The k-Selection Problem (Talk 1)

[Notes for the Training Camp]

Yufei Tao

ITEE
University of Queensland

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.

Easy to do $O(n \log n)$ time—how?

We will now describe a simple randomized algorithm to solve the median selection problem in $O(n)$ expected time.

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 Randomized Algorithm

1. Randomly pick an integer v from S.
2. Get the rank of v—let it be r.
3. If r is not in $[n/3, 2n/3]$, repeat from Step 1.
4. Otherwise:
   - 4.1 If $k = r$, return v.
   - 4.2 If $k < r$, produce an array A containing all the integers of S strictly smaller than v. Recur on A by looking for the k-th smallest element in A.
   - 4.3 If $k > r$, produce an array A containing all the integers of S strictly larger than v. Recur on A by looking for the $(k - r)$-th smallest element in A.

Observation: A has at most $2n/3$ elements left!

Running Time Analysis

Step 1 takes $O(1)$ time.
Step 2 takes $O(n)$ time.

How many times do we have to repeat the above two steps?

At Step 3, with a probability $1/3$ we can proceed to Step 3 $\Rightarrow$ need to repeat only 3 times in expectation!

Running Time Analysis

Let $f(n)$ be the expected running time of our algorithm on an array of size n.

We know:

$$f(n) \quad \leq \quad c \cdot n + f(\lceil 2n/3 \rceil).$$

for some constant c.

Solving the recurrence gives $f(n) = O(n)$.