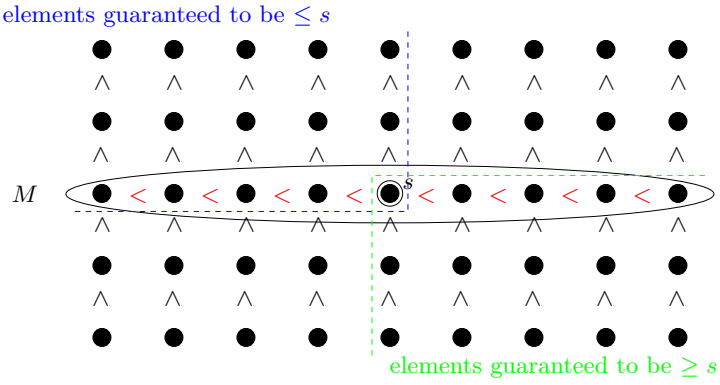
Deterministic order-statistics algorithm

Vassos Hadzilacos

Shown below is pseudocode for the deterministic order-statistics algorithm. It takes as input a non-empty array of numbers $A[1..n]$ and an integer k, $1 \le k \le n$, and returns the k-th smallest element of A. It is similar to the randomized order-statistics algorithm, but takes much greater care to select the splitter s (lines 5-8) so that it is guaranteed to be "good": at least a quarter of the elements of A are smaller than s and at least a quarter are larger; this way, the algorithm recurses on at most three-quarters of the elements.

The selection of the splitter is done as follows: The algorithm first partitions the elements of A into $\lfloor n/5 \rfloor$ groups of five elements each (line 5). There may be up to four elements left out, and these are ignored for the purposes of selecting the splitter. The algorithm them sorts each group of five elements, selects the median of each group (line 6), and forms the array M of these $\lfloor n/5 \rfloor$ medians (line 7). It then (recursively) finds the median of these medians, and this becomes the splitter s (line 8).

```
    D-SELECT(A, k)
1   if n < 50 then
2       sort A
3       return the k-th element of A
4   else
5       partition A into ⌊n/5⌋ groups of five elements each (up to four elements left out)
6       find the median or each group by sorting its five elements
7       M := array of ⌊n/5⌋ medians of the groups
8       s := D-SELECT(M, ⌈|M|/2⌉)
9       A⁻ := elements of A (including the up to four left out in line 5) that are less than s
10      A⁺ := elements of A (including the up to four left out in line 5) that are greater than s
11      if k ≤ |A⁻| then return D-SELECT(A⁻, k)
12      elsif k = |A⁻| + 1 then return s
13      else return D-SELECT(A⁺, k − |A⁻| − 1)
```



The figure above is a visualization to help explain why the splitter selected in this manner is guaranteed to have at least one quarter of the elements less than or equal to it, and at least one quarter greater than or equal to it. The dots represent the elements of A partitioned into groups of five (the up to four elements left out in line 5 are not shown). The "vertical" inequalities (shown in black) are established by the algorithm

in line 6, when it sorts each group of five elements. We emphasize that the "horizontal" inequalities (shown in red among the medians of the groups) are not actually established by the algorithm: the algorithm does ***not*** sort M! We ***imagine*** that the groups are sorted by their medians to help us see why about one quarter of the elements of A (those above and to the left of the broken line shown in blue) are guaranteed to be less than or equal to the splitter; and about one quarter of the elements of A (those below and to the right of the borken line shown in green) are guaranteed to be greater than or equal to the splitter. Therefore the number of elements in the array on which the algorithm recurses has at most three-quarters of the elements. A more careful argument follows.

Claim 1 *The number of elements in each of A^- and A^+ (lines 9–10) is at most $\lfloor 3n/4 \rfloor$.*

PROOF. The number of elements less than or equal to the splitter s is at least $3 \cdot \lceil \lfloor n/5 \rfloor / 2 \rceil$. To see why, first note that there are $\lfloor n/5 \rfloor$ medians of groups, and (the ceiling of) half of them are less than or equal to their median s. So, $\lceil \lfloor n/5 \rfloor / 2 \rceil$ medians are less than or equal to s. Each of these medians is less than or equal to three of the five elements of its group. So, there are at least $3 \cdot \lceil \lfloor n/5 \rfloor / 2 \rceil$ elements less than or equal to the splitter. By the same argument there are equally many elements greater than or equal to the splitter. Therefore, the larger of the arrays A^- and A^+ on which the algorithm may recurse has at most $n - 3 \cdot \lceil \lfloor n/5 \rfloor / 2 \rceil$ elements. (Note that here we are <u>not</u> ignoring the up to four elements of A that may have been left out when partitioning A into $\lfloor n/5 \rfloor$ groups of five elements each.)

Since we are computing A^- and A^+ in lines 9 and 10, we have that $n \geq 50$ (see line 1). We will show that, for $n \geq 50$, $n - 3 \cdot \lceil \lfloor n/5 \rfloor / 2 \rceil \leq \lfloor 3n/4 \rfloor$.[1] We have:

$$
\begin{aligned}
n - 3 \cdot \left\lceil \frac{\lfloor n/5 \rfloor}{2} \right\rceil &\leq n - 3 \cdot \frac{\lfloor n/5 \rfloor}{2} && [\text{because } \lceil x \rceil \geq x, \forall x] \\
&\leq n - 3 \cdot \frac{n/5 - 1}{2} && [\text{because } \lfloor x \rfloor \geq x - 1, \forall x] \\
&\leq n - \frac{3n - 15}{10} \\
&= \frac{7n + 15}{10} \\
&\leq \frac{3n}{4} - 1 && [\text{true } \forall n \geq 50] \\
&\leq \left\lfloor \frac{3n}{4} \right\rfloor. && [\text{because } x - 1 \leq \lfloor x \rfloor, \forall x]
\end{aligned}
$$

□

Therefore the running time $T(n)$ of the algorithm satisfies the following recurrence:

$$
T(n) \leq \begin{cases} c, & \text{if } n < 50 \\ T(\lfloor n/5 \rfloor) + T(\lfloor 3n/4 \rfloor) + cn, & \text{otherwise} \end{cases} \qquad (*)
$$

for some constant c. In the recursive case, the term $T(\lfloor n/5 \rfloor)$ is the time used by the first recursive call to D-SELECT, which finds the median of M; the term $T(\lfloor 3n/4 \rfloor)$ is (an upper bound on) the time used by the second recursive call, made to the array A^- or A^+; and cn is the linear term that accounts for the

[1] This bound actually holds, not just for all $n \geq 50$, but for all $n \geq 5$ with the exception of $n = 13$ and $n = 14$. At small scales, however, bounding the rounding functions (floor and ceiling) with the simple inequalities $x - 1 \leq \lfloor x \rfloor \leq x \leq \lceil x \rceil$ has an outsize effect and does not yield the desired bound. The only way I could show that this inequality holds for all $n \geq 5$ except $n = 14$ and $n = 15$ was to consider many cases, resulting in an inelegant and laborious proof. Luckily, when $n \geq 50$, the effects of the simple bounds on the rounding functions are small enough, yielding the less laborious proof shown here.

time to sort the $\lfloor n/5 \rfloor$ groups of five elements each and to form the array of medians M (lines 5-7), and the time to form the arrays A^- and A^+ (lines 9 and 10), once the splitter is determined.

Using induction we can prove that $T(n) \leq 20cn$. This is obvious by $(*)$ for the base cases $n < 50$. For the induction step, for any $n \geq 50$, assume that the claim holds for values less than n. Then we have

$$\begin{aligned}
T(n) &\leq T(\lfloor n/5 \rfloor) + T(\lfloor 3n/4 \rfloor) + cn && \text{[by } (*)\text{]} \\
&\leq 20c\lfloor n/5 \rfloor + 20c\lfloor 3n/4 \rfloor + cn && \text{[by the induction hypothesis]} \\
&\leq 20c(n/5) + 20c(3n/4) + cn && \text{[since } \lfloor x \rfloor \leq x, \forall x\text{]} \\
&= 4cn + 15cn + cn \\
&= 20cn,
\end{aligned}$$

as wanted.

Therefore, the (worst-case) running time of D-SELECT is $O(n)$, where n is the number of elements in the array A. This is in contrast to the randomized algorithm R-SELECT, whose ***expected*** running time is $O(n)$ but whose ***worst-case*** running time is $O(n^2)$.