

5. Selection

Goals

- Learn average-case $O(n)$ time selection algorithm.
- Learn worst-case $O(n)$ time selection algorithm.
- Understand the relationship between two algorithms.

Selection Problem

- Input: $A[1 \dots n]$ and integer i ($1 \leq i \leq n$)
- Output: i -th smallest element of $A[1 \dots n]$
- Algorithms for selection problem
 - Average-case $O(n)$ time algorithm
 - Worst-case $O(n)$ time algorithm

Average-Case $O(n)$ Time Algorithm

select (A, p, r, i)

▷ find i -th smallest element in $A[p \dots r]$

{

if ($p = r$) **then return** $A[p]$; ▷ if there is one element, i must be 1.

$q \leftarrow \text{partition}(A, p, r)$;

$k \leftarrow q - p + 1$; ▷ pivot is k -th smallest element

if ($i < k$) **then return** **select**($A, p, q-1, i$) ; ▷ left part

else if ($i = k$) **then return** $A[q]$; ▷ pivot is the answer

else return **select**($A, q+1, r, i-k$) ; ▷ right part

}

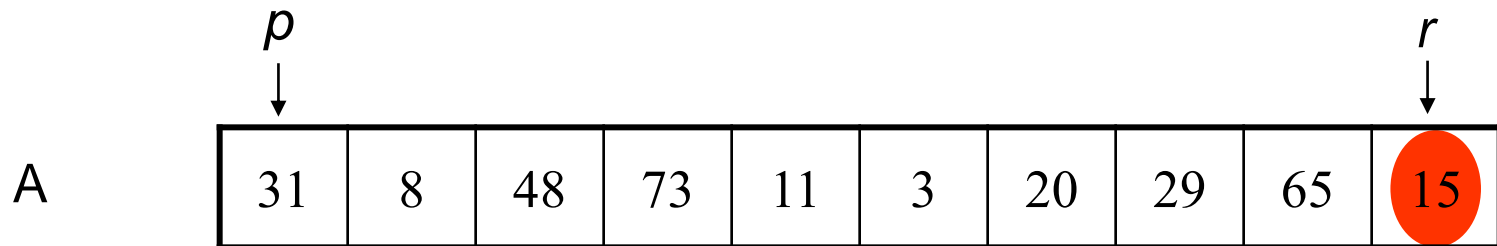
✓ Average-case time: $\Theta(n)$

✓ Worst-case time: $\Theta(n^2)$

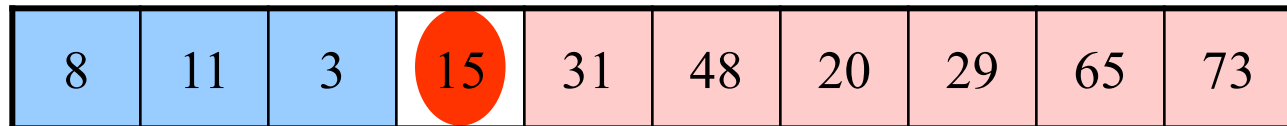
Randomized selection algorithm: $\text{randomized-partition}(A, p, r)$

Selection Algorithm

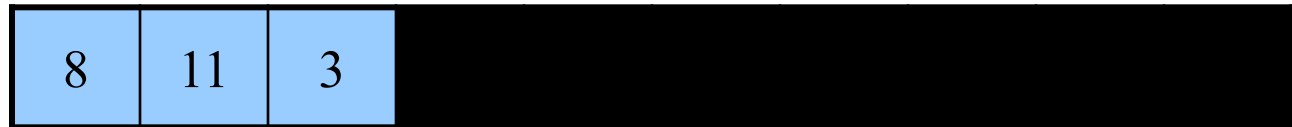
Find 2nd smallest element



partition

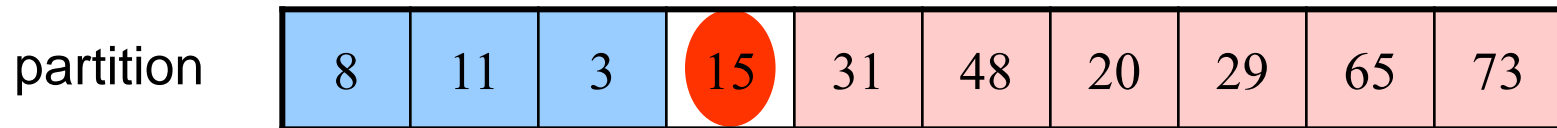
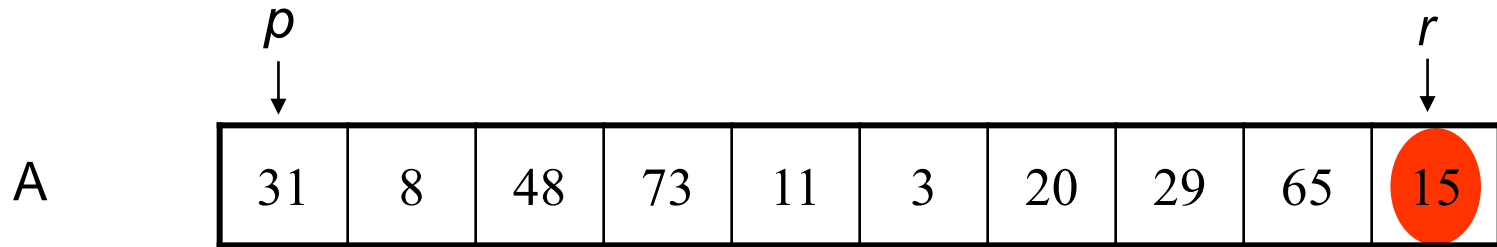


Find 2nd smallest element in left part

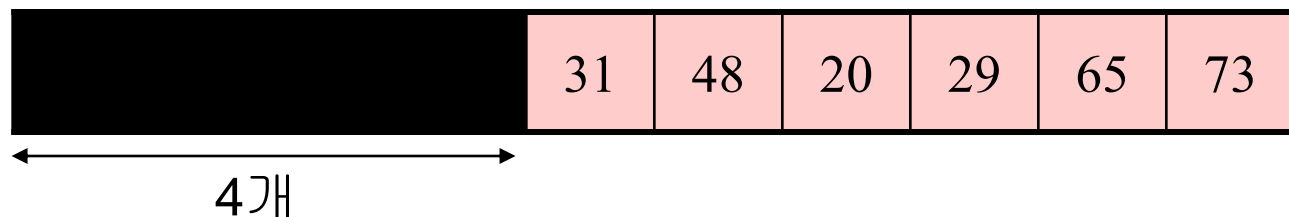


Selection Algorithm

Find 7th smallest element



Find 3rd smallest element in right part



Average Time

$$T(n) \leq \frac{1}{n} \sum_{k=1}^n \max[T(k-1), T(n-k)] + \Theta(n)$$

Time for larger among two parts

Overhead (mostly partition)

Prove $T(n) \leq cn$ by guess-and-prove.

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

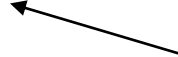
$$\text{Since } T(n) = \Omega(n), \quad T(n) = \Theta(n)$$

Worst Case

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



Time for larger when partition is $0:n-1$



Overhead (mostly partition)

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

Worst-Case $O(n)$ Time Algorithm

- Previous selection algorithm
 - Time is affected by balance in partition
- This algorithm
 - Guarantee some balance in partition in worst case so that worst-case time is $\Theta(n)$.
 - Overhead to find such a balance shouldn't be too much

Worst-Case $O(n)$ Time Algorithm

linearSelect (A, p, r, i)

▷ find i -th smallest element in $A[p \dots r]$

{

① if $n \leq 5$, find answer by insertion sort and return.

② Divide n elements into $\lceil n/5 \rceil$ groups of 5 elements.

(if n is not a multiple of 5, size of one group is < 5 .)

③ Find median in each group by insertion sort.

Let $m_1, m_2, \dots, m_{\lceil n/5 \rceil}$ be these medians.

④ Find median M of $m_1, m_2, \dots, m_{\lceil n/5 \rceil}$ recursively

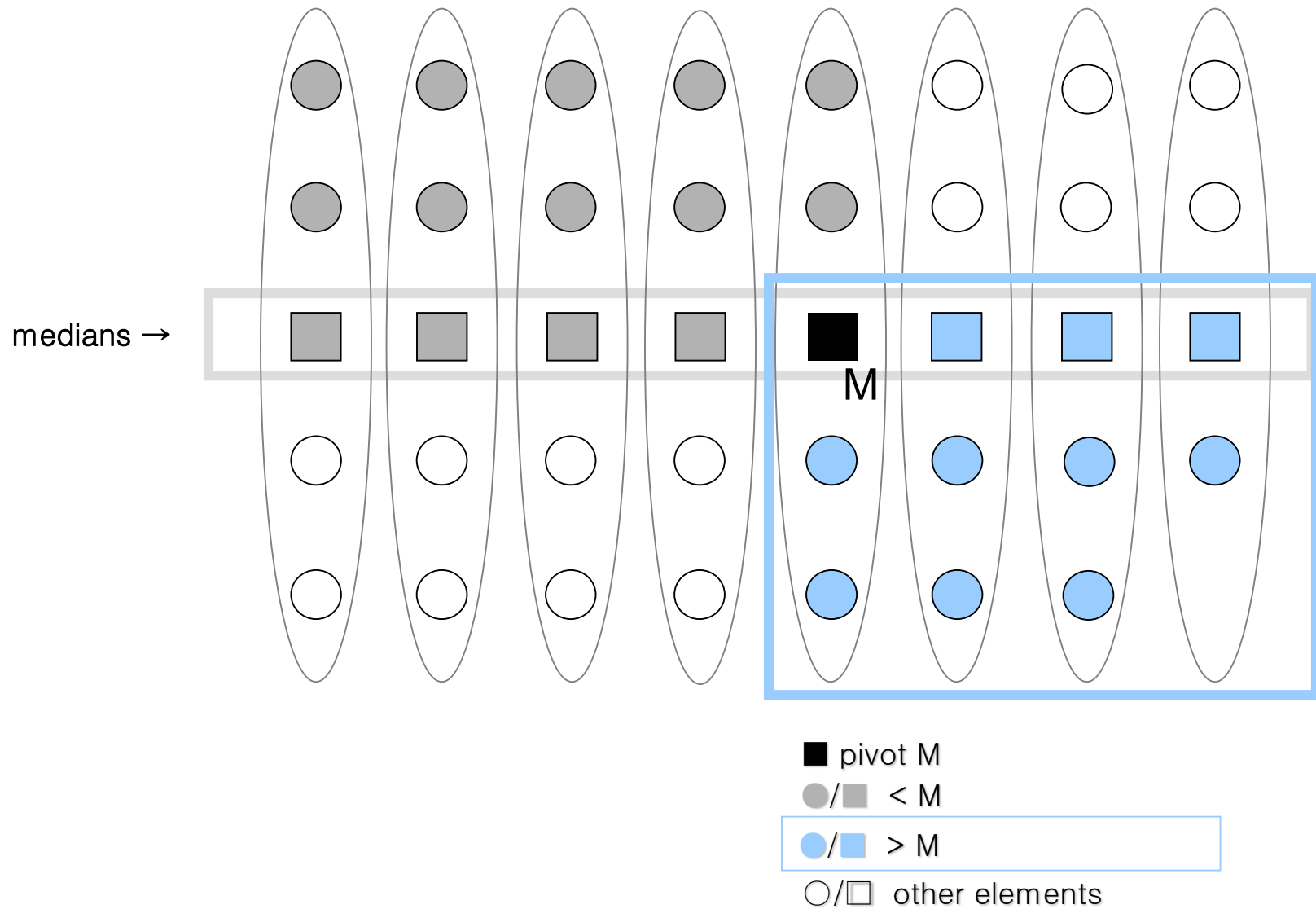
(if number of elements is even, find lower median.) ▷ call **linearSelect**()

⑤ partition n elements using M as pivot

⑥ same as select(). ▷ call **linearSelect**()

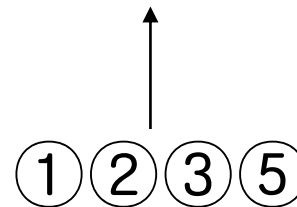
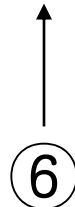
}

linearSelect



Worst-Case Time

$$T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 4) + \Theta(n)$$



Prove $T(n) \leq cn$ by guess-and-prove

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

Since $T(n) = \Omega(n)$, $T(n) = \Theta(n)$



Thank you
