Quicksort

Julián García

Sorting

Input:

A sequence of *n* numbers

$$\langle a_1, a_2, \ldots, a_n \rangle$$

Output:

A permutation of the input sequence

$$\langle a'_1, a'_2, \ldots, a'_n \rangle$$

do it efficiently.

such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Quicksort

Quicksort

- Divide and conquer algorithm.

Quicksort

- Divide and conquer algorithm.
- Recursive

Quicksort

- Divide and conquer algorithm.
- Recursive
- Sorts in place (no extra arrays!).

Quicksort

- Divide and conquer algorithm.
- Recursive
- Sorts in place (no extra arrays!).
- Popular: Top-10 algorithms 20th century (SIAM).

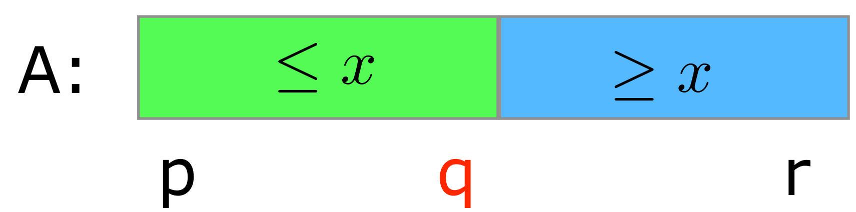
Quicksort strategy

<u>Input:</u>

Quicksort strategy

<u>Input:</u>

<u>Divide:</u>

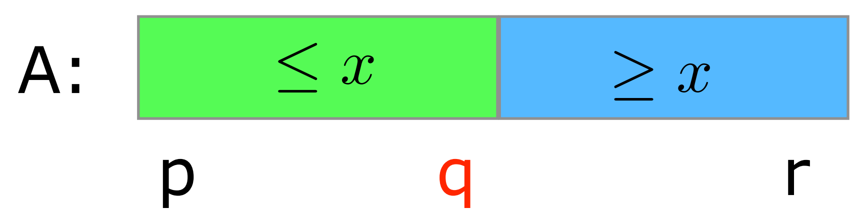


(partition step)

Quicksort strategy

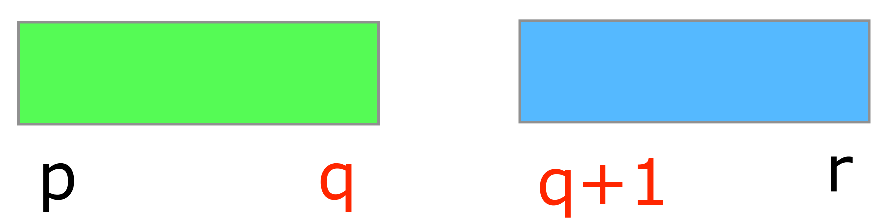
Input:

Divide:



(partition step)

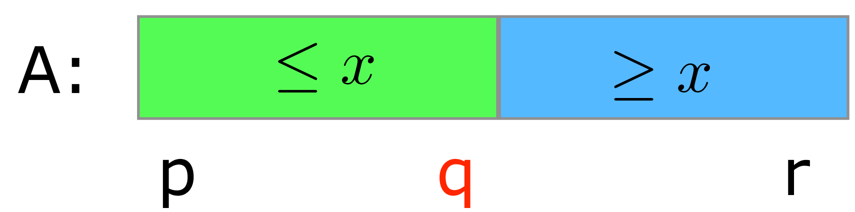
Conquer: Recursively solve two smaller problems.



Quicksort strategy

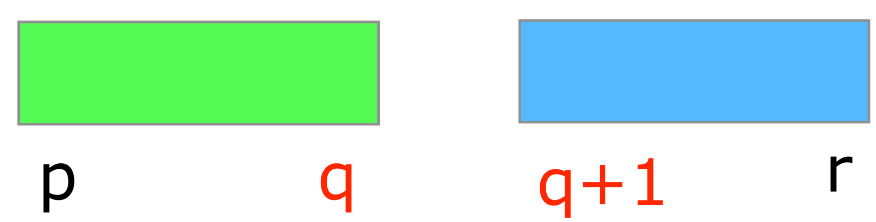
<u>Input:</u>

<u>Divide:</u>



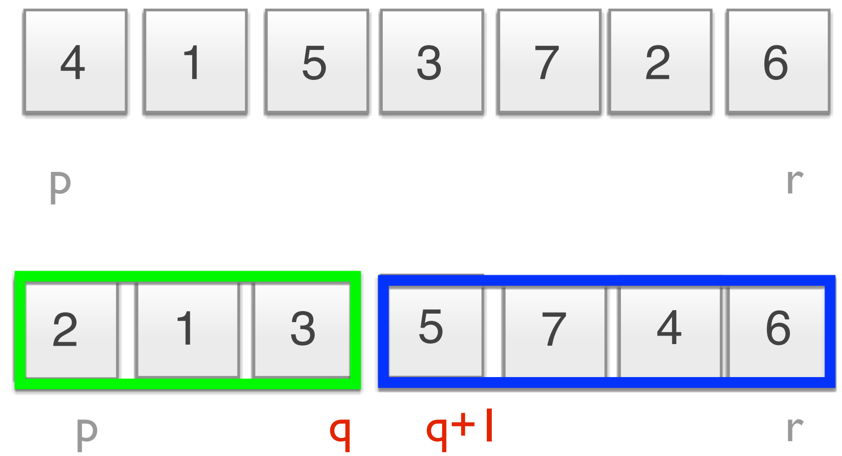
(partition step)

<u>Conquer:</u> Recursively solve two smaller problems.

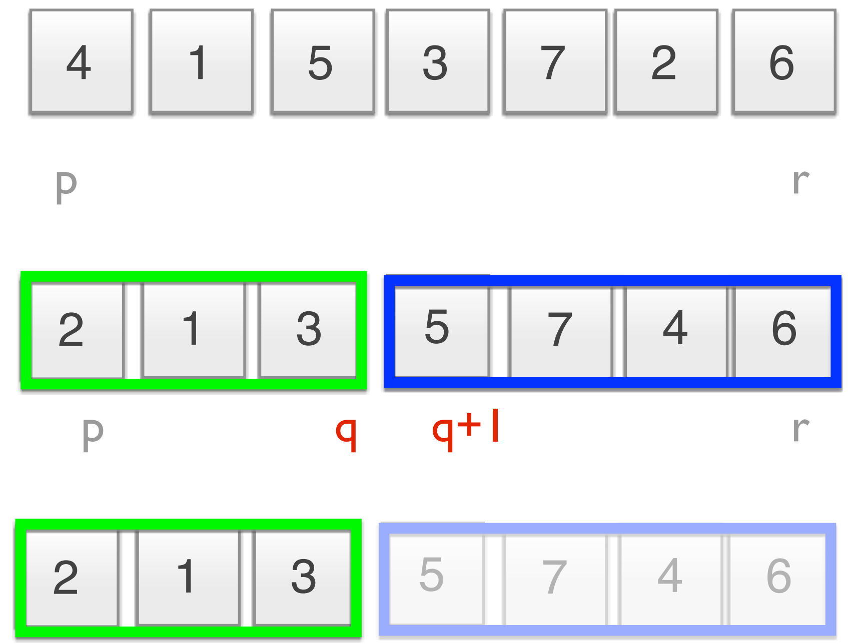


<u>Combine:</u> In-place! so we are done.

4	1	5	3	7	2	6
p						r



(partition step)

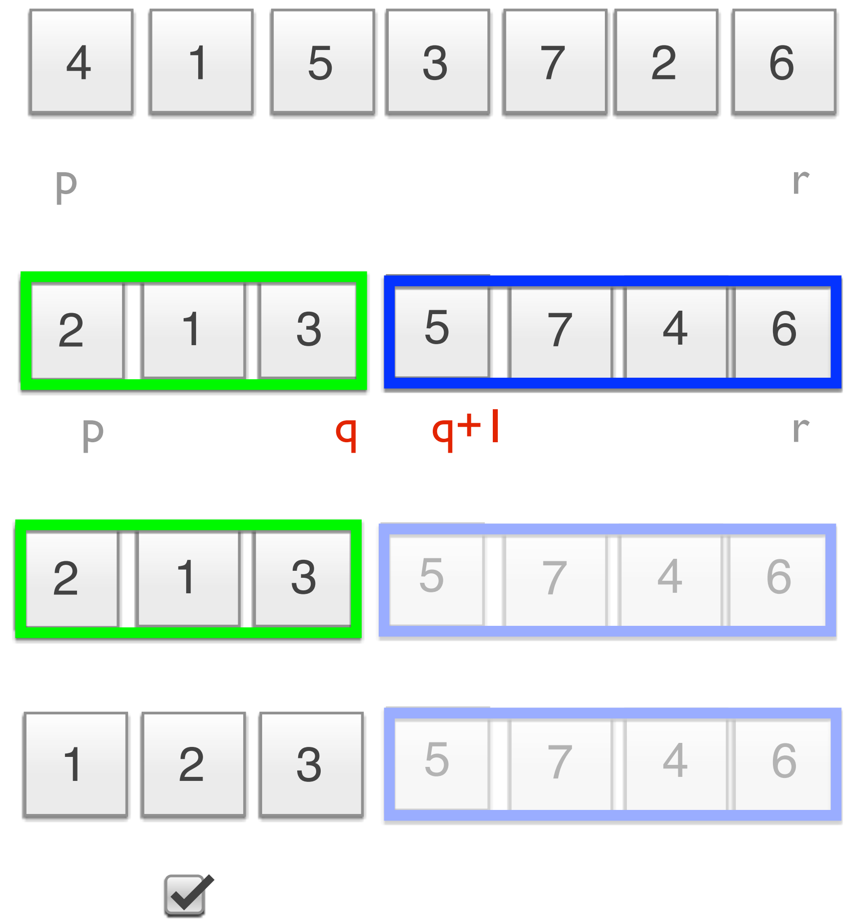


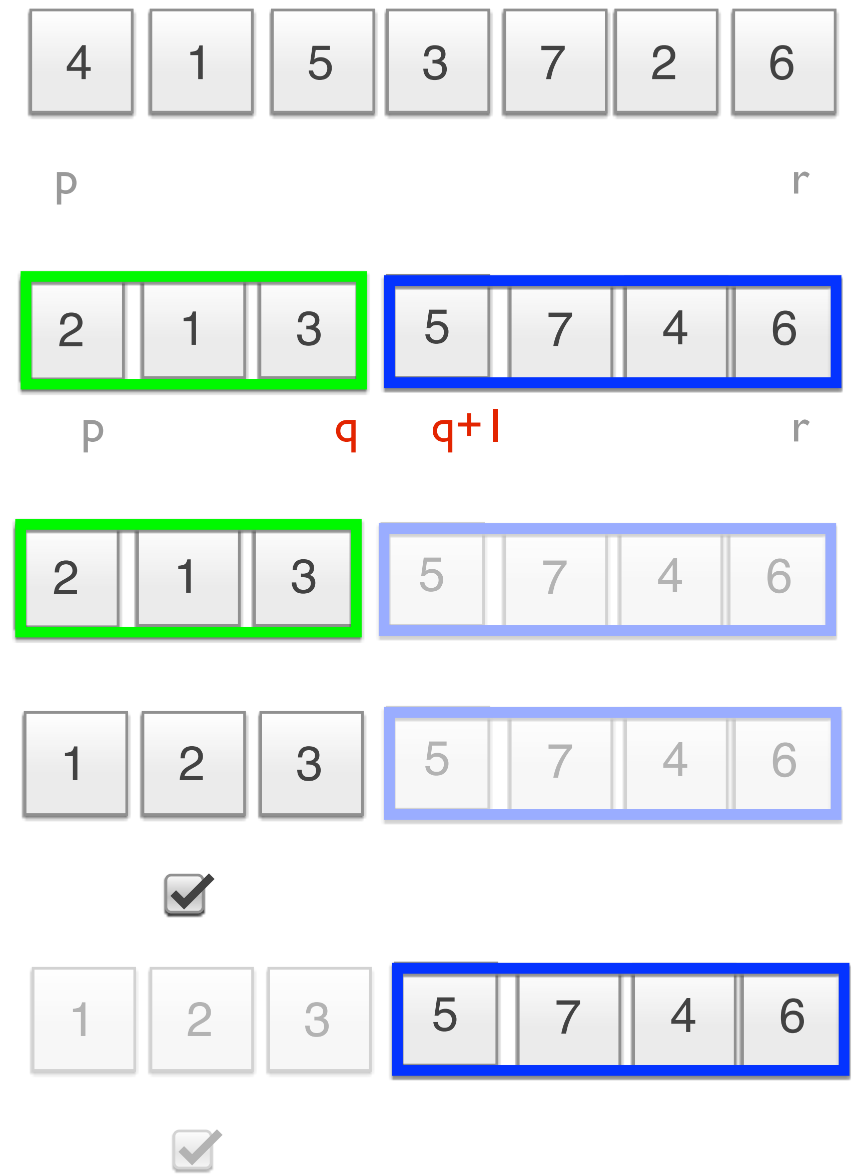
(partition step)

sort left

(partition step)

sort left

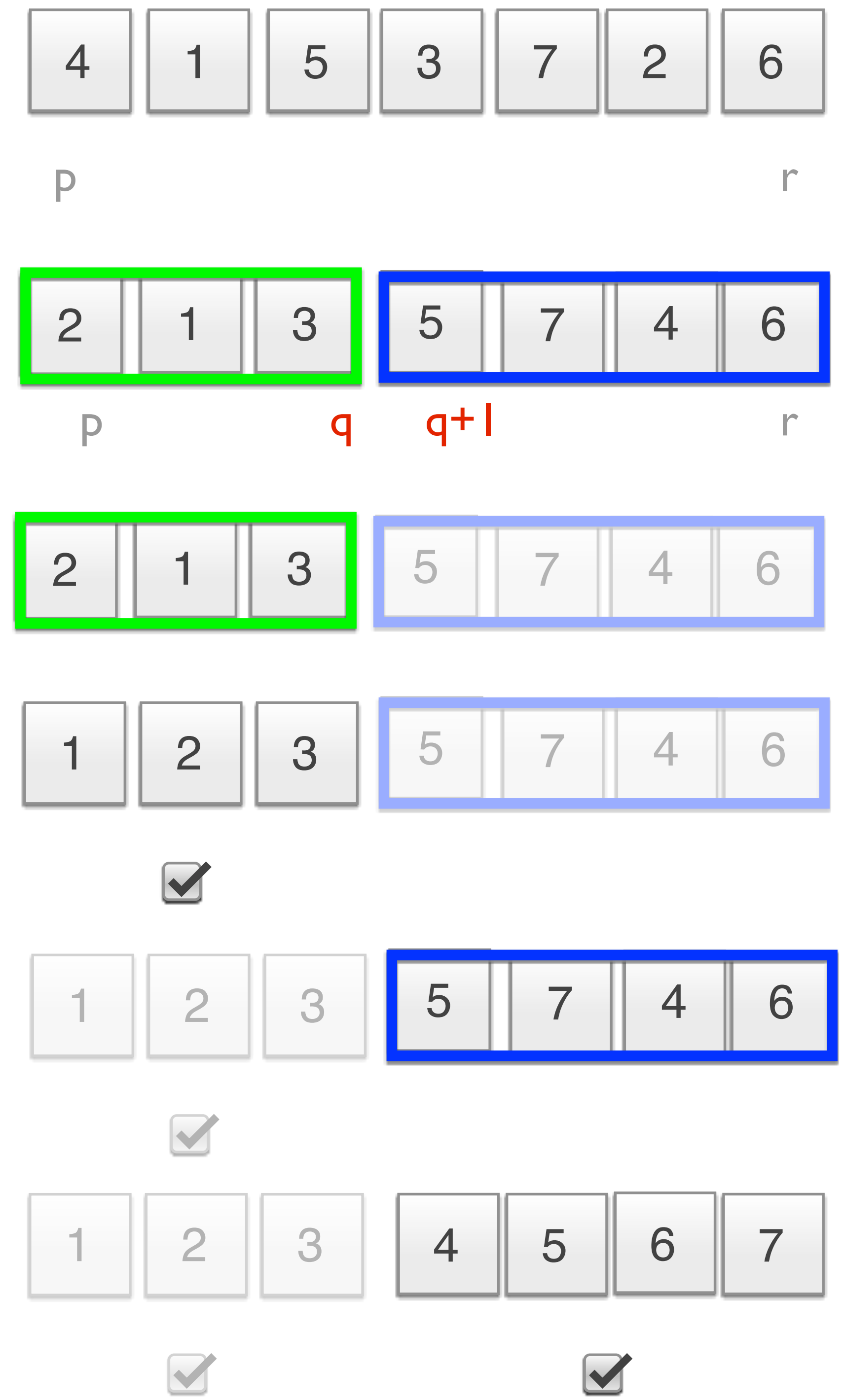




(partition step)

sort left

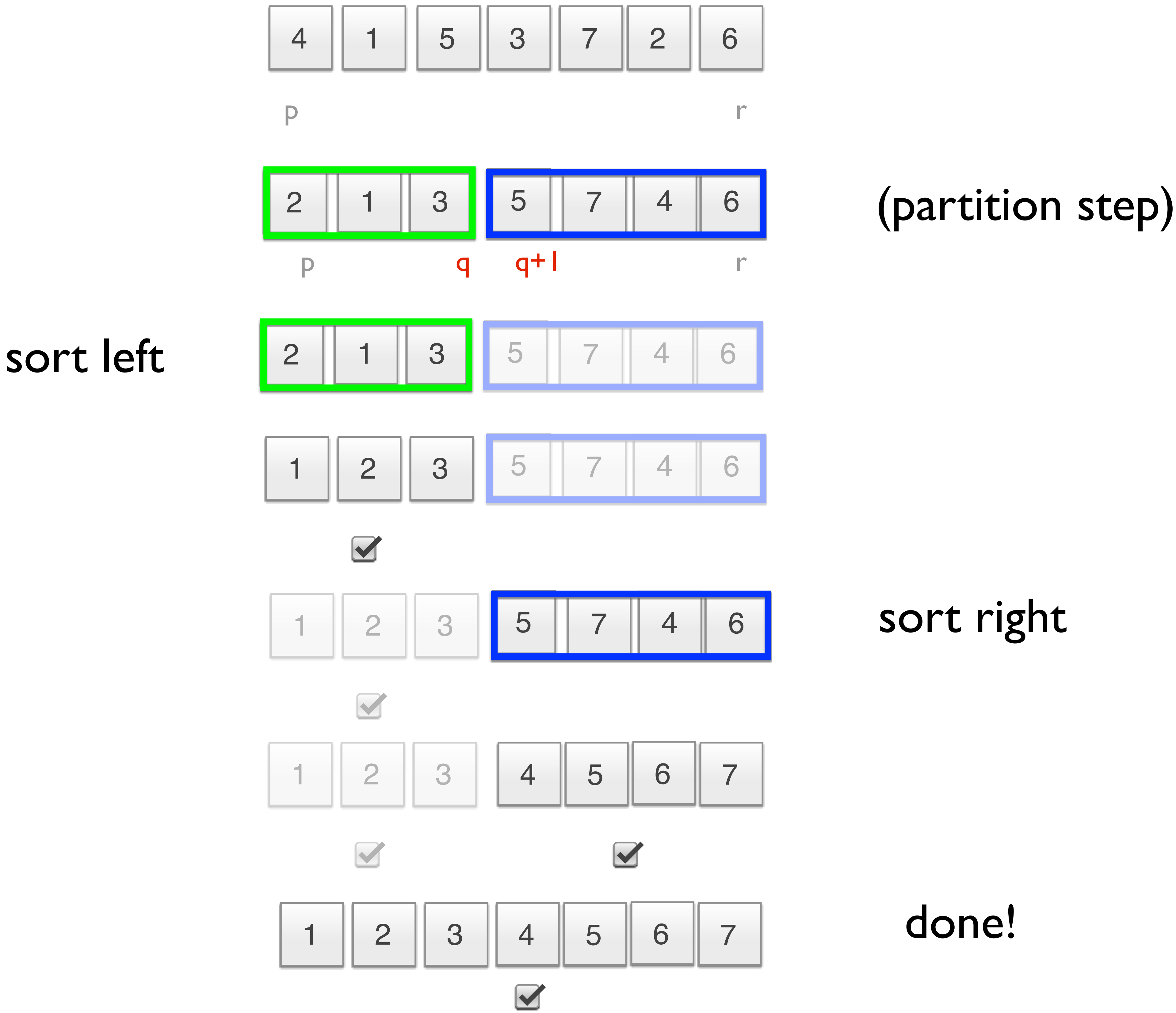
sort right



(partition step)

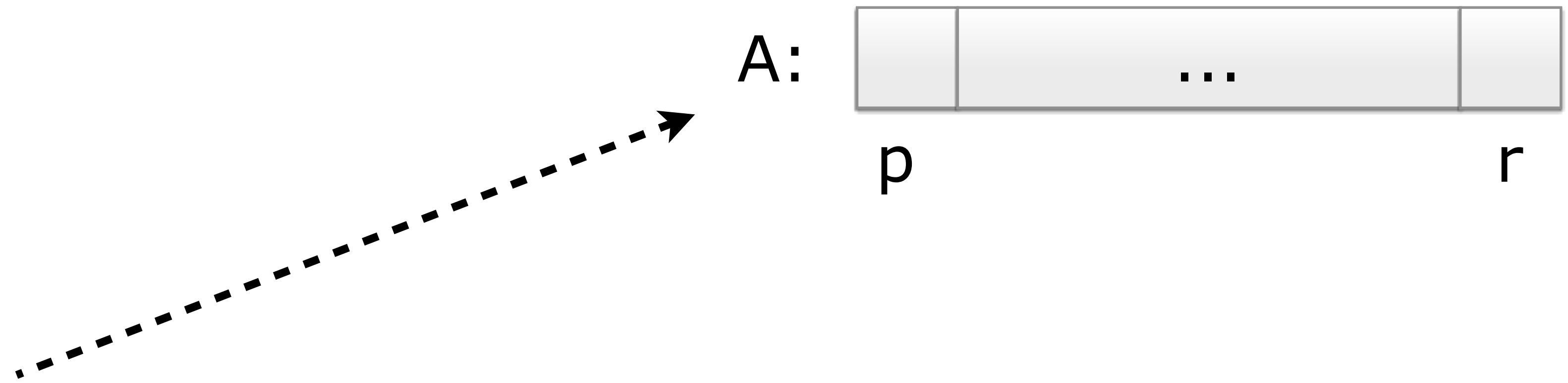
sort left

sort right



4 1 5 3 7 2 6
p
r
2 1 3 5 7 4 6
(partition step)
p
q
q+1
r
sort left
2 1 3 5 7 4 6
1 2 3 5 7 4 6
1 2 3 5 7 4 6
sort right
1 2 3 4 5 6 7
1 2 3 4 5 6 7
done!

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q)
    QUICKSORT(A, q + 1, r)
```



$\textsc{Quicksort}(A, p, r)$

```
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q)
    QUICKSORT(A, q + 1, r)
```

Is it non-trivial?
(exit door)

```
QUICKSORT(A, p, r)
    if p < r
        q = PARTITION(A, p, r)
        QUICKSORT(A, p, q)
        QUICKSORT(A, q + 1, r)
```

Is it non-trivial?
(exit door)

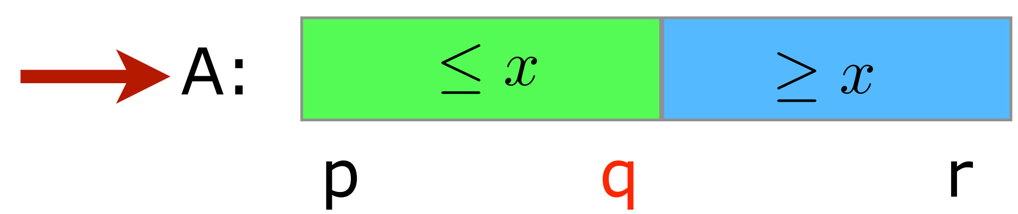
$\text{Quicksort}(A, p, r)$

$\quad \textbf{if } p < r$

$\qquad q = \text{Partition}(A, p, r)$

$\qquad \text{Quicksort}(A, p, q)$

$\qquad \text{Quicksort}(A, q + 1, r)$



Is it non-trivial?
(exit door)

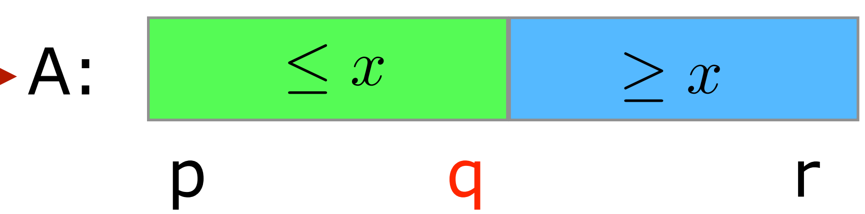
QUICKSORT(A, p, r)

if $p < r$

$q =$ PARTITION(A, p, r)

QUICKSORT(A, p, q)

QUICKSORT$(A, q+1, r)$



Is it non-trivial?
(exit door)

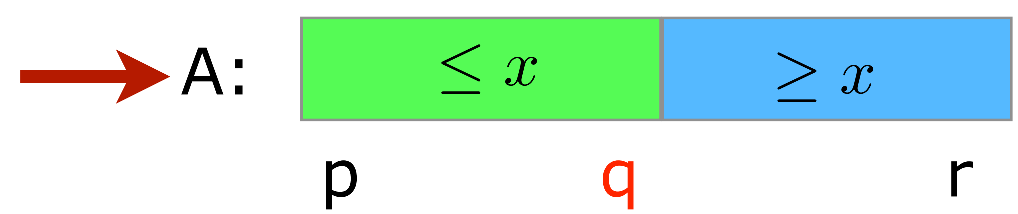
$\textsc{Quicksort}(A, p, r)$

if $p < r$

$\quad q = \textsc{Partition}(A, p, r)$

$\quad \textsc{Quicksort}(A, p, q)$

$\quad \textsc{Quicksort}(A, q+1, r)$



Is it non-trivial?
(exit door)

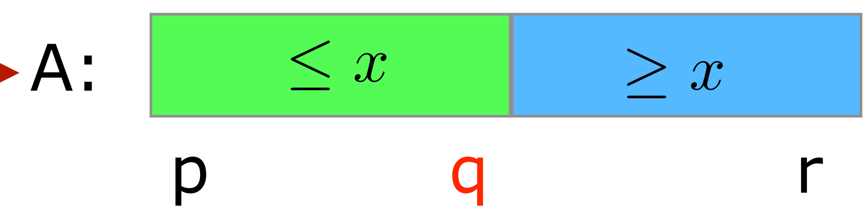
$\textsc{Quicksort}(A, p, r)$

$\quad \textbf{if } p < r$

$\quad\quad q = \textsc{Partition}(A, p, r)$

$\quad\quad \textsc{Quicksort}(A, p, q)$

$\quad\quad \textsc{Quicksort}(A, q+1, r)$

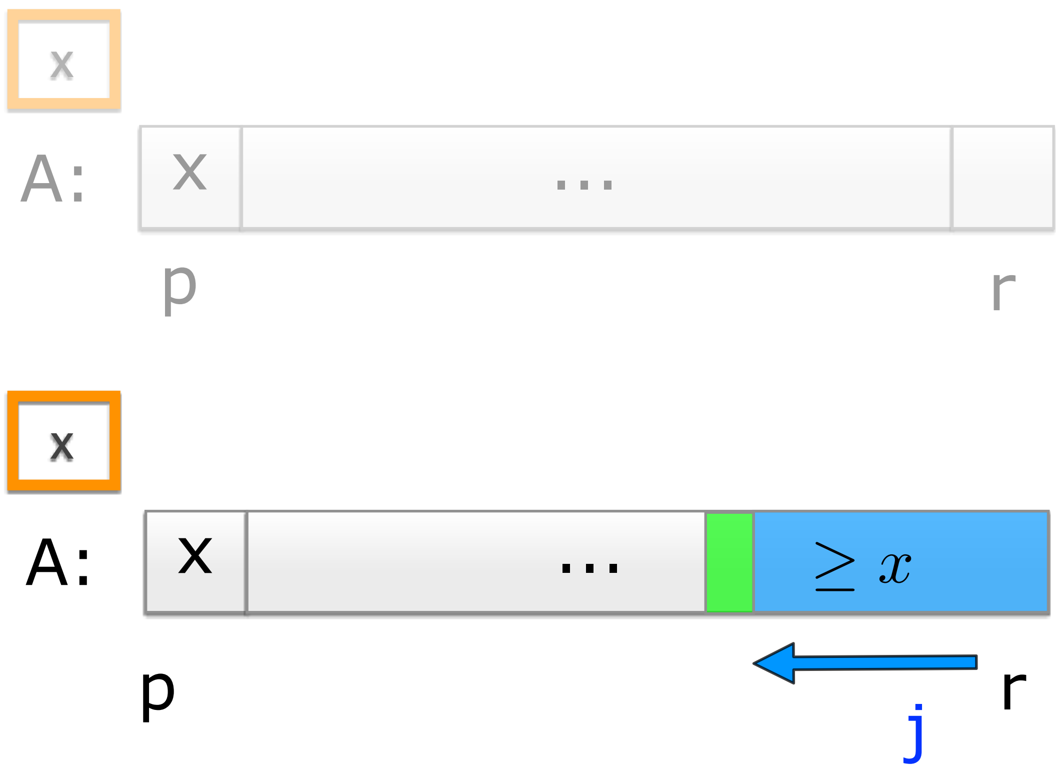


partition really does all the work!

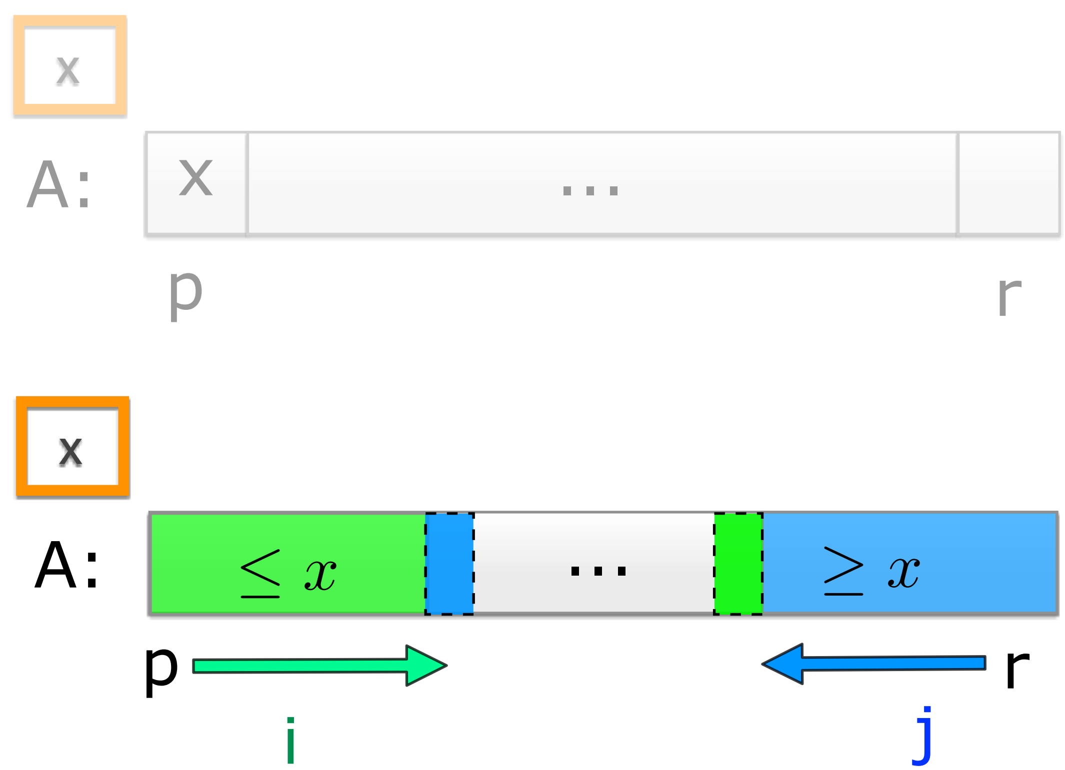
$\textsc{Partition}(A, p, r)$

$\textsc{Partition}(A, p, r)$

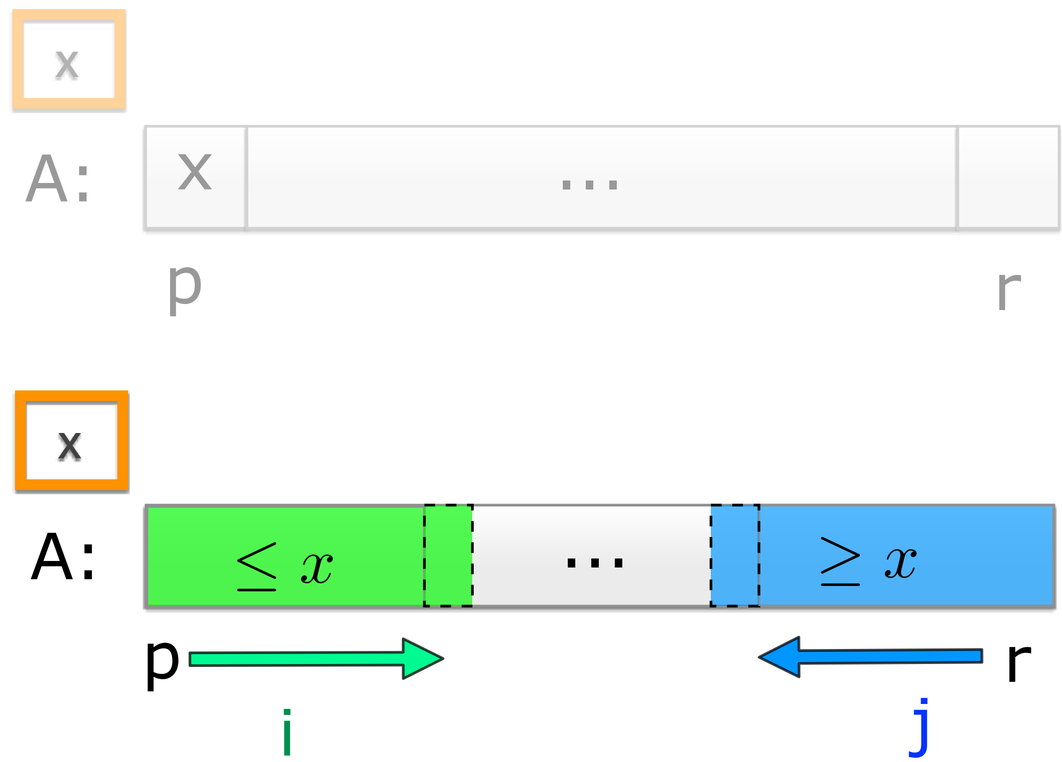
$\textsc{Partition}(A, p, r)$



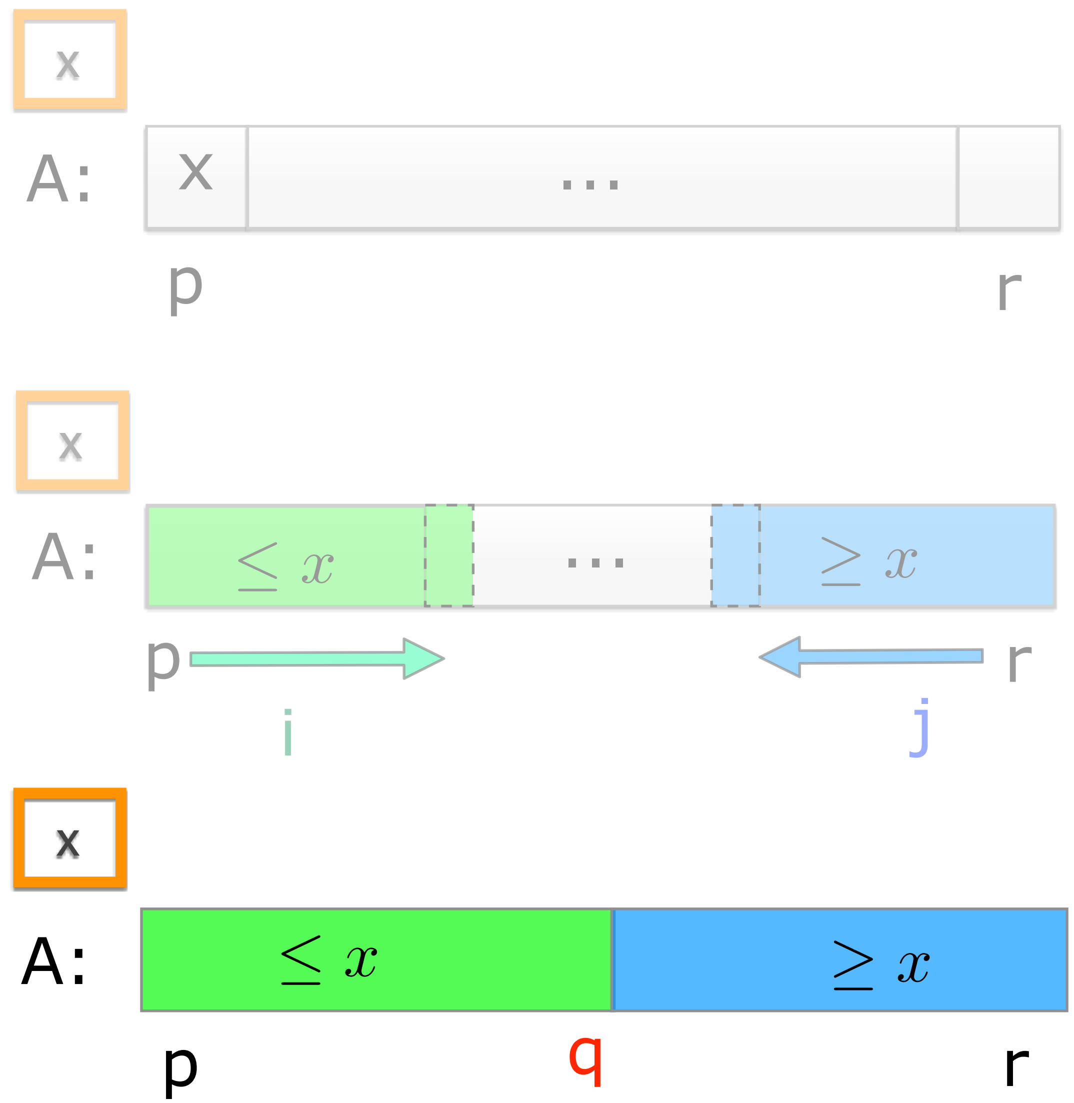
$\textsc{Partition}(A, p, r)$



$\textsc{Partition}(A, p, r)$

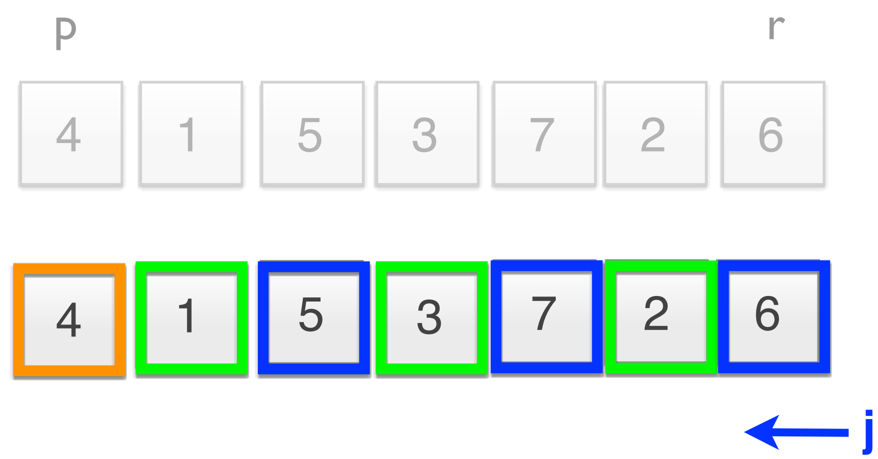


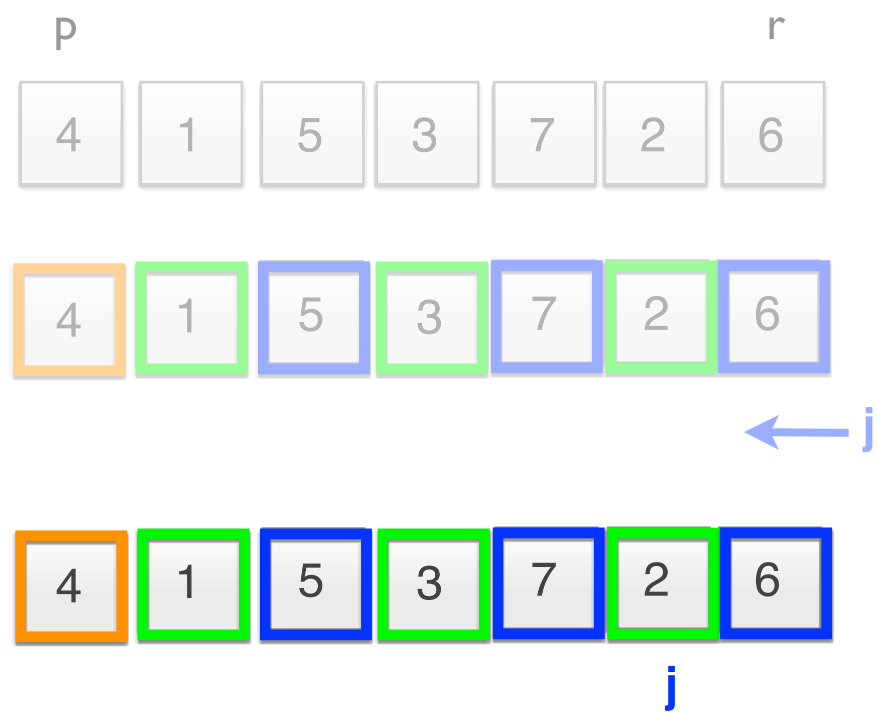
$\text{PARTITION}(A, p, r)$



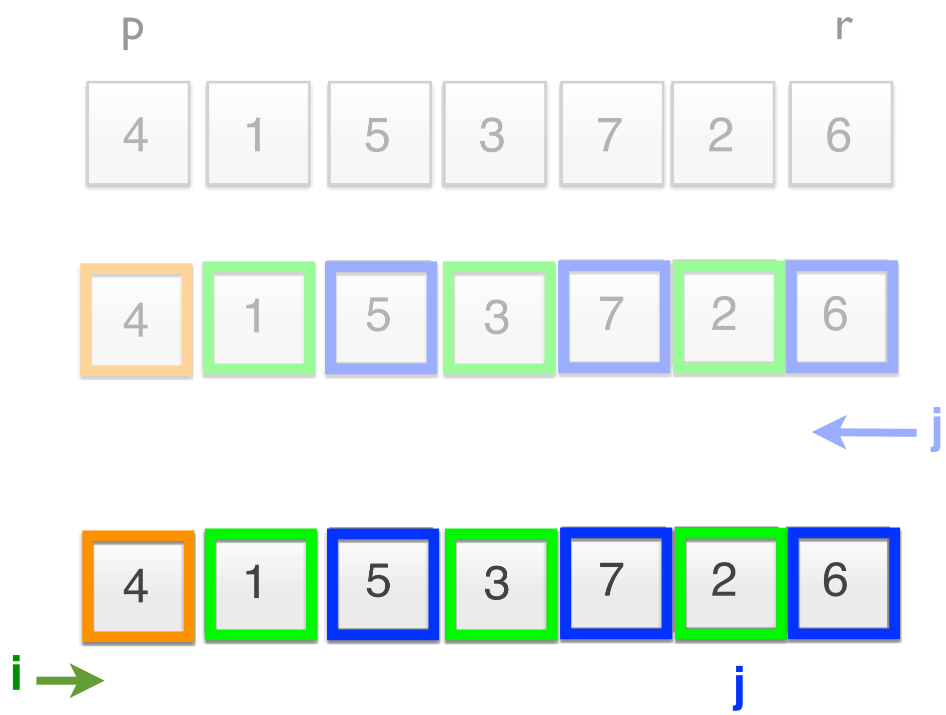
p						r
4	1	5	3	7	2	6

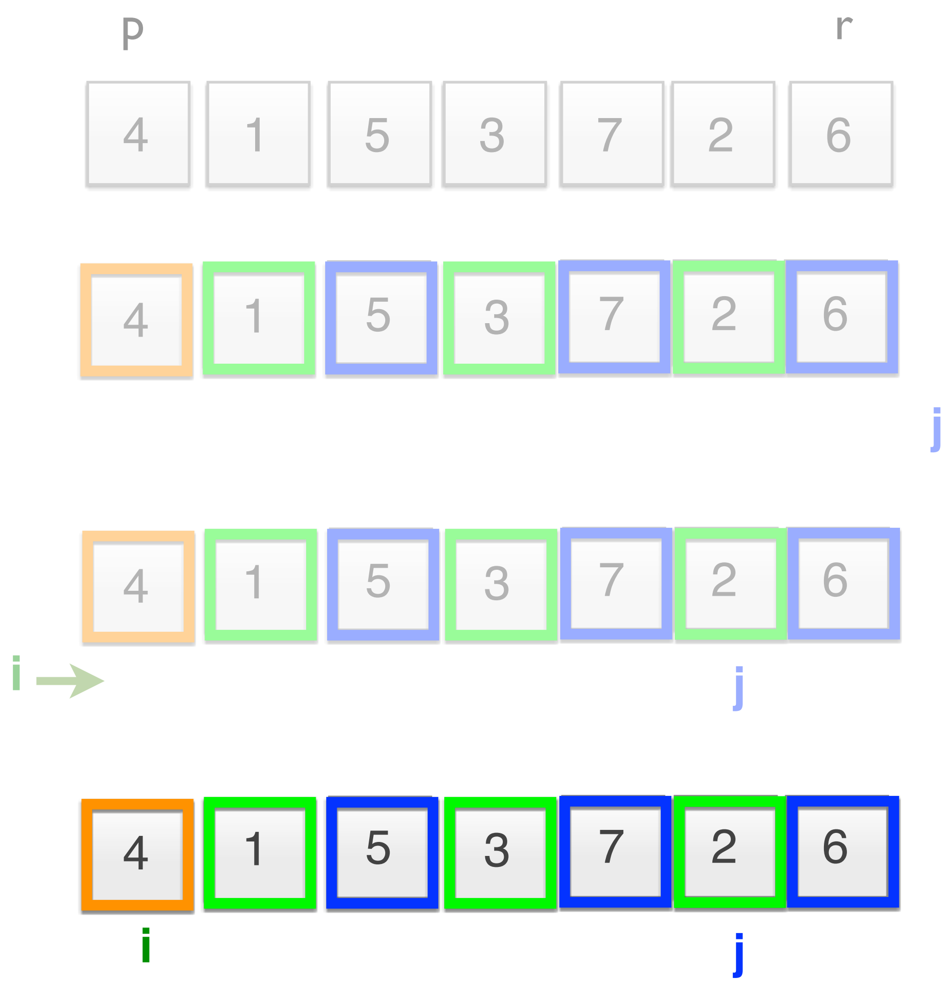

p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6

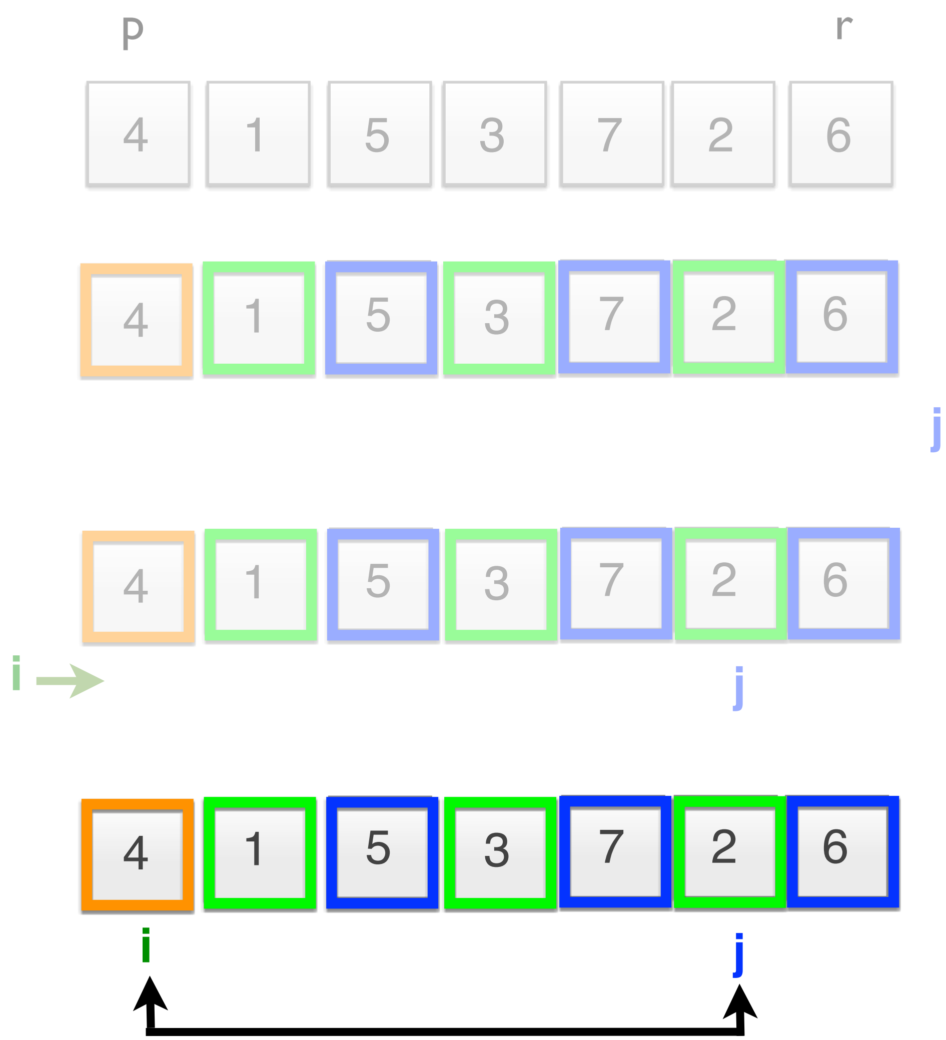

p
r
4
1
5
3
7
2
6
4
1
5
3
7
2
6
j

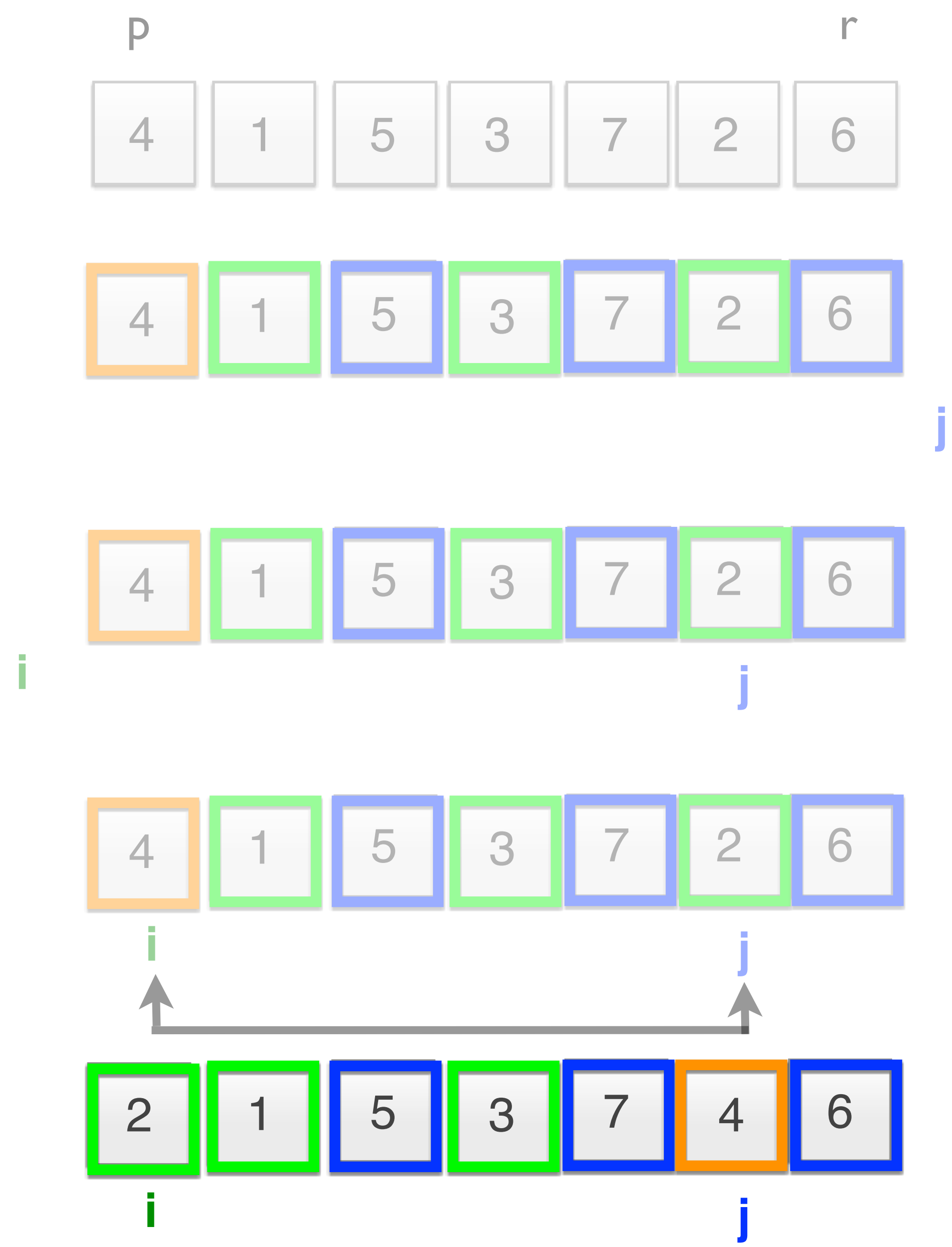


p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
j

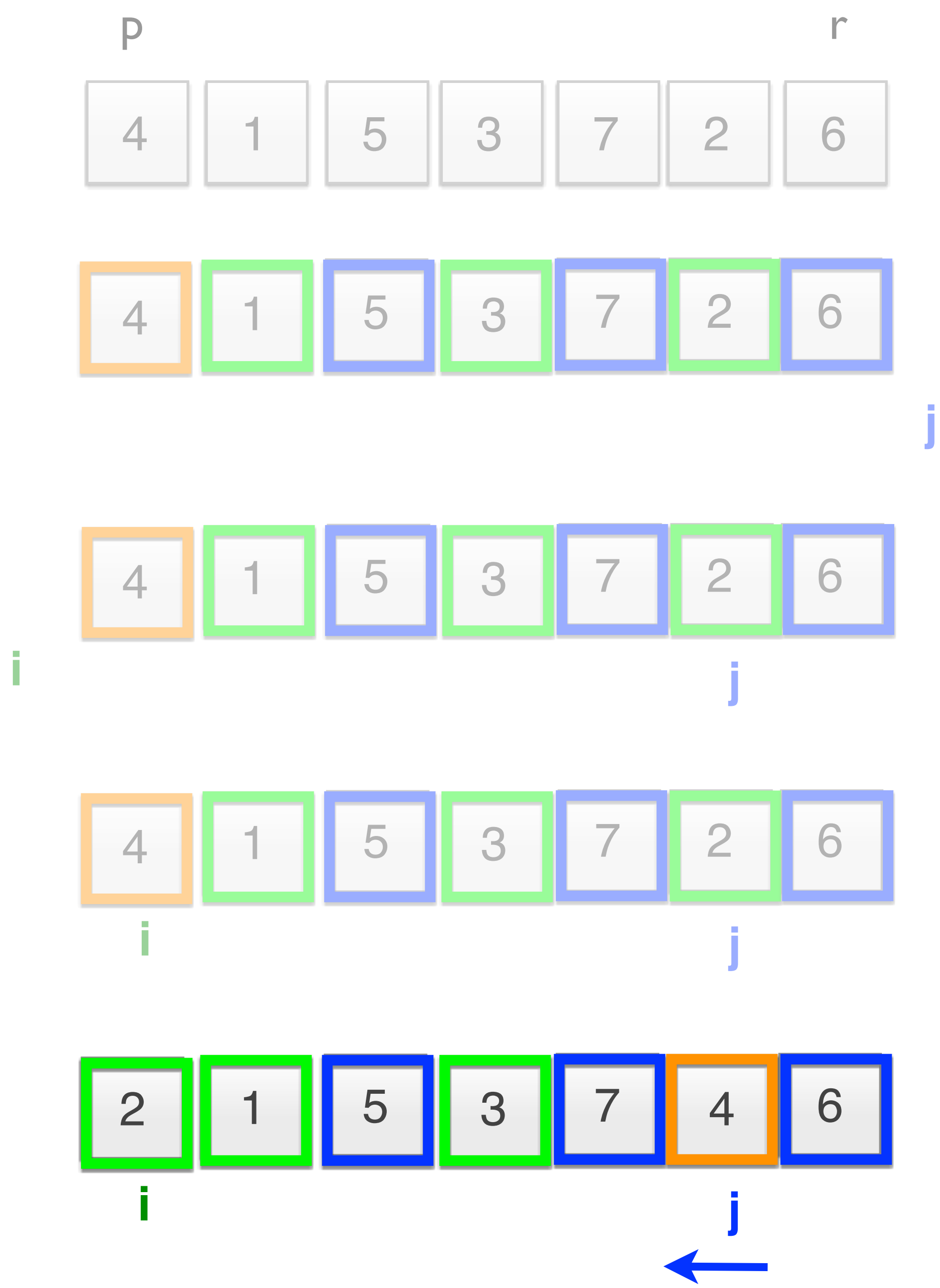

p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j

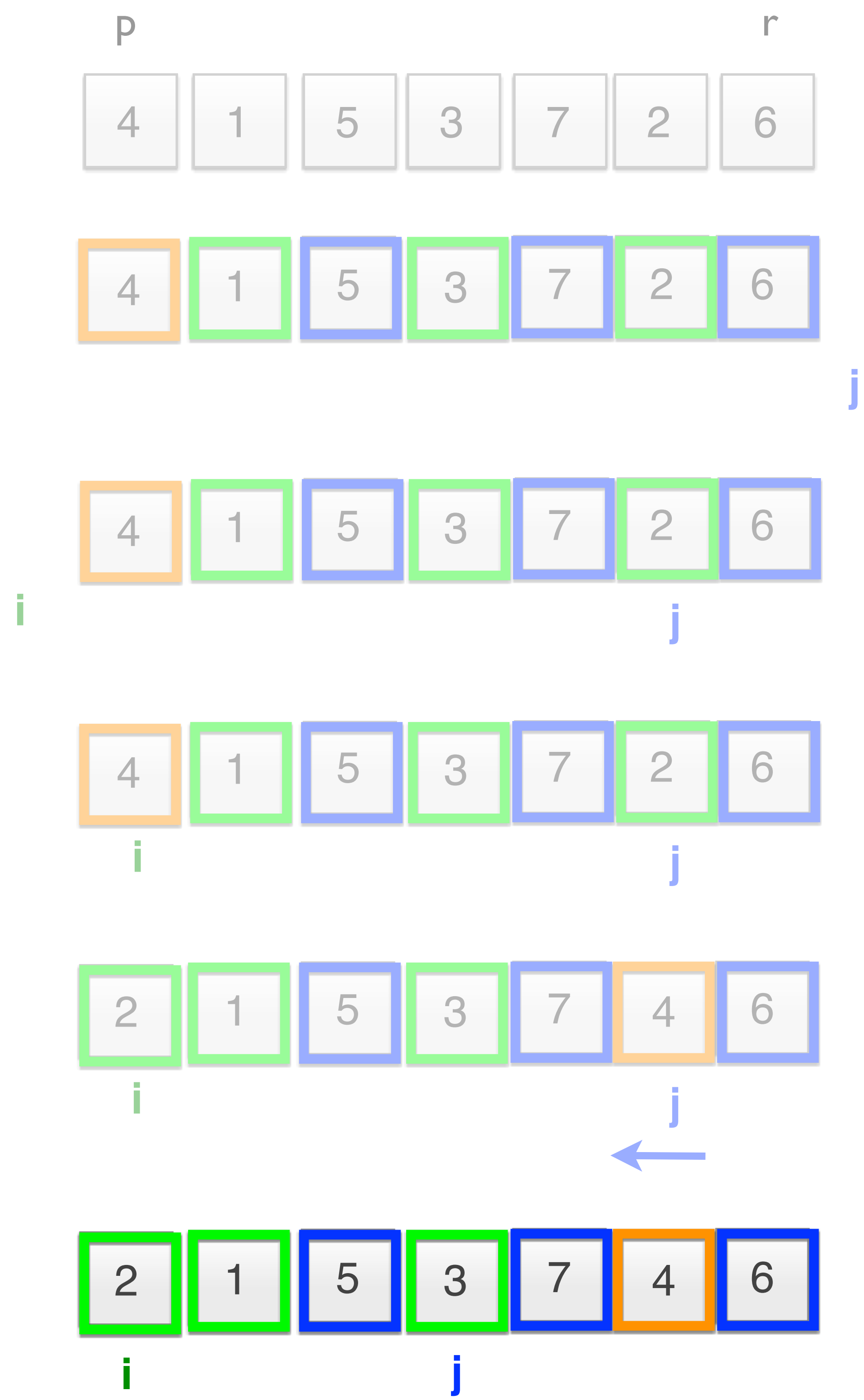

p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j


p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j

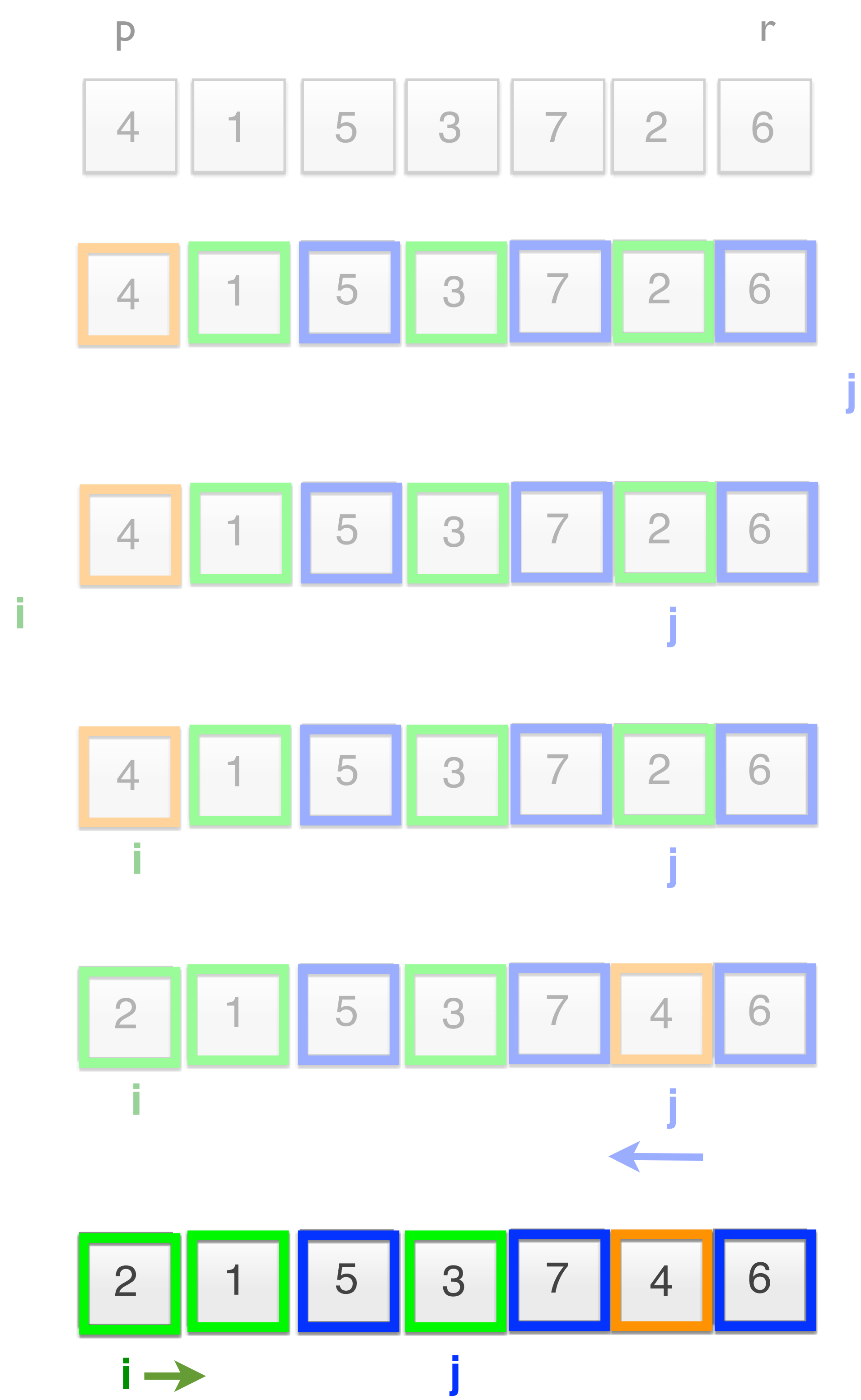


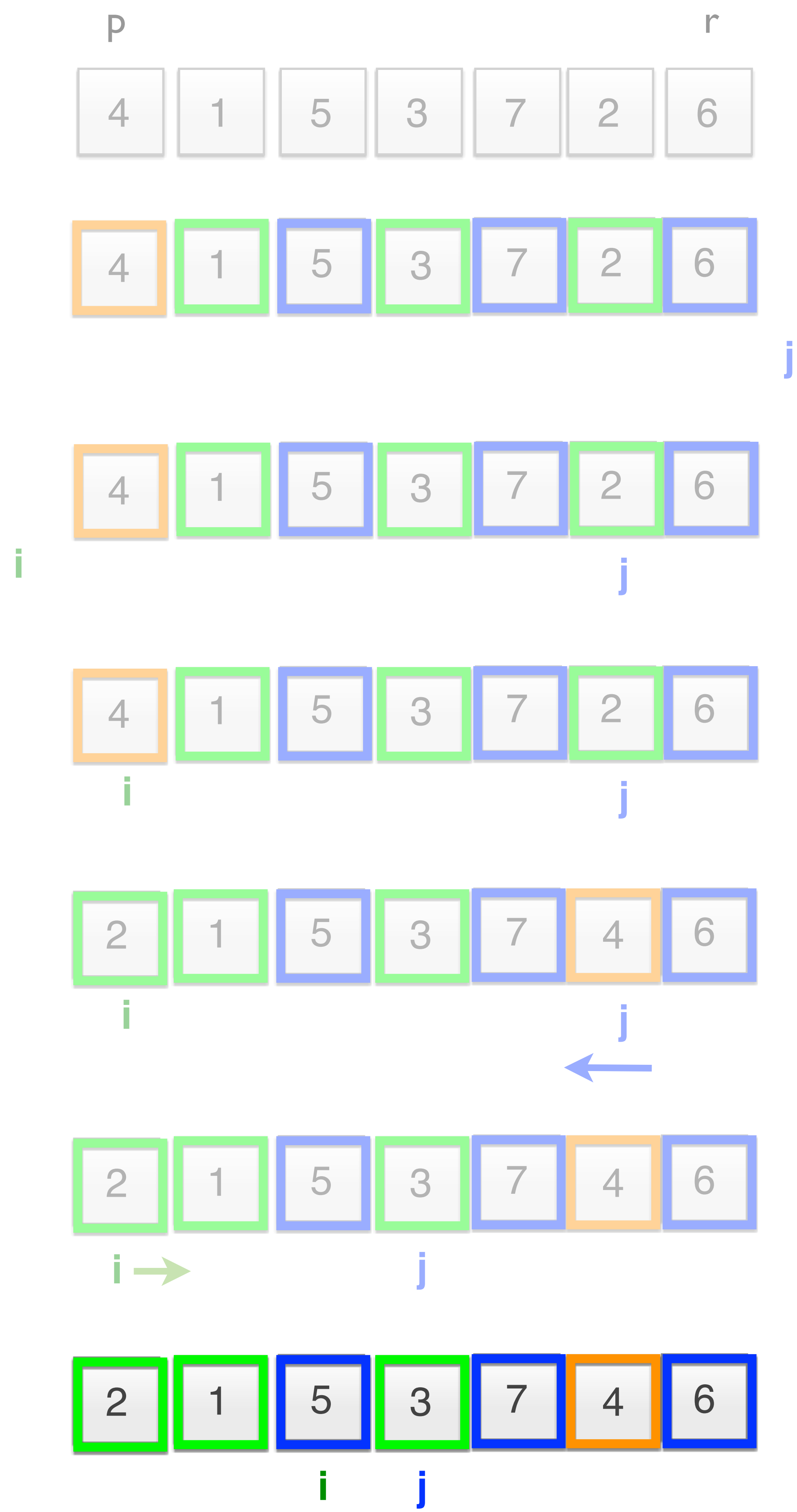
p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j


p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j

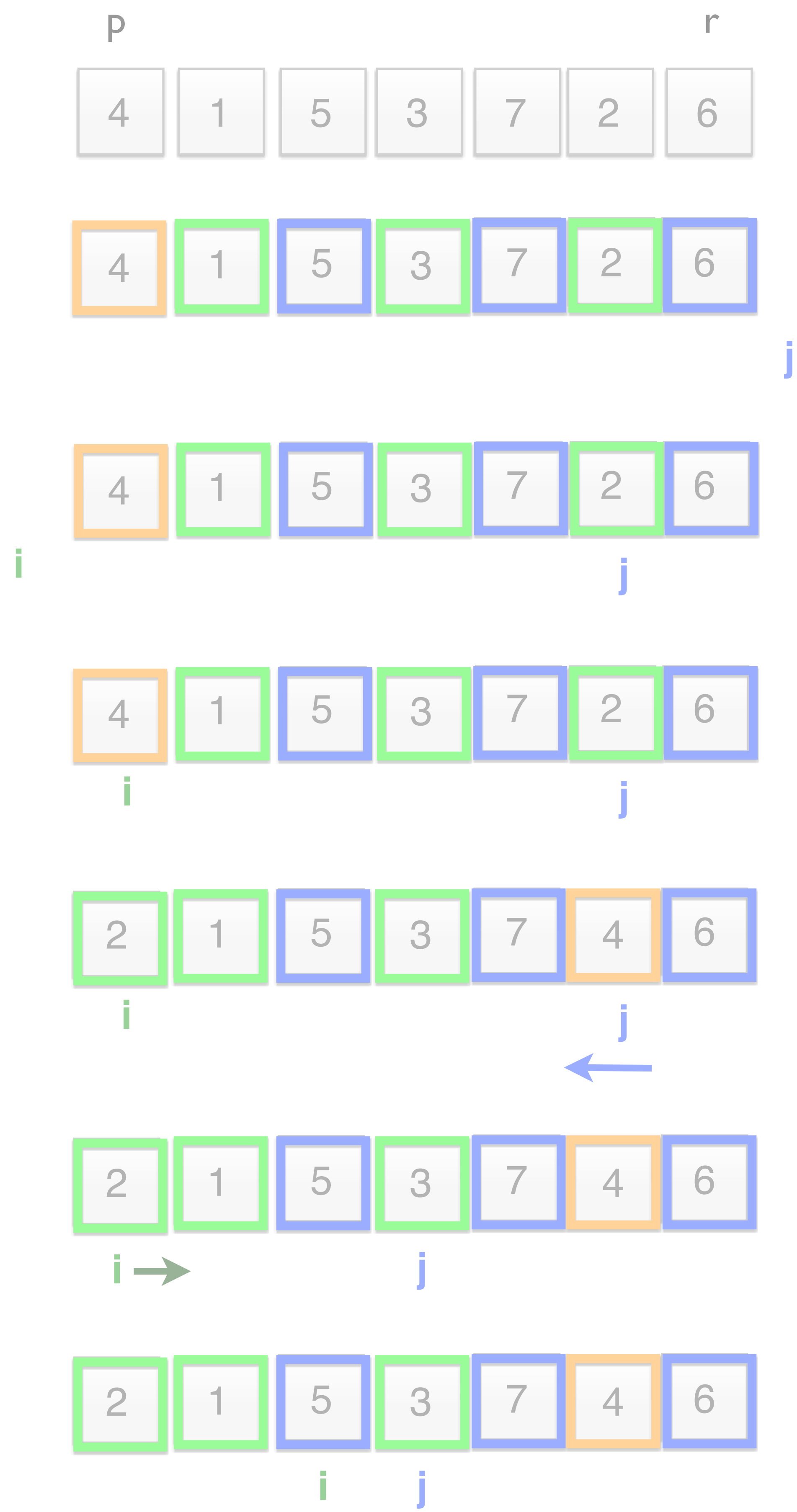
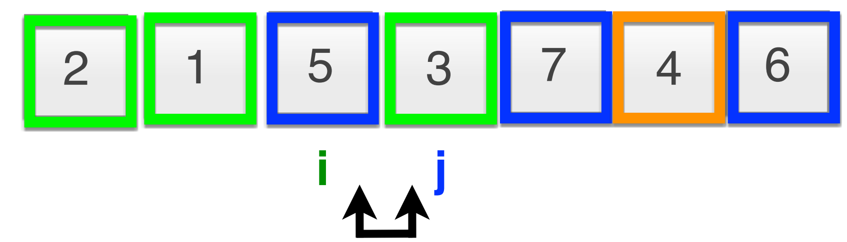


p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j


p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j



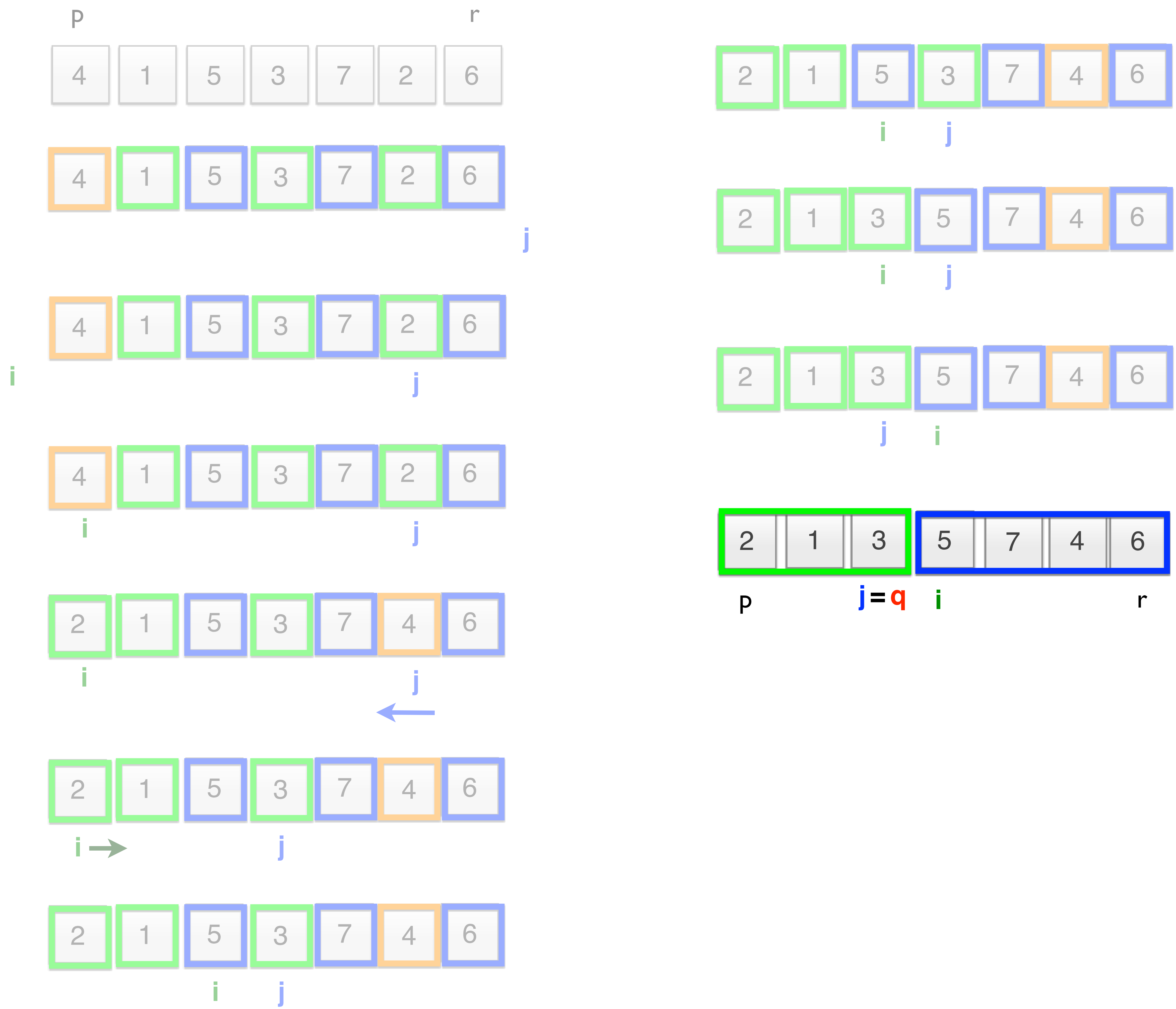
p
r
4 1 5 3 7 2 6
j
i
i
j
2 1 5 3 7 4 6
i
j


p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j

2 1 5 3 7 4 6
i
j

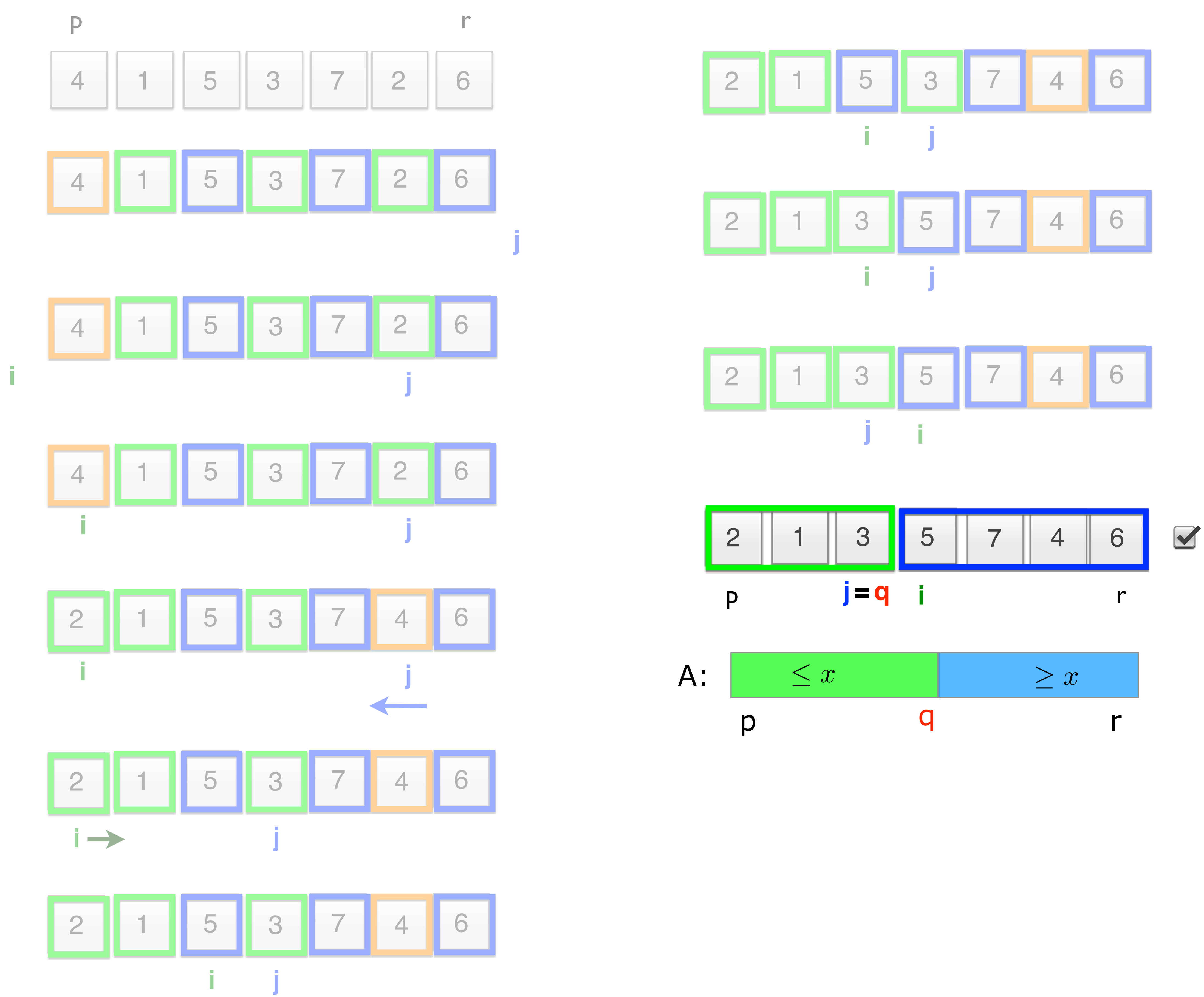

p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 3 5 7 4 6
i
j

p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 3 5 7 4 6
i
j

p
r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i
j
4 1 5 3 7 2 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 5 3 7 4 6
i
j
2 1 3 5 7 4 6
i
j
2 1 3 5 7 4 6
j
i



p r
4 1 5 3 7 2 6
4 1 5 3 7 2 6
j
4 1 5 3 7 2 6
i j
4 1 5 3 7 2 6
i j
2 1 5 3 7 4 6
i j
2 1 5 3 7 4 6
i j
2 1 5 3 7 4 6
i j
2 1 5 3 7 4 6
i j
2 1 3 5 7 4 6
i j
2 1 3 5 7 4 6
j i
2 1 3 5 7 4 6
p j=q i r



p
r
4 1 5 3 7 2 6
j
i
j
2 1 5 3 7 4 6
i
j
2 1 3 5 7 4 6
j=q
i
A:
$\leq x$
$\geq x$
p
q
r

```
PARTITION(A, p, r)
x = A[p]
i = p - 1
j = r + 1
while TRUE
    repeat
        j = j - 1
    until A[j] ≤ x
    repeat
        i = i + 1
    until A[j] ≥ x
    if i < j
        exchange A[i] with A[j]
    else return j
```

PARTITION(A, p, r)

$x = A[p]$ $\boxed{x}$
$i = p - 1$
$j = r + 1$
while TRUE
    repeat
        $j = j - 1$
    until $A[j] \leq x$
    repeat
        $i = i + 1$
    until $A[j] \geq x$
    if $i < j$
        exchange $A[i]$ with $A[j]$
    else return j

Partition(A, p, r)

```
x = A[p]
i = p - 1
j = r + 1
while TRUE
    repeat
        j = j - 1
    until A[j] ≤ x
    repeat
        i = i + 1
    until A[j] ≥ x
    if i < j
        exchange A[i] with A[j]
    else return j
```

$\boxed{x}$

$\textsc{Partition}(A, p, r)$

$x = A[p]$

$i = p - 1$

$j = r + 1$

while TRUE

    repeat

        $j = j - 1$

    until $A[j] \leq x$

    repeat

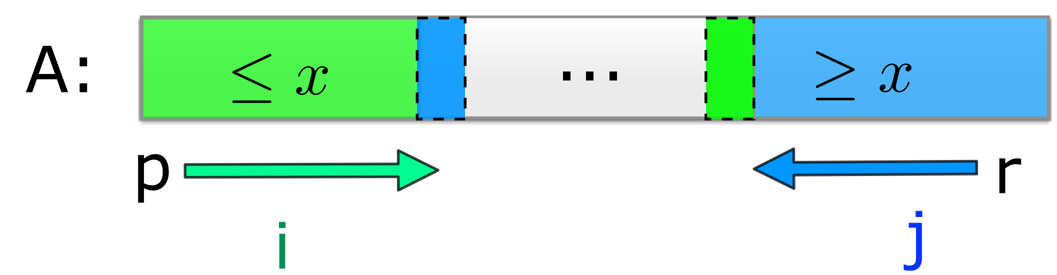
        $i = i + 1$

    until $A[j] \geq x$

    if $i < j$

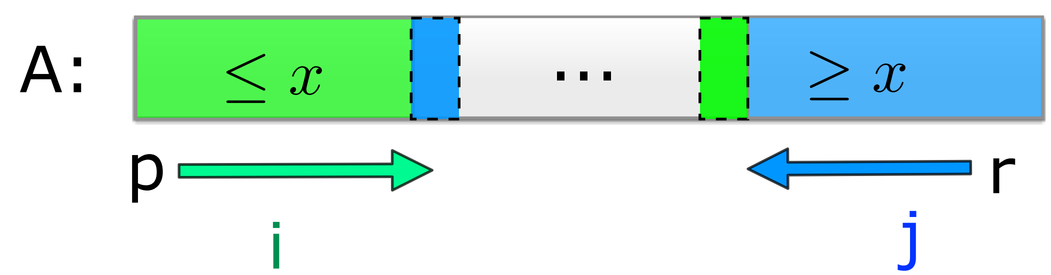
        exchange $A[i]$ with $A[j]$

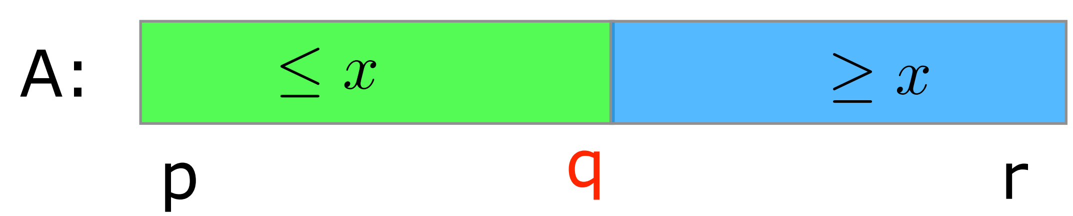
    else return j



PARTITION(A, p, r)

```
x = A[p]
i = p - 1
j = r + 1
while TRUE
    repeat
        j = j - 1
    until A[j] ≤ x
    repeat
        i = i + 1
    until A[j] ≥ x
    if i < j
        exchange A[i] with A[j]
    else return j
```





- Details of partition matter (tinker it, check other partitions, experiment)

- Details of partition matter (<u>tinker it, check other partitions, experiment</u>)
- Number of comparisons depends on the choice of the pivot, i.e., data.

- Details of partition matter (tinker it, check other partitions, experiment)
- Number of comparisons depends on the choice of the pivot, i.e., data.
    - Best case: we partition in halves all along

- Details of partition matter (<u>tinker it, check other partitions, experiment</u>)
- Number of comparisons depends on the choice of the pivot, i.e., data.
  - Best case: we partition in halves all along
  - Worst case: ordered list (partition sizes: 1, n-1)

- Details of partition matter (tinker it, check other partitions, experiment)
- Number of comparisons depends on the choice of the pivot, i.e., data.
  - Best case: we partition in halves all along
  - Worst case: ordered list (partition sizes: 1, n-1)
- The best case is to be expected.
  On average quicksort is very efficient.

Summing up

Summing up

- Divide and conquer
  - Divide: Partition *in-place*
  - Conquer: Order each partition recursively

Summing up

- Divide and conquer
    - Divide: Partition *in-place*
    - Conquer: Order each partition recursively
- Partition algorithms vary, and affect the process dramatically

Summing up

- Divide and conquer
    - Divide: Partition *in-place*
    - Conquer: Order each partition recursively
- Partition algorithms vary, and affect the process dramatically
- Efficient on average

Thank you.

More info: lecture handout

Literature:

- Leiserson, Charles E., Ronald L. Rivest, and Clifford Stein. ***Introduction to algorithms*.** Edited by Thomas H. Cormen. The MIT press, 2001.
- Sedgewick, Robert. and Wayne, Kevin. ***Algorithms*.** Pearson Education, 2011.

garcia@evolbio.mpg.de

http://garciajulian.com