# Quicksort

Slides from Heejin Park

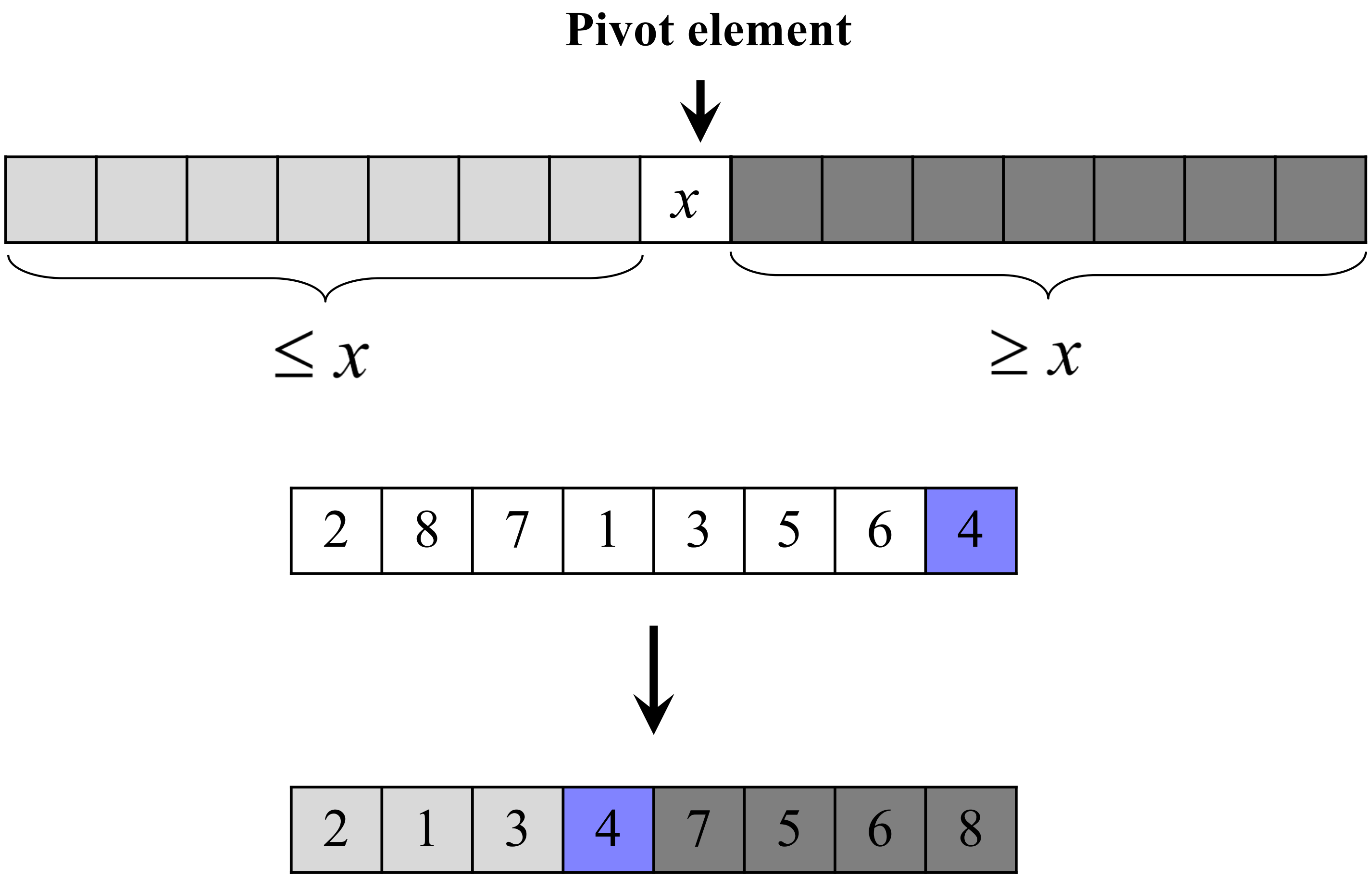
# Contents

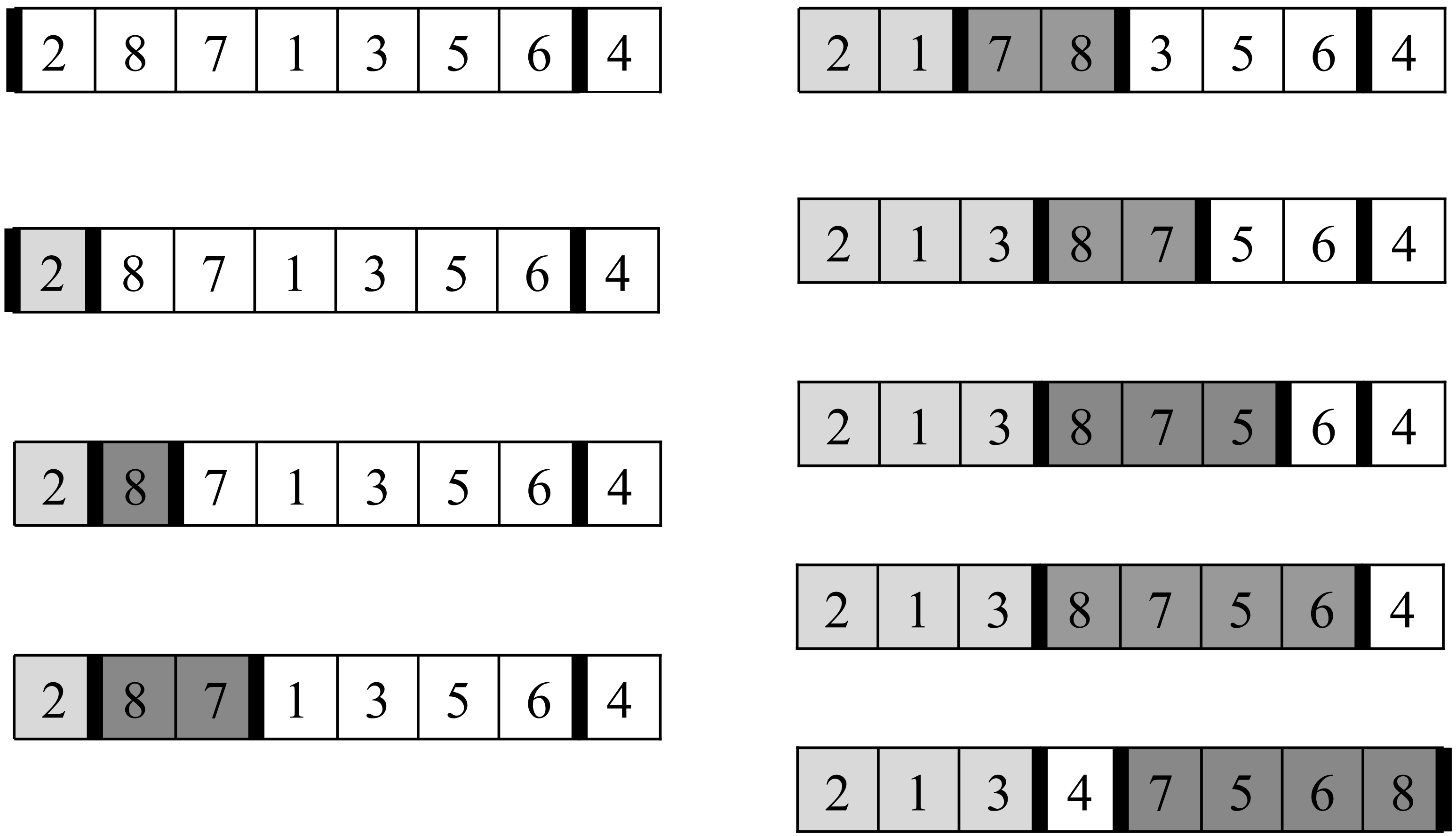
- **Quicksort**
- **Randomized quicksort**

- **Divide-and-Conquer paradigm**

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

- **Partition**




2 8 7 1 3 5 6 4
2 8 7 1 3 5 6 4
2 8 7 1 3 5 6 4
2 8 7 1 3 5 6 4
2 1 7 8 3 5 6 4
2 1 3 8 7 5 6 4
2 1 3 8 7 5 6 4
2 1 3 8 7 5 6 4
2 1 3 4 7 5 6 8

# Partition

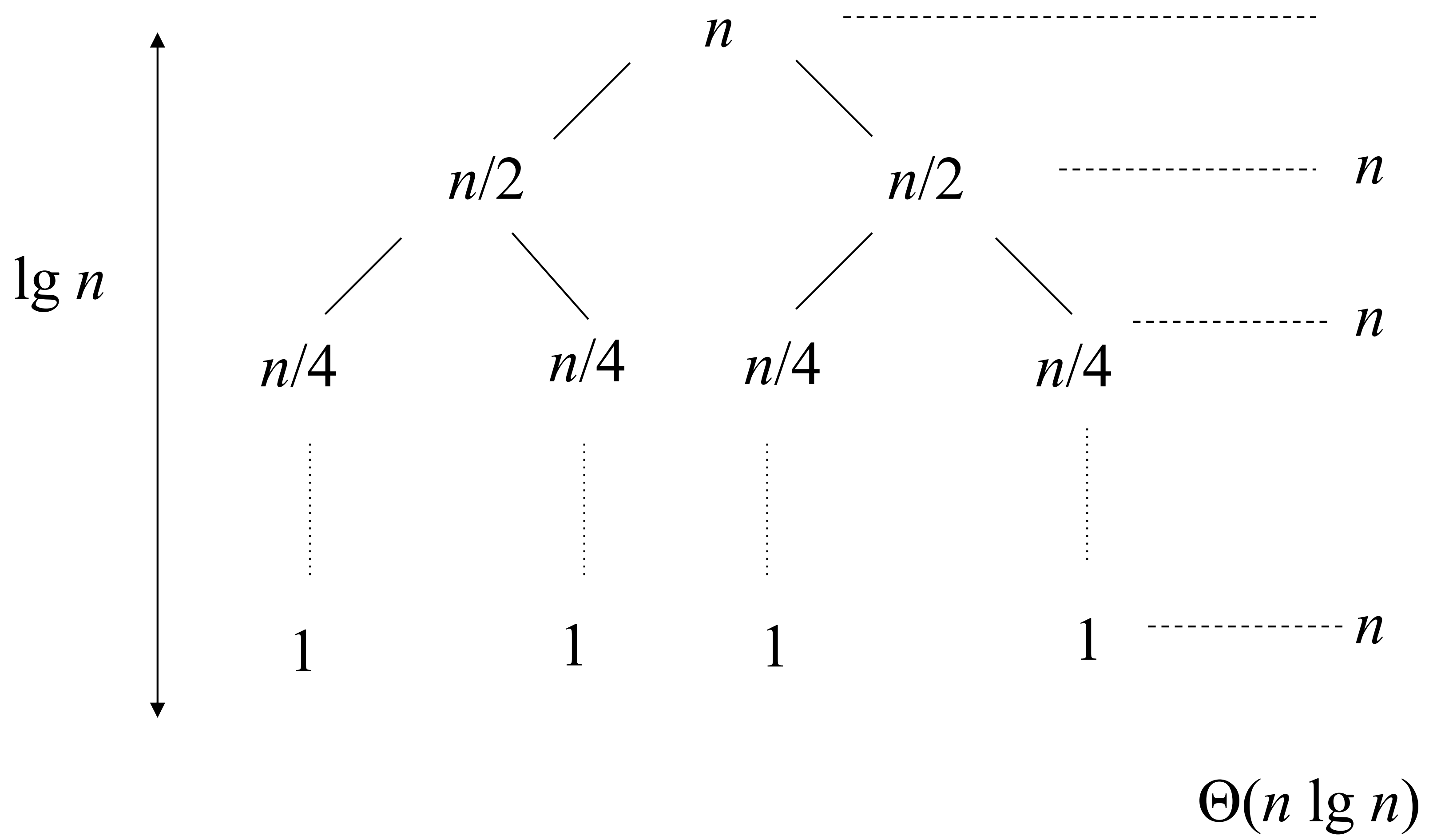
```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to  r − 1
    do if A[j] ≤ x
        then  i = i + 1
              exchange A[i] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return  i + 1
```

- **Partition**
  - $\Theta(n)$ time.

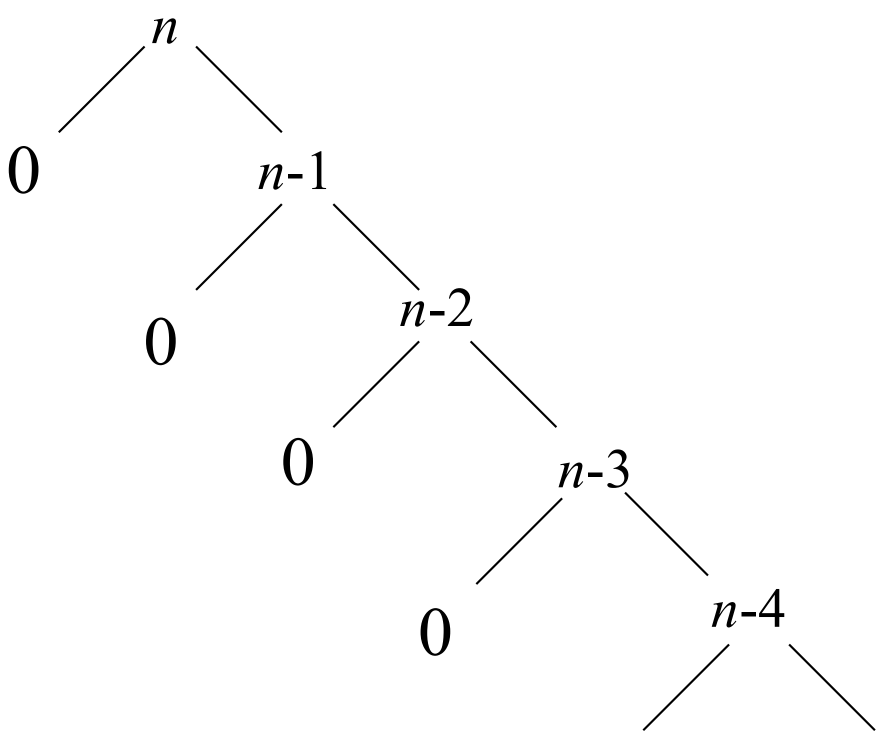
- ***Balanced partitioning*** vs. ***unbalanced partitioning***

- **Balanced partitioning**
  - When PARTITION produces two subproblems of sizes $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil - 1$.
  - $T(n) \le 2T(n/2) + \Theta(n) = O(n \lg n)$

- **Balanced partitioning**



- **Unbalanced partitioning**



- **Unbalanced partitioning**

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

$$= \sum_{k=1}^{n} \Theta(k)$$

$$= \Theta\left(\sum_{k=1}^{n} k\right)$$

$$= \Theta(n^2)$$

# Worst-case Analysis

- **Worst-case analysis**
  - Quicksort takes $\Omega(n^2)$ time in worst case.
    - Consider the unbalanced partitioning.
  - Is the unbalanced partitioning the worst case?

- **Worst-case analysis**
  - Show that the running time of quicksort is $O(n^2)$ by substitution method.

$$T(n) = \max_{0 \le q \le n-1} \big(T(q) + T(n-q-1)\big) + \Theta(n)$$

  - Show that $T(n) \le cn^2$ for some constant $c$.

$$T(n) \le \max_{0 \le q \le n-1} (cq^2 + c(n-q-1)^2) + \Theta(n)$$

$$= c \cdot \max_{0 \le q \le n-1} (q^2 + (n-q-1)^2) + \Theta(n)$$

$$= c \cdot \max_{0 \le q \le n-1} (2q^2 - 2q(n-1) + (n-1)^2) + \Theta(n)$$

$$= c \cdot \max_{0 \le q \le n-1} (2(q-(n-1)/2)^2 + (n-1)^2/2) + \Theta(n)$$

- **Worst-case analysis**
  - The internal expression is maximized when $q = 0$ or $n - 1$.

$$T(n) \le c \cdot \max_{0 \le q \le n-1} \left( 2\left(q - \frac{n-1}{2}\right)^2 + \frac{(n-1)^2}{2} \right) + \Theta(n)$$
$$= c \cdot (n-1)^2 + \Theta(n)$$
$$= cn^2 - c(2n-1) + \Theta(n)$$
$$\le cn^2$$

  - We can pick the constant $c$ large enough so that the $c(2n - 1)$ term dominates the $\Theta(n)$ term.
  - Thus, $T(n) = O(n^2)$.

- **Average-case analysis**

$$E[T(n)] = \frac{1}{n}\left(\sum_{q=1}^{n}(E[T(q-1)] + E[T(n-q)])\right) + \Theta(n)$$

$$= \frac{2}{n}\sum_{q=0}^{n-1} E[T(q)] + \Theta(n)$$

  – By substitution method, show $T(n) \le cn\lg n$ for some $c$.
    - Problem 7-3.

- **Average Case Analysis II**
  - Let $X$ be the number of comparisons over the entire execution of QUICKSORT on an $n$-element array.
  - Then the average running time of QUICKSORT is
    - $O(n + \mathrm{E}[X])$.
  - We will not attempt to analyze how many comparisons are made in *each* PARTITION.
  - Rather, we will derive an overall bound on the total number of comparisons.

- Let $z_i$ denote the $i$th smallest element in the sorted array.
- $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$
- Each pair of elements $z_i$ and $z_j$ is compared at most once.
  - An element is compared only to the pivot element in each PARTITION.
  - The pivot element used in a PARTITION is never again compared to any other elements.

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ is compared to } z_j\}$$

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr\{z_i \text{ is compared to } z_j\}$$

- $\Pr\{z_i$ is compared to $z_j\}$
  - $\Pr\{z_i$ or $z_j$ is the first pivot chosen from $Z_{ij}\}$

$$= \frac{2}{j-i+1}$$

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$k = j - i$, the harmonic series

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$$= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1}$$

$$< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k}$$

$$= \sum_{i=1}^{n-1} O(\lg n) \qquad \text{Equation (A.7)}$$

$$= O(n \lg n)$$

RANDOMIZED-PARTITION($A$, $p$, $r$)

1. $i$ = RANDOM($p$, $r$)
2. exchange $A[r]$ with $A[i]$
3. **return** PARTITION($A$, $p$, $r$)

# Randomized quicksort

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

- **Exercise 7.1-2**
  - Balanced partition with same elements
- **Exercise 7.2-4**
  - Sorting almost-sorted input