Quick Sort

Ju-Won Seo
2019.07.17

Table of Contents

1. Introduction

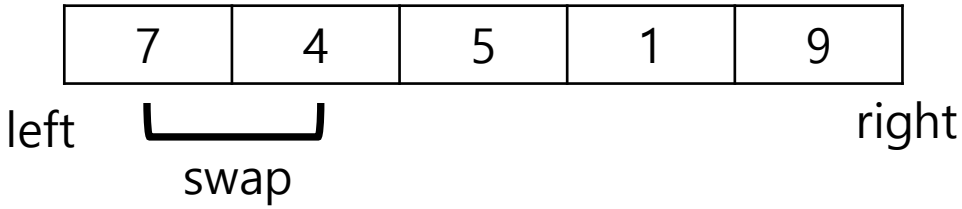
Different types of sorting

- **Bubble Sort**

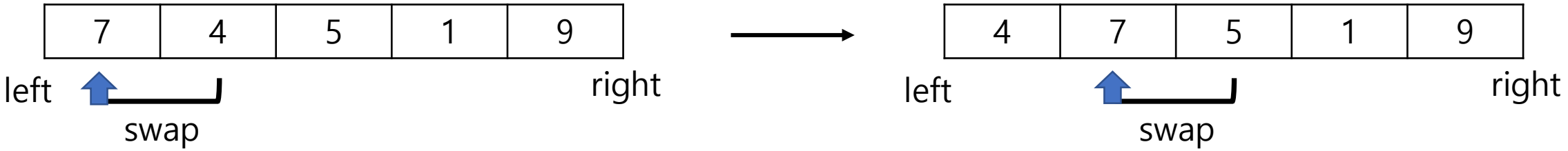
- **Merge Sort**

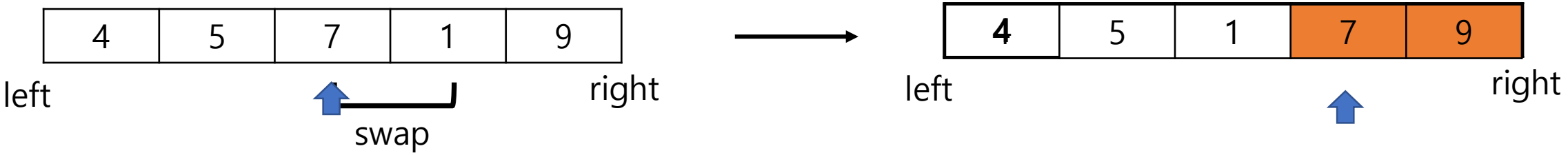
- Bubble Sort

- **Check and rearrange the two adjacent data.**
- **If data is not arranged in order, swap them.**
- T(n) = O(n^2)

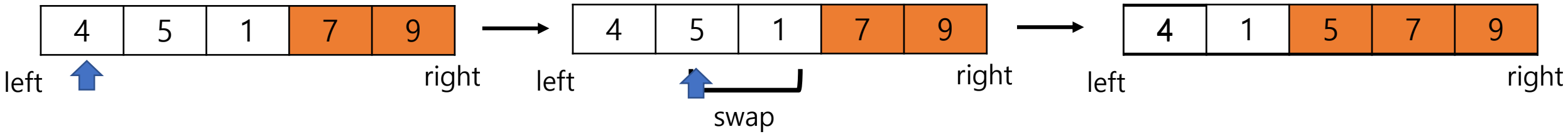


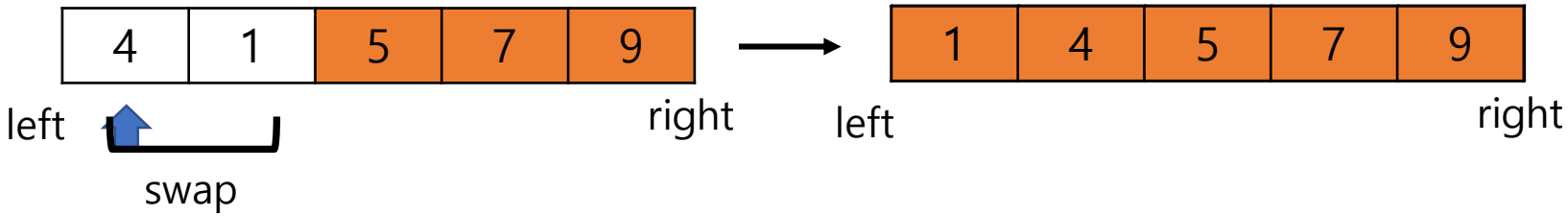
- Bubble Sort

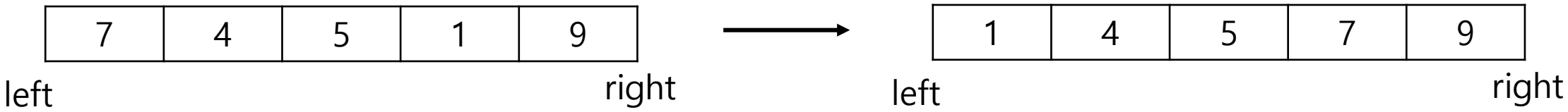

7
4
5
1
9
left
right
swap
4
7
5
1
9
left
right
swap


4
5
7
1
9
left
right
swap
4
5
1
7
9
left
right

- Bubble Sort


4 5 1 7 9
left
right
4 5 1 7 9
left
swap
right
4 1 5 7 9
left
right


4 1 5 7 9
left
swap
right
1 4 5 7 9
left
right


7 4 5 1 9
left
right
1 4 5 7 9
left
right

- Merge Sort

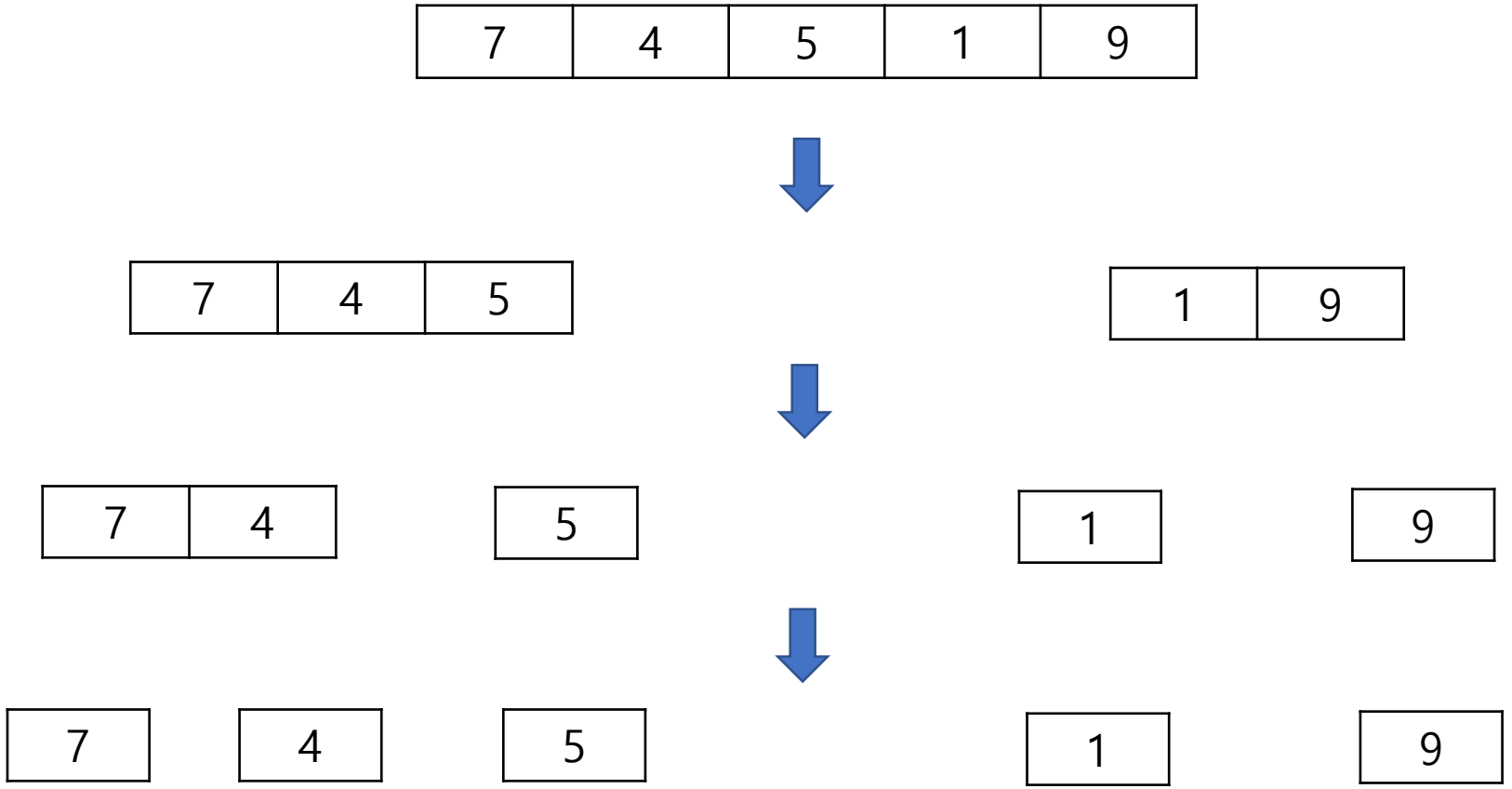
- **One of the divide and conquer algorithm.**

- **3 step process**

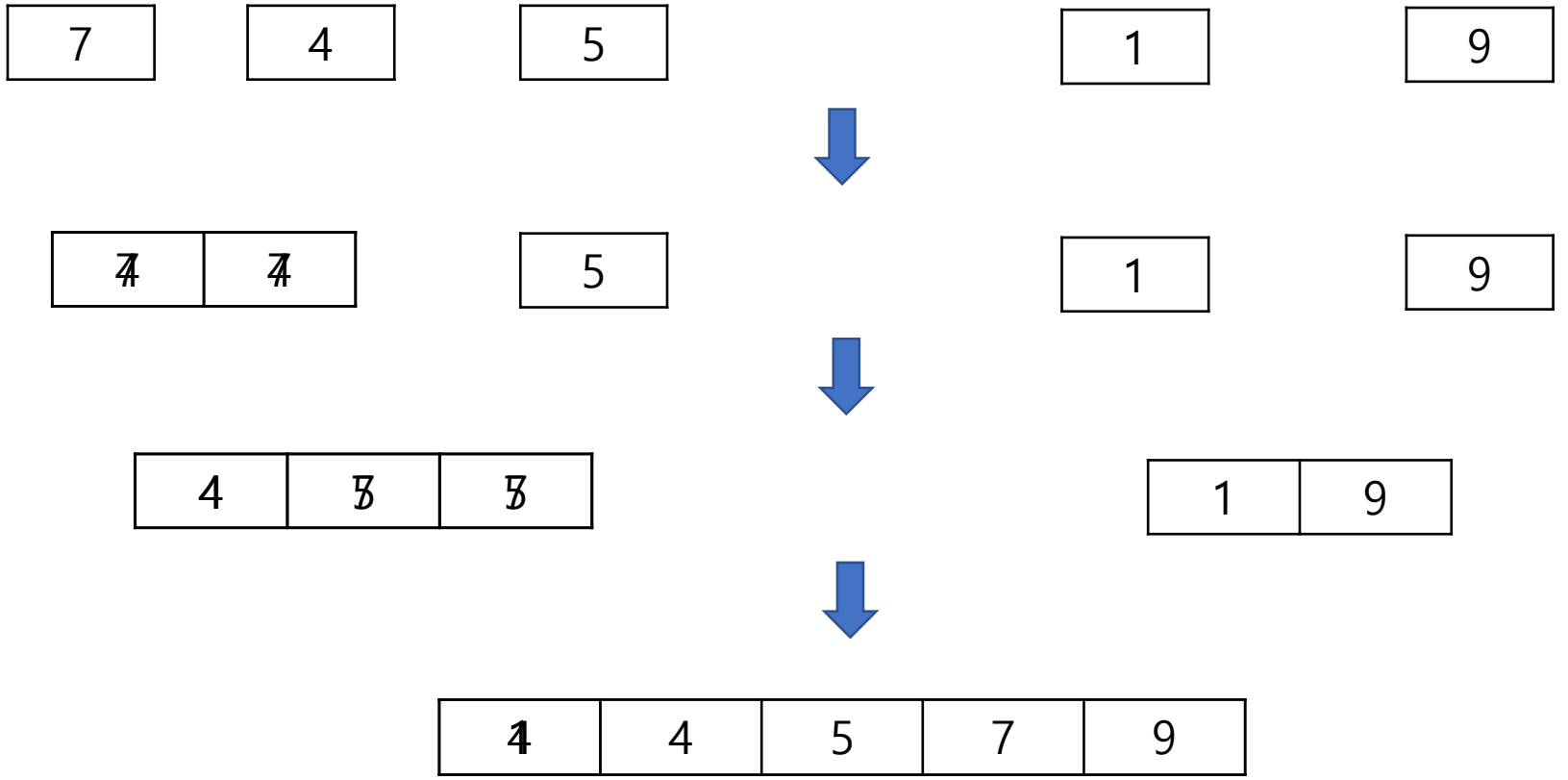
1. Divide : Divide unsorted list to two unsorted sub-list.
2. Conquer : Sort each sub-lists.
3. Combine : Merge the sorted sub-lists to one list.

- **To sorting we need a temporary list**.

Merge Sort
(Divide)



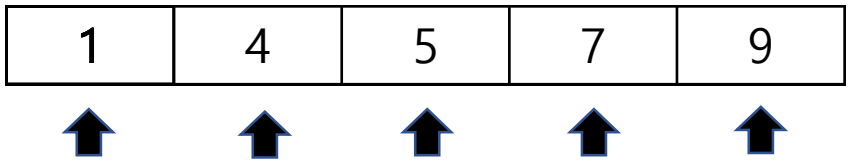
Merge Sort
(Conquer & Combine)



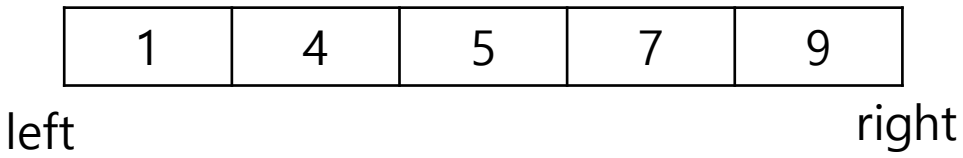
Merge Sort
(Conquer & Combine)

Two sub-lists

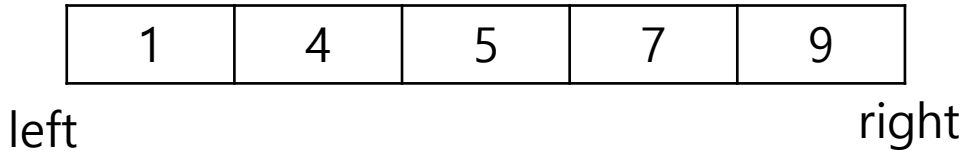
Temporary list



Temporary list



Original list



Time Complexity : $T(n) = 2T(\frac{n}{2}) + \theta(n)$

$= O(nlogn)$

2. Quick Sort

- Quick Sort

- **One of the divide and conquer algorithm.**

- **A method of sorting known to be very fast on average.**

- **3 step process**

1. Divide : Divide unsorted list to two unsorted sub-list.
2. Conquer : Sort each sub-lists.
3. Combine : There is nothing to do.

- Quick Sort

```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)
```

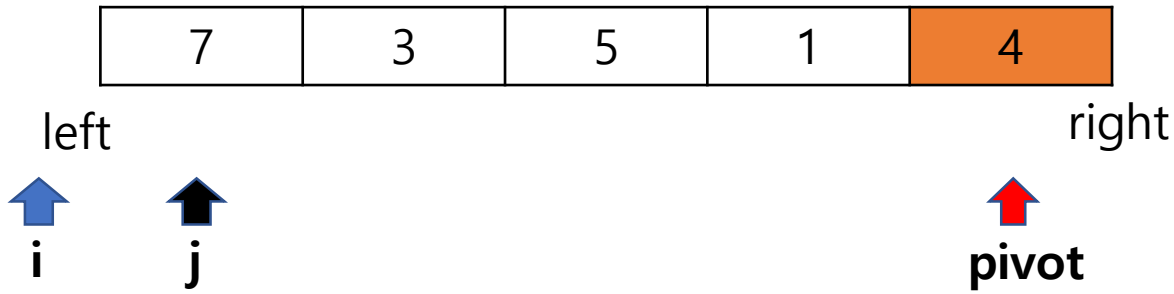
```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Quick Sort

pivot **Reference Index**

j **Current Index**

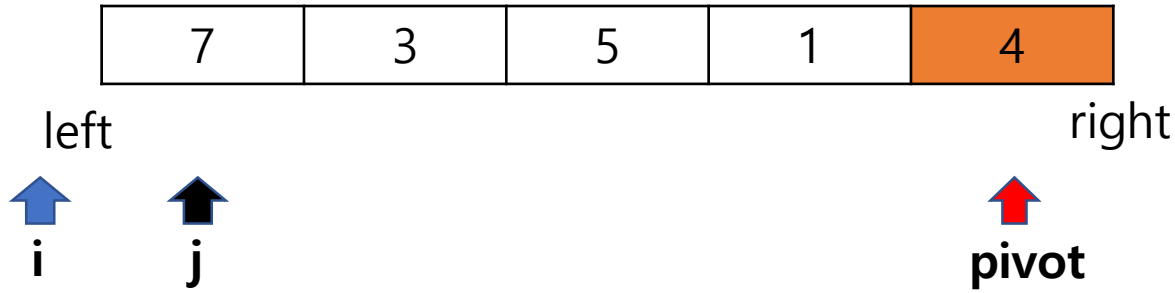
i **To be swapped Index**



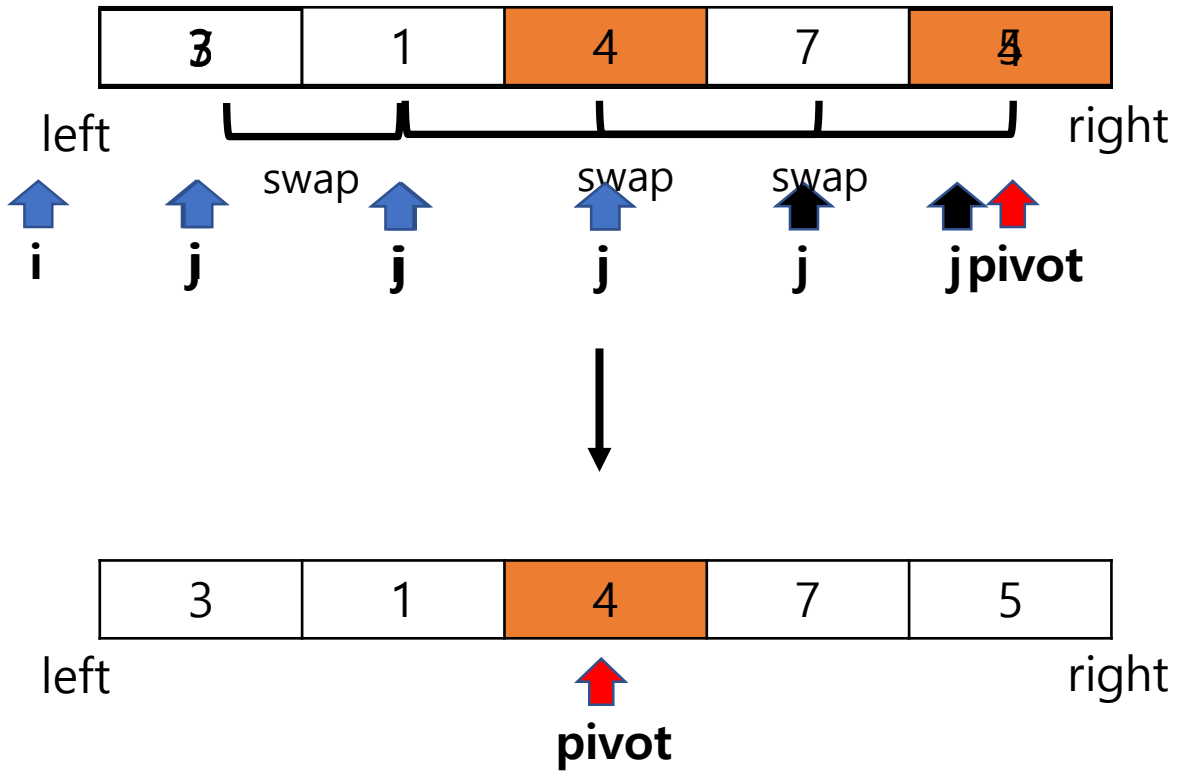
Quick Sort

Sorting process

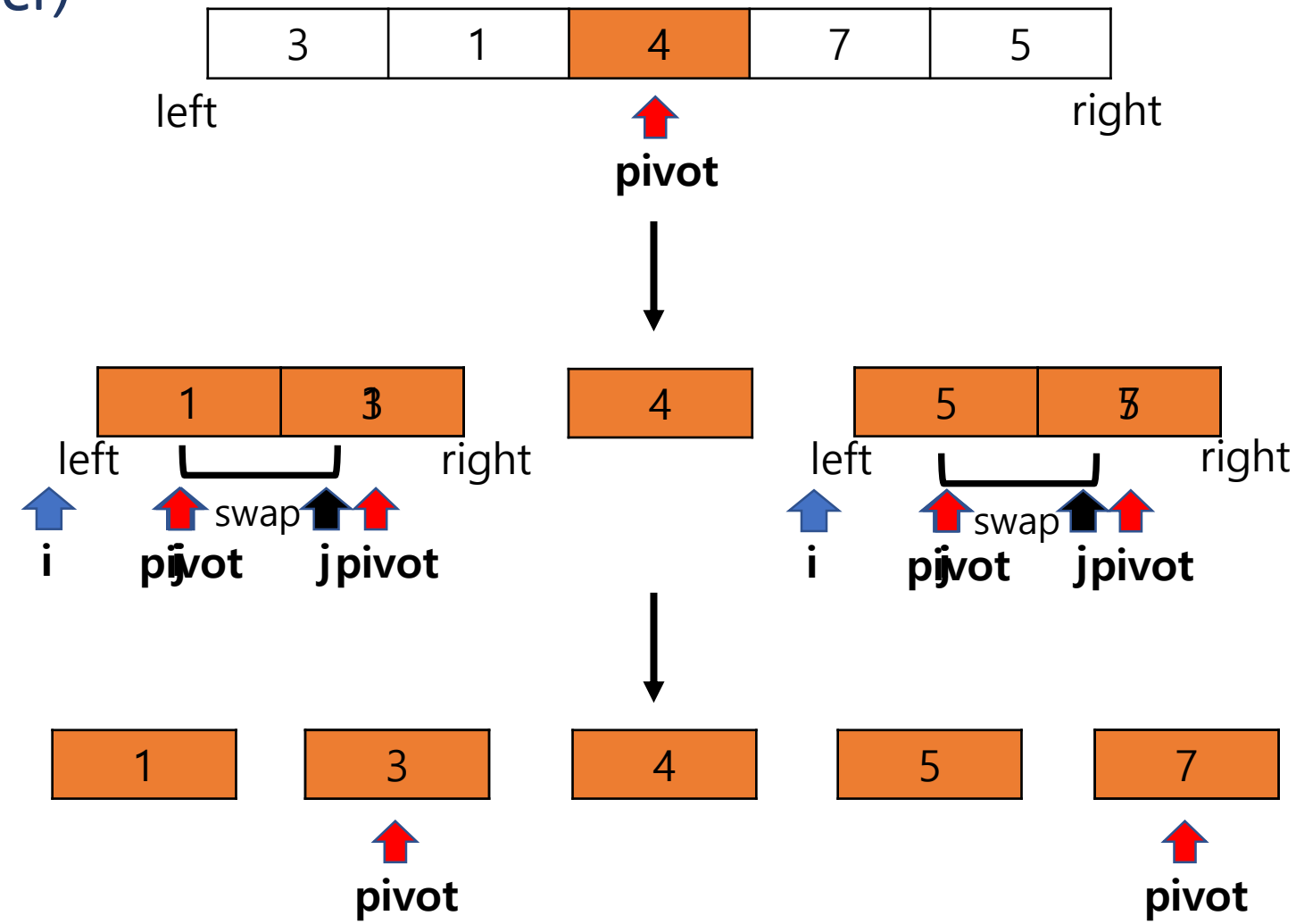
1. Initializing : i -> left – 1
   j -> left
2. Pivot index is each list's right index.
3. If current index's value smaller than pivot value, increase i index and swap(i, j).
4. When j==pivot, increase i index and then swap(i, pivot).



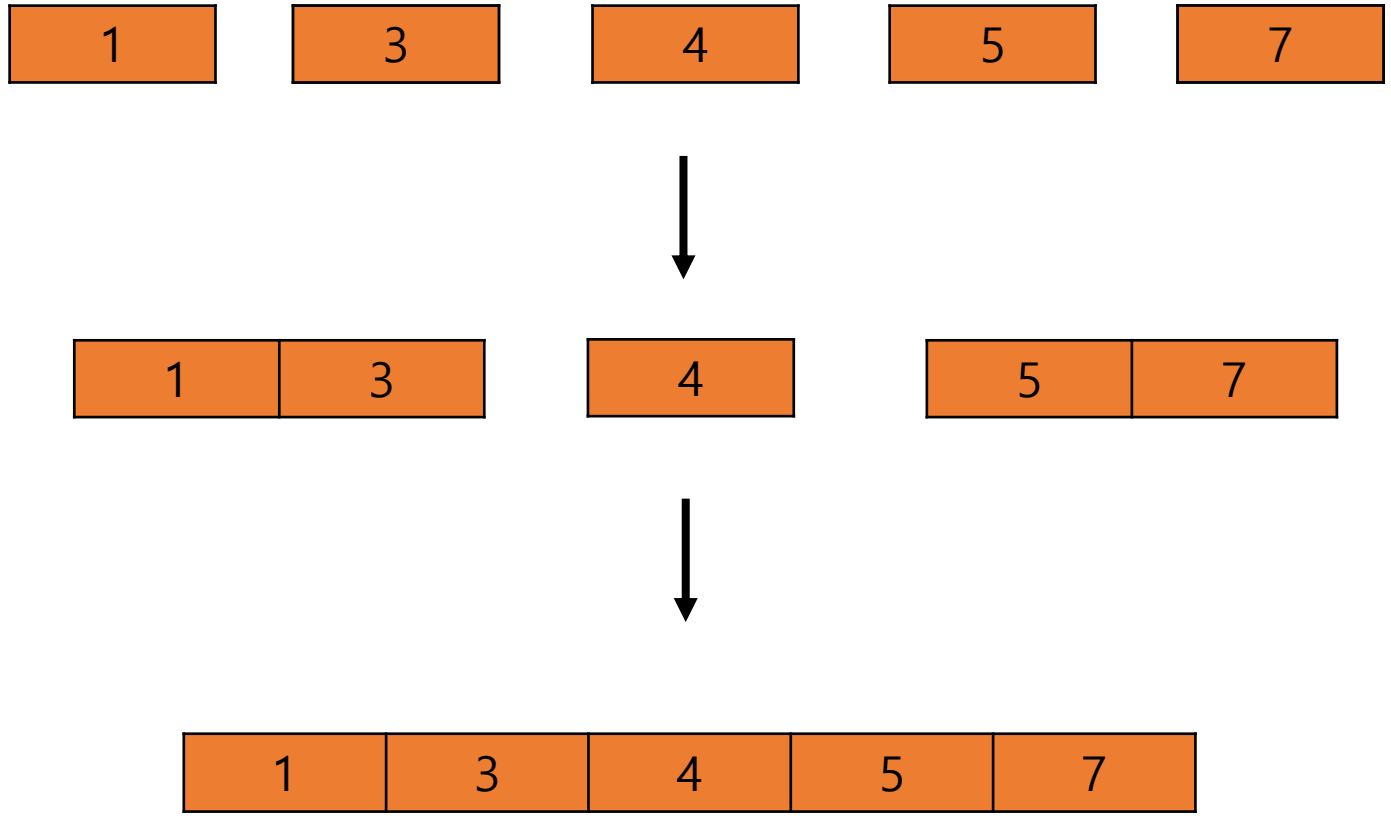
Quick Sort
(Conquer)



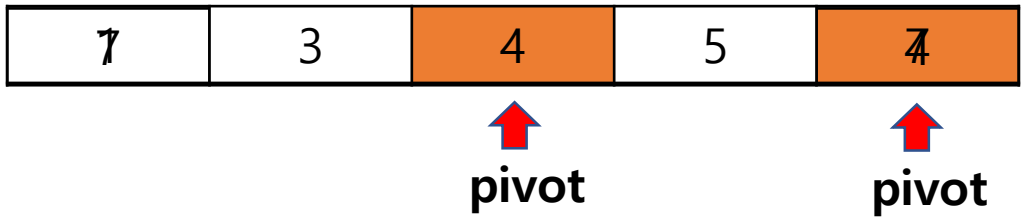
Quick Sort
(Divide & Conquer)

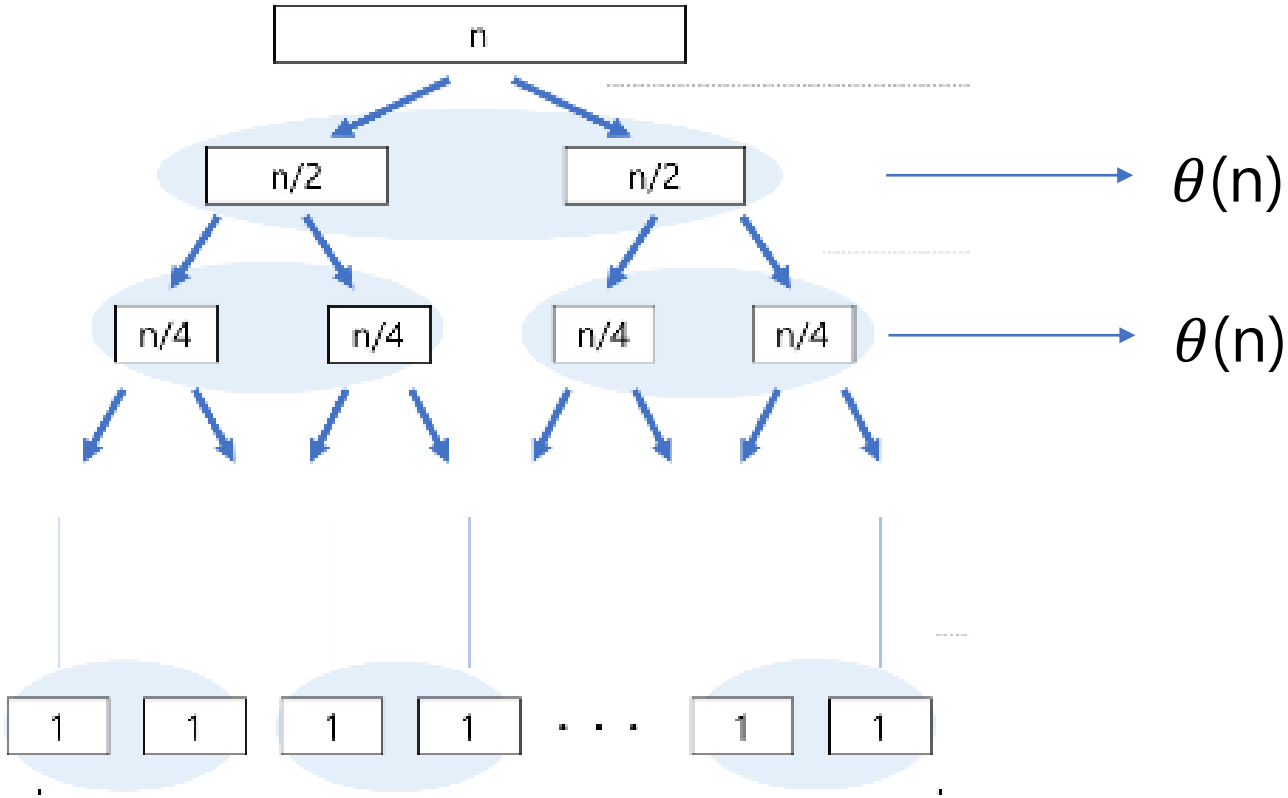


Quick Sort
(Combine)



Analysis in best case





$\theta(n)$

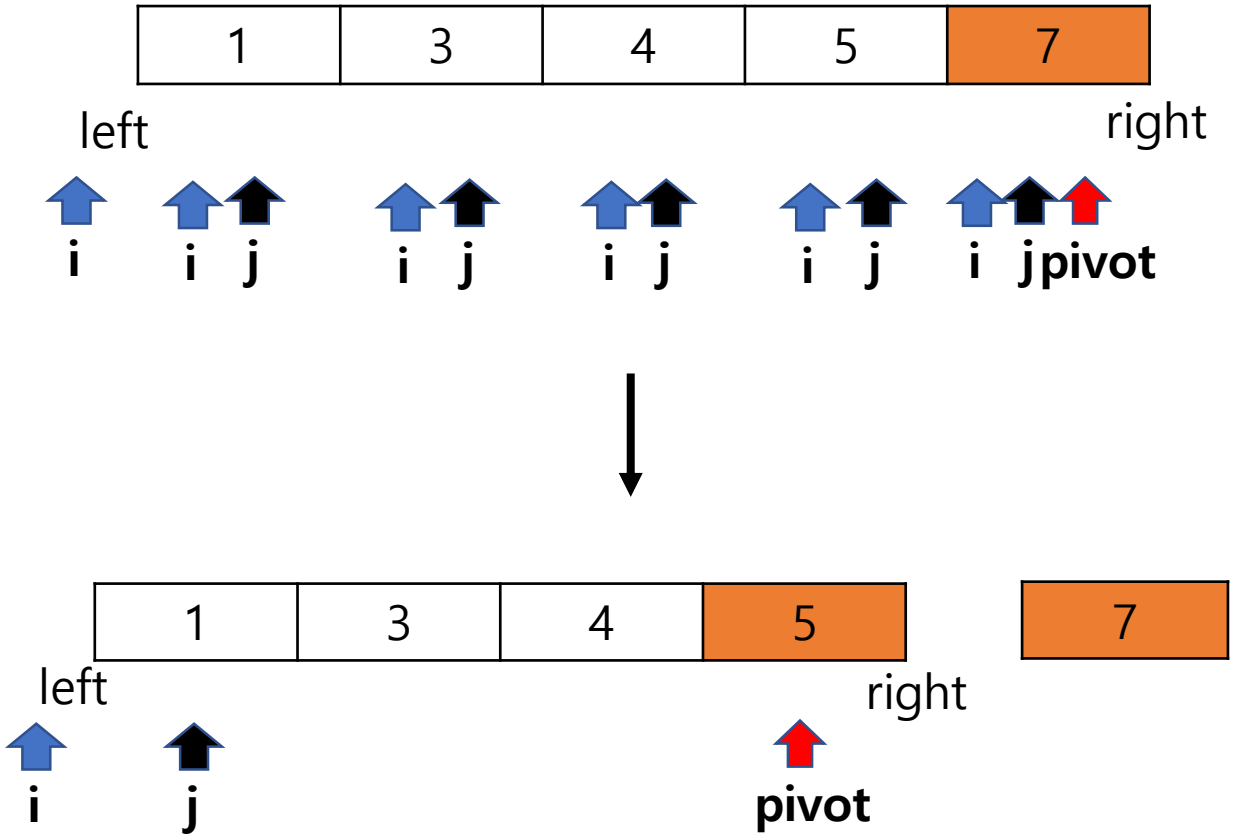
$\theta(n)$

$$T(n) = 2T(n/2) + f(n)$$

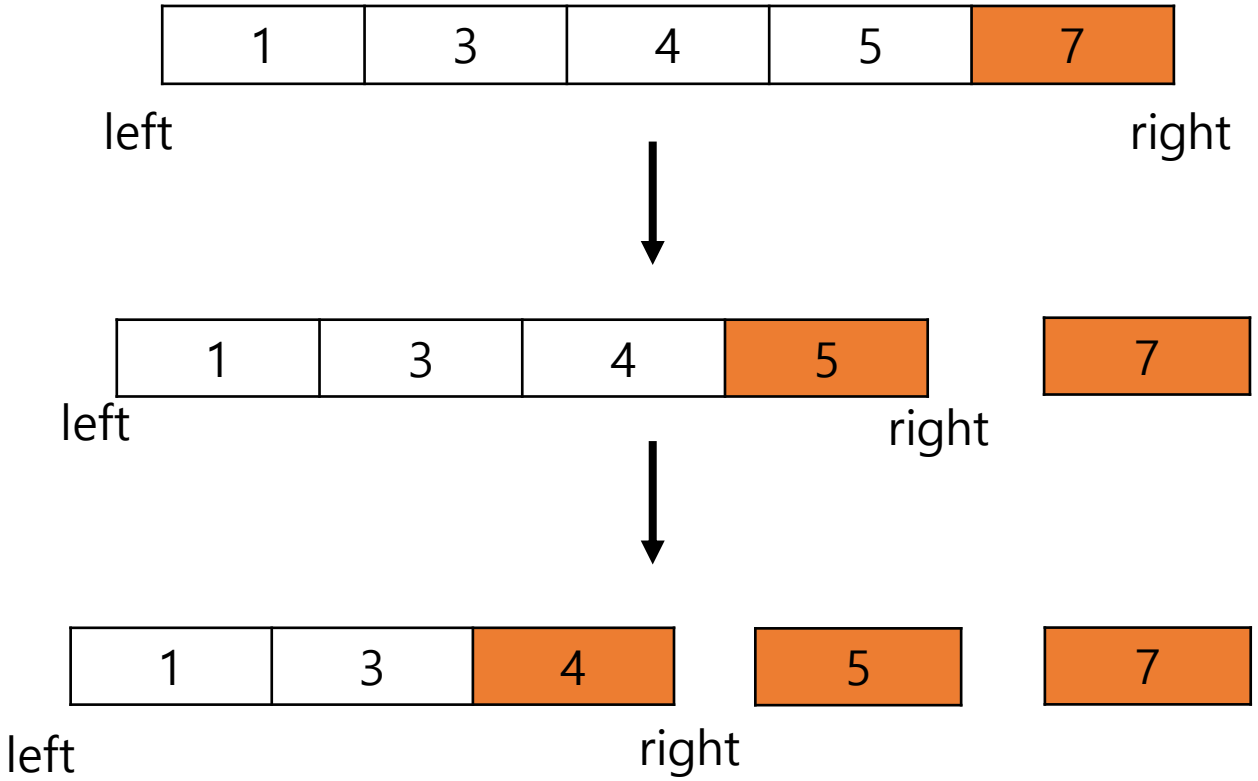
$$= 2T(n/2) + \theta(n)$$

$$= \theta(nlogn)$$

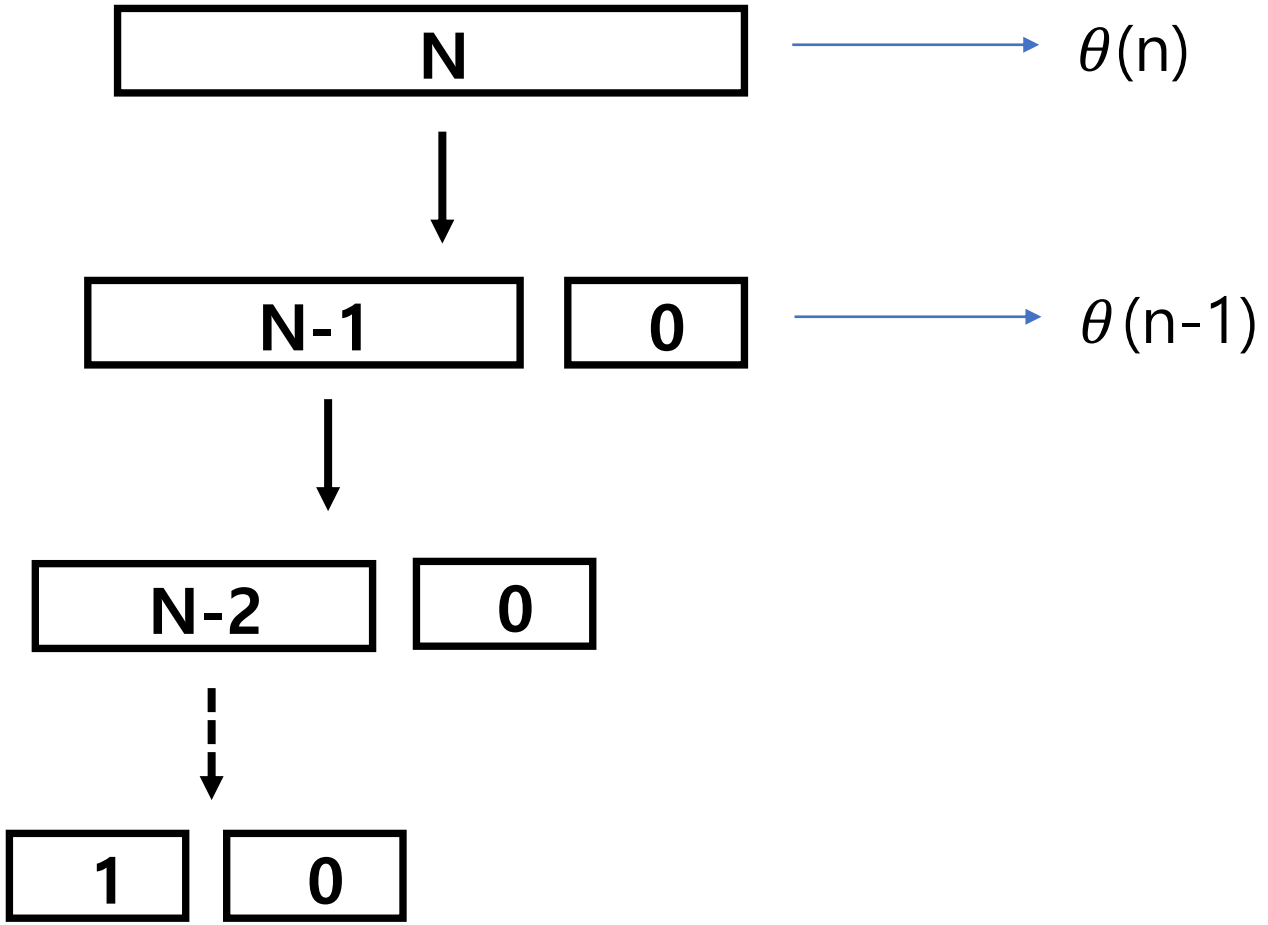
Problem


1
3
4
5
7
left
right
i
i j
i j
i j
i j
i j pivot
1
3
4
5
7
left
right
i
j
pivot

Problem



Analysis in worst case



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

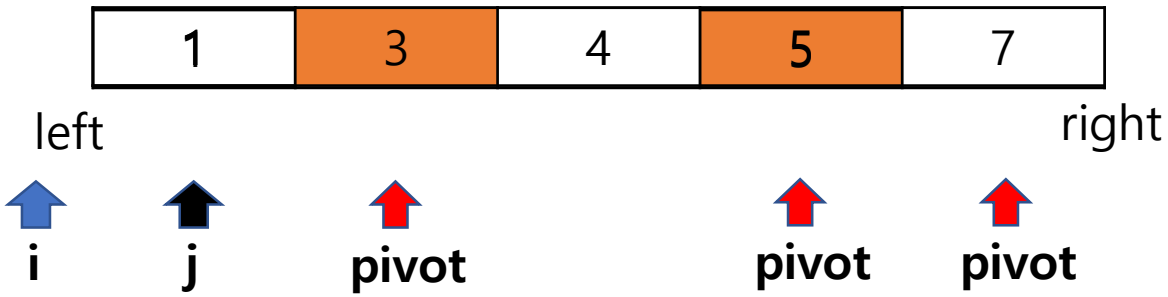
$= T(n-1) + T(n-2) + \theta(n) + \theta(n-1)$

$= \ldots\ldots\ldots$

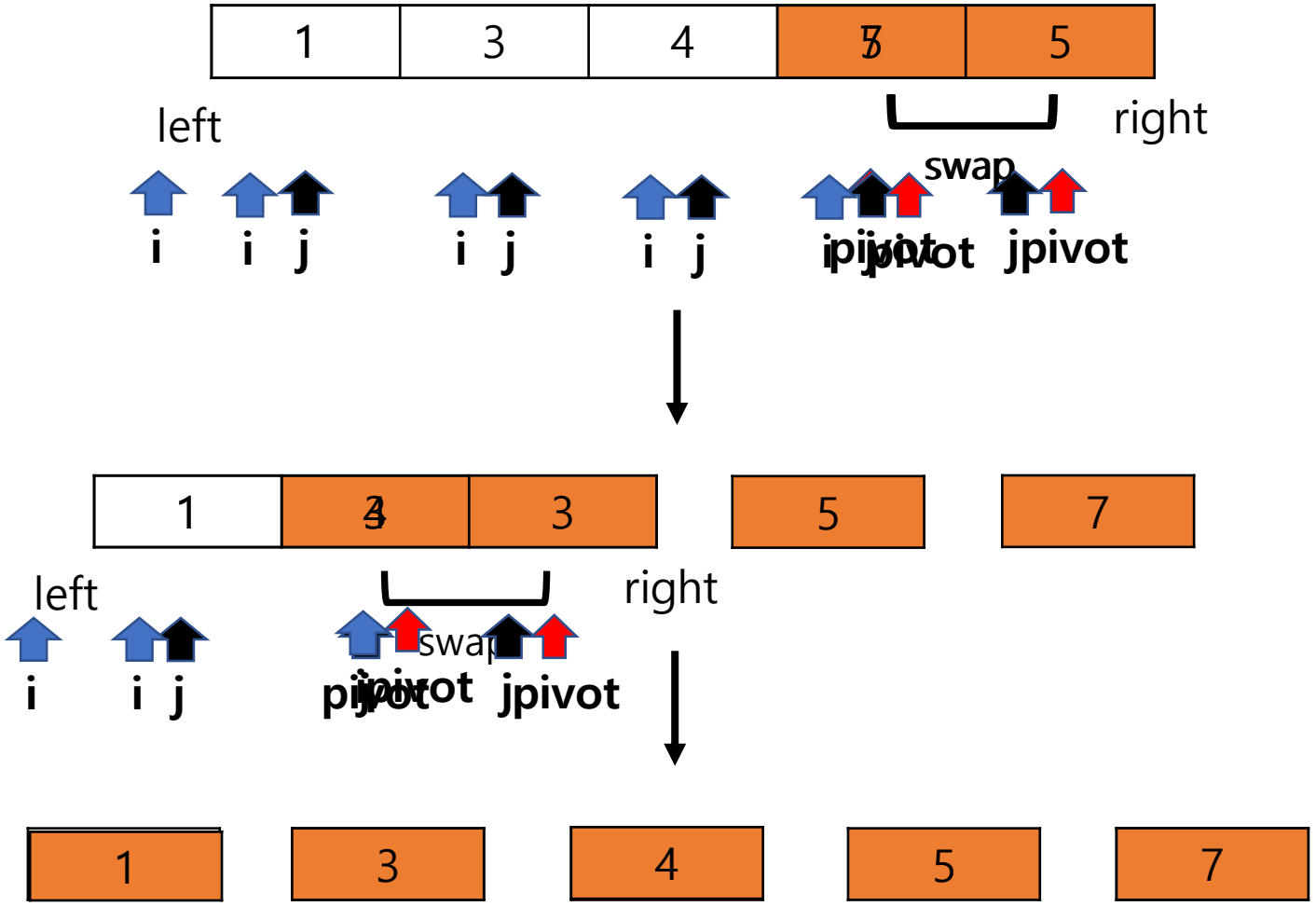
$= \theta(1) + \theta(2) + \ldots. + \theta(n-1) + \theta(n)$

$= \theta(n^2)$

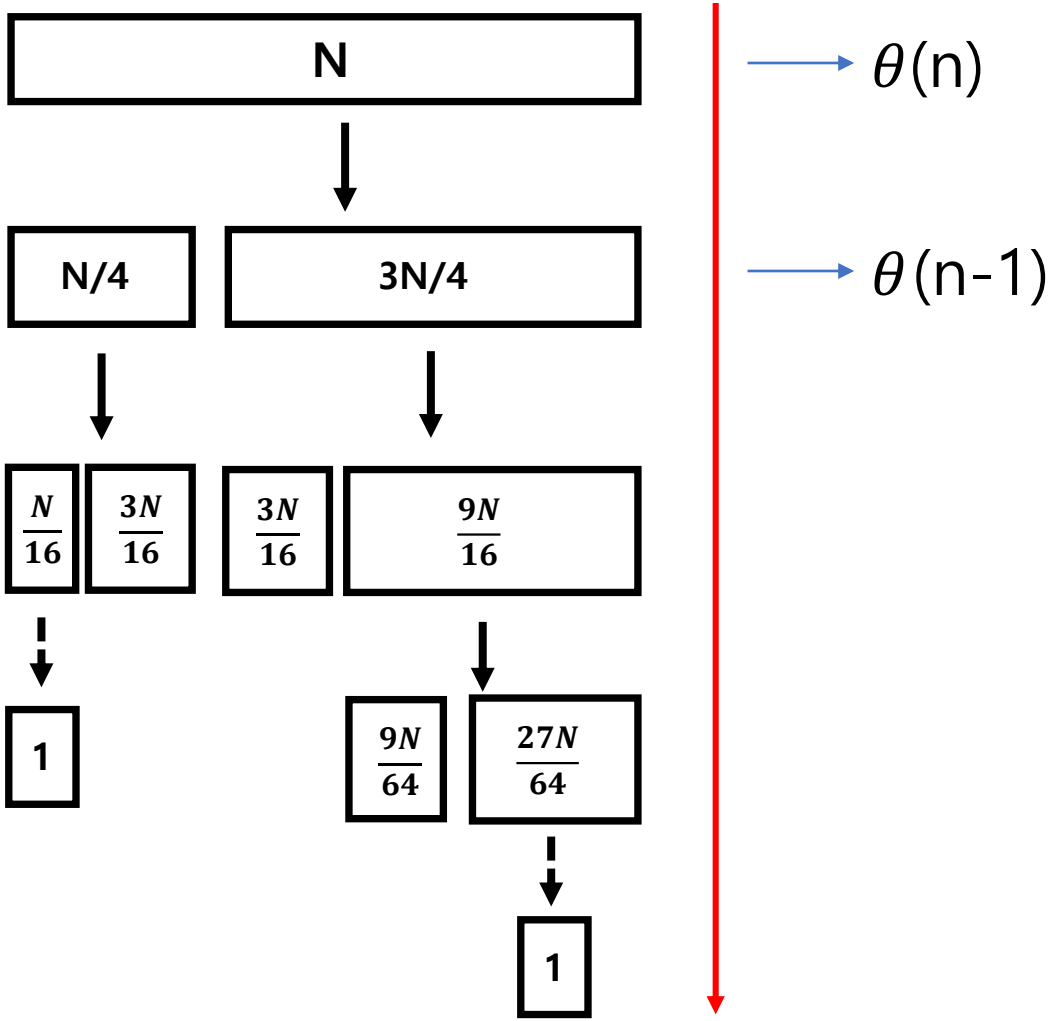
Solution-Randomized



Randomized algorithm



Analysis in not worst case



Longest depth

$(\frac{3}{4})^k$*n = 1 (k is recursive depth)

n = $(\frac{4}{3})^k$

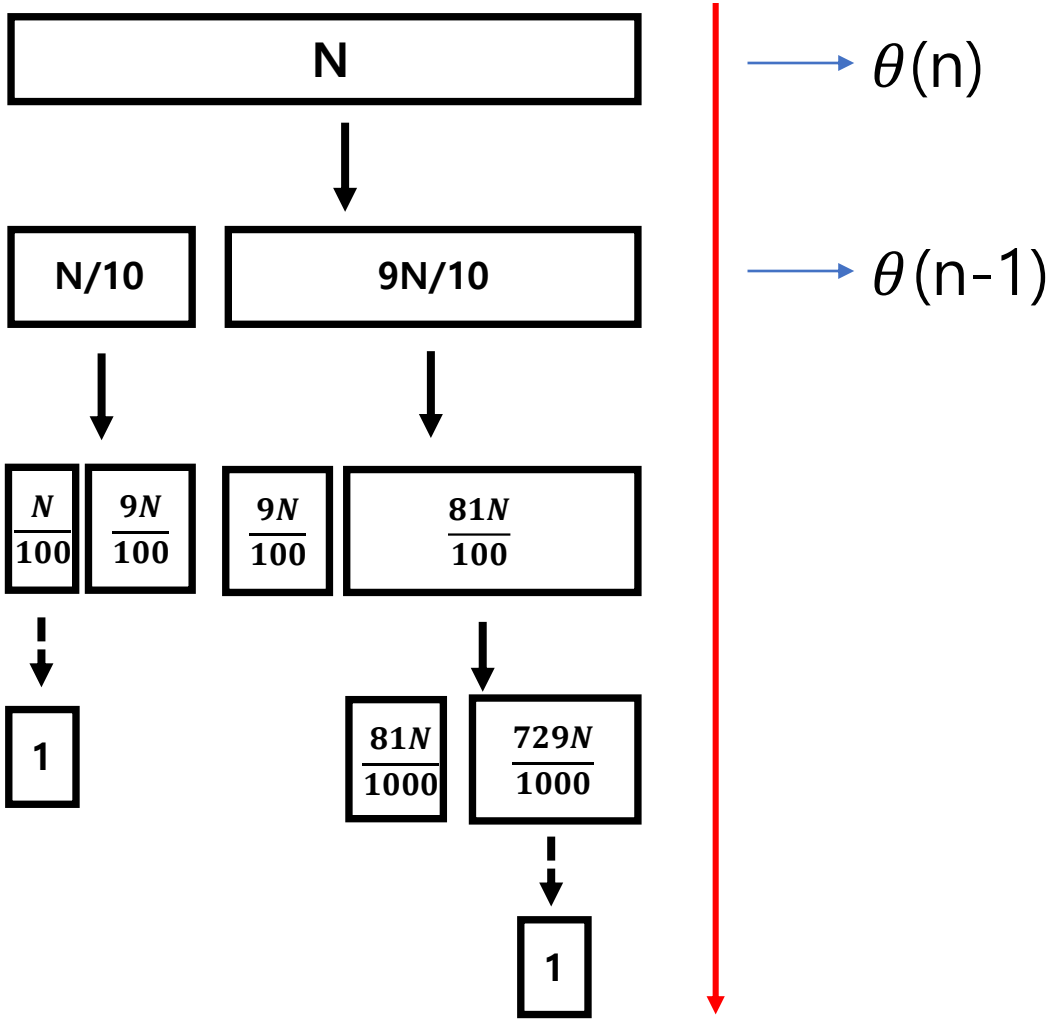
$K = \log_{\frac{4}{3}} n = \frac{\log_2 n}{\log_2 \frac{4}{3}} = c\log_2 n$

$\therefore k = \theta(\log n)$

f(n) = θ(n) -> Linear time for each partition

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

Analysis in not worst case



θ(n)

θ(n-1)

Longest depth

$(\frac{9}{10})^k * n = 1$ (k is recursive depth)

$n = (\frac{10}{9})^k$

$K = \log_{\frac{10}{9}} n = \frac{\log_2 n}{\log_2 \frac{10}{9}} = c\log_2 n$

$\therefore k = \theta(\log n)$

f(n) = θ(n) -> Linear time for each partition

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

3. Conclusion

Summary

Time complexity in worst case **=>** **O(n^2)**

Time complexity in average case (including best case) **=>** $\boldsymbol{\theta(n \log n)}$

Conclusion

- **Advantages**

- It is fast compared to several sorting algorithms such as bubble sort in not worst case.

- Unlike merge sort, quick sort is no need a new list in combine level.

- **Disadvantages**

- For an ordered list, the unequally division of quick sort rather takes more time to perform.

Q & A

Thank you!