Quicksort

Heejin Park

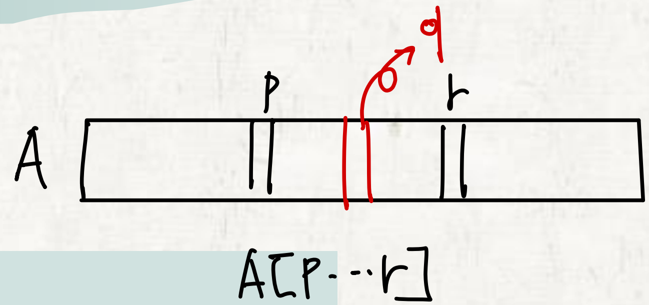
Hanyang University

Contents

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

Quicksort

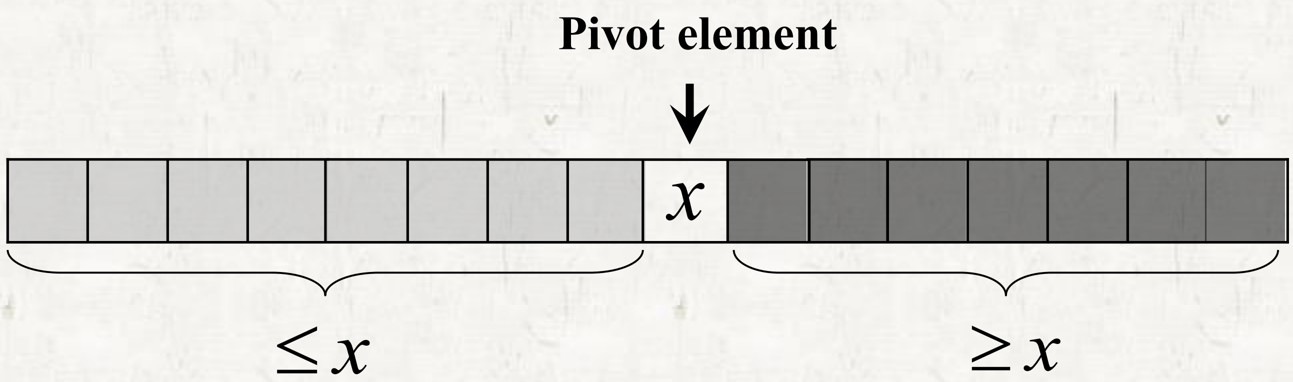
- **Divide-and-Conquer paradigm**




```
QUICKSORT(A, p, r)
 if p < r
pivot    q = PARTITION(A, p, r)
         QUICKSORT(A, p, q - 1)  LEFT
         QUICKSORT(A, q + 1, r)  RIGHT
```

Quicksort

- **Partition**



Input

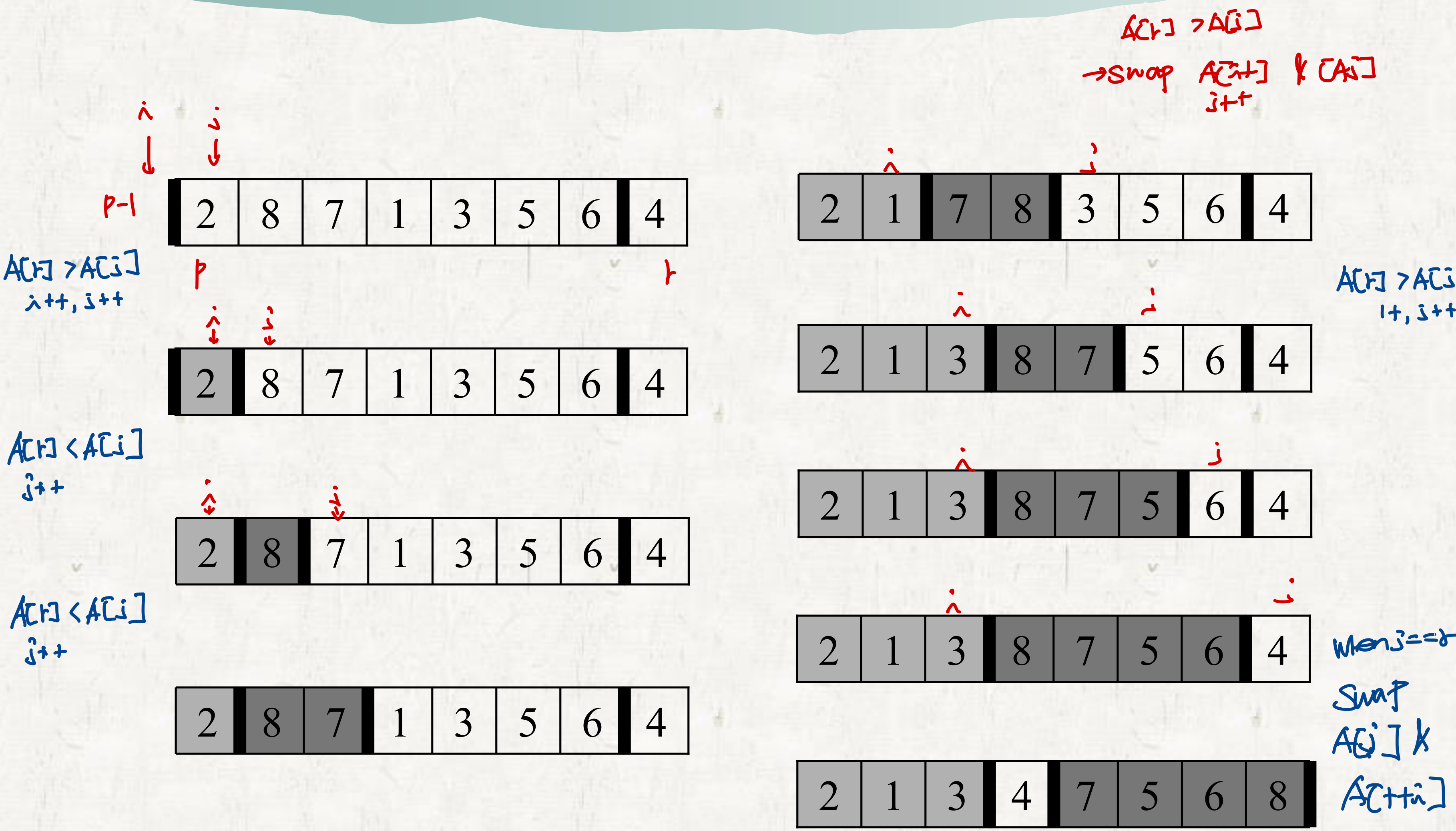
2	8	7	1	3	5	6	4

pivot

partitioning

2	1	3	4	7	5	6	8

Quicksort



A[r] > A[j]
→ swap A[i+1] & [A[j]]
j++
i
j
p-1
2 8 7 1 3 5 6 4
p
r
A[r] > A[j]
i++, j++
2 8 7 1 3 5 6 4
A[r] < A[j]
j++
2 8 7 1 3 5 6 4
A[r] < A[j]
j++
2 8 7 1 3 5 6 4
2 1 7 8 3 5 6 4
A[r] > A[j]
i++, j++
2 1 3 8 7 5 6 4
2 1 3 8 7 5 6 4
2 1 3 8 7 5 6 4
when j==r
swap
A[j] &
A[++i]
2 1 3 4 7 5 6 8

Quicksort

- **Partition**

q is the index of the pivot after pivoting

PARTITION(A, p, r)
$x = A[r]$ → x is pivot
$i = p - 1$
for $j = p$ **to** $r - 1$ → $\Theta(r-p) = \Theta(n)$
    if $A[j] \le x$
        $i = i + 1$
        exchange $A[i]$ with $A[j]$

swap A[++i] & A[j]

exchange $A[i+1]$ with $A[r]$
return $i+1$

equal to q

Quicksort

- **Partition**
  - $\Theta(n)$ time. → $n = p - r$ (size of partition array)

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

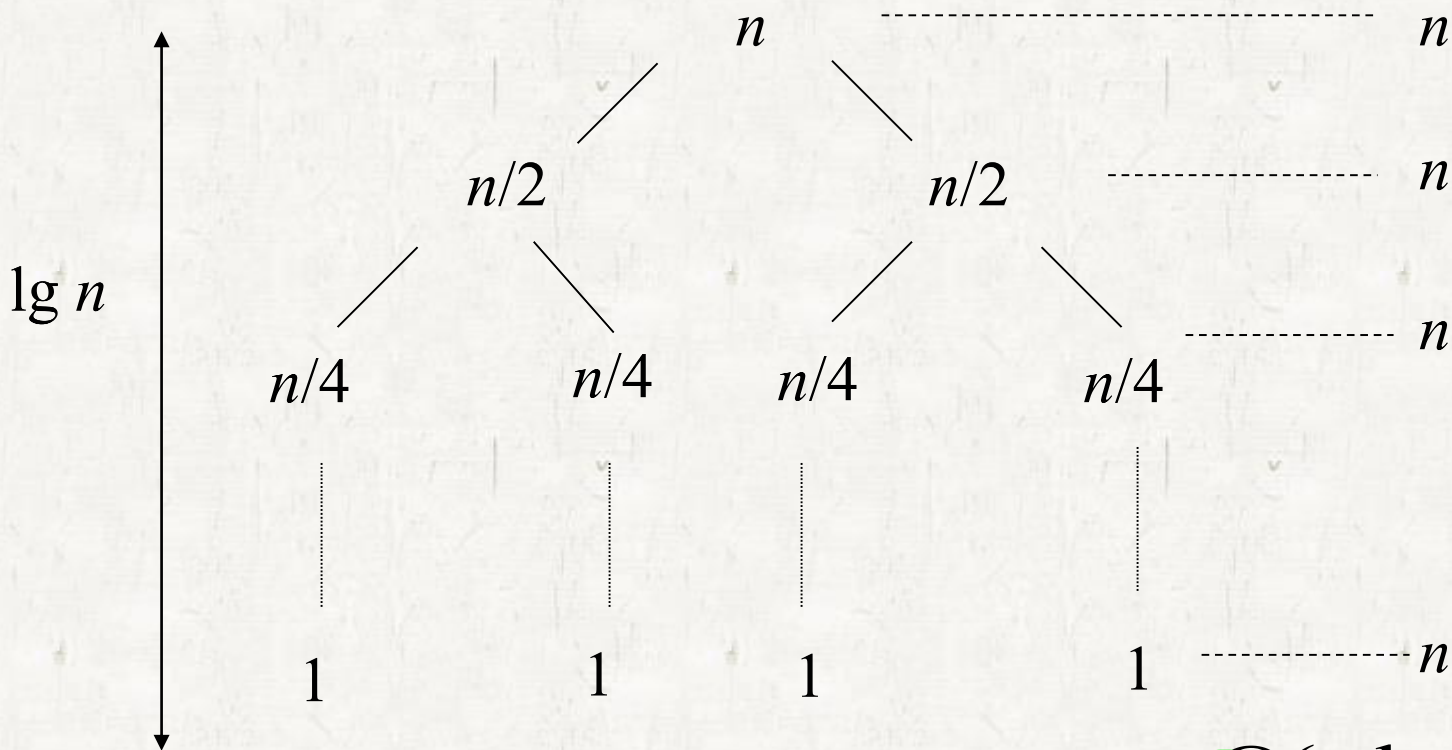
→ partition 한 후의 좌, 우 배열이 크기가 심하게 비대칭인 경우

Performance of quicksort

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

Performance of quicksort

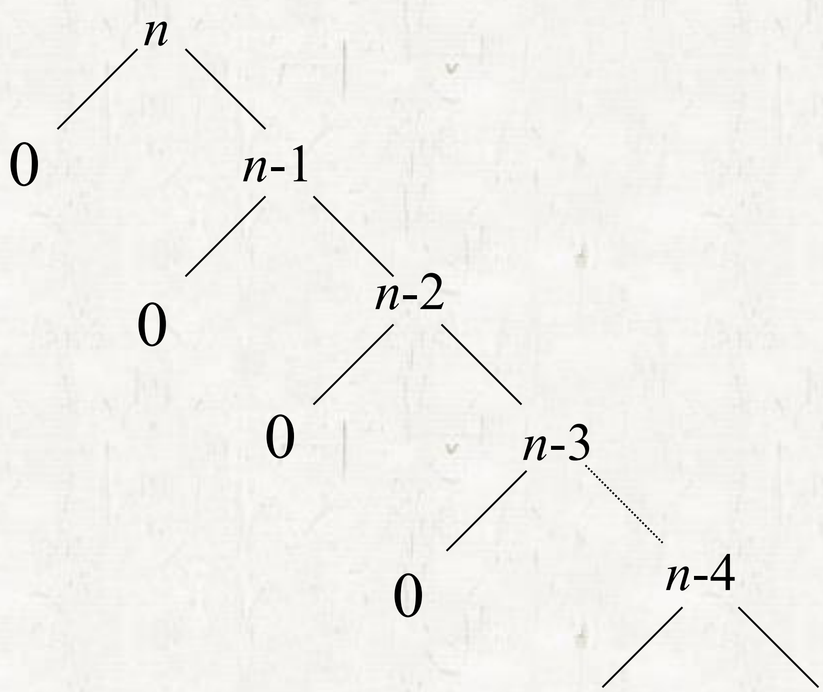
- **Balanced partitioning**



$\Theta(n \lg n)$

Performance of quicksort

- **Unbalanced partitioning**



Performance of quicksort

- **Unbalanced partitioning**

$$T(n) = \underbrace{T(n\text{-}1)}^{\text{Quick sort}} + \overbrace{\Theta(n)}^{\text{partitioning}} + \cancel{T(0)}$$

무시

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

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

$$= \boxed{\Theta(n^2).}$$

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

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

$\vdots$

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

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

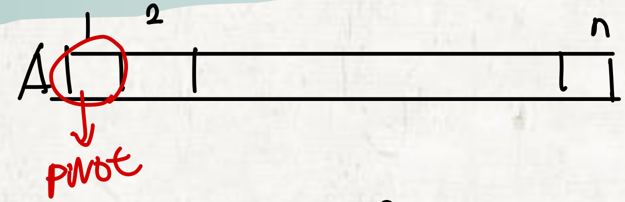
Worst-case Analysis

- **Worst-case analysis**

  $\Theta(n^2)$에 대한 증명은 아니다.

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

Worst-case Analysis



Worst-case analysis

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

T(0)+T(n-1)
T(1)+T(n-2)
⋮
T(n-1)+T(0)

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

worst case running time

partitioning

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

가정

$\Rightarrow T(n) = O(n^2)$

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

$((n-1)-q)^2$

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

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

$2(q^2 - (n-1)q) + (n-1)^2$

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

Worst-case Analysis

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

$$T(n) \le c \bullet \boxed{\max_{0\le q\le n\text{-}1}} \boxed{(2(q \;-(n\text{-}1)/2)^2 + (n\text{-}1)^2/2) + \Theta(n)}$$

Minimize: $q = \frac{n-1}{2}$

Maximize : $q=0$ or $q=n-1$

$\to q=0$

$$= c \bullet (n\text{-}1)^2 + \Theta(n)$$

$$= cn^2 \underline{- c(2n-1) + \Theta(n)}_{\le 0}$$

$$\le cn^2$$

$\Theta(n) \Rightarrow d_1 n \le \Theta(n) \le d_2(n)$

if c large enough so the

$C(2n-1) \ge d_1 n$

  - 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).

In worst Case

Average-case Analysis

- **Average-case analysis**

Blanced O(nlogn)
Unbalanced O(n^2)
⇒ Average O(nlogn)

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

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

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

Average-case Analysis II

- **Average Case Analysis II**
  - Let X be the total number of comparisons over the entire execution of QUICKSORT on an n-element array.
  - Then the average running time of QUICKSORT is
    - $O(n + \underline{E[X]})$.
  - 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.

Average-case Analysis II

- Let z_i denote the ith smallest element in the sorted array.
- $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

  → 모든 element는 pivot과 최대 한번이상 비교되기 때문

- 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. → pivot은 더 이상 비교되지 않는다.

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

Average-case Analysis II

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

(j-i+1)개

z_i	z_{i+1}	z_{i+2}	...	z_j

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

z_i나 z_j를 pivot으로
고를 확률

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

Z_{ij}에서 pivot 하나를
선택하면 다른 비교없이
partitioning 종료
=> Already Sorted!!

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

Average-case Analysis II

$k = j - i$, the harmonic series

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

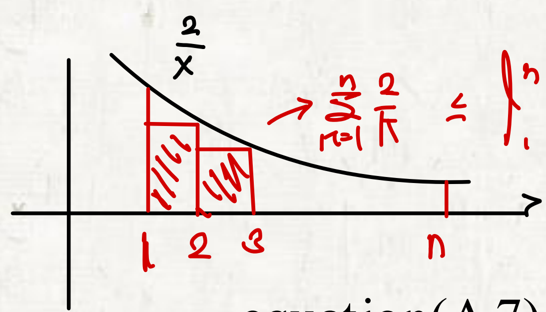
$$= \sum_{i=1}^{n\text{-}1} \sum_{k=1}^{n-i} \frac{2}{k+1}$$

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

$$= \sum_{i=1}^{n-1} O(\lg n)$$

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

$j - i = k,\ n = i + k.\ k = n - i$



$\rightarrow \sum_{k=1}^{n} \frac{2}{k} \le \int_1^n \frac{2}{x} dx$

$= 2\ln n$

equation(A.7)

$\sum_{k=1}^{n} \frac{2}{k} \le 2\ln n + 2$

$= O(\log n)$

Randomized quicksort

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)
```

Self-study

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