DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

SELECTION IN WORST-CASE LINEAR TIME

AGENDA

- Quick review on the time-complexity of RANDOMIZED-SELECT algorithm
- Select in worst-case scenario
- A two-dimensional array has 5 rows and 7 columns. How many elements are contained in the two-dimensional array?

RANDOMIZED-SELECT TIME COMPLEXITY

- The **best-case** scenario of the normal case $T(n) = \underline{\boldsymbol{\Theta(n) + T(n/2)}} = O(\underline{\ \boldsymbol{n}\ })$.
- Recursion tree

- The **worst-case** scenario of the normal case $T(n) = \underline{\boldsymbol{\Theta(n) + T(n-1)}} = O(\underline{\ \boldsymbol{n^2}\ })$.
- Recursion tree

Takeaway?

IMPROVING RANDOMIZED SELECT

- If we can find the ***median (close to median)*** of the input in $\boldsymbol{O(n)}$, we can guarantee a good split.

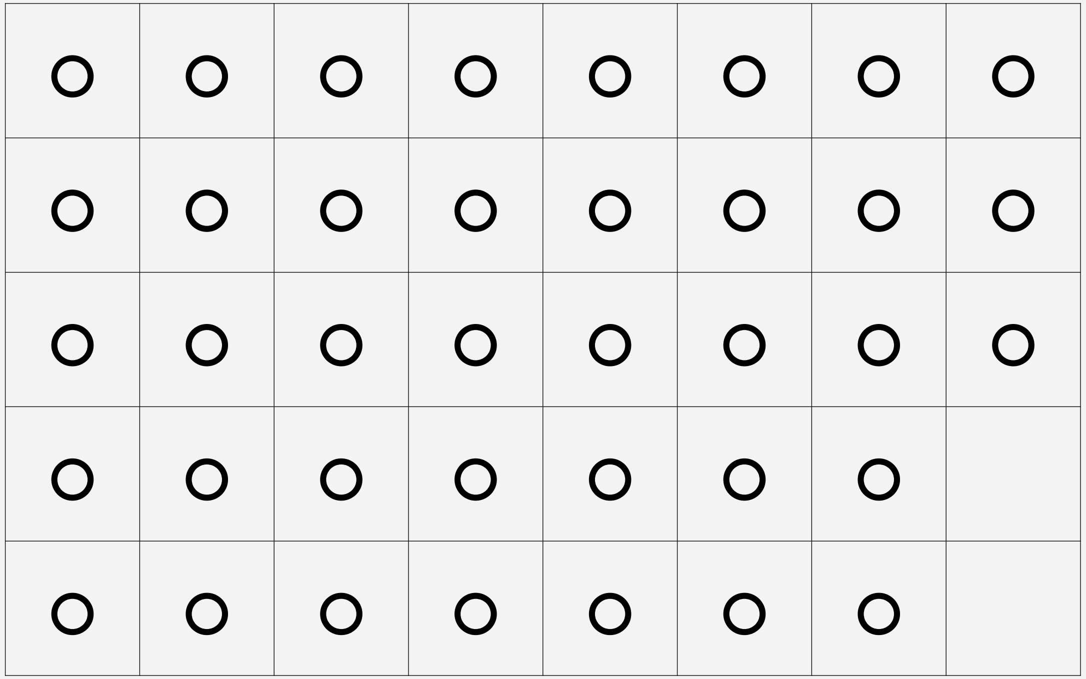
- The algorithm is SELECT.
  - Input array has n distinct numbers.
  - Determines the ith smallest number of the input array.
  - Select in **worst**-case linear time.

LOGIC PROBLEM

- Three geniuses A, B, and C.
  - Give each a hat.
  - They can see the colors of the hats of the other two.
  - They can't see the color of their own hat.
- I ask them a question, "what is the color of your hat?"
  - No answer
- Ask the same question, again.
  - No answer
- Ask the same question for the 3rd time.
  - They all shout out the color of their own hat.

SELECT ALGORITHM
STEP 1

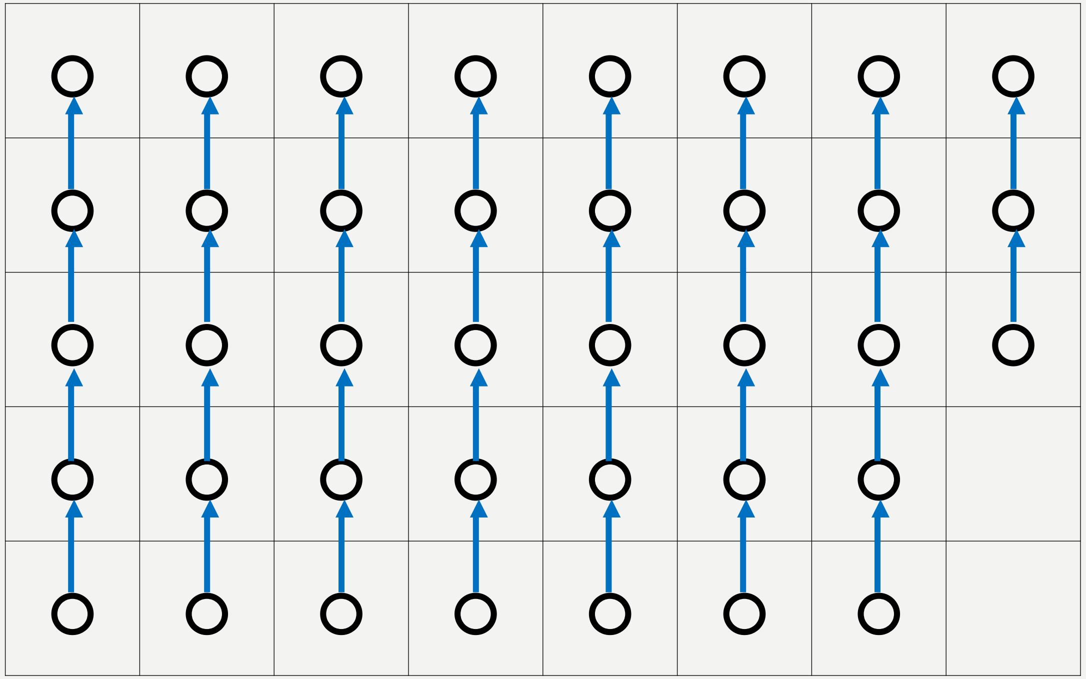
- **Step 1**: Divide the n elements of the input array into groups of 5 elements.
  - The number of groups = ______________.
  - The last group has ____________ elements.
  - The running time of this step is __________.



The input array after the grouping

SELECT ALGORITHM
STEP 2

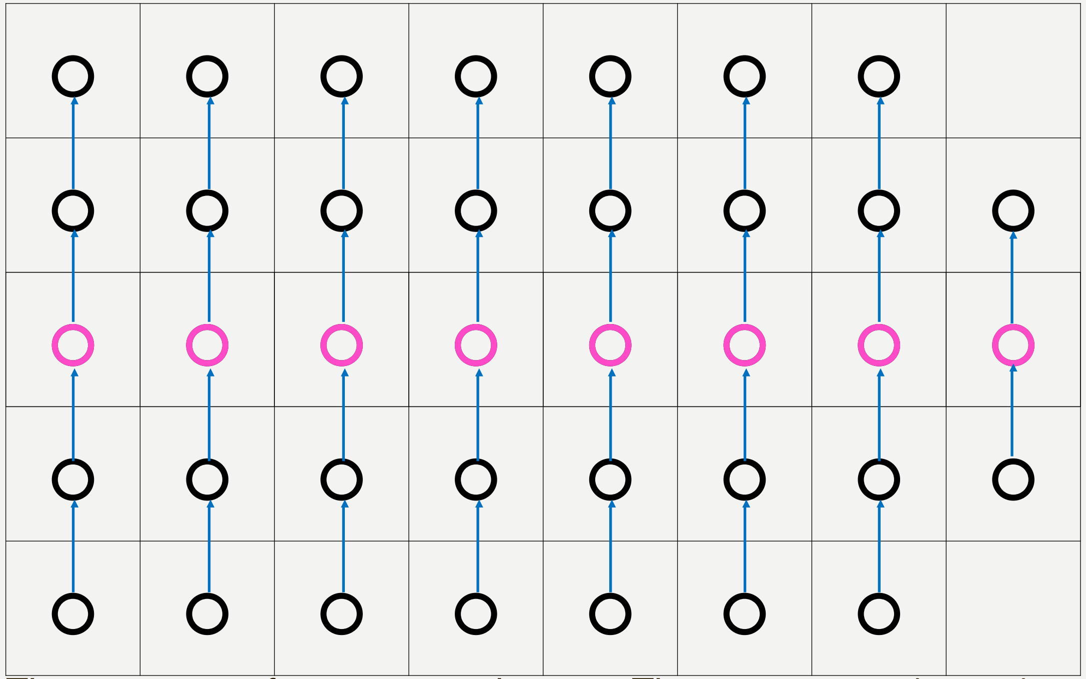
- **Step 2**: Find the median of each group by
  - **2a**: first insertion-sorting the elements of each group.
    - The cost of insertion-sorting **ONE** group is ______________________________.
    - The cost of insertion-sorting **ALL** groups is ____________________________________.



The input array after sorting each group. Greater element ⟶ smaller

SELECT ALGORITHM
STEP 2 (CONT'D)

- **Step 2**: Find the median of each group by
  - **2b**: then picking the **median** from the sorted list of group elements.
    - The cost of finding the **median** of **ONE** group is ________________________.
    - The cost of finding the **median** of **ALL** groups is ________________________.
  - The **overall** running time of **step 2** is ________________________________.



The input array after sorting each group. The ○ represents the median of each group.

SELECT ALGORITHM
STEP 2 (A CLOSER LOOK)

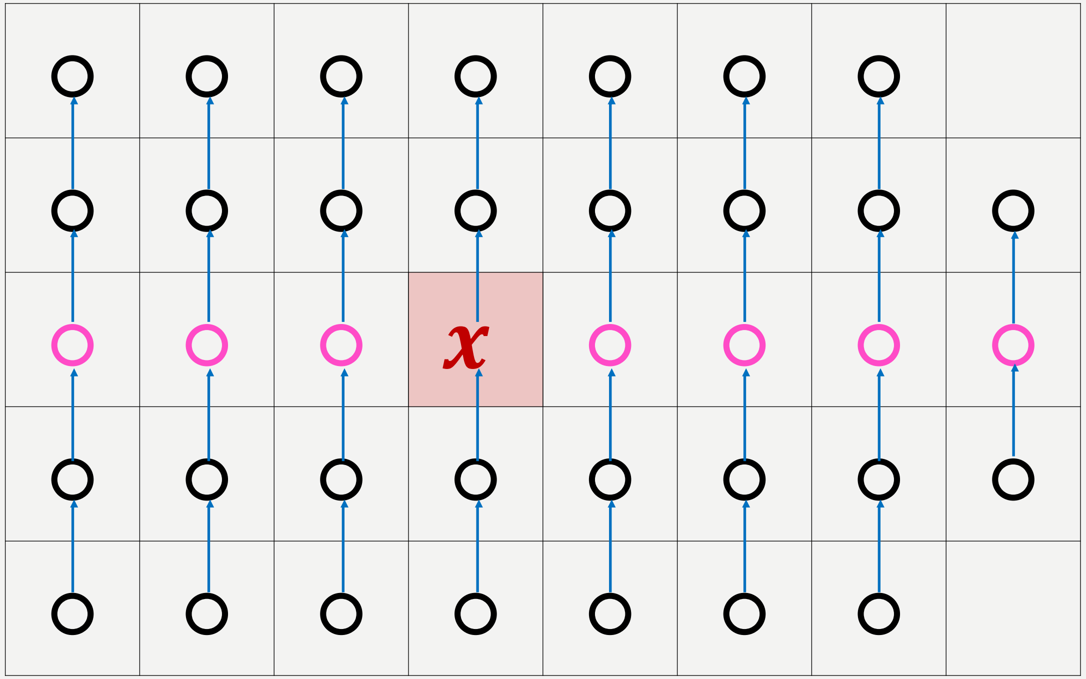
- **Step 2**: Find the median of each group.
  - On an input instance
  - Each vertical group has been sorted in increasing order from top to bottom.

9	25	17	1	4	22	6	
10	26	18	2	5	23	14	7
11	27	19	3	36	24	15	8
12	28	20	32	37	30	16	33
13	29	21	34	38	31	35	

The input array after sorting each group.

SELECT ALGORITHM
STEP 3

- **Step 3**: Use SELECT recursively to find the median x of the $\lceil n/5 \rceil$ **medians** found in step 2.
  - If there are two medians, then by convention, m is the lower median.
  - The **overall** running time of step 3 is ______________________.



The array after finding the median x of the **medians**. Assume that the x is located at the midpoint of the **medians** .

SELECT ALGORITHM
STEP 3 (A CLOSER LOOK)

- **Step 3**: Use SELECT <u>recursively</u> to find the median $\boldsymbol{x}$ of the $\lceil n/5 \rceil$ **medians** found in step 2.
  - On an input instance
  - There are 8 **medians**, the **median** of the 8 **medians** should be the $\left\lfloor \frac{1+8}{2} \right\rfloor = \mathbf{4}$th order statistic of all **medians**.

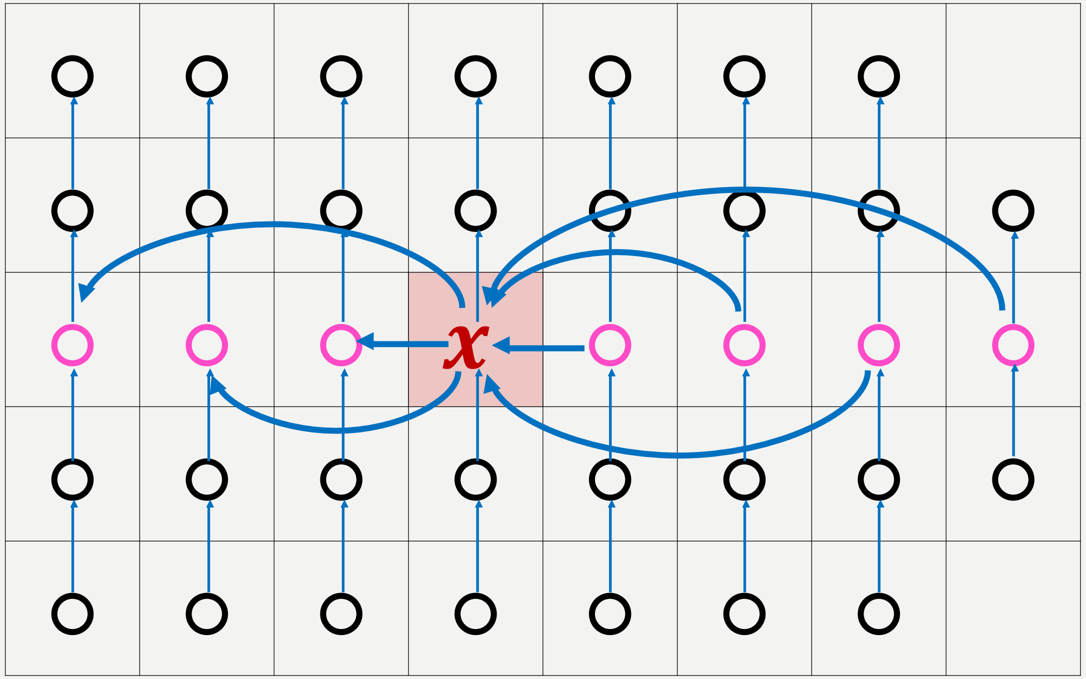
9	25	17	1	4	22	6	
10	26	18	2	5	23	14	7
11	27	19	3	36	24	15	8
12	28	20	32	37	30	16	33
13	29	21	34	38	31	35	

The array after finding the median $\boldsymbol{x}$ of the **medians**.

SELECT ALGORITHM
STEP 4

- **Step 4**: Partition the **GROUPS** around the median-of-medians $\boldsymbol{x}$ using the modified PARTITION algorithm.
  - The PARTITION-PIVOT **take the element to partition around** as an input parameter.

PARTITION-PIVOT $(A, p, r, \boldsymbol{q})$
$pivot = A[q]$
$i = p - 1$
for $j = p$ **to** $r - 1$
if $A[j] < pivot$
$i = i + 1$
exchange $A[i]$ with $A[j]$
exchange $A[i + 1]$ with $A[q]$
return $i + 1$



The array after applying PARTITION around the median $\boldsymbol{x}$.
Larger element ⟶ smaller

SELECT ALGORITHM
STEP 4 (A CLOSER LOOK)

9	25	17	1	4	22	6	
10	26	18	2	5	23	14	7
11	27	19	3	36	24	15	8
12	28	20	32	37	30	16	33
13	29	21	34	38	31	35	

- **Step 4**: Partition the **GROUPS** around the median-of-medians x using the modified PARTITION algorithm.

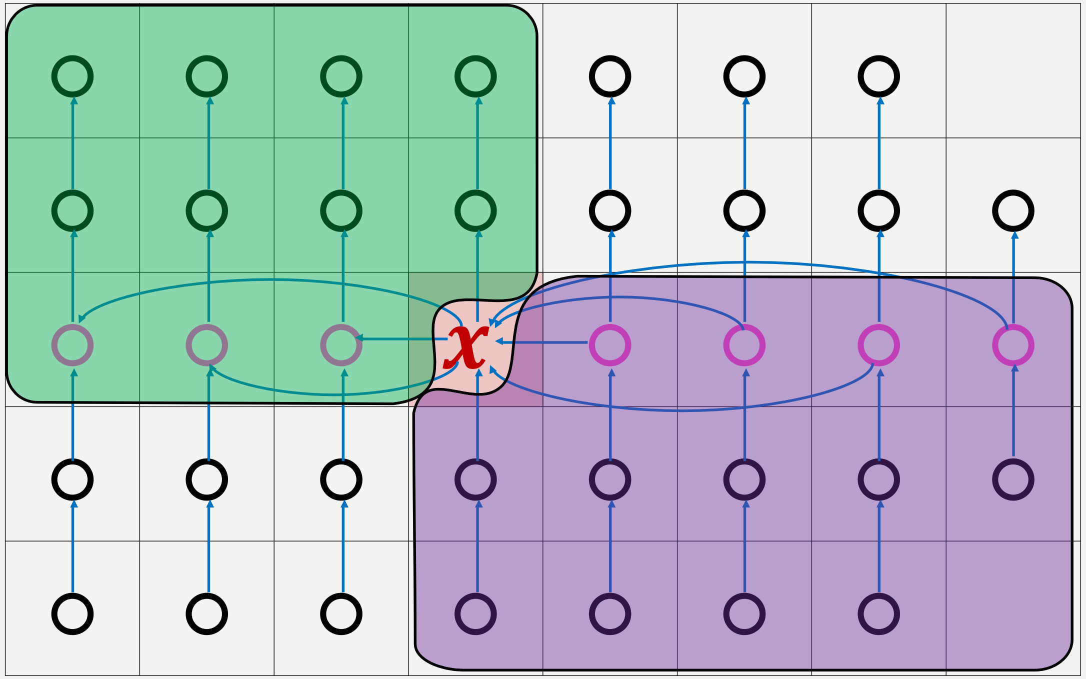
```
PARTITION-PIVOT (A, p, r, q)
pivot = A[q]
i = p − 1
for j = p to r − 1
        if A[j] < pivot
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[pivot]
return i + 1
```

9	1		6	4	22	25	17
10	2	7	14	5	23	26	18
11	3	8	15	36	24	27	19
12	32	33	16	37	30	28	20
13	34		35	38	31	29	21

The array after applying PARTITION around the median x.
Larger element ⟶ smaller

SELECT ALGORITHM
STEP 4 (CONT'D)

- **Step 4**: Partition the **GROUPS** around the median-of-medians x using the modified PARTITION algorithm.
  - *Low side* of the partition convers all the elements known to be *less than* the median-of-medians x.
  - *High side* of the partition convers all the elements known to be *greater than* the median-of-medians x.

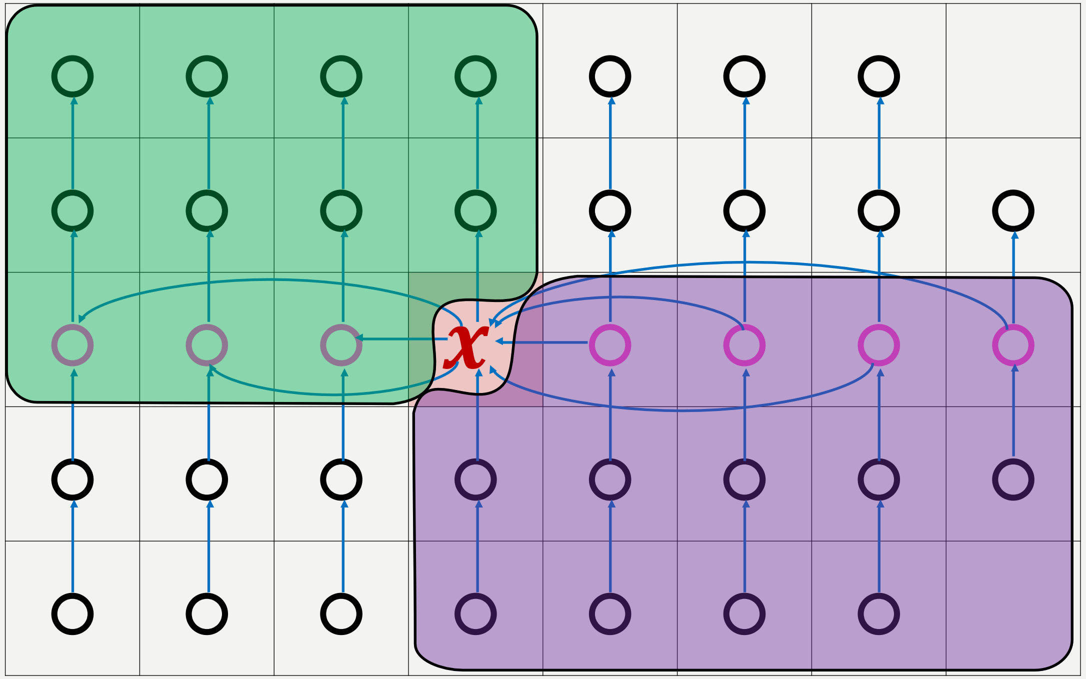


The array after applying PARTITION around the median x.
Larger element ⟶ smaller

SELECT ALGORITHM
STEP 4 (CONT'D)

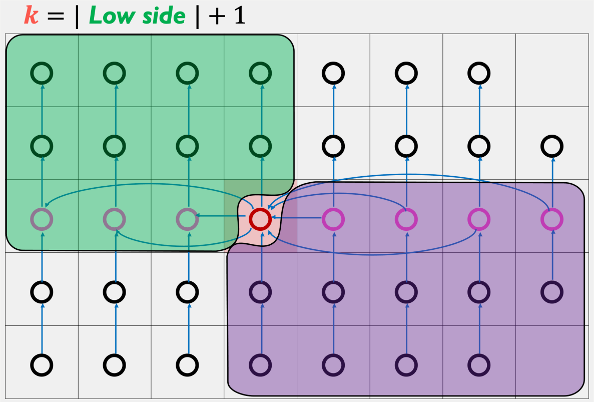
- **Step 4**: Partition the **GROUPS** around the median-of-medians $\boldsymbol{x}$ using the modified PARTITION algorithm.
  - Let k be one less than the index of median of medians $\boldsymbol{x}$.

$k = |\ \textbf{Low side}\ | + 1$



The array after applying PARTITION around the median $\boldsymbol{x}$.
Larger element ⟶ smaller

SELECT ALGORITHM
STEP 4 (RUNNING TIME)



- **Step 4**: Partition the **GROUPS** around the median-of-medians $\boldsymbol{x}$ using the modified PARTITION algorithm.
  - The running time of step 4 is $\boldsymbol{O(n)}$.

PARTITION-PIVOT $(A, p, r, \boldsymbol{q})$
$pivot = A[q]$
$i = p - 1$
for $j = p$ **to** $r - 1$
if $A[j] \leq pivot$
$i = i + 1$
exchange $A[i]$ with $A[j]$
exchange $A[i+1]$ with $A[q]$
return $i + 1$

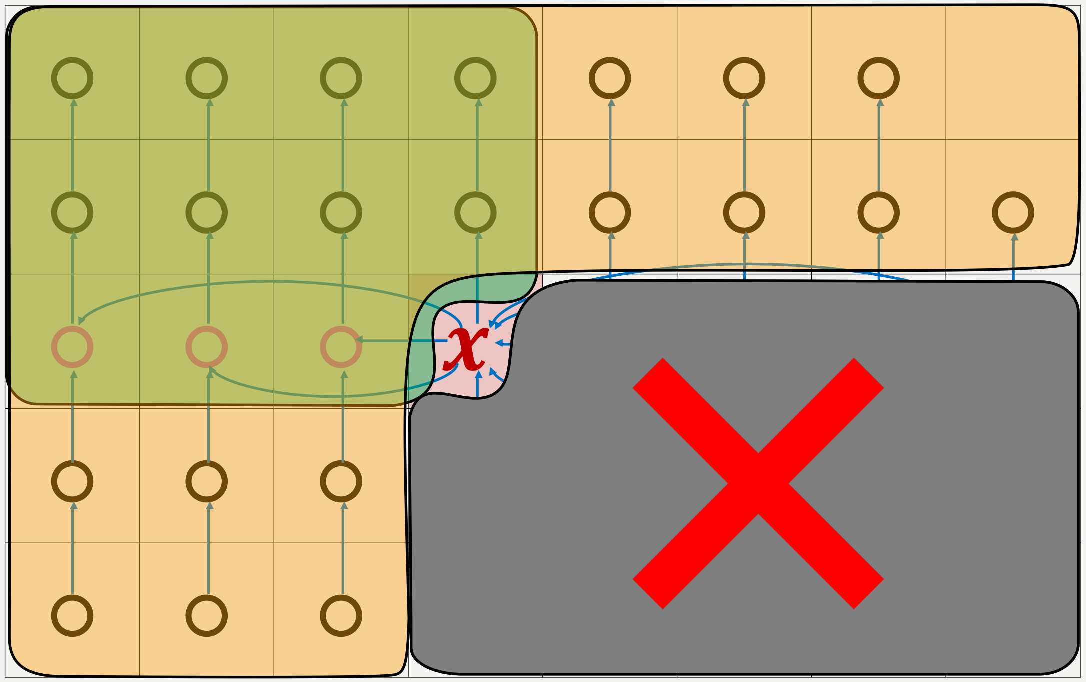
PARTITION-GROUPS $(A, p, r, \boldsymbol{q})$
$pivot = A[\boldsymbol{q}]$
$i = ?$
for $j = ?$ **to** ? **by** ?
if $A[j] \leq pivot$
$i = ?$
exchange $A[?]$ with $A[?]$
exchange $A[?]$ with $A[?]$
return ?

Completing this code is part of the HW.

SELECT ALGORITHM
STEP 5

- **Step 5**: Depending on the relationship between i and $\boldsymbol{k}$, proceed accordingly.
  - Case 1: $i \leq \boldsymbol{k}$
    - The ith order statistic does NOT exist in ____________________.
    - Recurse on **Set** $\boldsymbol{A}$ − {***high side***} to find the $\boldsymbol{i}$ th order statistic;

$\boldsymbol{k} = |$ ***Low side*** $| + 1$

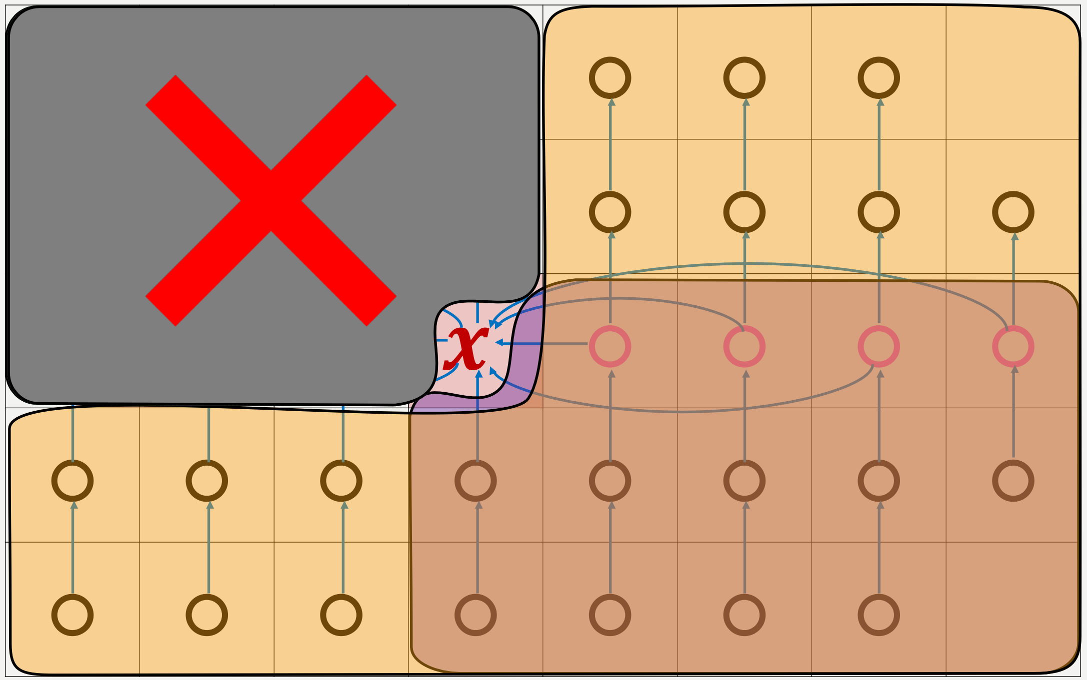


The array after applying PARTITION around the median $\boldsymbol{x}$.
Larger element ⟶ smaller

SELECT ALGORITHM
STEP 5

- **Step 5**: Depending on the relationship between i and k, proceed accordingly.
  - Case 2: $i > k$
    - The ith order statistic does NOT exist in __________________.
    - Recurse on **Set A − {low side}** to find the **$i - k$** th order statistic.

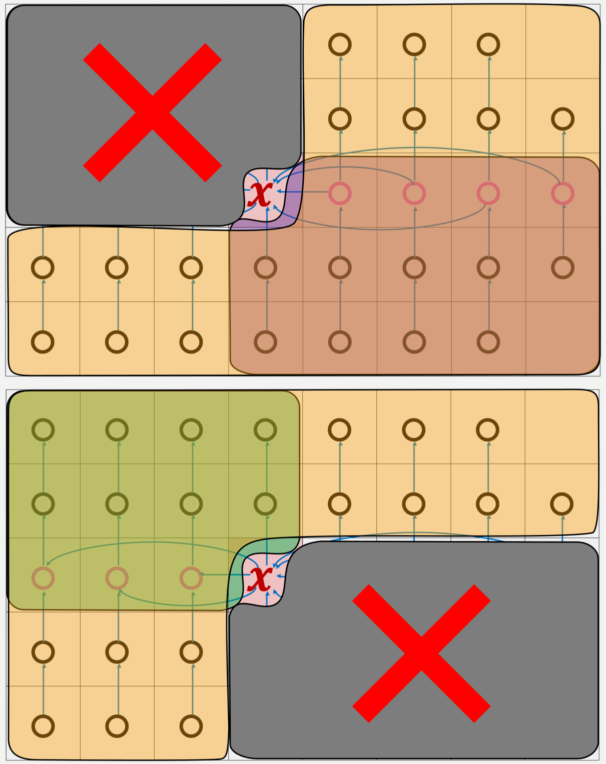
$k = |\ \textbf{Low side}\ | + 1$



The array after applying PARTITION around the median x.
Larger element → smaller

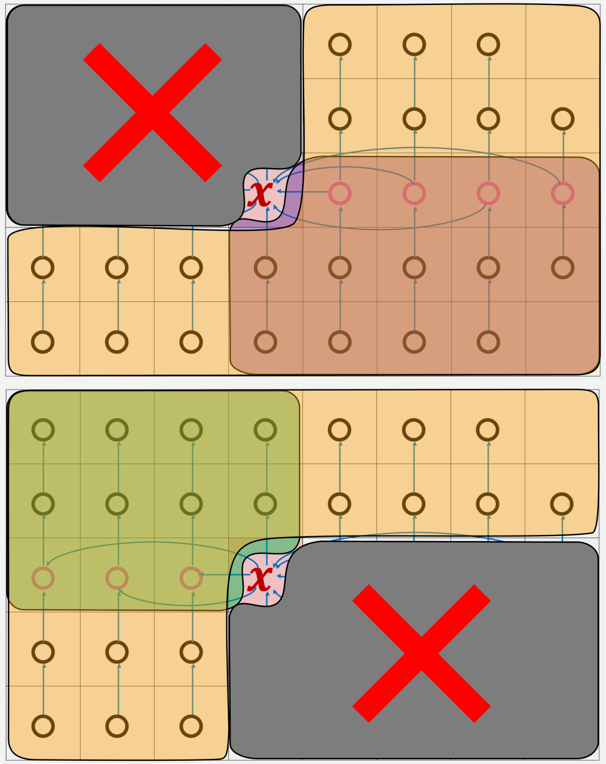
SELECT ALGORITHM
STEP 5 RUNNING TIME

- **Step 5:** Depending on the relationship between i and k, proceed accordingly.
  - Calculate a **lower**-bound on the number of elements that are in either the ***low side*** or the ***high side*** of the partition.
    - At least ____________________ groups contribute at least ____________ elements to either the ***low side*** or the ***high side***.



SELECT ALGORITHM
STEP 5 WORST-CASE RUNNING TIME

- **Step 5:** Depending on the relationship between i and k, proceed accordingly.
  - The **lower**-bound on the number of elements that are in the ***low side*** or the ***high side*** of the partition is $\underline{3\left(\left\lceil \left\lceil \frac{n}{5} \right\rceil /2 \right\rceil - 2\right)} \geq$ ________________.
  - The algorithm needs to recurse on

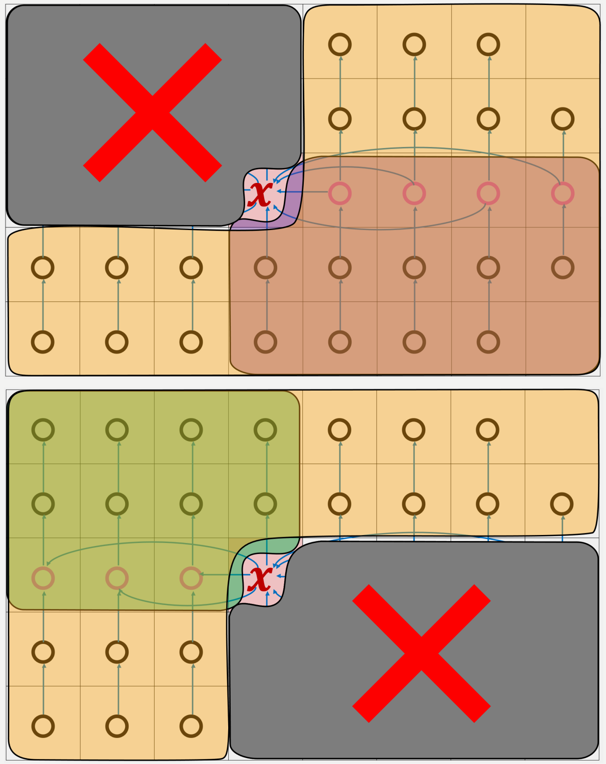


SELECT ALGORITHM
STEP 5 WORST-CASE RUNNING TIME

- **Step 5:** Depending on the relationship between i and k, proceed accordingly.
  - The algorithm needs to recurse on ***at most***

    ____________________ $\leq$ _______________ elements.
  - In the **worst**-case scenario, the running time of step 5 is

    $T\left(\frac{7n}{10}+6\right)$.



SELECT ALGORITHM
TOTAL RUNNING TIME

- **Step 1**: Divide into groups of 5 elements. ______________________.
- **Step 2**: Find medians of the groups. ______________________.
- **Step 3**: Recurse on the medians to find the median-of-medians x. ______________.
- **Step 4**: Partition the GROUPS around x. ______________________.
- **Step 5**: Return or recurse. ______________________.

SELECT ALGORITHM
RUNNING TIME FUNCTION

- The running time function of the SELECT algorithm is

$$T(n) \leq \begin{cases} O(1) & if\ n < 140 \\ \boldsymbol{T(\lceil n/5 \rceil) + T\left(\frac{7n}{10} + 6\right) + O(n)} & if\ n \geq 140 \end{cases}$$

- The function $T(n) = O(\ \_\_\_\ )$.
  - Detailed proof can be found on page 222 of the textbook.

NEXT UP
SORTING

REFERENCE