# Introduction to Algorithms

Mong-Jen Kao (高孟駿)

Tuesday 10:10 – 12:00

Thursday 15:30 – 16:20

# ***Divide-and-Conquer***

# – More Examples

More on recursion for problem solving.

# Median & Order Statistics

Select the k-th smallest element in $O(n)$ time.

# The k-th Order Selection Problem

- Given a sequence of numbers $A = \{a_1, a_2, \dots, a_n\}$ and an integer $k \in [1, n]$, find the $k^{th}$-smallest element in $A$.
  - The naïve approach runs in $O(kn) = O(n^2)$ time in the worst-case.
  - With sorting, we can do it in $O(n \log n)$ time.

# The k-th Order Selection Problem

- Given a sequence of numbers $A = \{a_1, a_2, \dots, a_n\}$ and an integer $k \in [1, n]$, find the $k^{th}$-smallest element in $A$.
  - The naïve approach runs in $O(kn) = O\left(n^2\right)$ time in the worst-case.
  - With sorting, we can do it in $O(n \log n)$ time.
- We will introduce two algorithms that solves this problem in
  - ***Worst-case*** $\boldsymbol{O(n)}$ ***time***, and
  - ***Expected*** $\boldsymbol{O(n)}$ ***time***.

# Selection in worst-case $O(n)$ time

To ***prune an $\boldsymbol{\Omega(1)}$-fraction*** of input ***in each round***.

# Selection in Deterministic $O(n)$ Time

- Let $A = \{a_1, \ldots, a_n\}$ be the input numbers and $k \in [1, n]$ be an integer.

  W.L.O.G., we may assume that

  – $\boldsymbol{k \geq 5}$. If $k \leq 4$, the answer can be computed in $O(n)$ time.

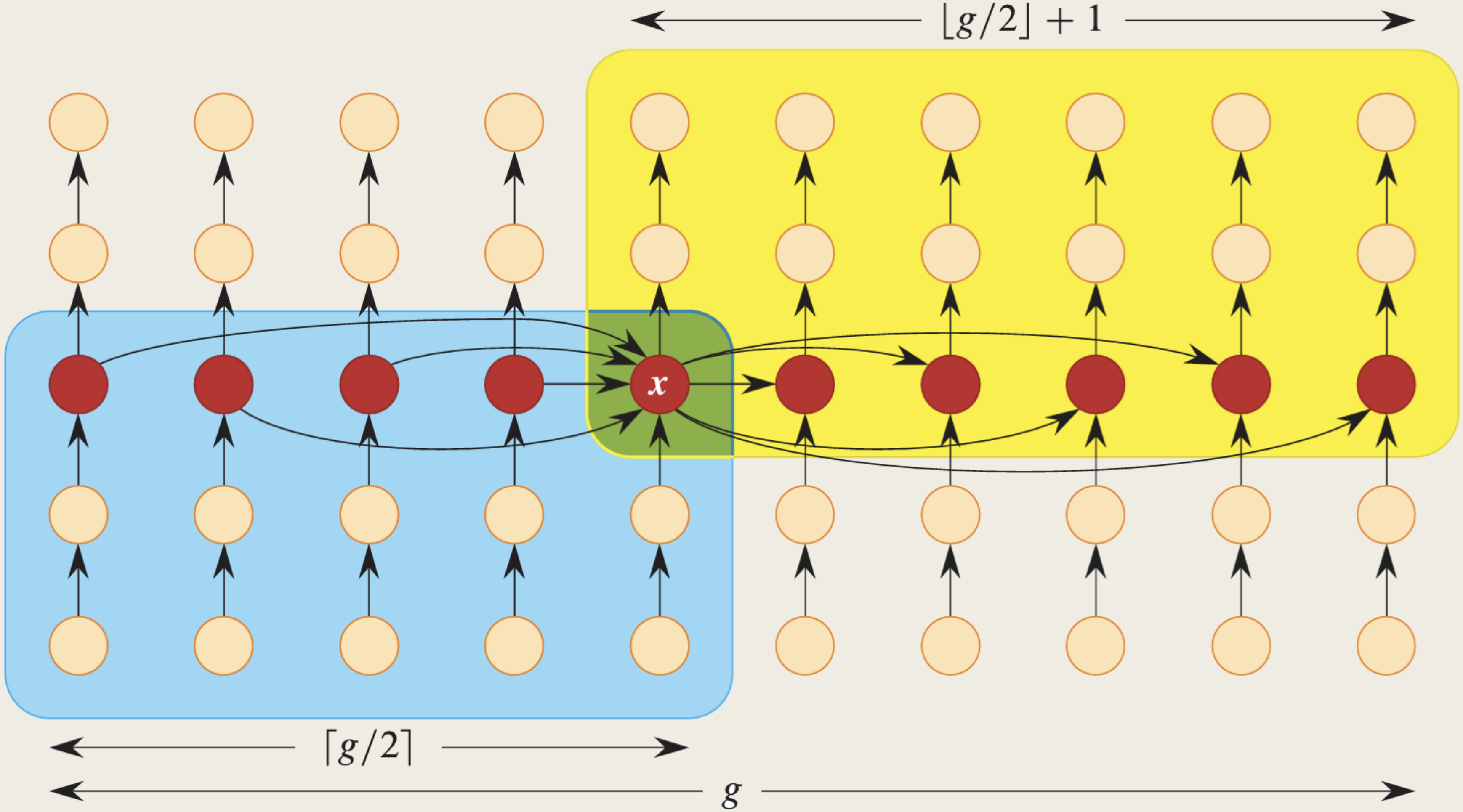
  – $\boldsymbol{n}$ **is a multiple of** $\mathbf{5}$, i.e., $n = 5g$ for some $g \in \mathbb{N}$.

    If not, remove the $r$ smallest elements from $A$ in $O(n)$ time, where $r \coloneqq n \bmod 5$, and

    consider the selection problem for $n' \coloneqq n - r$ and $k' \coloneqq k - r$.

# Selection in Deterministic $O(n)$ Time

- Let $A = \{a_1, \ldots, a_n\}$ be the input **with $\boldsymbol{n = 5g}$ for some $\boldsymbol{g}$** $\in \mathbb{N}$ and $k \in [1, n]$ be an integer.
  - Partition $A$ into $g$ groups of size $5$.
  - Sort the elements in each group in $O(n)$ time, and let $A'$ be the set of median elements in these groups.
  - ***Suppose that*** we have ***the median element $\boldsymbol{x}$*** in $A'$, then…




Roughly **3/10-fraction of the input <u>can be discarded</u>** from consideration!

- Recursive-Section($A, \ell, r, k$) -- Select the $k^{th}$ element from $A[\ell \ldots r]$.

A. // Preprocess $A[\ell \ldots r]$.

- Let $n \coloneqq r - \ell + 1$ and $r \coloneqq n \bmod 5$.
- For each $1 \leq i \leq r$,
  swap the $i^{th}$-smallest element in $A[\ell \ldots r]$ with $A[\ell + i - 1]$.
- If $k \leq r$, then return $A[\ell + k - 1]$.
- Otherwise,
  set $\ell \leftarrow \ell + r$, $n \leftarrow n - r$, $g = n/5$, and $k \leftarrow k - r$.

- Recursive-Section($A, \ell, r, k$) -- Select the $k^{th}$ element from $A[\ell \dots r]$.

A. Preprocess $A[\ell \dots r]$ such that $n := r - \ell + 1 = 5g$.

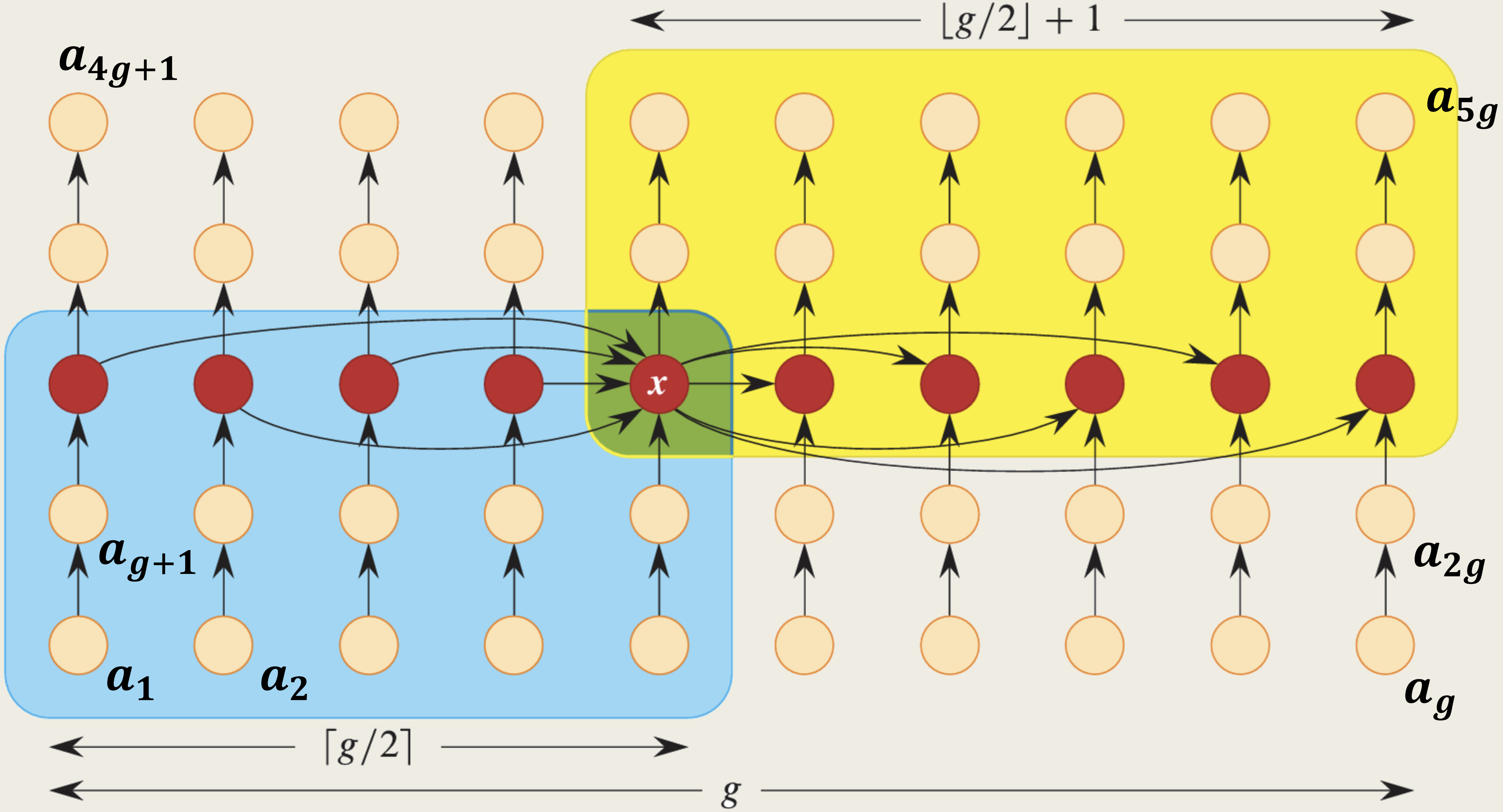
B. For each $j = \ell, \dots, \ell + g - 1$,
sort the 5 elements $\{ A[j + ig] \}_{0 \le i < 5}$ in place.

C. Let $x \leftarrow$ Recursive-Selection( $A, \ \ \ell + 2g, \ \ \ell + 3g - 1, \ \ \lceil g/2 \rceil$ ).

D. // Partition $A[\ell \dots r]$ according to $x$.
Set $q \leftarrow$ In-Place-Partition($A, \ell, r, x$) and $t \leftarrow q - \ell + 1$.

E. If $k == t$, then return $A[q]$.

F. If $k < t$, then return Recursive-Selection($A, \ell, q, k$).
Otherwise, return Recursive-Selection($A, q + 1, r, k - t$).



⌊g/2⌋ + 1
$a_{4g+1}$
$a_{5g}$
x
$a_{g+1}$
$a_{2g}$
$a_1$
$a_2$
$a_g$
⌈g/2⌉
g

# Analysis of Recursive-Selection

- For the recursion call in step F,

  the number of remaining elements is at most

$$n - 3 \cdot \left\lceil \frac{g}{2} \right\rceil \leq n - \frac{3}{10}n = \frac{7}{10}n.$$

- Hence, the time complexity of this algorithm is

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + \Theta(n),$$

  which has a solution $T(n) = O(n)$.

# Selection in expected $O(n)$ time

The quick-select algorithm in ProgHW-III.

# Analysis of Quick-Select

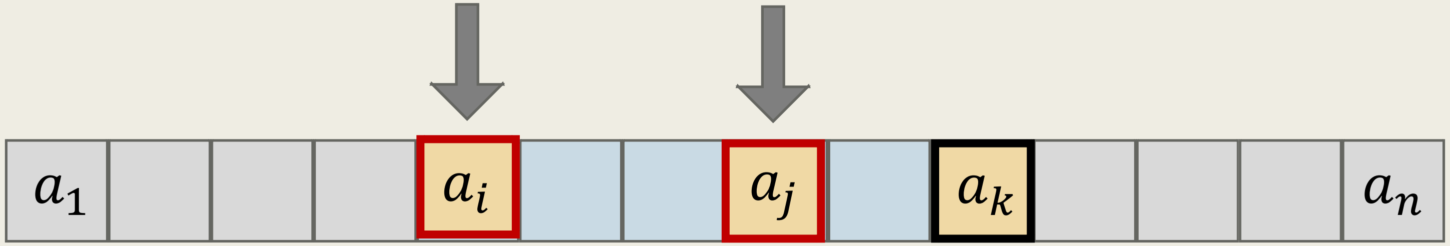
- Let $k$ be the input order to find and

$$a_1 < a_2 < \cdots < a_n$$

  be the input numbers re-indexed in non-descending order.

- For any $1 \le i < j \le n$, let $\boldsymbol{X_{i,j}}$ ***be the <u>indicator variable</u>*** for the event that $\boldsymbol{a_i}$ ***and*** $\boldsymbol{a_j}$ ***are compared*** during the execution.
  - According to the relative order between $i, j$ and $k$, we consider three cases.

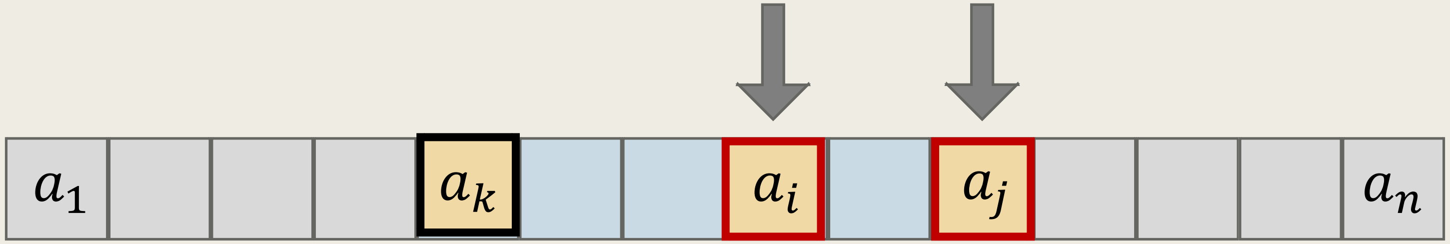
# Case I - $i < j \leq k$




- The value of $X_{i,j}$ is determined <u>only when</u> some number between $a_i$ to $a_k$ is selected to be the pivot.
  - In this case, $a_i$ and $a_j$ is compared only when $a_i$ or $a_j$ is picked.
  - Hence,

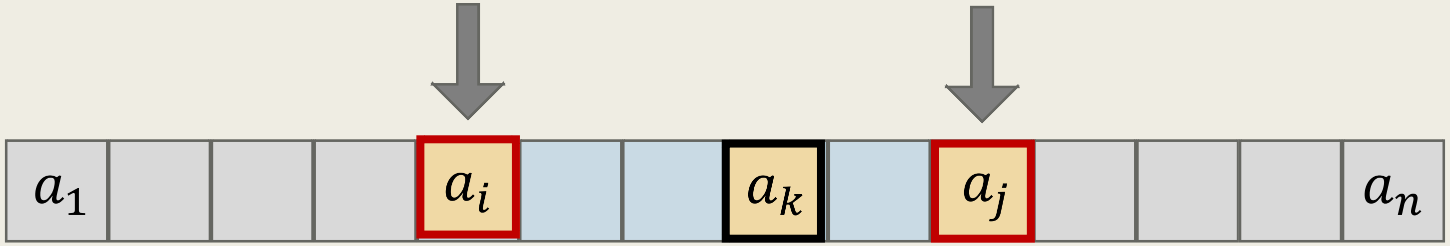
$$E[\,X_{i,j}\,] = \frac{2}{k-i+1}\,.$$

# Case II - $k \leq i < j$




- This case is symmetric to the previous one.
  - We have $$E[X_{i,j}] = \frac{2}{j-k+1}.$$

# Case III - $i < k < j$




- In this case, the value of $X_{i,j}$ is determined <u>only when</u> some number between $a_i$ to $a_j$ is selected to be the pivot.
  - We have $$E[X_{i,j}] = \frac{2}{j - i + 1}.$$

# Analysis of Quick-Select

- The total number of comparisons for case I is hence

$$\sum_{i<j\leq k} \frac{2}{k-i+1} = \sum_{i\leq k} (k-i)\cdot\frac{2}{k-i+1} \leq 2(k-1) = O(n).$$

- Similarly, for case II, we have

$$\sum_{k\leq i<j} \frac{2}{j-k+1} \leq 2(n-k) = O(n).$$

# Analysis of Quick-Select

- For case III, the total number of comparisons is

$$\sum_{i<k<j} \frac{2}{j-i+1} = \sum_{1\le i<k} \sum_{d=k-i+1}^{n-i} \frac{2}{d+1}$$

$$\le 2 \cdot \sum_{1\le i<k} H_{n-i+1} \le 2 \cdot \sum_{1\le i<k} \ln(n-i+1)$$

$$= 2 \cdot \ln\binom{n}{k-1} \le 2 \cdot \ln(2^n) = O(n).$$

# Analysis of Quick-Select

- Let $k$ be the input order to find and

$$a_1 < a_2 < \cdots < a_n$$

be the input numbers re-indexed in non-descending order.

- For any $1 \le i < j \le n$, let $\boldsymbol{X_{i,j}}$ ***be the indicator variable*** for the event that $\boldsymbol{a_i}$ ***and*** $\boldsymbol{a_j}$ ***are compared*** during the execution.
  - We have
$$E\left[\sum_{i,j} X_{i,j}\right] = O(n)\,.$$

# Second Proof

- Let $k$ be the input order to find and

$$a_1 < a_2 < \cdots < a_n$$

  be the input numbers ***re-indexed*** in non-descending order.

- Consider the ***number of remaining elements*** after each recursion.
  - When a number between $a_{n/4}$ and $a_{3n/4}$ is picked as pivot, **at least $1/4$-fraction** of the numbers ***will be pruned***.
  - This happens with probability $1/2$.

# Second Proof

- Consider the *number of remaining elements* after each recursion.
  - When a number between $a_{n/4}$ and $a_{3n/4}$ is picked as pivot, **at least $1/4$-fraction** of the numbers ***will be pruned***.
  - This happens with probability $p \coloneqq 1/2$.
- Let $Y$ be the ***number of recursions*** before at least $1/4$-fraction of the elements are pruned.
  - Then $Y$ is a ***geometric distribution*** with parameter $p$ and

$$E[Y] = 1/p = 2.$$

# Second Proof

- Let $Y$ be the ***number of recursions*** before at least $1/4$-fraction of the elements are pruned.
  - Then $Y$ is a ***geometric distribution*** with parameter $p$ and

$$E[Y] = 1/p = 2.$$

- The running time of the algorithm is at most

$$\sum_{i \geq 0} 2 \cdot \left(\frac{3}{4}\right)^i \cdot n = 8n = O(n) .$$