The k-Selection Problem (Det.)

(Slides for ESTR2102)

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

The k-Selection Problem

Input

You are given a set S of n integers in an array, the value of n, and also an integer $k \in [1, n]$.

Output

The k-th smallest integer of S.

We will describe an algorithm solving the problem deterministically in $O(n)$ time.

Recall:

> Define the rank of an integer v in S as the number of elements in S smaller than or equal to v.

For example, the rank of 23 in $\{76, 5, 8, 95, 10, 31\}$ is 3, while that of 31 is 4.

A Deterministic Algorithm

We will assume that n is a multiple of 10 (if not, pad up to 9 dummy elements).

Step 1: Divide A into chunks of size 5, that is: (i) each chunk has 5 elements, and (ii) there are $n/5$ chunks.

Step 2: From each chunk, identify the median of the 5 elements therein. Collect all the $n/5$ medians into an array B.

Step 3: Recursively run the algorithm to find the median p of B.

A Deterministic Algorithm

Step 4: Find the rank r of p in A.
Step 5:

- If $r = k$, return p.
- If $r < k$, produce an array A' containing all the elements of A strictly less than p. Recursively find the k-th smallest element in A'.
- If $r > k$, produce an array A' containing all the elements of A strictly greater than p. Recursively find the $(k - r)$-th smallest element in A'.

Analysis

Lemma 1.

The value of r falls in the range from $\lceil (3/10)n \rceil$ to $\lceil (7/10)n \rceil + 7$.

Proof: Let us first prove the lemma by assuming that n is a multiple of 10.

Let C_1 be the set of chunks whose medians are $\leq p$.
Let C_2 be the set of chunks whose medians are $> p$.

Hence: $|C_1| = |C_2| = n/10$.

Analysis

Every chunk in C_1 contains at least 3 elements $\leq p$. Hence:

$$r \quad \geq \quad 3|C_1| = (3/10)n.$$

Every chunk in C_2 contains at least 3 elements $> p$. Hence:

$$r \quad \leq \quad n - 3|C_1| = (7/10)n.$$

It thus follows that when n is a multiple of 10, $r \in [(3/10)n, (7/10)n]$.

Analysis

Now consider that n is not a multiple of 10. Let n' be the lowest multiple of 10 at least n. Hence, $n \le n' < n + 10$. By our earlier analysis:

$$
\begin{aligned}
& (3/10)n' \le r \le (7/10)n' \\
\Rightarrow \quad & (3/10)n \le r \le (7/10)(n+10) = (7/10)n + 7 \\
\Rightarrow \quad & \lceil (3/10)n \rceil \le r \le (7/10)(n+10) < \lceil (7/10)n \rceil + 7
\end{aligned}
$$

where the last step used the fact that r is an integer. $\square$

Analysis

Let $f(n)$ be the worst-case running time of our algorithm on n elements.

We know that when n is at most a certain constant, $f(n) = O(1)$.

For larger n:

$$\begin{aligned} f(n) &= f(\lceil (n+10)/5 \rceil) + f(\lceil (7/10)n \rceil + 7) + O(n) \\ &= f(\lceil n/5 \rceil + 2) + f(\lceil (7/10)n \rceil + 7) + O(n) \end{aligned}$$

In the next talk, we will learn a powerful method for solving this recurrence, which gives $f(n) = O(n)$.