Design and Analysis of Algorithms

03-08
Divide – and – Conquer

Quick Sort

Imran Ihsan
Assistant Professor, Department of Computer Science
Air University, Islamabad, Pakistan
www.imranihsan.com

Quick Sort

comparison based algorithm

running time: O(n log n) (on average)

efficient in practice

Example: Quick Sort

6	4	8	2	9	3	9	4	7	6	1

Example: Quick Sort

6	4	8	2	9	3	9	4	7	6	1

partition with respect to x = A[1] in particular,

x is in its final position

1	4	2	3	4	6	6	9	7	8	9

$\leq$ 6 $\geq$ 6

Example: Quick Sort

6	4	8	2	9	3	9	4	7	6	1

partition with respect to x = A[1] in particular,

x is in its final position

1	4	2	3	4	6	6	9	7	8	9

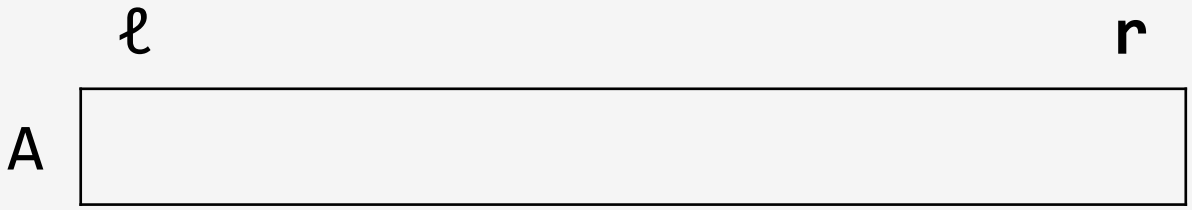
sort the two parts recursively

1	2	3	4	4	6	6	7	8	9	9

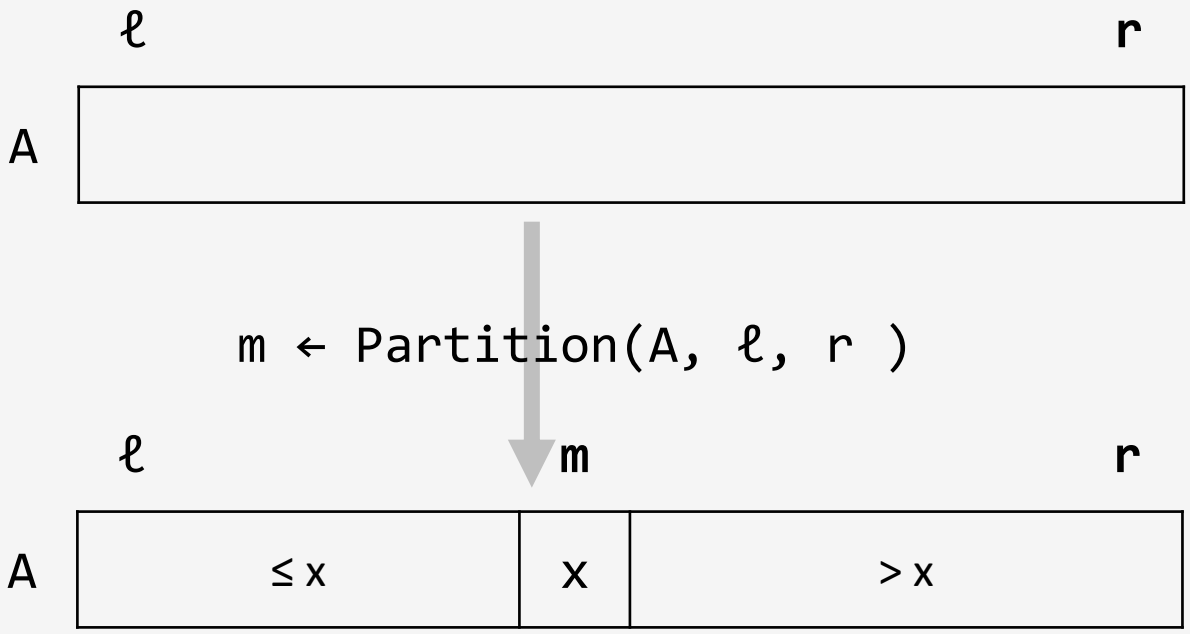
Quick Sort

```
QuickSort(A, ℓ, r)
      if ℓ ≥ r:
         return
      m ← Partition(A, ℓ, r )
      {A[m] is in the final position}
      QuickSort(A, ℓ,m – 1)
      QuickSort(A,m + 1, r )
```

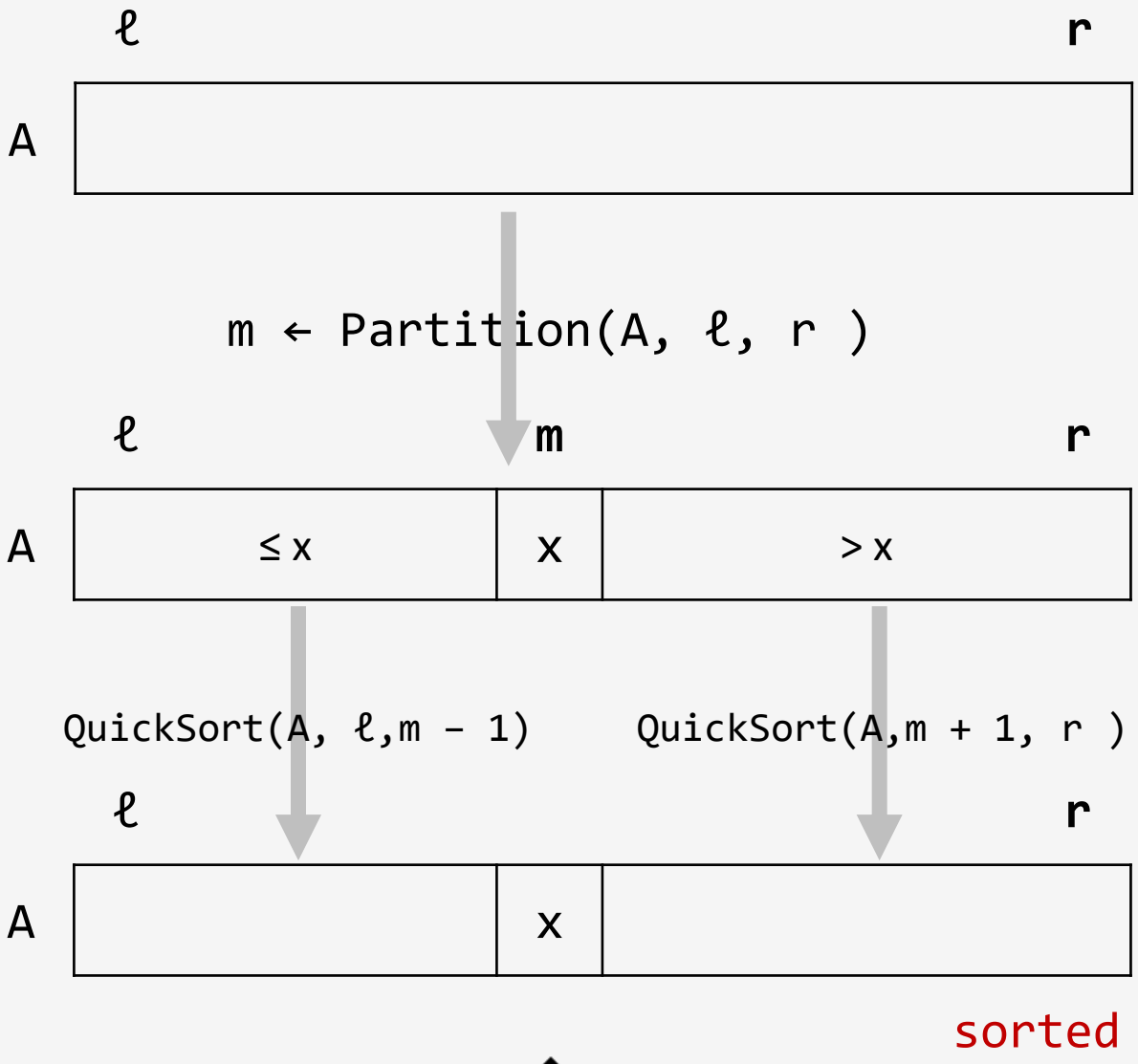
Quick Sort



Quick Sort



Quick Sort


ℓ
r
A
m ← Partition(A, ℓ, r)
ℓ
m
r
A
≤ x
x
> x
QuickSort(A, ℓ,m – 1)
QuickSort(A,m + 1, r)
ℓ
r
A
x
sorted

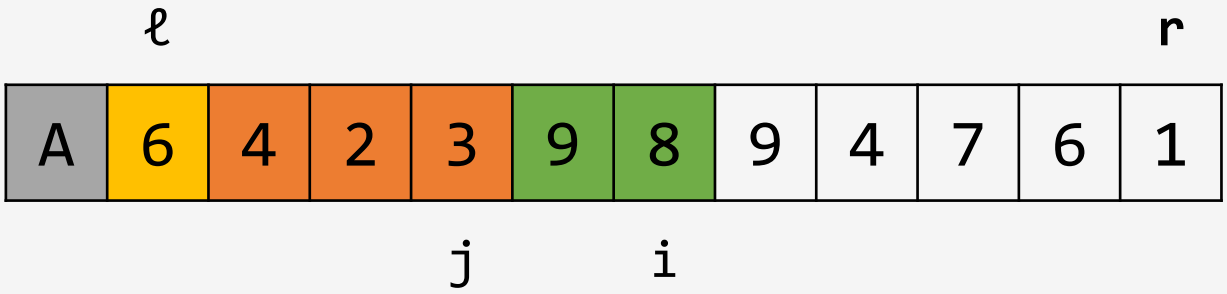
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



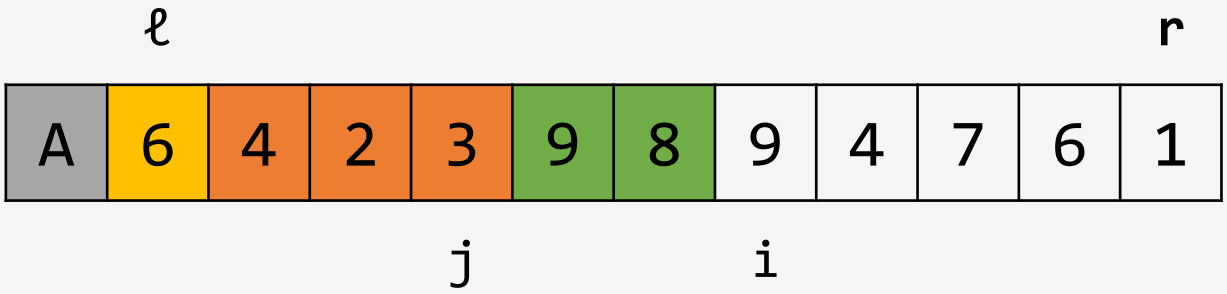
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



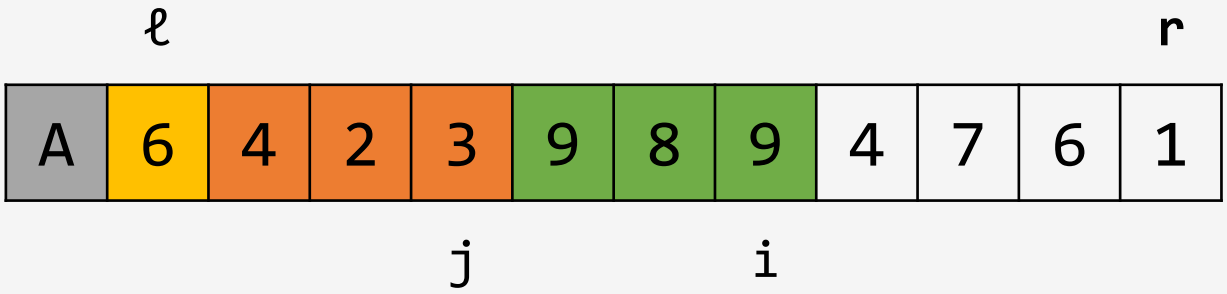
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



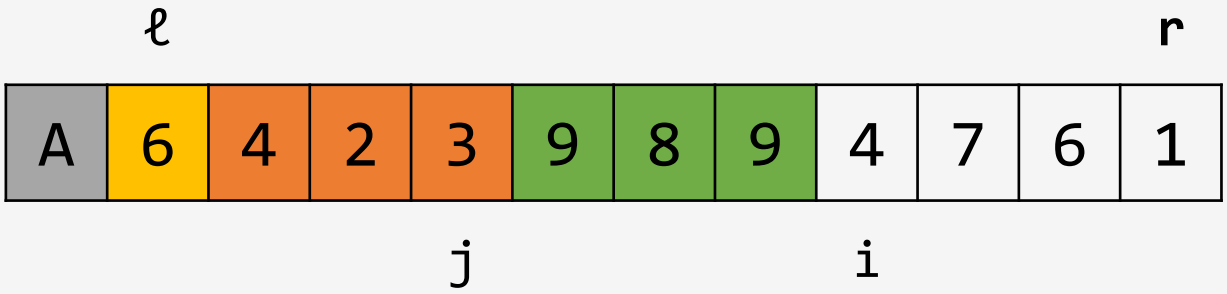
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



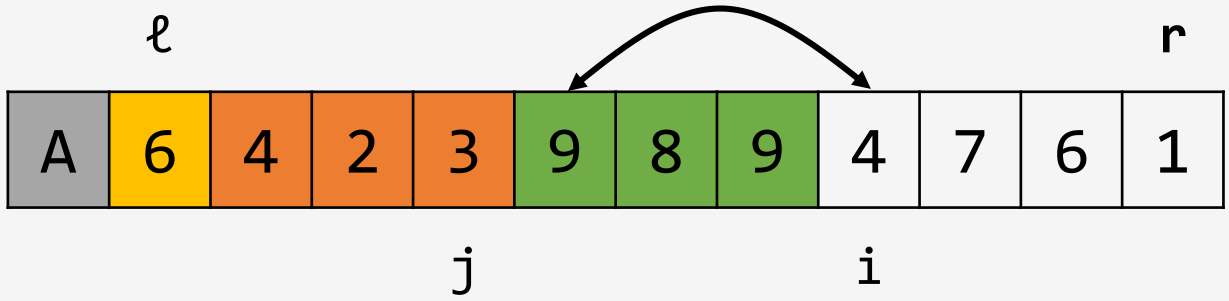
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



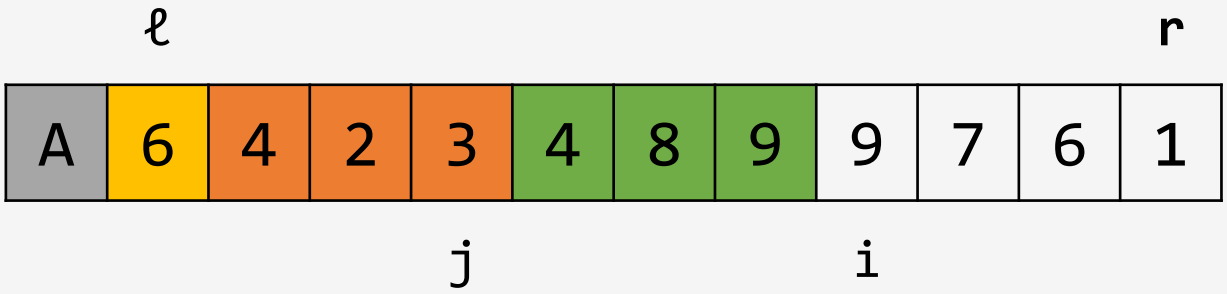
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



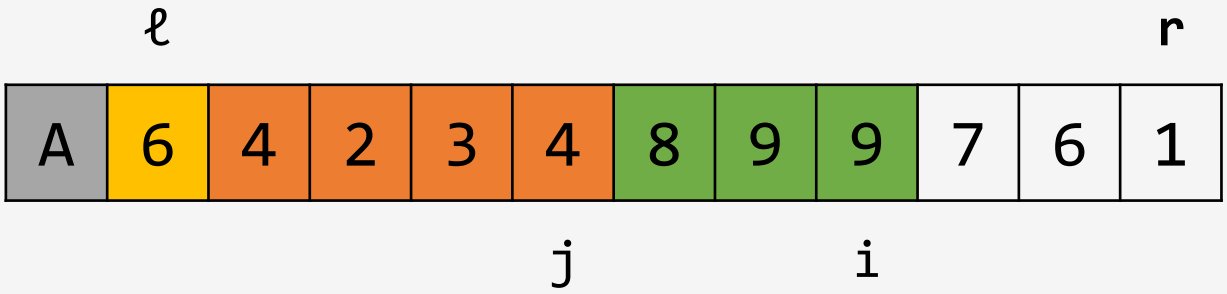
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



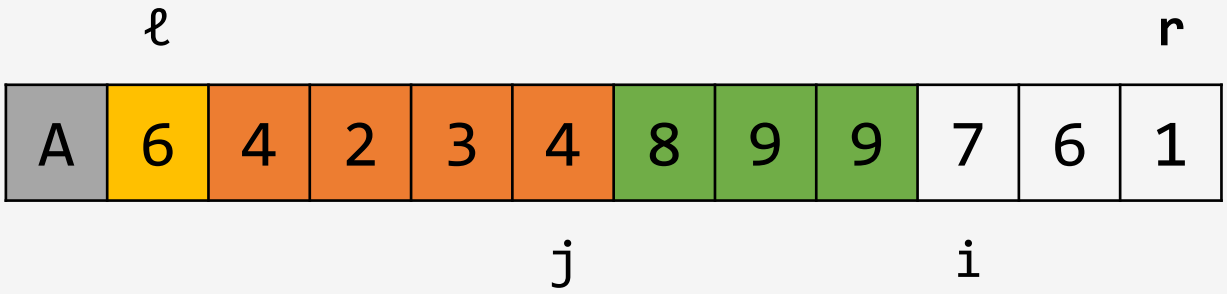
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



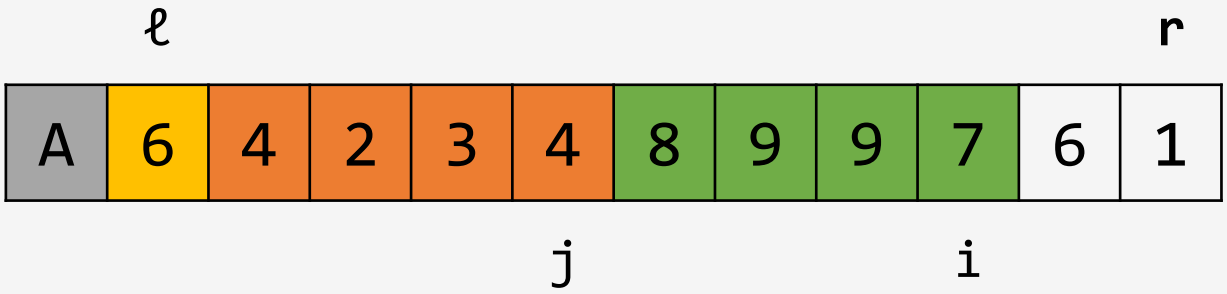
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



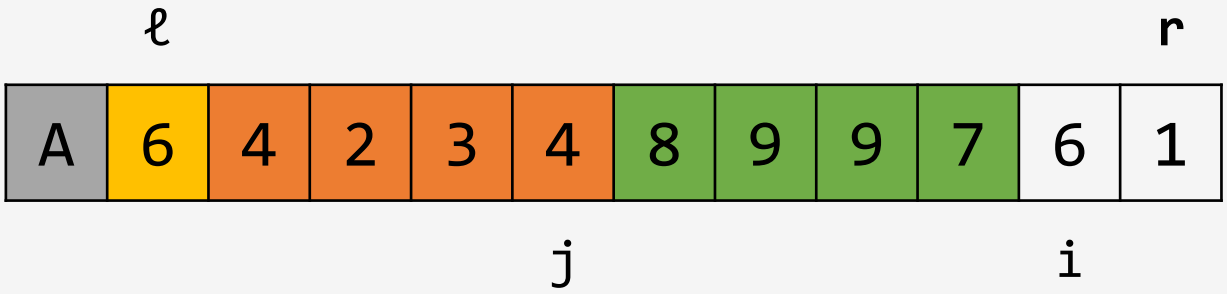
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



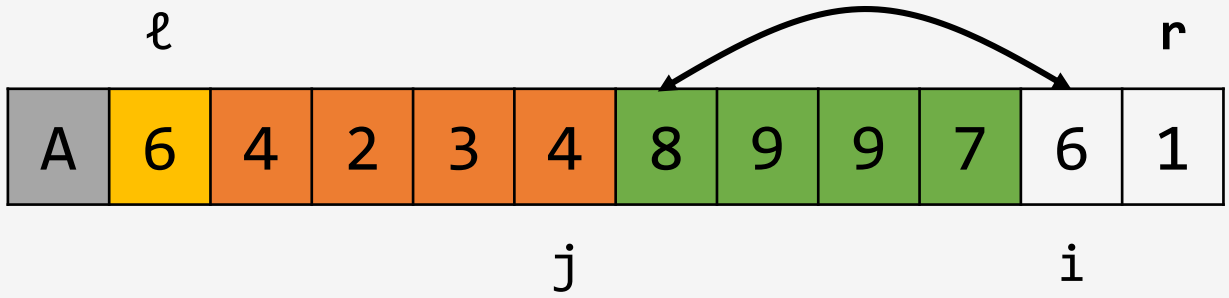
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



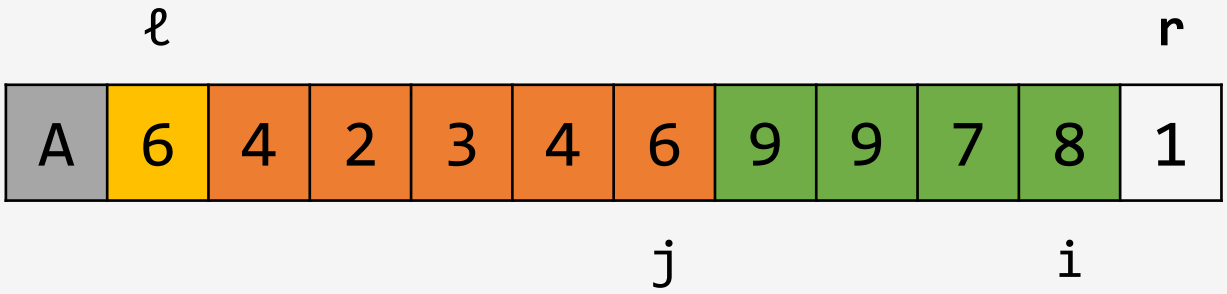
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



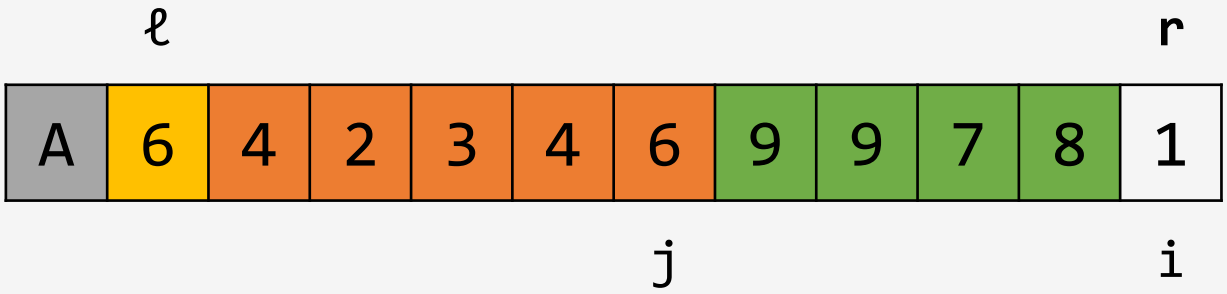
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



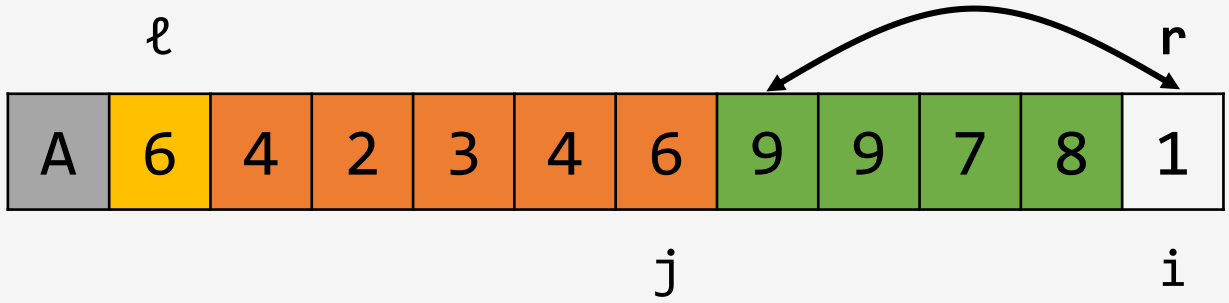
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



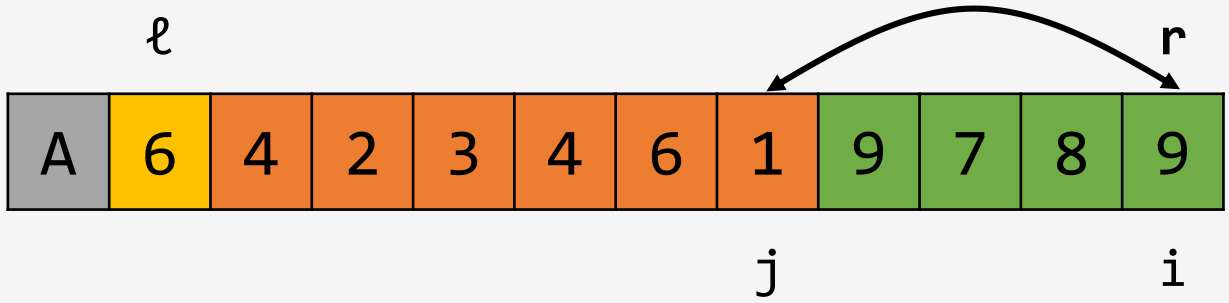
Partitioning: Example

the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ i



Partitioning: Example

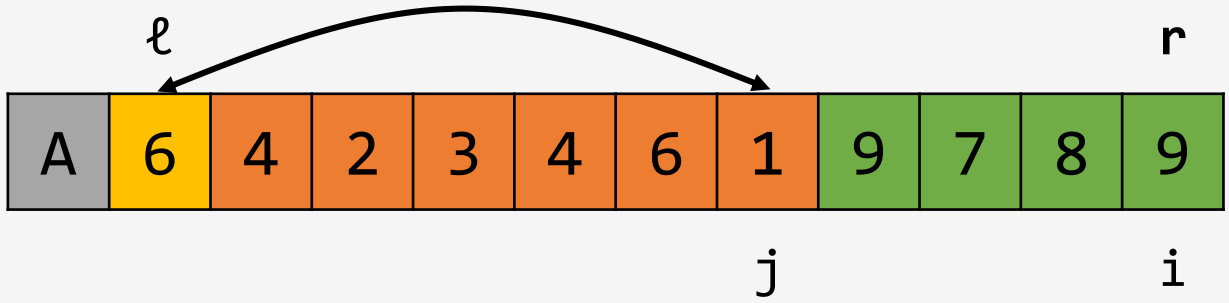
the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ I

in the end, move A[ℓ] to its final place



Partitioning: Example

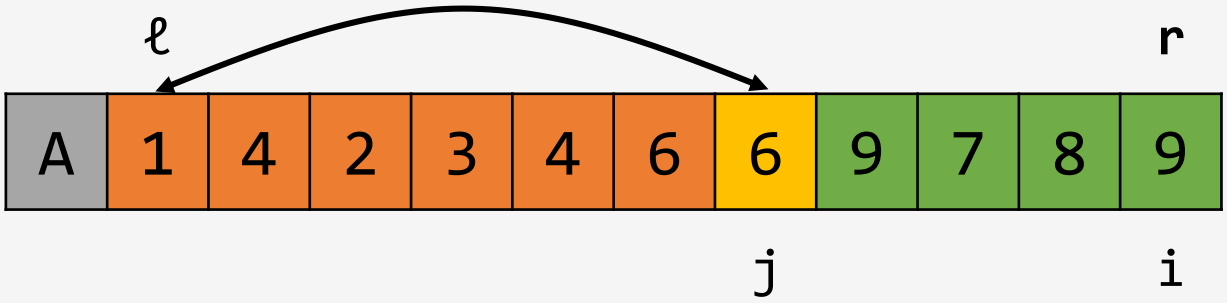
the pivot is x = A[ℓ]

move i from ℓ + 1 to r maintaining the following invariant:

A[k] ≤ x for all ℓ + 1 ≤ k ≤ j

A[k] > x for all j + 1 ≤ k ≤ I

in the end, move A[ℓ] to its final place



Partition

```
Partition(A, ℓ, r )
      x ← A[ℓ] {pivot}
      j ← ℓ
      for i from ℓ + 1 to r:
         if A[i] ≤ x:
            j ← j + 1
            swap A[j] and A[i]
         {A[ℓ + 1 . . . j] ≤ x, A[j + 1 . . . i] > x}
      swap A[ℓ] and A[j ]
      return j
```

Unbalanced Partitions

$$T(n) = n + T(n-1)$$

$$T(n) = n + (n-1) + (n-2) + \cdots = \theta(n^2)$$

$$T(n) = n + T(n-5) + T(4)$$

$$T(n) \geq n + (n-5) + (n-10) + \cdots = \theta(n^2)$$

Balanced Partitions

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

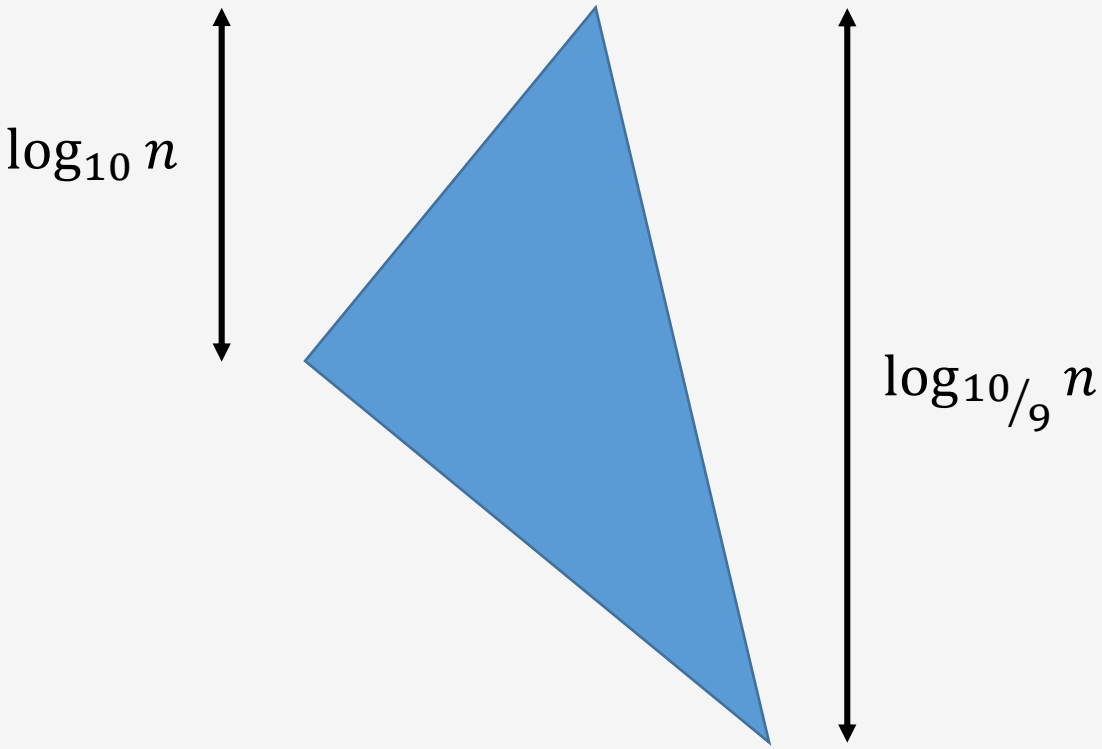
$$T(n) = \theta(n \log n)$$

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$

$$T(n) = \theta(n \log n)$$

Balanced Partitions

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + O(n)$$



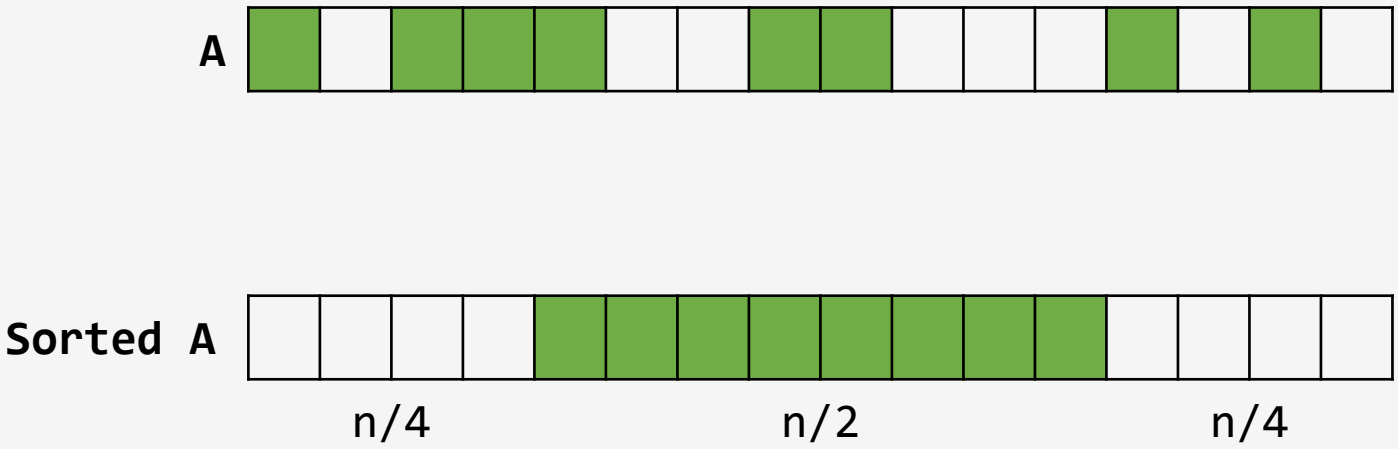
$$T(n) = O(n \log n)$$

Random Pivot

```
RandomizedQuickSort(A, ℓ, r )
      if ℓ ≥ r:
         return
      k ← random number between ℓ and r
      swap A[ℓ] and A[k]
      m ← Partition(A, ℓ, r )
      {A[m] is in the final position}
      RandomizedQuickSort(A, ℓ,m – 1)
      RandomizedQuickSort(A,m + 1, r )
```

Why Random?

half of the elements of A guarantees a balanced partition:



Theorem

Assume that all the elements of A[1 . . . n] are pairwise different. Then the average running time of RandomizedQuickSort(A) is O(n log n) while the worst-case running time is $O(n^2)$.

Remark

Averaging is over random numbers used by the algorithm, but not over the inputs.