Algorithms & Data Structures I
CSC 225

Ali Mashreghi

Fall 2018

Department of Computer Science, University of Victoria

Selection Problem

Input: Set A of n <u>distinct</u> numbers and an integer i, s.t. $1 \leq i \leq n$
Output: The element $x \in A$ that is the i^{th} <u>smallest</u> element.

input: $A = \{7, 3, 12, 5, 6, 8, 2, 9\}, i = 1$ (same as finding minimum)
output: 2

input: $A = \{7, 3, 12, 5, 6, 8, 2, 9\}, i = 4$ (the 4^{th} smallest element)
output: 6

- The i^{th} smallest element is also called the i^{th} ***order statistic***.

Selection problem

- Minimum is the **first** order statistic.
- Maximum is the $\boldsymbol{n}$th order statistic.
- Median is the $\lfloor(\boldsymbol{n}+\mathbf{1})/\mathbf{2}\rfloor th$ order statistic, which is informally "halfway point" of the set.

Selection problem

- Minimum is the **first** order statistic.
- Maximum is the $\boldsymbol{n}$th order statistic.
- Median is the $\lfloor(\boldsymbol{n}+\mathbf{1})/\mathbf{2}\rfloor th$ order statistic, which is informally "halfway point" of the set.

- **Question:** What is the easiest way to find ith smallest element?

Selection problem

- Minimum is the **first** order statistic.
- Maximum is the $\boldsymbol{n}$th order statistic.
- Median is the $\lfloor(\boldsymbol{n}+\mathbf{1})/\mathbf{2}\rfloor th$ order statistic, which is informally "halfway point" of the set.

- **Question:** What is the easiest way to find ith smallest element?
- **Answer:** First sort the array A, then return $A[i]$

- The trivial solution takes $O(n \log n)$ time, but we show that this can be done in linear time.

Selection problem

- An elegant algorithm is as follows:
- Partition the input array A, and let x be the pivot:

$$\underbrace{\boxed{\quad < x \quad}}_{k-1 \text{ elements}}\boxed{x}\underbrace{\boxed{\quad > x \quad}}_{n\ - k \text{ elements}}$$

- If $i = k$, the answer is x.
- If $\mathrm{i} < \ k$, recurse on the left part
- If $i > k$, recurse on the right part, looking for $(i\ - k)^{th}$ order statistic

Selection problem

Example: Find the **7th** smallest element ($i = 7$)

6	**14**	**3**	**9**	**2**	**5**	**11**	**8**

say **6** is the **pivot**, and the partition returns **index 4 (k = 4)**

3	**2**	**5**	**6**	**14**	**9**	**11**	**8**

< 6 $\qquad\qquad$ > 6

Selection problem

Example: Find the **7th** smallest element ($i = 7$)

6	**14**	**3**	**9**	**2**	**5**	**11**	**8**

say **6** is the **pivot**, and the partition returns **index 4 (k = 4)**

3	**2**	**5**	**6**	**14**	**9**	**11**	**8**

< 6 > 6

We recurse on the right side, since anything on or to the left of the pivot is 4th order statistic or lower. Therefore, we looking for the 3rd order statistic on the right side which is 7th order in the original array.

Selection problem

We recurse on right and find the **3rd** smallest element ($i = 3$)

14	**9**	**11**	**8**

Say this time 9 becomes the pivot, **index 2** in the subarray

8	**9**	**11**	**14**

< 9 > 9

Selection problem

We recurse on right and find the **1st** smallest element ($i = 1$)

11	**14**

Say this time 14 becomes the pivot, **index 2** in the subarray

11	**14**

$\underbrace{11}_{< 14}$

Selection problem

We recurse on left and look for the **1st** smallest element ($i = 1$)

$\mathbf{11}$

- Since there is only element left, 11 is the answer.
- 11 is the **7th order statistic** in the original array.

Selection problem

```
RANDOMIZED-SELECT(A, p, r, i)
    if p == r
        return A[p]
    q = PARANOID-PARTITION(A, p, r)
    k = q - p + 1
    if i == k
        return A[q]
    elseif i < k
        return RANDOMIZED-SELECT(A, p, q - 1, i)
    else return RANDOMIZED-SELECT(A, q + 1, r, i - k)
```

Analysis

- $\textsc{Paranoid-Partition}$ has an **expected running time** of $\Theta(n)$
- It partitions the array with a pivot that is greater than or equal to
  1. At least $n/4$ elements in A
  2. At most $3n/4$ elements in A

Analysis

- PARANOID-PARTITION has an **expected running time** of $\Theta(n)$
- It partitions the array with a pivot that is greater than or equal to
  1. At least $n/4$ elements in A
  2. At most $3n/4$ elements in A

- So, the **maximum size** for the part **we recurse on** is $3n/4$

$$T(n) = T\left(\frac{3n}{4}\right) + \Theta(n)$$

By Master method: $f(n) = \Theta(n), g(n) = n^{\log_{\frac{4}{3}} 1} = n^0 = 1$

Therefore $T(n) = \Theta(n)$ is the worst-case expected running time.

Deterministic algorithm

- It’s possible to solve the selection problem **deterministically** in **linear time**, as well.

Deterministic algorithm

- It's possible to solve the selection problem **deterministically** in **linear time**, as well.

- The algorithm was proposed in 1973 by

Blum, Floyd, Pratt, Rivest, and Tarjan

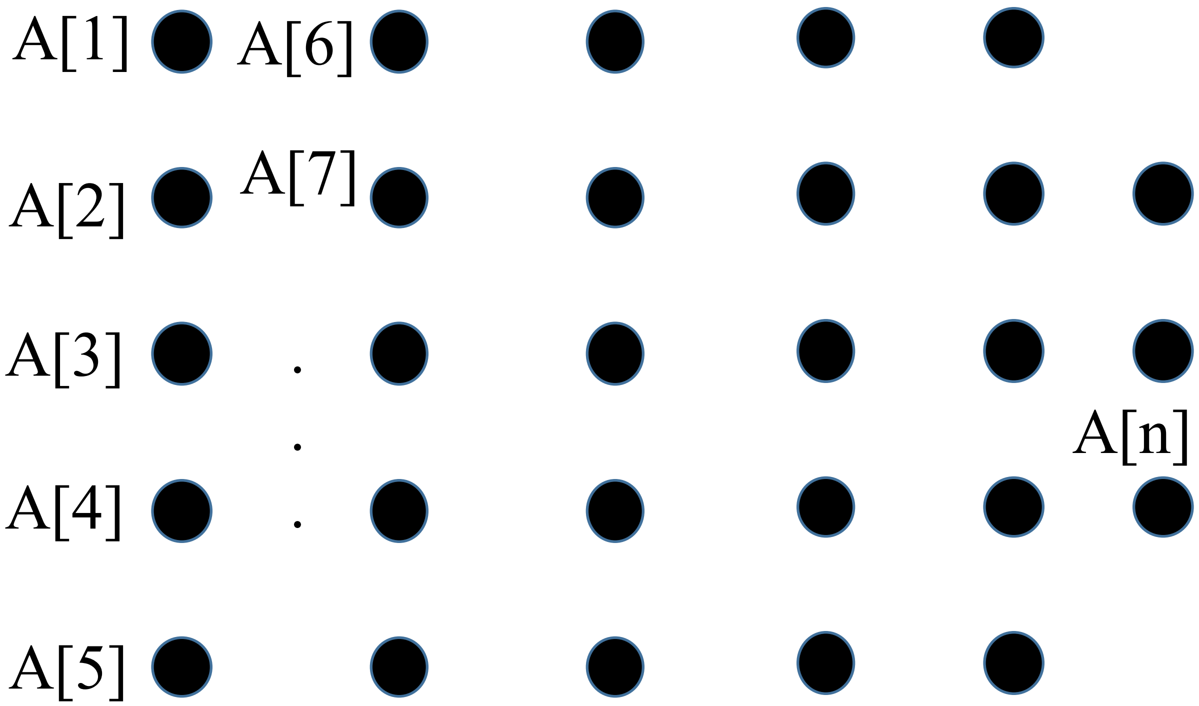
Turing award winners

Inventor of CAPTCHA

Inventor of RSA encryption

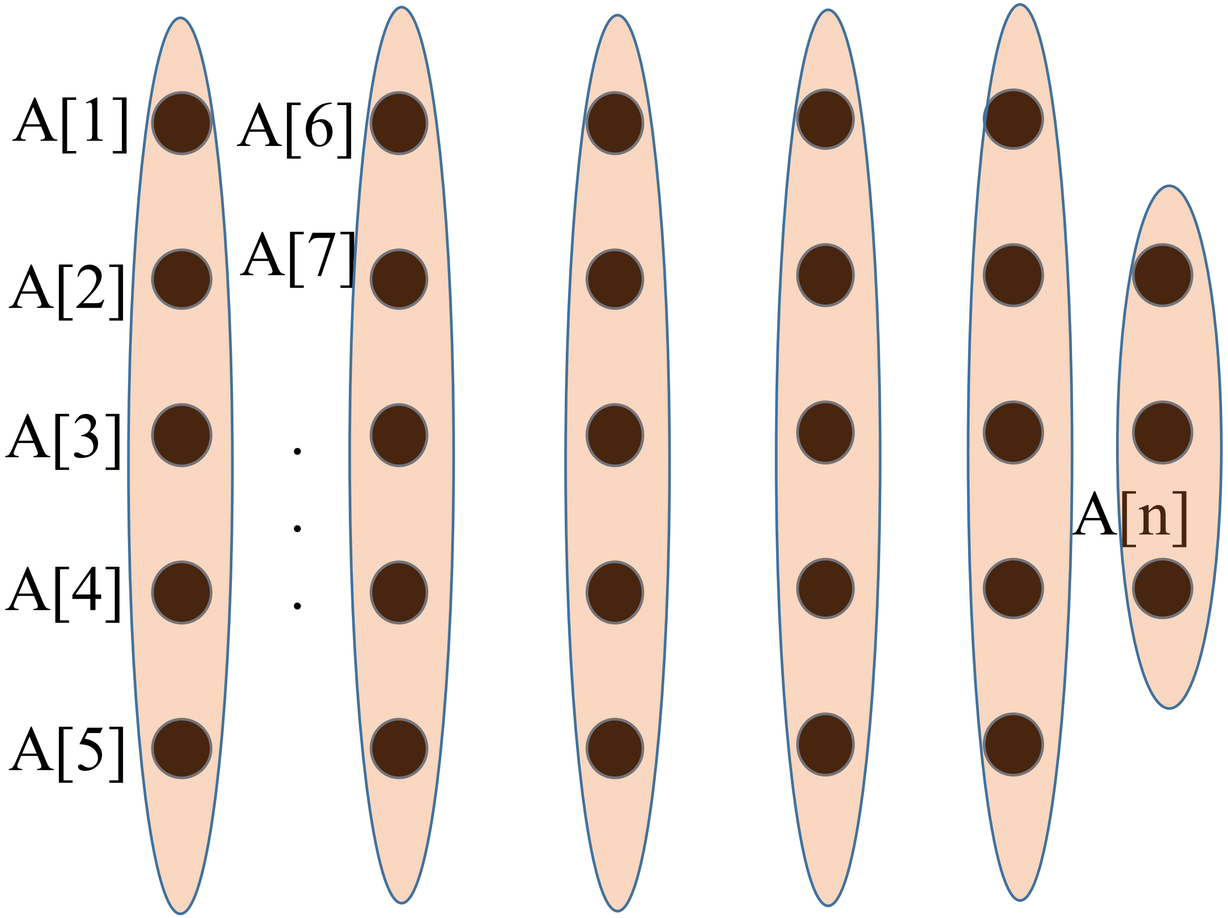
Deterministic algorithm

- Imagine the input array A like this (28 elements)



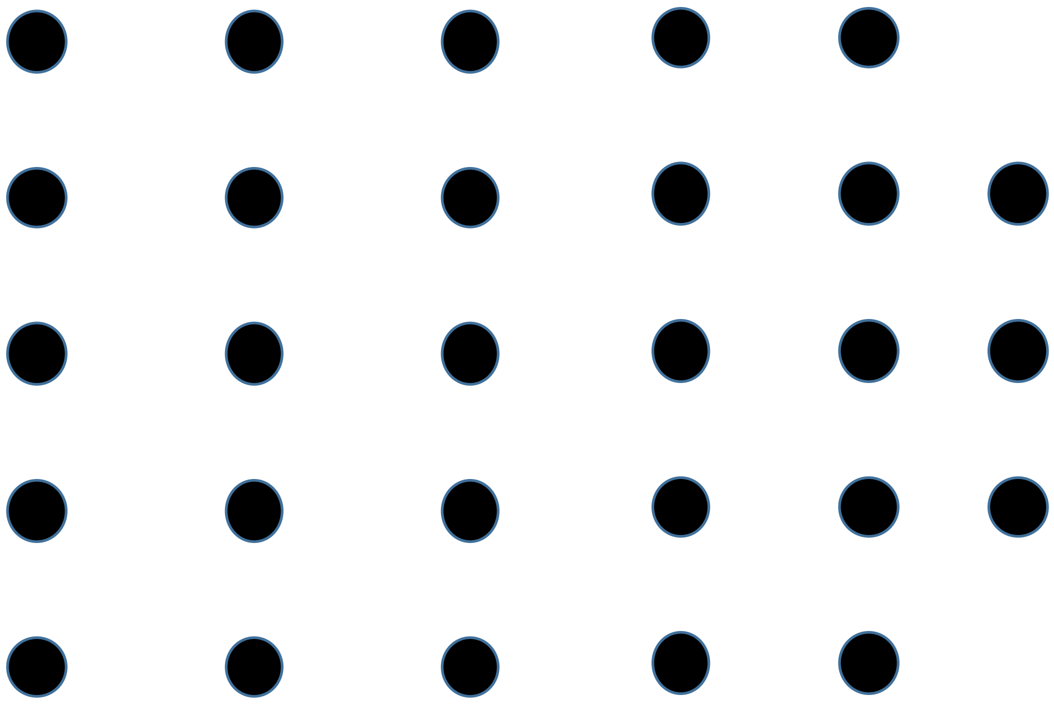
Deterministic algorithm

- Imagine the input array A like this



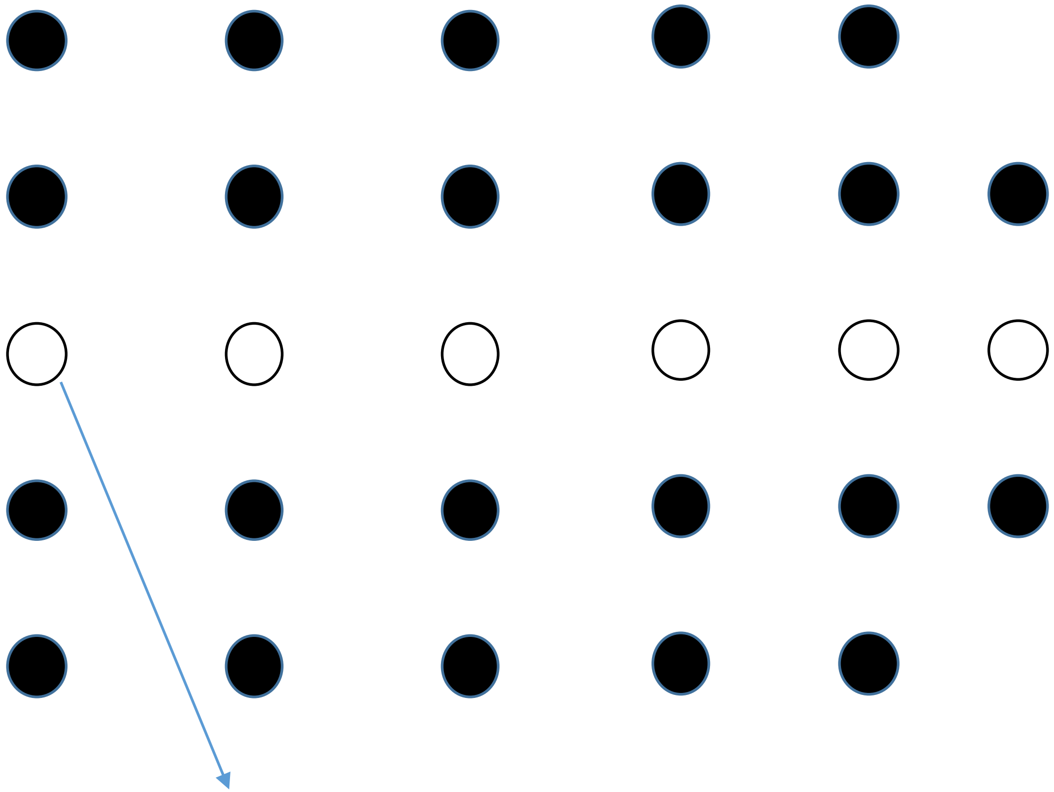
Deterministic algorithm

- We have $\lceil n/5 \rceil$ groups



Algorithm Select

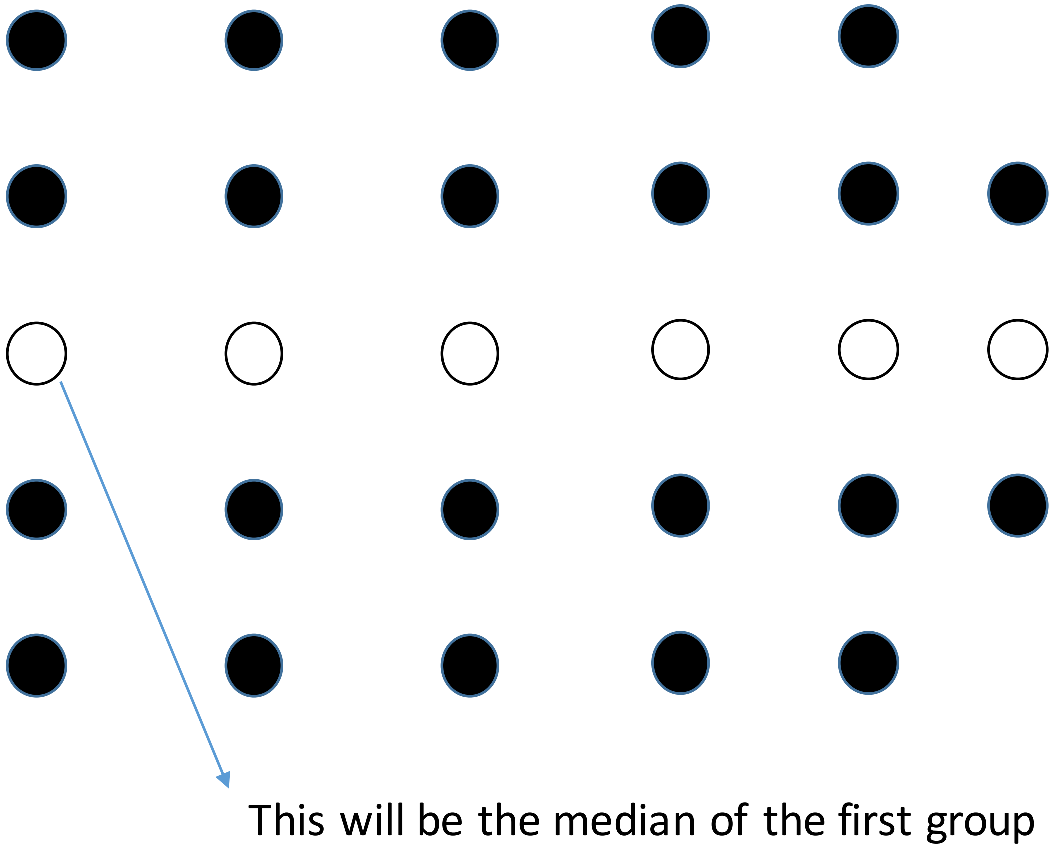
- **Step 1:** Divide the elements into $\lceil n/5 \rceil$ groups and find the median of each group. This can be done by sorting and picking the middle element (white circles).



This will be the median of the first group

Algorithm SELECT

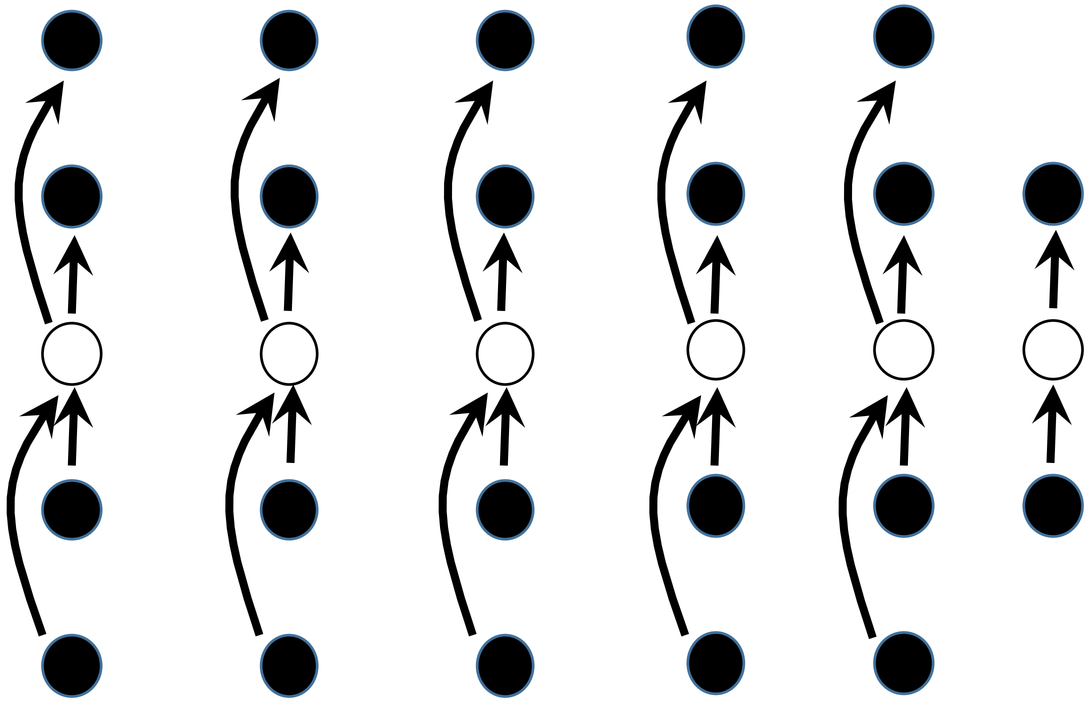
- **Step 1:** Divide the elements into $\lceil n/5 \rceil$ groups and find the median of each group. This can be done by sorting and picking the middle element (white circles).



This takes $\Theta(n)$ time, $\Theta(1)$ time for each group.

Algorithm Select

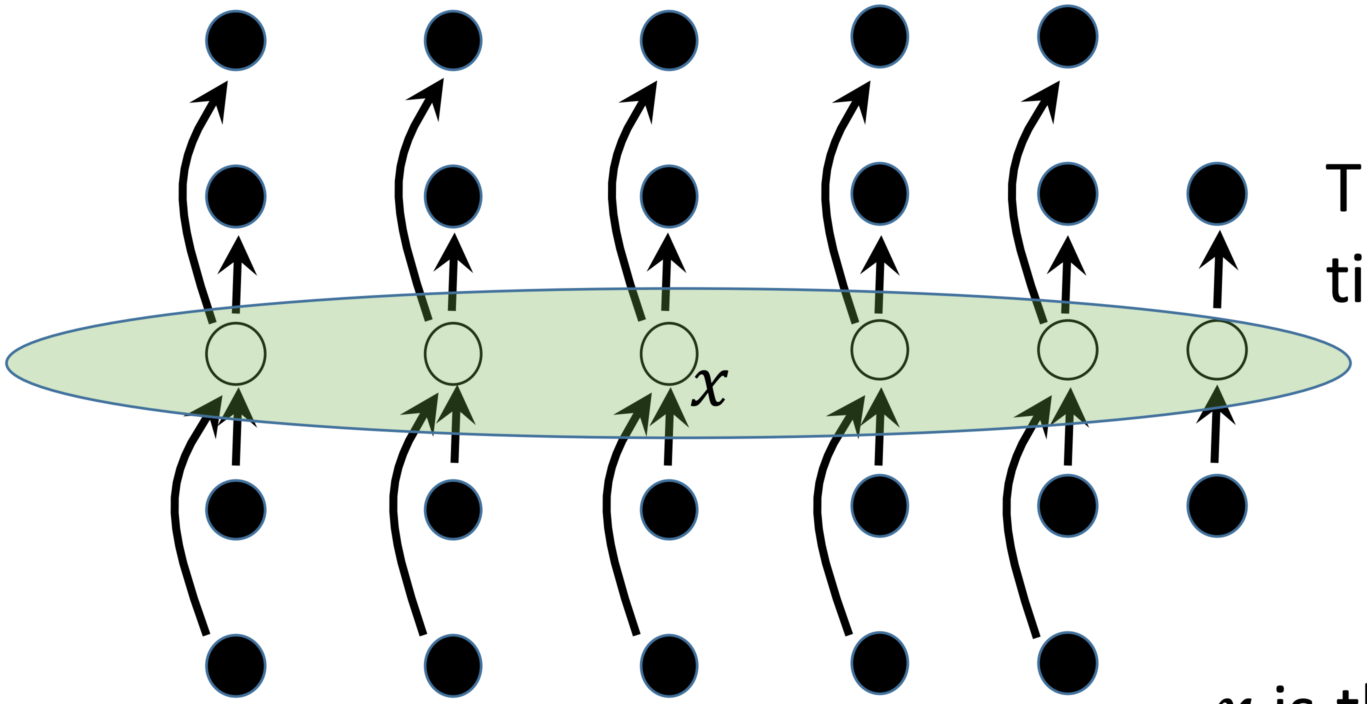
- **Step 1:** Divide the elements into $\lceil n/5 \rceil$ groups and find the median of each group. This can be done by sorting and picking the middle element (white circles).



$a \rightarrow b$ means
a is larger than b

Algorithm $\textsc{Select}$

- **Step 2:** Recursively call $\textsc{Select}$ to find the median of these $\lceil n/5 \rceil$ medians and call it x.

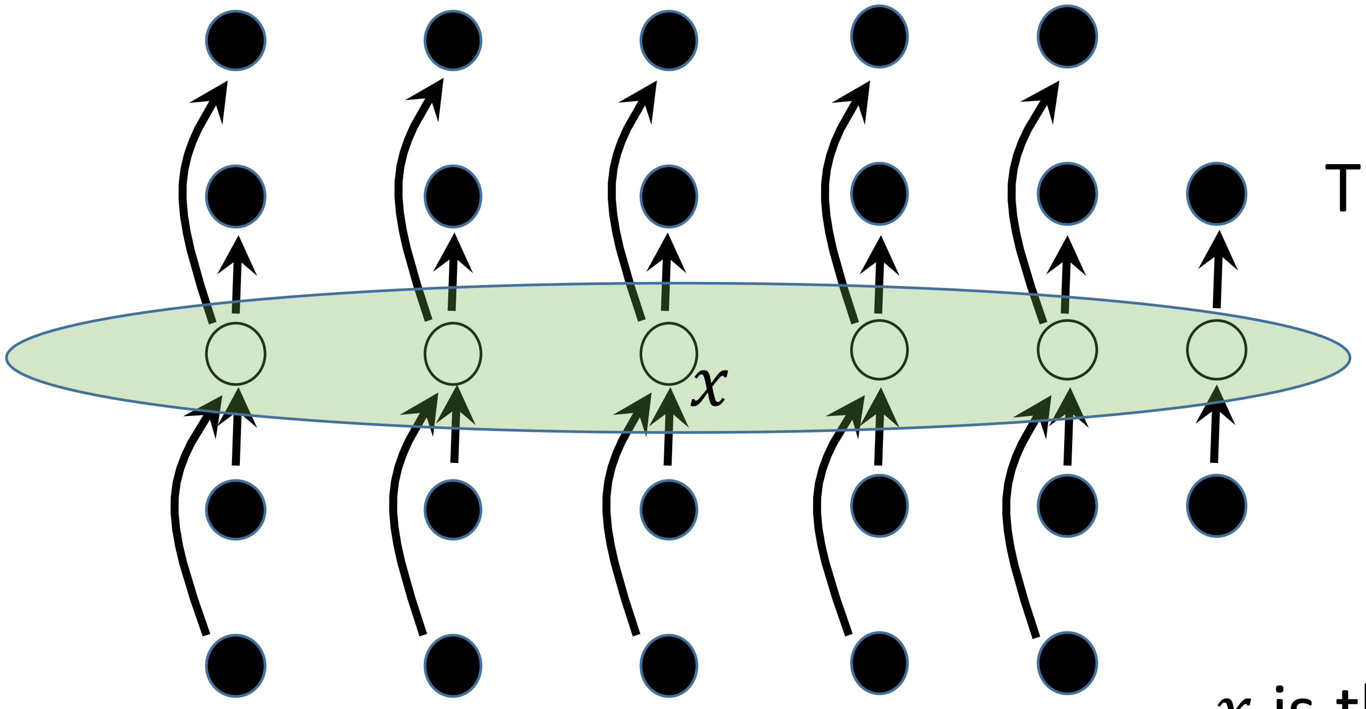


This takes $\mathrm{T}(\lceil n/5 \rceil)$ time

x is the median-of-medians

Algorithm SELECT

- **Step 3:** Use x to partition all n elements in array A



This takes $\Theta(n)$ time

x is the median-of-medians

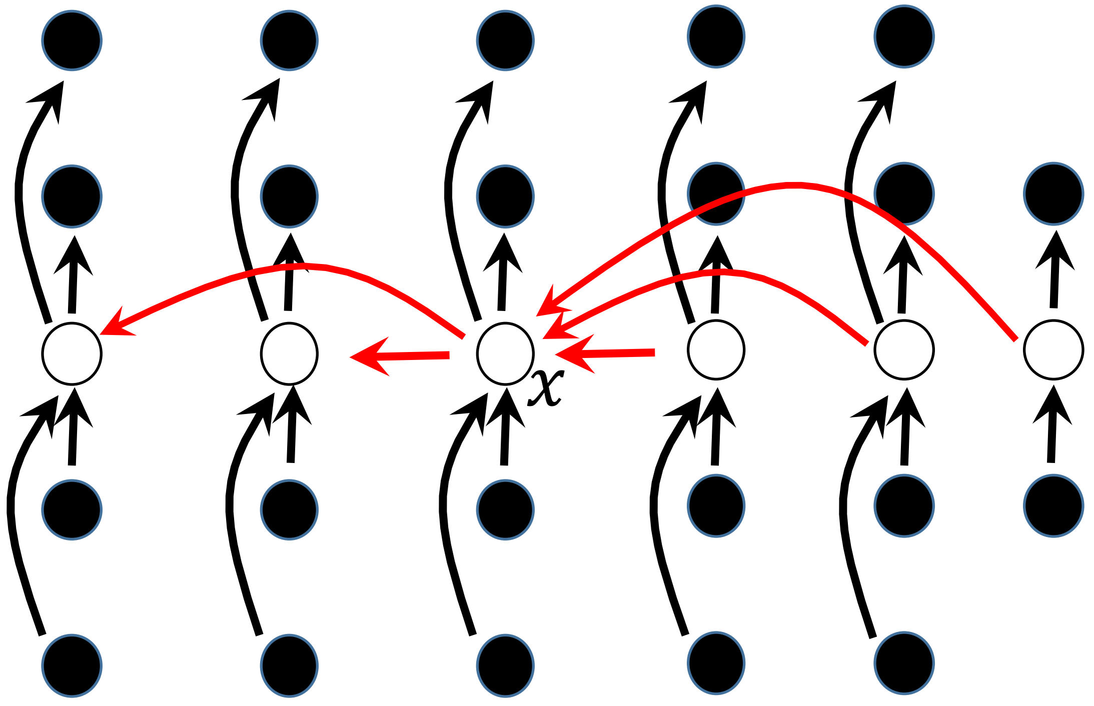
Algorithm Select

- The whole point of the algorithm is that the pivot x is not chosen randomly.

- So, we have to somehow argue that even using the median-of-medians as the pivot we will still get pretty decent partitions (not too big).

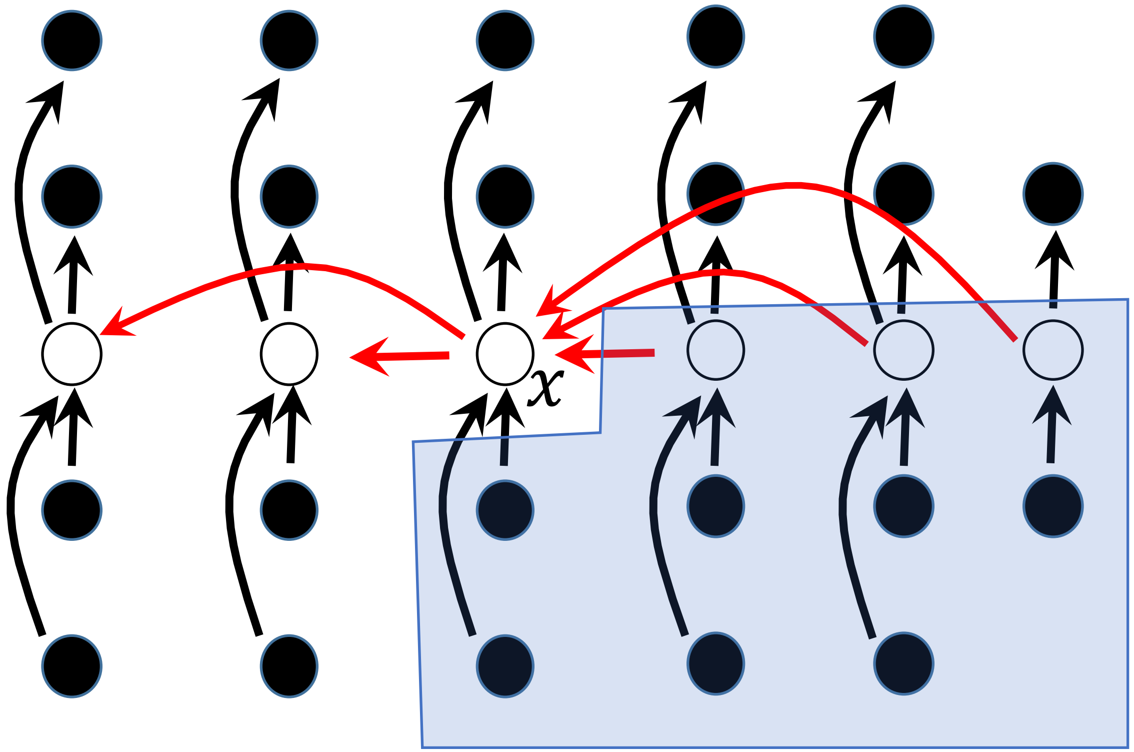
Algorithm Select

- We just know that x **is in the middle of the medians** (white circles).
- So, we can **draw** the groups of 5 like below.



- This picture is NOT the array after the partition; it just shows the relation between x and other elements.
- This is all imagination and not part of the algorithm.

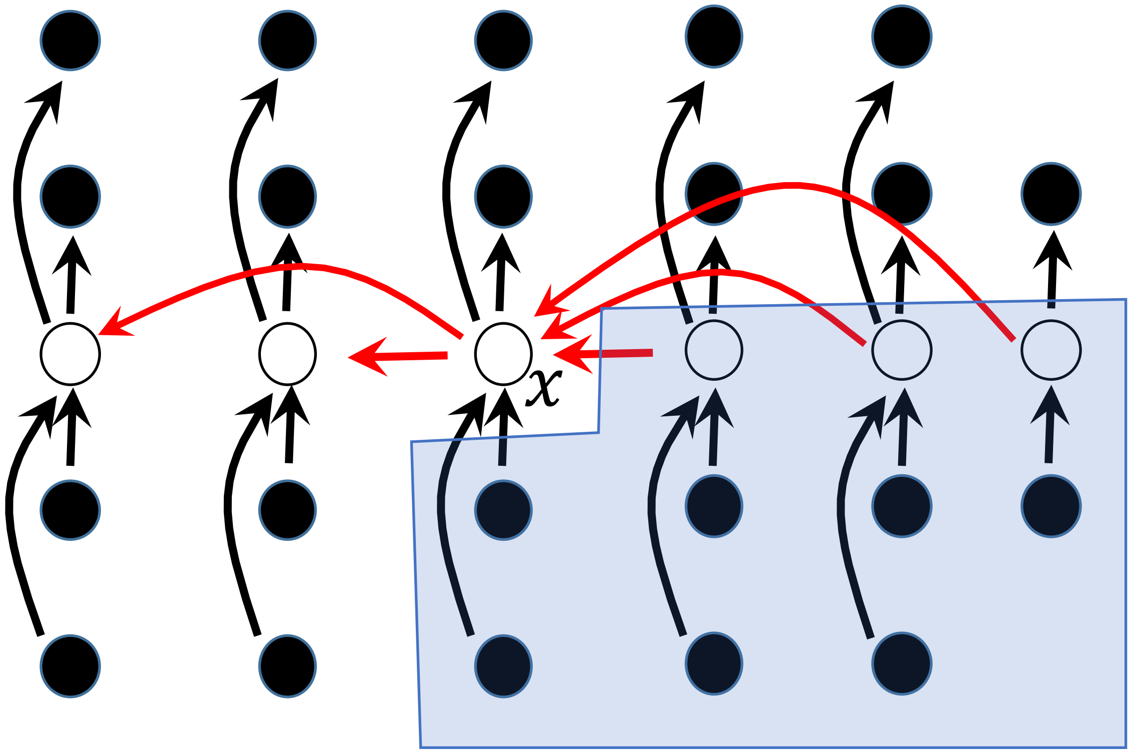
Algorithm Select



These elements are bigger than x because they have an arrow to the median of their group, and their median has an arrow x

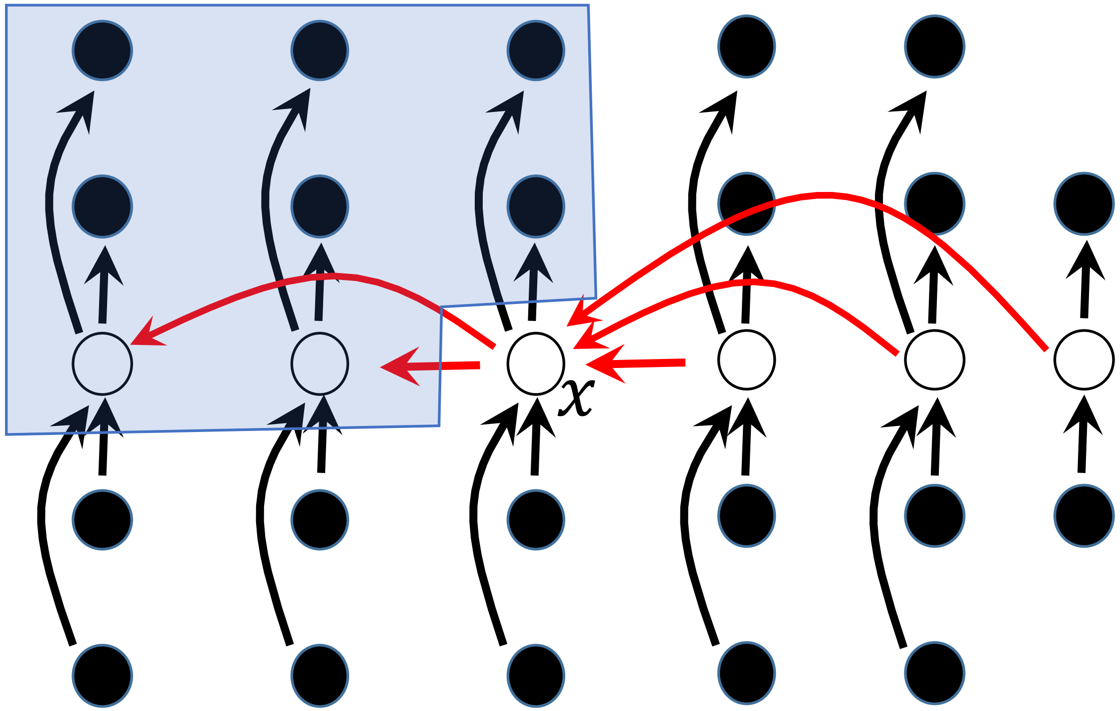
Algorithm Select

At least $\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2$ groups contribute 3 elements (excluding the last group and x's group); so, at least $3\left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$ elements here.



These elements are bigger than x because they have an arrow to the median of their group, and their median has an arrow x

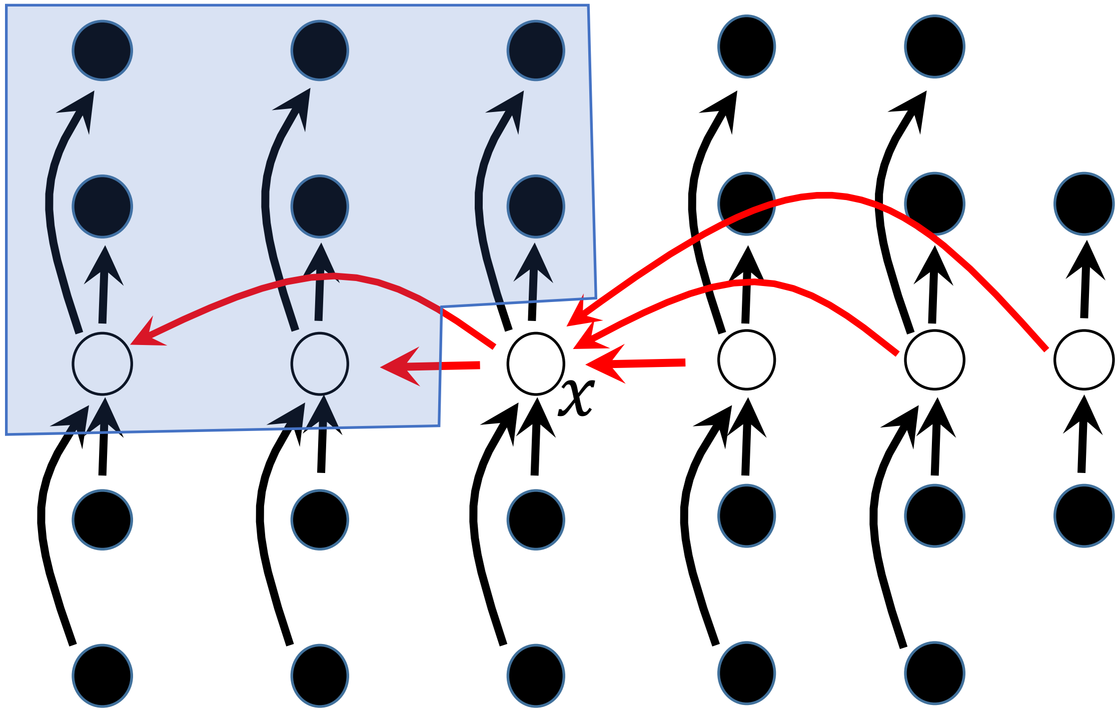
Algorithm Select



These elements are less than x because x has an arrow the the median of their group ,and their median has an arrow to them

Algorithm Select

Similarly, this part has a size of at least $\frac{3n}{10} - 6$



These elements are less than x because x has an arrow the the median of their group ,and their median has an arrow to them

Algorithm Select

- **Step 4:** Assume that after partition x is at index k, and we are looking for the ith smallest element.

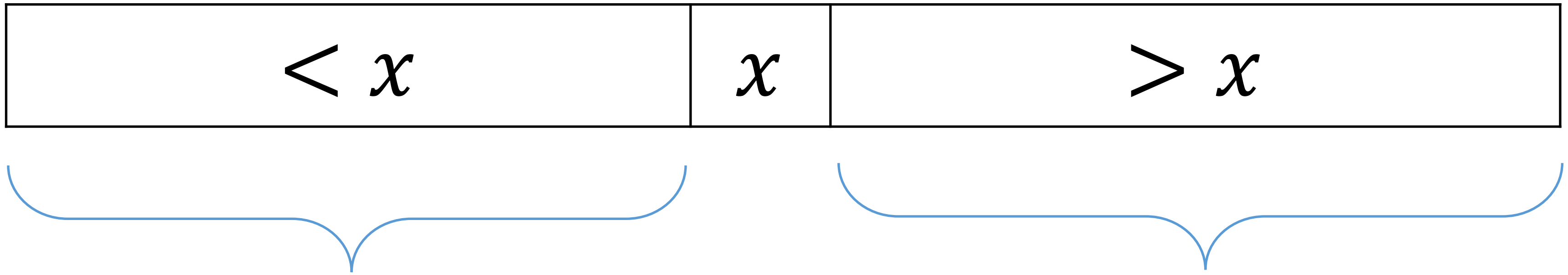
  if i $= k$: return x

  else if $i < k$: recurse on the left of x

  else: recurse on the right of x

$$\underbrace{\boxed{\quad < x \quad}}_{k-1 \text{ elements}}\boxed{x}\underbrace{\boxed{\quad > x \quad}}_{n \;-k \text{ elements}}$$

Algorithm Select



- **Question:** Knowing that x is roughly bigger than at least $\frac{3n}{10} - 6$ elements and less than at least $\frac{3n}{10} - 6$ elements, what is the maximum size of the part that we recurse on?

Algorithm Select

$< x$	x	$> x$

- **Question:** Knowing that x is roughly bigger than at least $\frac{3n}{10} - 6$ elements and less than at least $\frac{3n}{10} - 6$ elements, what is the maximum size of the part that we recurse on?
- **Answer:** $\frac{7n}{10} + 6$, this happens when for example when x is bigger than ***exactly*** $\frac{3n}{10} - 6$, and less than the other $n - \left(\frac{3n}{10} - 6\right) = \frac{7n}{10} + 6$

Algorithm Select

- **Step 4:** Assume that after partition x is at index k, and we are looking for the ith smallest element.
  if $k == i$: return x
  else if $k < i$: recurse on the left of x
  else: recurse on the right of x

- Therefore the time for this step is at most $T(\frac{7n}{10} + 6)$.

Analysis of $\textsc{Select}$

- $\boldsymbol{T(n) =}$
- **Step 1:** Divide into groups and find median

  $\boldsymbol{\Theta(n)}$
- **Step 2:** Find the median-of-medians x

  $\boldsymbol{T}(\lceil n/5 \rceil)$
- **Step 3:** Use x to partition array A

  $\boldsymbol{\Theta(n)}$
- **Step 4:** Compare i and k and recurse

  $\boldsymbol{T(\frac{7n}{10} + 6)}$

Analysis of $\text{S}\scriptstyle{\text{ELECT}}$

- $\boldsymbol{T(n)} =$
- **Step 1:** Divide into groups and find median

  $\boldsymbol{\Theta(n)}$
- **Step 2:** Find the median-of-medians x

  $\boldsymbol{T}(\lceil n/5 \rceil)$
- **Step 3:** Use x to partition array A

  $\boldsymbol{\Theta(n)}$
- **Step 4:** Compare i and k and recurse

  $\boldsymbol{T(\frac{7n}{10} + 6)}$

$$\boldsymbol{T(n) = T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + \Theta(n)}$$

Analysis of $\textsc{Select}$

- $\boldsymbol{T(n)} =$
- **Step 1:** Divide into groups and find median
  $\boldsymbol{\Theta(n)}$
- **Step 2:** Find the median-of-medians x
  $\boldsymbol{T}(\lceil n/5 \rceil)$
- **Step 3:** Use x to partition array A
  $\boldsymbol{\Theta(n)}$
- **Step 4:** Compare i and k and recurse
  $\boldsymbol{T}(\frac{\boldsymbol{7n}}{\boldsymbol{10}} + \boldsymbol{6})$

$$\boldsymbol{T(n) = T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + \Theta(n)}$$

Analysis of Select

$$\boldsymbol{T(n) = T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + \Theta(n)}$$

- This can be simplified as since ceiling and the constant 6 will only make a constant difference.

$$\boldsymbol{T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + c'n}$$

- We can use the substitution method to show $T(n) = O(n)$. The exact form of induction is $T(n) \leq cn$.

- Since $T(n)$ is also at least n, then $T(n) = \Theta(n)$.

```
SELECT(A, p, r, i)
1   n = r − p + 1
2   if n == 1
3       return A[1]
4   Divide A into ⌈n/5⌉ groups of size 5. /* one
        group might have size less than 5.
        A[1..5] is the first group, A[6..10] is the
        second, and so on */
5   Simply find the median of each group by sorting
6   Bring these medians to the beginning of array A
7   // j is the position of median in an array of size ⌈n/5⌉
8   j = ⌊(⌈n/5⌉ + 1)/2⌋
9   // x is the median-of-medians
10  x = SELECT(A, p, p + ⌈n/5⌉, j)
11  Use x as a pivot, and partition A /* need to modify
        the partition subroutine */
12  Let k be the index of x after partition
13  Based on whether i == k, i < k, or i > k,
        return x, recuse on the left, or recurse on right,
        respectively // just like RANDOMIZED-SELECT
```

This is a more detailed pseudocode based on CLRS section 9.3.

Note that we are calling Select for two different purposes, one is finding the median-of-medians (line 10), and the other is finding the ith smallest element (line 13).