CP312
Algorithm Design and Analysis I

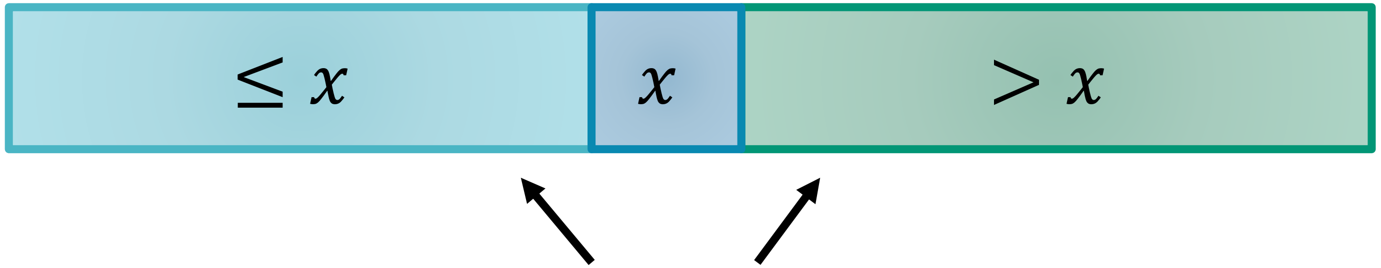
LECTURE 8: QUICKSORT

Quicksort: At a glance

- Based on the divide-and-conquer paradigm
- Recursive algorithm (like merge-sort)
- Quicksort: $\Theta(n^2)$ worst-case, $O(n \lg n)$ **average-case**
- Mergesort: $\Theta(n \lg n)$ worst-case, $\Theta(n \lg n)$ **average-case**

Quicksort Algorithm

- **Divide:** Partition the n-element array around some **pivot** x into two subarrays representing elements $\leq x$ and $> x$.



- **Conquer**: Sort the two subarrays recursively

- **Combine**: automatically happens (in-place)

Quicksort Algorithm

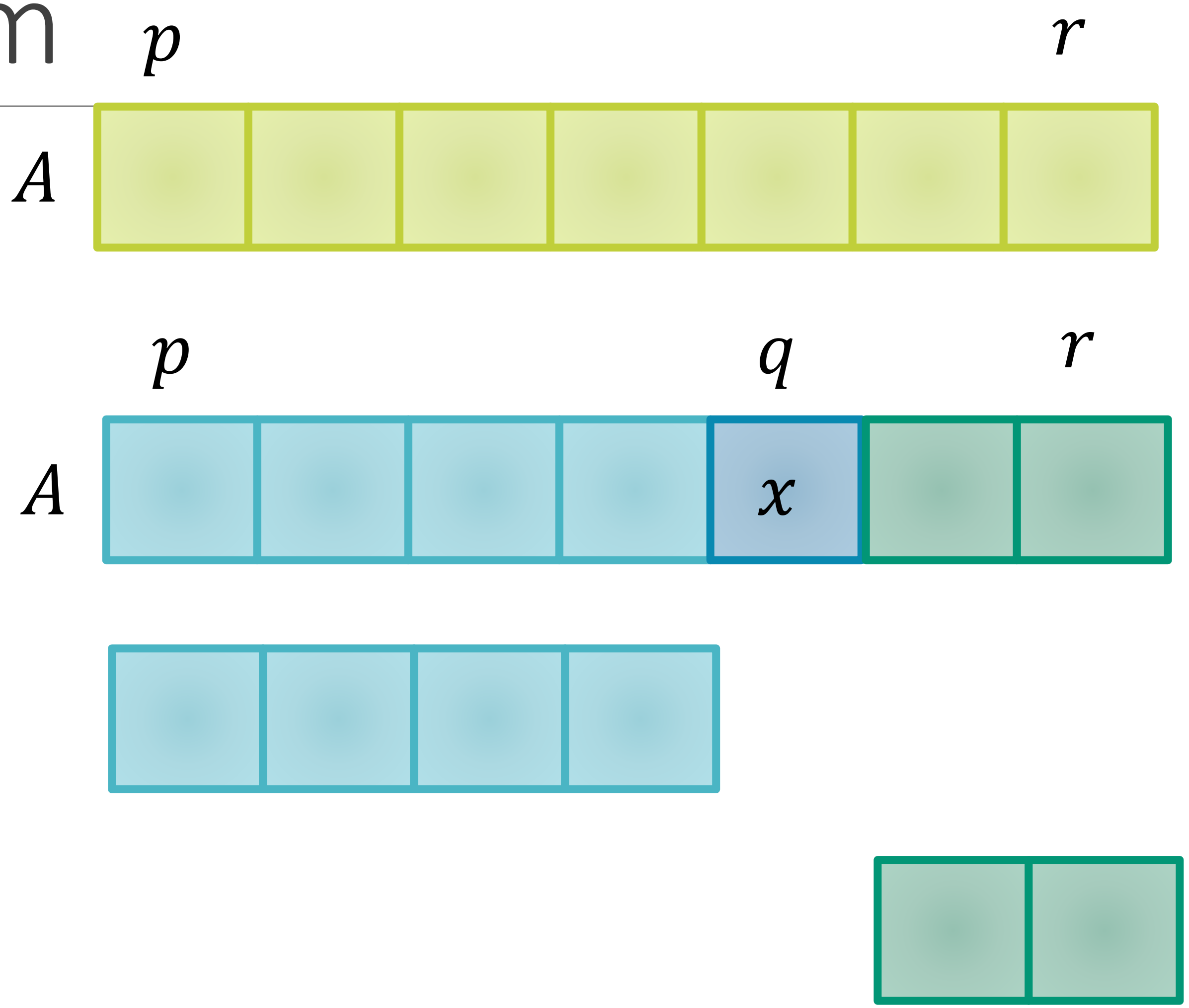
Quicksort(A, p, r):

if $p < r$ **then**

$q \leftarrow$ Partition(A, p, r)

Quicksort$(A, p, q - 1)$

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



Quicksort Algorithm

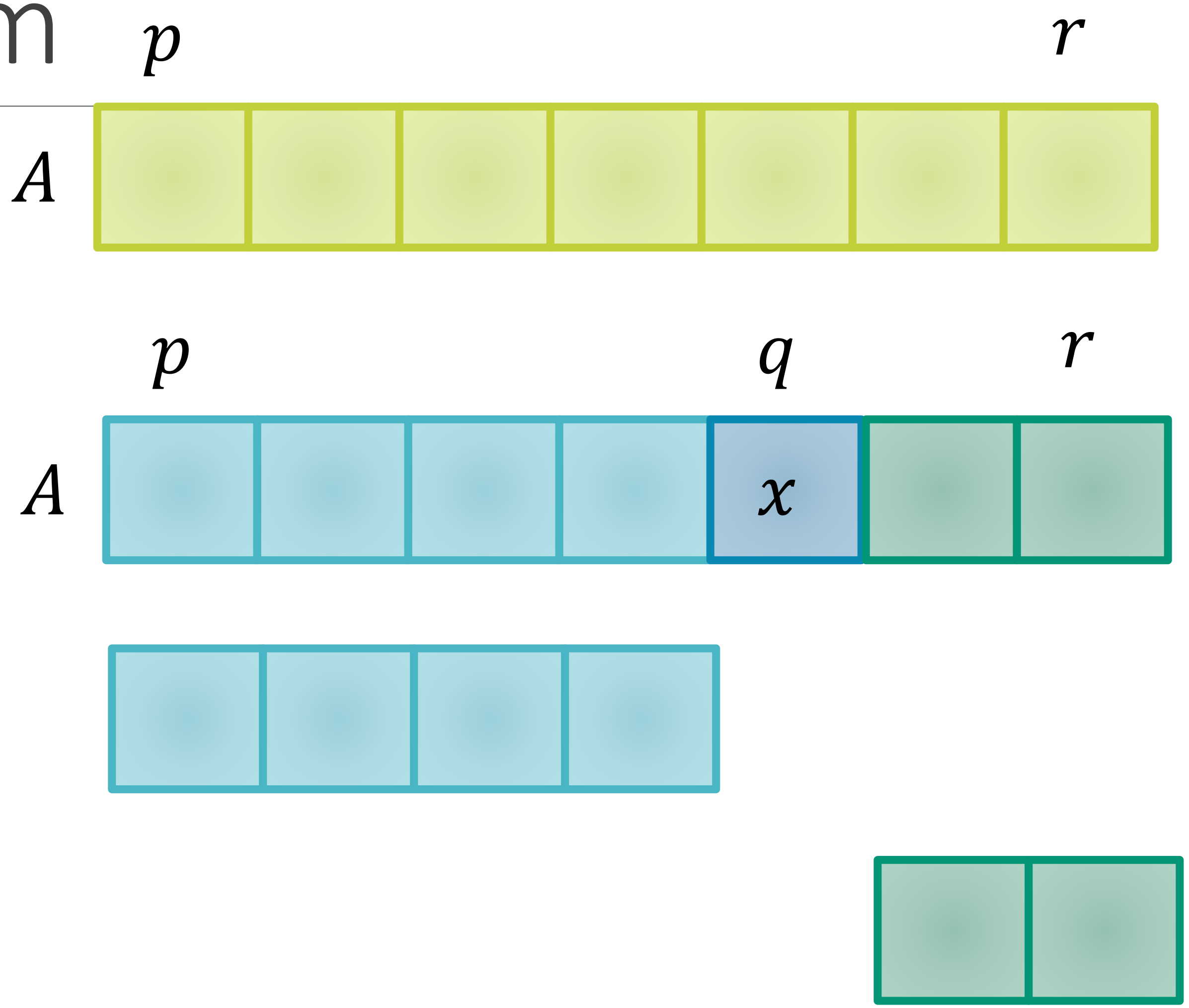
Quicksort(A, p, r):

if $p < r$ **then**

$q \leftarrow$ Partition(A, p, r)

Quicksort$(A, p, q - 1)$

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

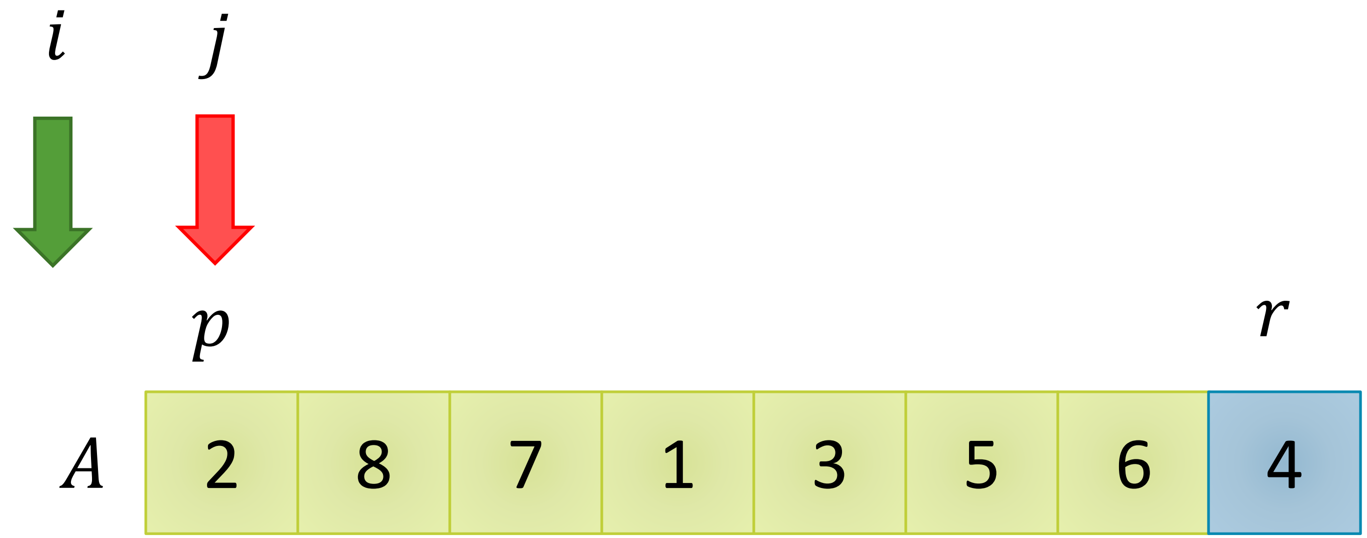


Partition(A, p, r)

	p							r
A	2	8	7	1	3	5	6	4

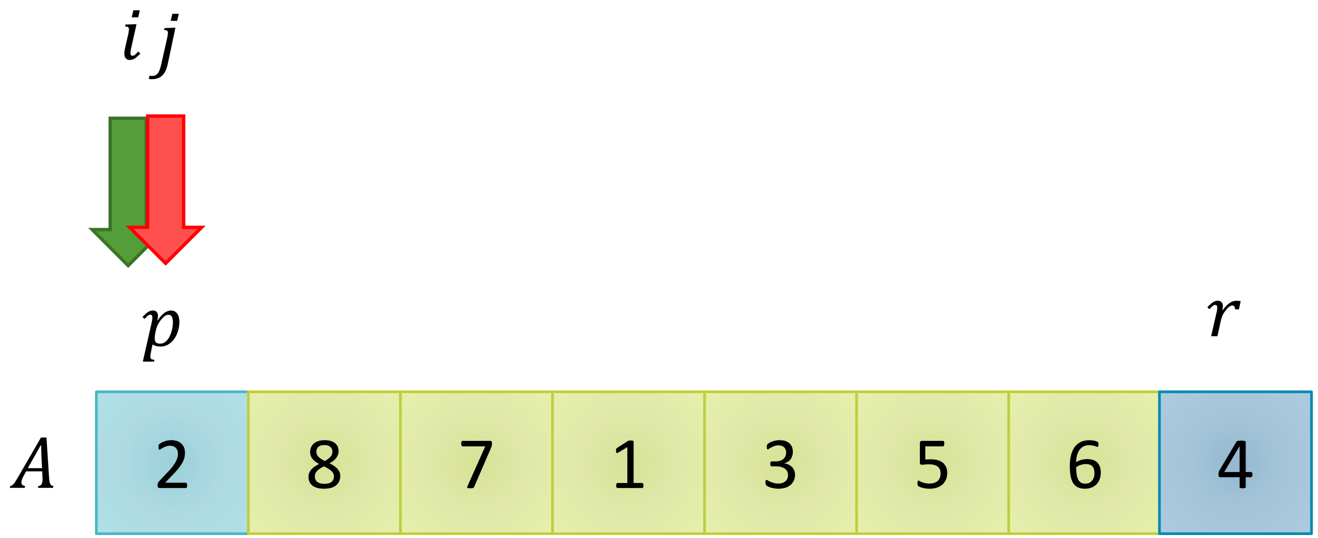
Partition(A, p, r):
$x = A[r]$
$i = p - 1$
for $j = p$ **to** $r - 1$
    if $A[j] \leq x$
        $i = i + 1$
        swap$(A[i], A[j])$
swap$(A[i + 1], A[r])$
return $i + 1$

Partition(A, p, r)



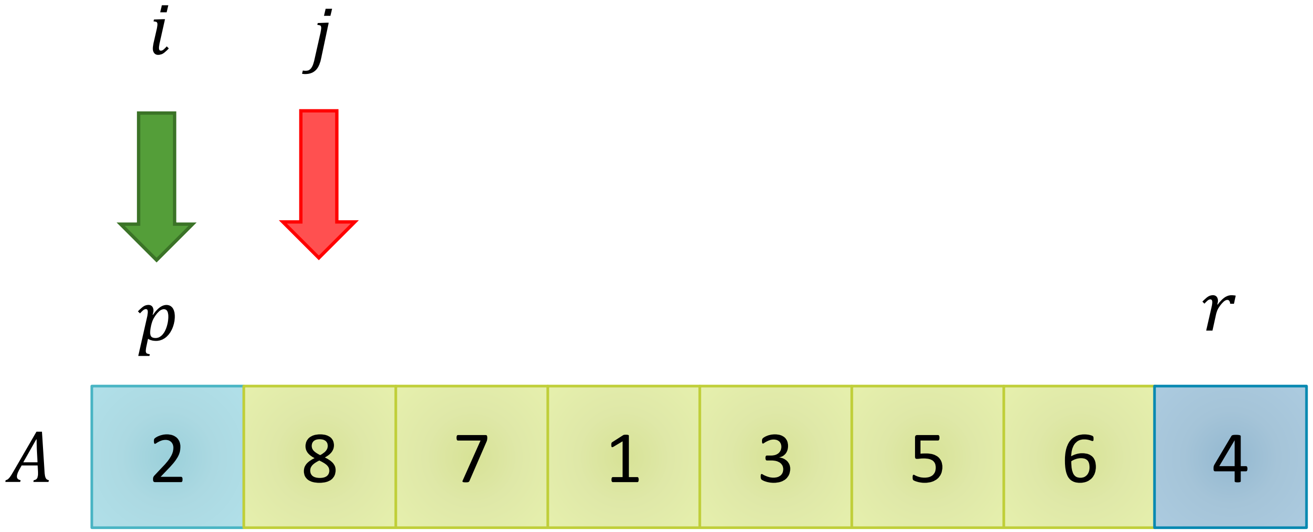
$A[j] \leq x?$ Yes

Partition(A, p, r)



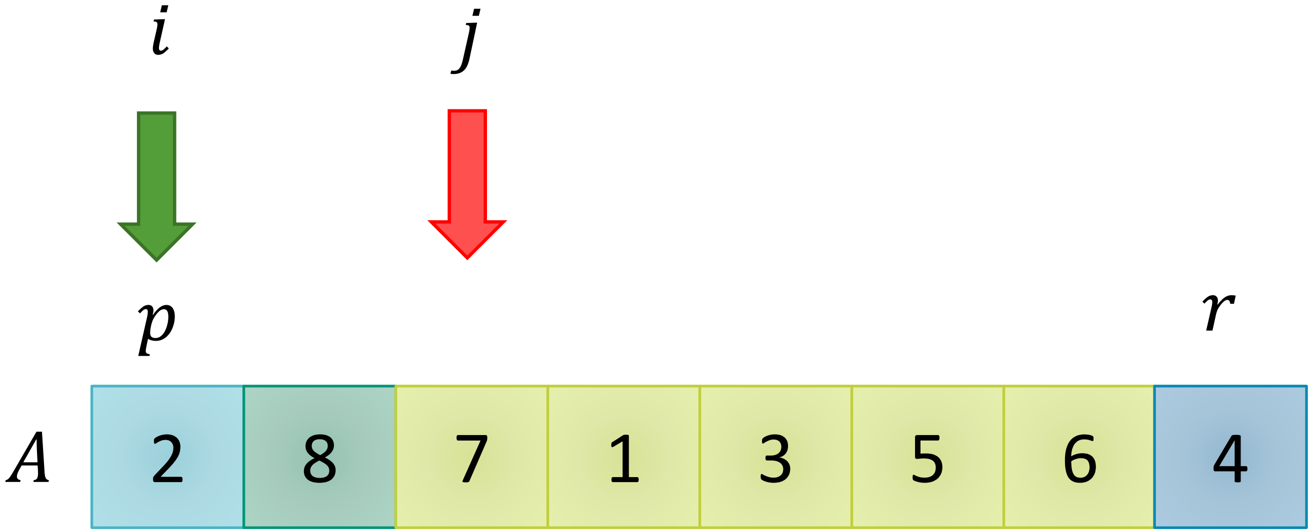
$A[j] \le x?$ Yes

Partition(A, p, r)



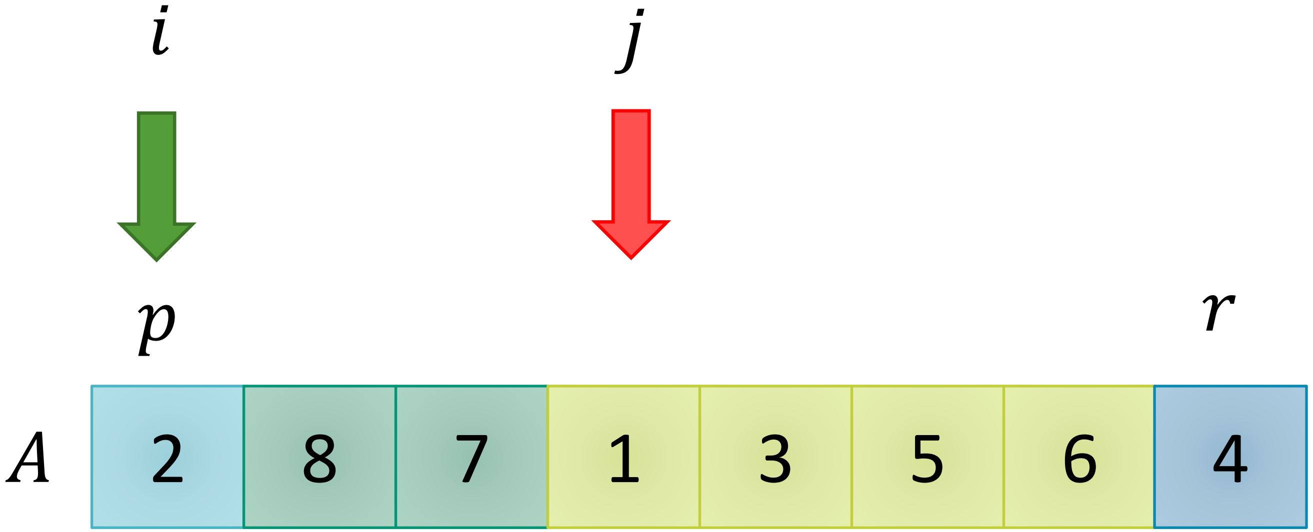
$A[j] \leq x?$ No

Partition(A, p, r)



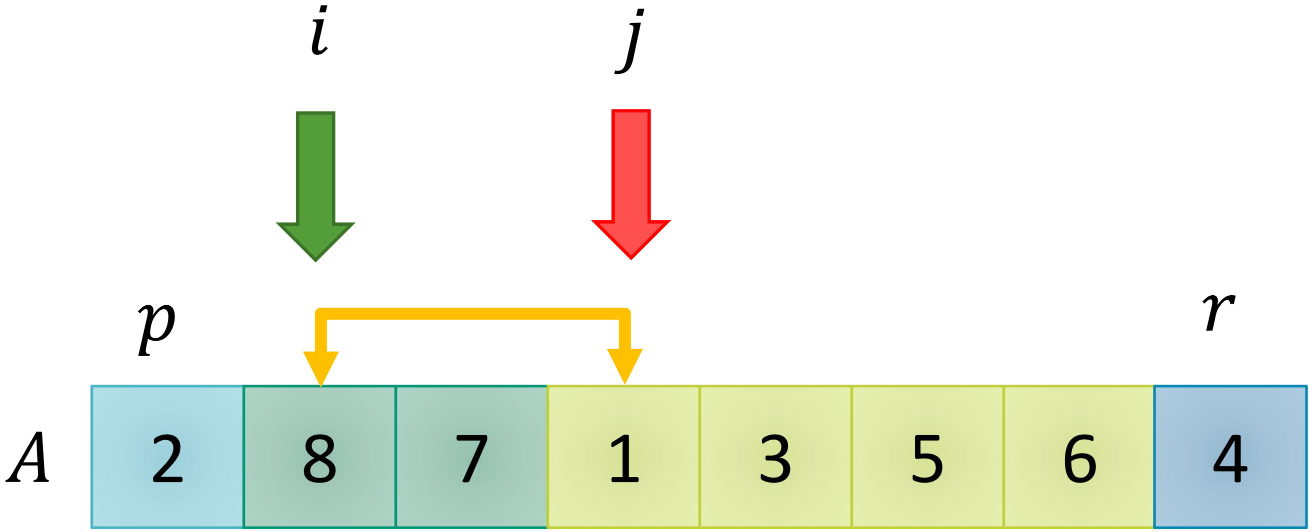
$A[j] \leq x?$ No

Partition(A, p, r)



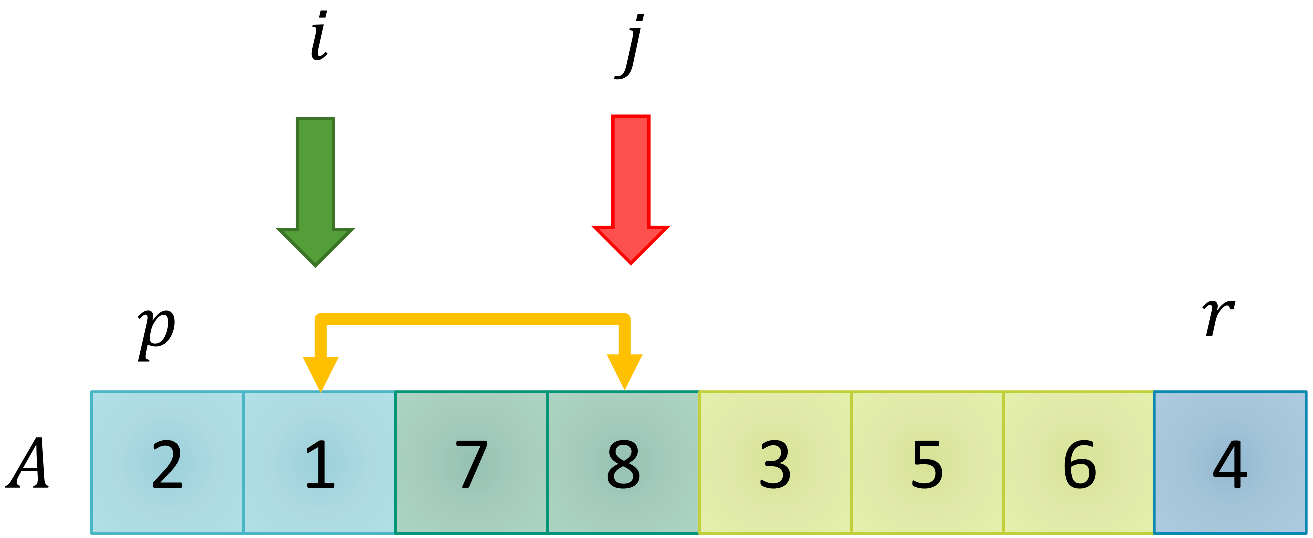
$A[j] \leq x?$ Yes

Partition(A, p, r)



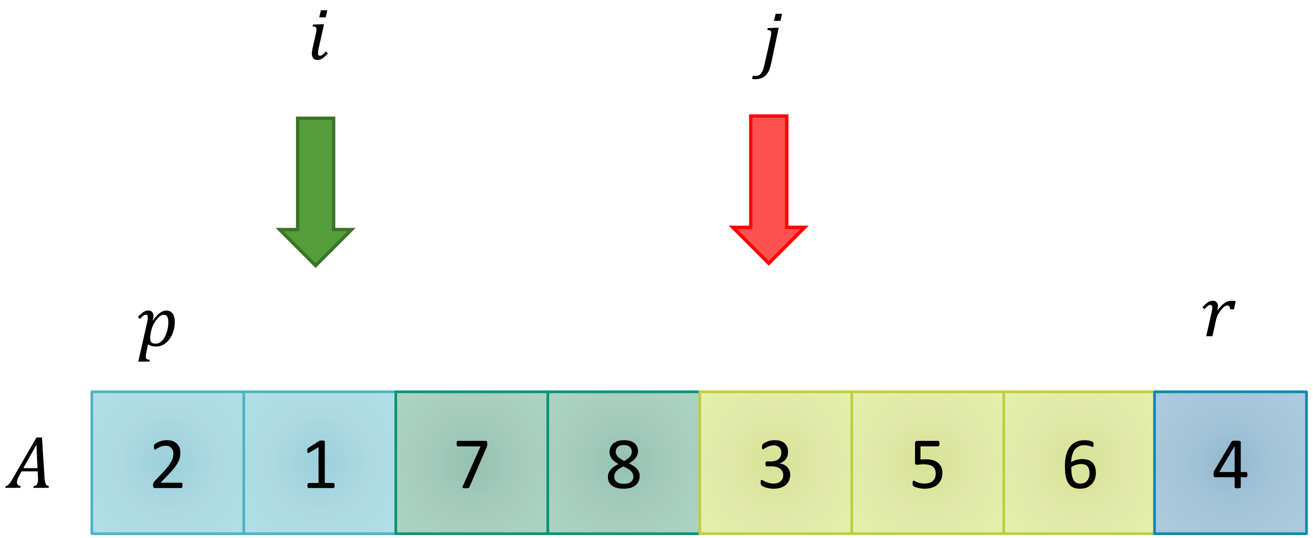
$A[j] \leq x?$ Yes

Partition(A, p, r)



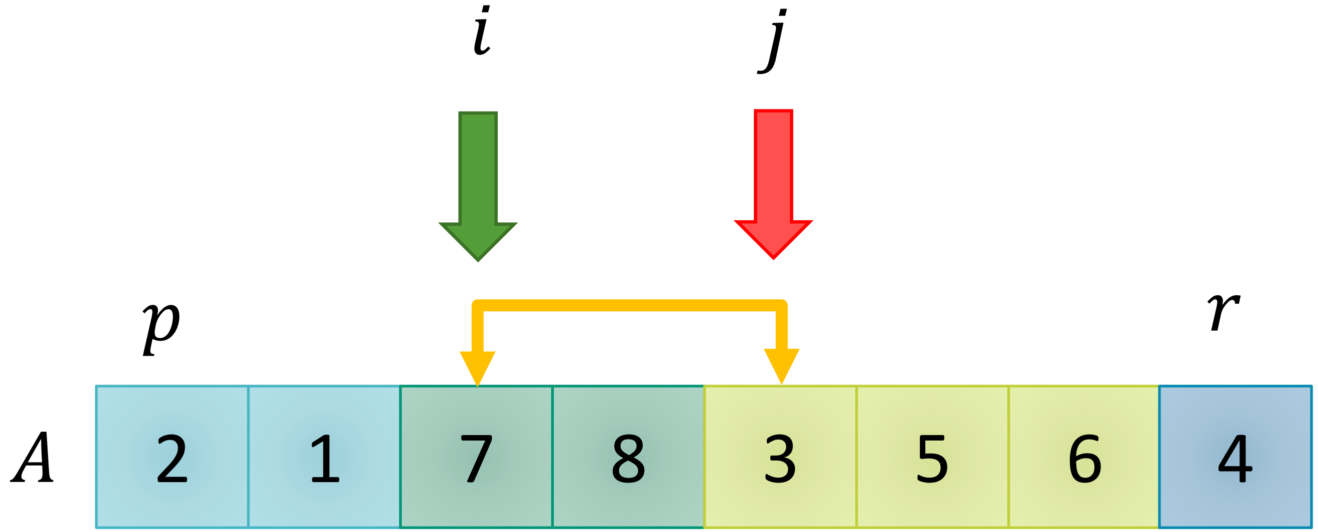
$A[j] \leq x?$ Yes

Partition(A, p, r)



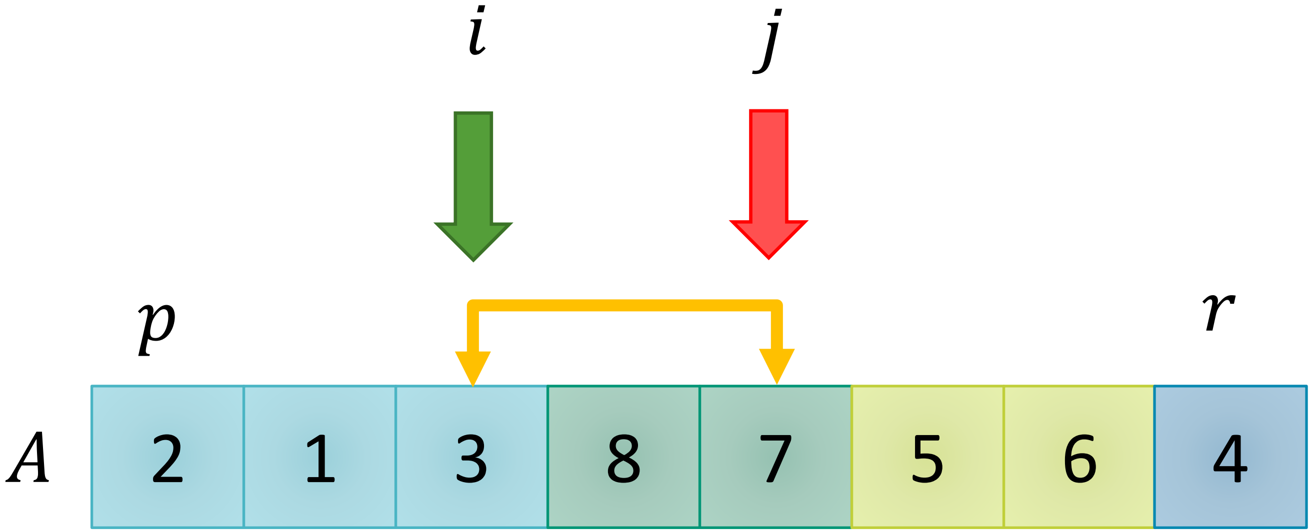
$A[j] \leq x$?   Yes

Partition(A, p, r)



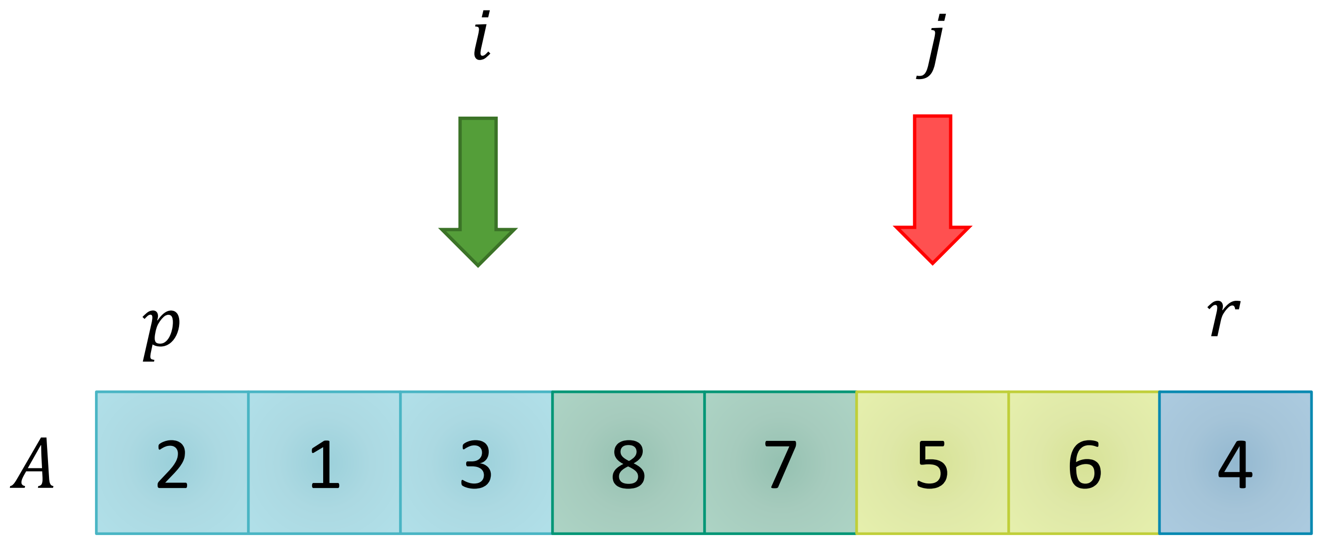
$A[j] \leq x?$ Yes

Partition(A, p, r)



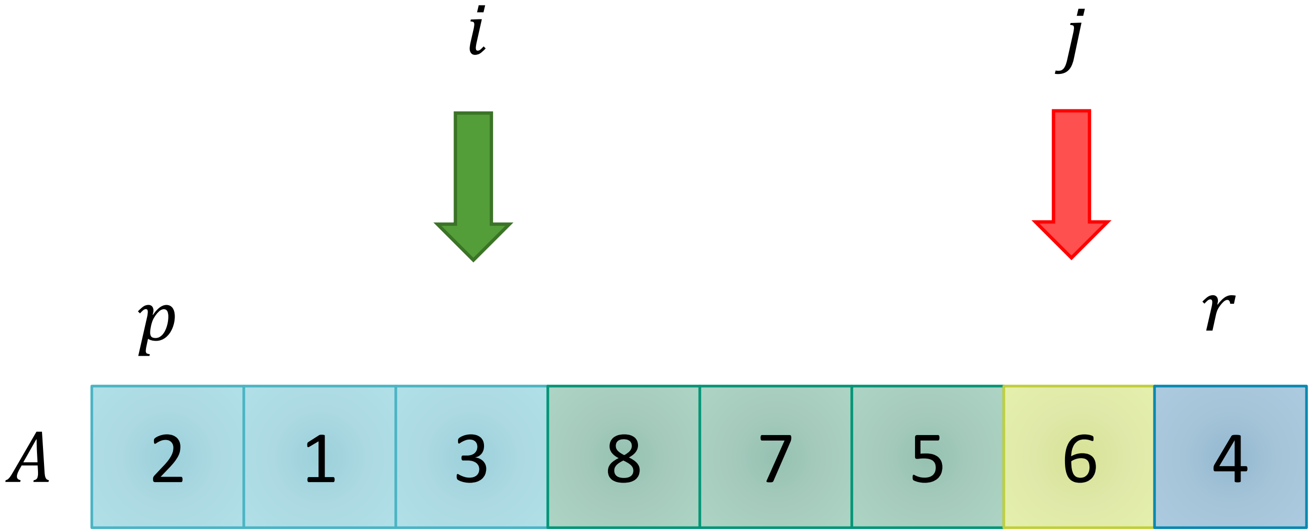
$A[j] \leq x?$ Yes

Partition(A, p, r)



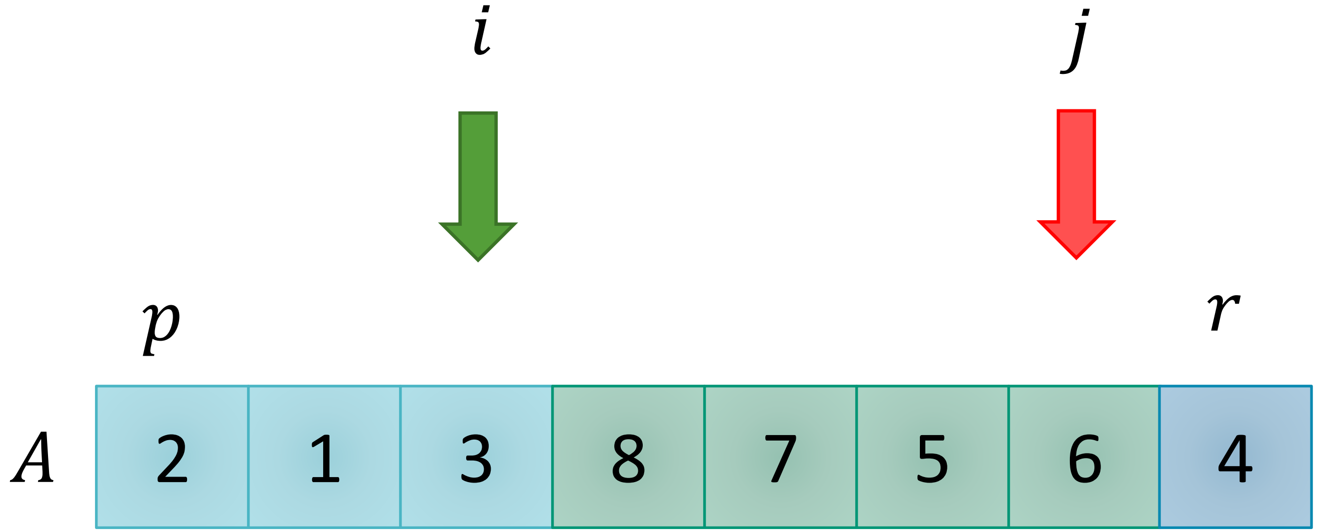
$A[j] \leq x?$ No

Partition(A, p, r)

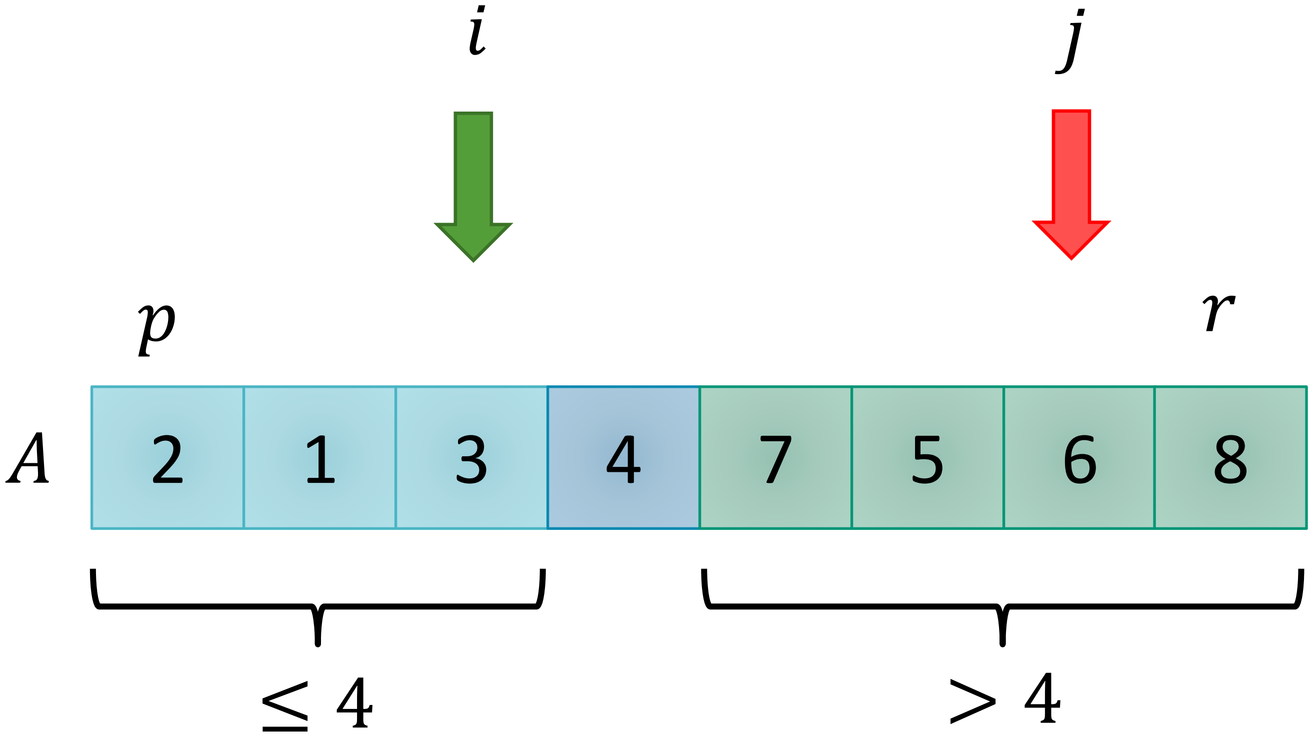


$A[j] \leq x?$ No

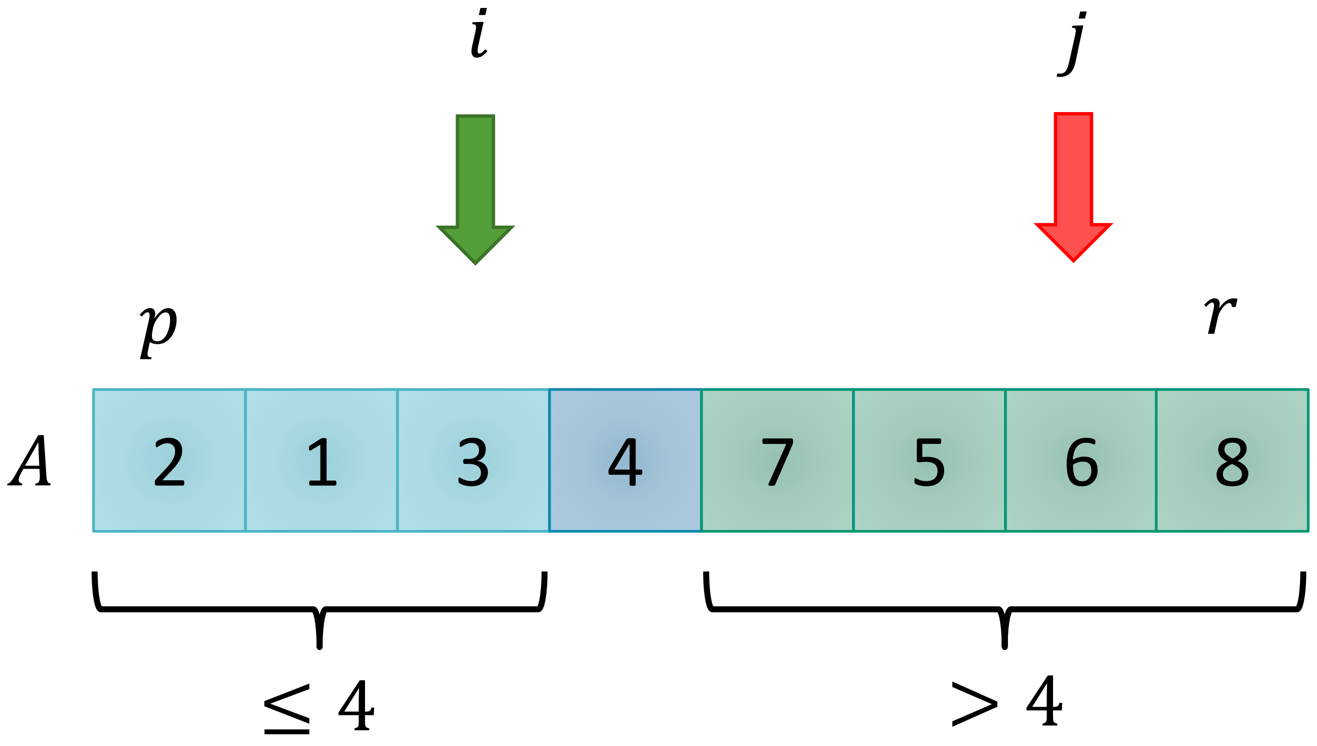
Partition(A, p, r)



Partition(A, p, r)



Partition(A, p, r) Running Time? $\boldsymbol{\Theta(n)}$



Partition(A, p, r):
$x = A[r]$
$i = p - 1$
for $j = p$ **to** $r - 1$
    if $A[j] \leq x$
        $i = i + 1$
        swap$(A[i], A[j])$
swap$(A[i+1], A[r])$
return $i + 1$

if $p < r$ **then**

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

  $\text{Quicksort}(A, p, q - 1)$

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

Analysis of Quicksort

- Worst-Case Analysis
  - Input sorted or reverse-sorted
  - Partition around minimum or maximum element
  - One side of partition always has no elements

	p							r
A	1	2	3	4	5	6	7	8

Analysis of Quicksort

if $p < r$ **then**
  $q \leftarrow$ Partition(A, p, r)
  Quicksort$(A, p, q-1)$
  Quicksort$(A, q+1, r)$

- Worst-Case Analysis

	p							r
A	1	2	3	4	5	6	7	8

$\underbrace{1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7}_{n-1 \text{ elements}}$

$$T(n) = T(0) + T(n-1) + \Theta(n)$$

Analysis of Quicksort

- Worst-Case Analysis $T(n) = T(0) + T(n-1) + \Theta(n)$

$$T(n)$$

Analysis of Quicksort

- Worst-Case Analysis $T(n) = T(0) + T(n-1) + \Theta(n)$

$$\begin{tikzcd} & cn \arrow[dl] \arrow[dr] & \\ T(0) & & T(n-1) \end{tikzcd}$$

Analysis of Quicksort

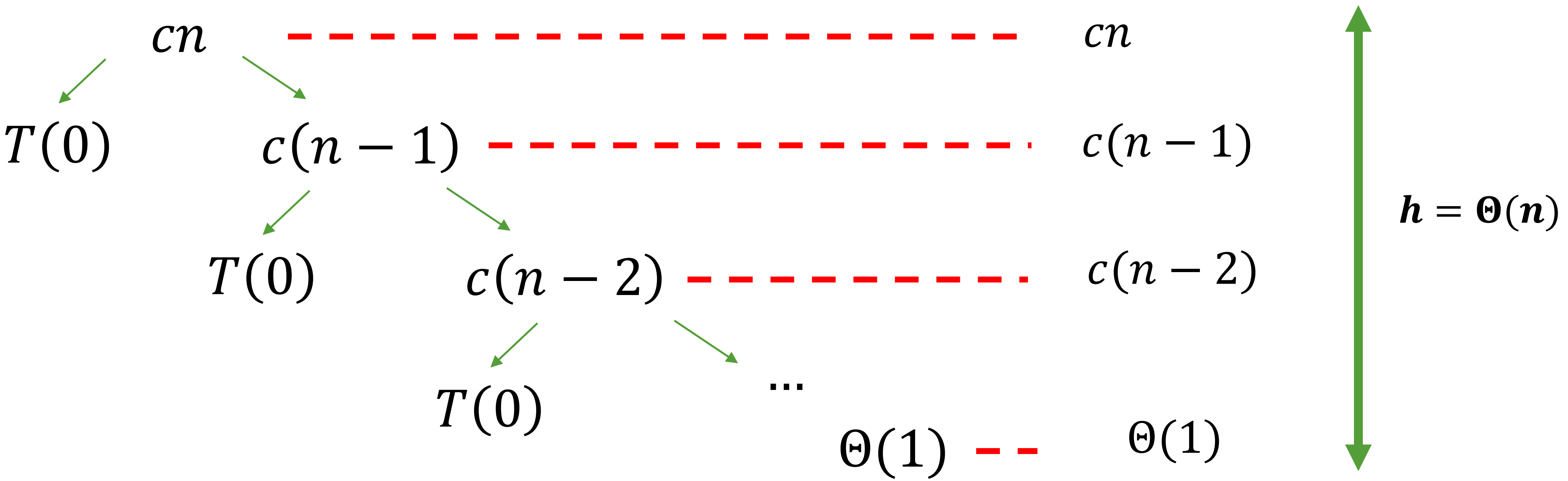
- Worst-Case Analysis $\quad T(n) = T(0) + T(n-1) + \Theta(n)$

$$
\begin{tikzcd}
 & cn \arrow[dl] \arrow[dr] & & \\
T(0) & & c(n-1) \arrow[dl] \arrow[dr] & \\
 & T(0) & & T(n-2)
\end{tikzcd}
$$

$$T(n) = \sum_{i=0}^{h} O(n-i) = O(n^2)$$

Analysis of Quicksort

- Worst-Case Analysis $\quad T(n) = T(0) + T(n-1) + \Theta(n)$



if $p < r$ **then**
 $q \leftarrow \text{Partition}(A, p, r)$
 $\text{Quicksort}(A, p, q-1)$
 $\text{Quicksort}(A, q+1, r)$

Analysis of Quicksort

- Best-Case Analysis
  - Half of the elements are less than the pivot and half are greater than the pivot

	p						r
A	1	5	3	7	2	6	4

$$T(n) = 2T(n/2) + \Theta(n)$$

$$T(n) = \Theta(n \lg n)$$

if $p < r$ **then**
$q \leftarrow \text{Partition}(A, p, r)$
$\text{Quicksort}(A, p, q-1)$
$\text{Quicksort}(A, q+1, r)$

Analysis of Quicksort

- "Almost" Best-Case Analysis
  - 9/10 of the elements are less than the pivot and 1/10 are greater than the pivot

$$T(n) = T(n/10) + T(9n/10) + \Theta(n)$$

  - Recursion Tree yields $O(n)$ every level with height $h = O(\log_{10/9} n) = \Theta(\lg n)$
  - $T(n) = \sum_{i=0}^{h} O(n) = \sum_{i=0}^{\lg n} O(n) = O(n \lg n)$

Randomized Quicksort

- **Idea**: Pick a **random** pivot to avoid choosing a bad pivot for worst-case inputs.
- Running time is independent of the input order.
- No assumptions need to be made about the input distribution.
- No specific input elicits worst-case behavior.
- The worst-case is determined only by the output of the random number generator.

Randomized Quicksort

Quicksort(A, p, r):

if $p < r$ **then**

$q \leftarrow$ Partition(A, p, r)

Quicksort$(A, p, q - 1)$

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

Randomized Quicksort

Randomized-Quicksort(A, p, r):

if $p < r$ **then**

$q \leftarrow$ Randomized-Partition(A, p, r)

Randomized-Quicksort$(A, p, q-1)$

Randomized-Quicksort$(A, q+1, r)$

$i = RAND(p, r)$
Swap $A[r]$ with $A[i]$
return Partition(A, p, r)

Randomized Quicksort

Randomized-Quicksort(A, p, r):

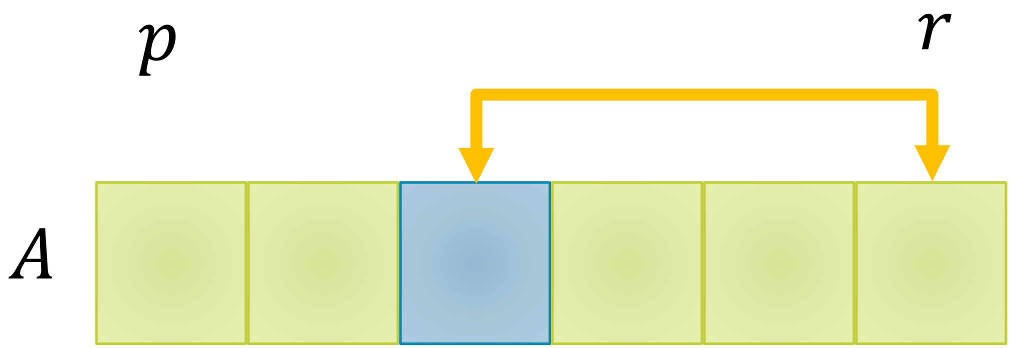
if $p < r$ **then**

$q \leftarrow$ Randomized-Partition(A, p, r)

Randomized-Quicksort$(A, p, q-1)$

Randomized-Quicksort$(A, q+1, r)$

$i = RAND(p, r)$
Swap $A[r]$ with $A[i]$
return Partition(A, p, r)



Randomized Quicksort

Randomized-Quicksort(A, p, r):

if $p < r$ **then**

  $q \leftarrow$ Randomized-Partition(A, p, r)

  Randomized-Quicksort$(A, p, q - 1)$

  Randomized-Quicksort$(A, q + 1, r)$

$i = RAND(p, r)$
Swap $A[r]$ with $A[i]$
return Partition(A, p, r)

Randomized Quicksort Analysis

- The **worst-case expected** running time is $O(n \lg n)$. That is, for *every* input of size n:

$$E[T(n)] = O(n \lg n)$$

- This is equivalent to saying that the **average-case** running-time of standard quicksort is $O(n \lg n)$**.**