Extended Algorithms Courses COMP3821/9801

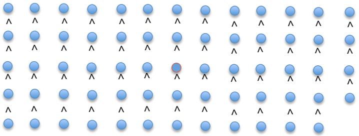
Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

Linear Time Deterministic Order Statistics

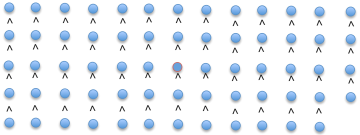
- **Problem:** Given n elements, select the i^{th} smallest element.
- **Main idea:** Use a recursive call of the very same algorithm to choose a good pivot!

- **Algorithm Select**(n, i) :
  1. Split the numbers in groups of five (the last group might contain less than 5 elements);
  2. Order elements of each group in an increasing order by brute force;

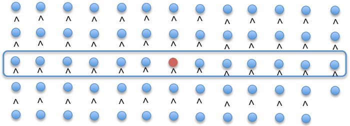


Deterministic Linear Time Algorithm for Order Statistics

- **Algorithm Select**(n, i) :
  - Split the numbers in groups of five (the last group might contain less than 5 elements);
  - Order them by brute force in an increasing order;



  - Take the collection of all $\lfloor \frac{n}{5} \rfloor$ middle elements of each group (i.e., the medians of each group of five)



  - Apply recursively SELECT algorithm to find the median p of this collection;

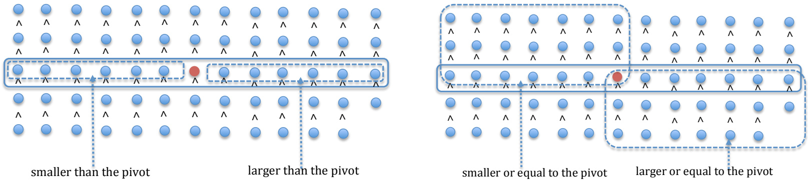
Deterministic Linear Time Algorithm for Order Statistics

- **Algorithm Select**(n, i) **continued:**
  - partition all elements using p as a pivot;
  - Let k be the number of elements in the subset of all elements smaller than the pivot p;
  - **if** $i = k$ **then return** p
  - **else if** $i < k$ **then** recursively SELECT the i^{th} smallest element of the set of elements smaller than the pivot;
  - **else** recursively SELECT the $(i - k)^{th}$ smallest element of the set of elements larger than the pivot;

- **Note:** This algorithm is the same as RAND-SELECT except for the way how we chose the pivot.
  - instead of choosing pivot randomly we called recursively the very same algorithm to pick the pivot as the median of the middle elements of the groups of five elements.

Deterministic Linear Time Algorithm for Order Statistics

- What have we accomplished by such a choice of the pivot?



- Note that at least $\lfloor (n/5)/2 \rfloor = \lfloor n/10 \rfloor$ group medians are smaller or equal to the pivot; and at least that many larger than the pivot;
- But this implies that at least $\lfloor 3n/10 \rfloor$ of the total number of elements are smaller than the pivot, and that many elements larger than the pivot.
- (the same caveat: we are assuming all elements are distinct; otherwise we have to slightly tweak the algorithm to split all elements equal to the pivot evenly between the two groups.)

Deterministic Linear Time Algorithm for Order Statistics

- What is the run time of our algorithm?

$$T(n) \leq T(n/5) + T(7n/10) + Cn.$$

- Let us show that $T(n) < 11Cn$ for all n. Assume that this is true for all $k < n$ and let us prove it is true for n as well.
  - Note: this is a proof using the following form of induction: $\varphi(0)$ & $(\forall n)((\forall k < n)\varphi(k) \rightarrow \varphi(n)) \rightarrow (\forall n)\varphi(n)$.
- Thus, assume $T(n/5) < 11C \cdot n/5$ and $T(7n/10) < 11C \cdot 7n/10$; then

$$\begin{aligned} T(n) &\leq T(n/5) + T(7n/10) + Cn < 11C \cdot \frac{n}{5} + 11C \cdot \frac{7n}{10} + Cn \\ &= 109\frac{Cn}{10} < 11C \end{aligned}$$

which proves out statement that $T(n) < 11C \cdot n$.

Deterministic Linear Time Algorithm for Order Statistics

- Note that this algorithm is a genuine recursion (rather than just an iteration) so its execution involves lots of traffic on the machine stack, which makes this algorithm slow in practice; the randomised version of it, Rand-Select, significantly outperforms it.

- Similarly Rand-QuickSort in practice outperforms MergeSort, which, unlike Rand-QuickSort, is guaranteed to run in time $O(n \log n)$.

Partitioning robust for having many repetitions in the array:

PARTITION(A, m, n, p) *(partitioning $A[m..n]$ around a pivot p)*

$i \leftarrow m - 1; \quad fl \leftarrow True;$

for $j = m$ **to** n

    if $A[j] < p$

        then $i++;$

            exchange $A[i] \leftrightarrow A[j];$

    else if $A[j] = p$

        then if $fl = True$

                $i++;$

                exchange $A[i] \leftrightarrow A[j];$

                $fl = False;$

            else $fl = True;$

return i