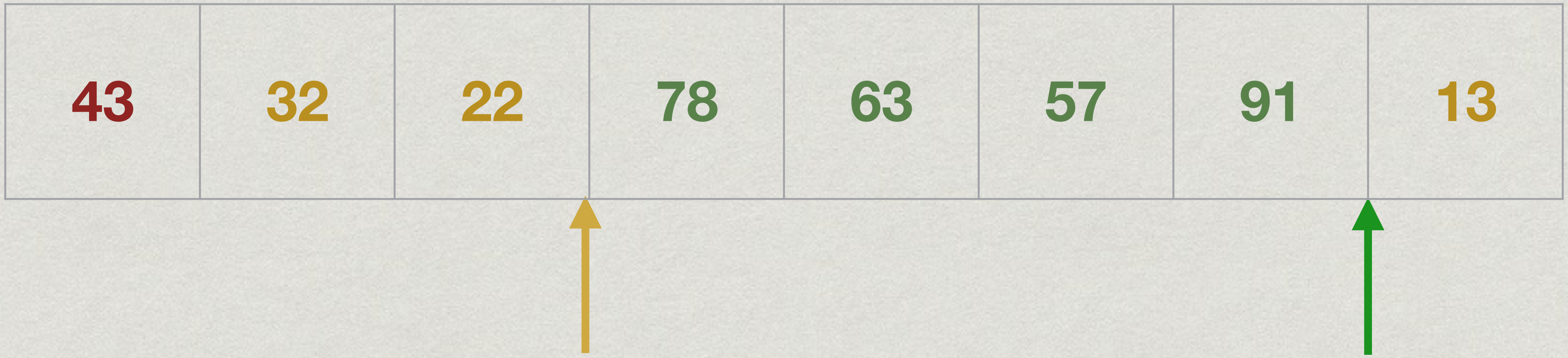
43	32	22	78	63	57	91	13

Quicksort: Partitioning



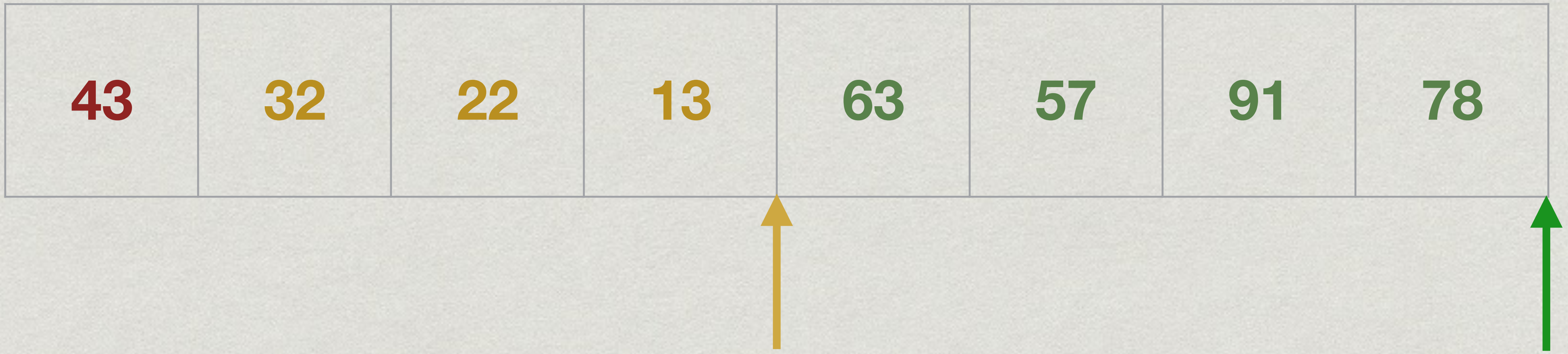
43	32	22	78	63	57	91	13

Quicksort: Partitioning



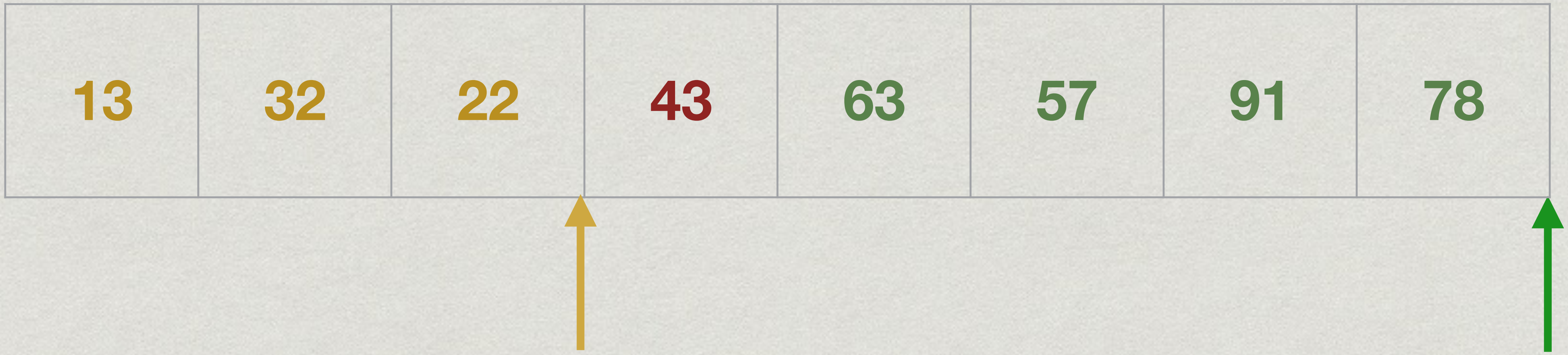
43	32	22	78	63	57	91	13

Quicksort: Partitioning



43	32	22	13	63	57	91	78

Quicksort: Partitioning



13	32	22	43	63	57	91	78

Quicksort in Python

```
def Quicksort(A,l,r): # Sort A[l:r]
  if r - l <= 1:   # Base case
    return ()

  # Partition with respect to pivot, a[l]
  yellow = l+1

  for green in range(l+1,r):
    if A[green] <= A[l]:
      (A[yellow],A[green]) = (A[green],A[yellow])
      yellow = yellow + 1

  # Move pivot into place
  (A[l],A[yellow-1]) = (A[yellow-1],A[l])

  Quicksort(A,l,yellow-1)  # Recursive calls
  Quicksort(A,yellow,r)
```