Finding the $k^{th}$-smallest in $\Theta(n)$ worst-case time

- Like the randomized algorithm, we recursively partition the array. But now we guarantee a good split in 5 steps:

1. Divide the $n$ elements into $\lceil \frac{n}{5} \rceil$ groups of 5 elements, and $\leq 1$ group of $<5$ elements.

   \[
   \begin{array}{c}
   5 \\
   \{  \cdot \cdot \cdot \cdot x \cdot \cdot \cdot \cdot \cdot \cdot \cdot \} \\
   \{  \cdot \cdot \cdot \cdot \cdot \cdot \cdot \} \\
   \{  \ \} \\
   \{  \ \} \\
   \{  \ \} \\
   \{  \ \} \\
   \end{array}
   \]
   \[
   \lceil \frac{n}{5} \rceil \text{ groups}
   \]

2. Find the median of each group (by say running insertion sort on the $\leq 5$ elements and taking the middle element).
Finding the $k^{\text{th}}$-smallest, cont'd

(3) Recursively find the median $x$ of the $\lceil \frac{n}{5} \rceil$ medians found in step (2) (by recursively calling the algorithm with $k' = \left\lceil \frac{\lceil \frac{n}{5} \rceil + 1}{2} \right\rceil$).

(4) Partition the input array $A$ around element $x$ from step (3):

\[
\begin{array}{c}
\text{A} \\
\text{\{low side\}  \{high side\}}
\end{array}
\]

(5) Let $i$ be the rank of $x$.

- $k = i$: Return $x$.
- $k < i$: Recursively find $k^{\text{th}}$-smallest in the low side of $A$.
- $k > i$: Recursively find $(k-i)^{\text{th}}$-smallest in the high side.