Selection

Input: sequence S of n numbers, and an integer k with $1 \leq k \leq n$

Output: k-th smallest element in S.

$k = 1$: smallest element in S

$k = n$: largest element in S

$k = n/2$: median in S

Algorithm: ① sort S

② return the element at position k in the sorted sequence.

Running time: $O(n \log n)$.

We will see: $O(n)$ time is possible.

∴ we can find the k-th smallest element <u>without</u> sorting S.

Algorithm Select(S, k):

// S is a sequence of numbers, $1 \leq k \leq |S|$

if $|S| = 1$: return the only element of S

if $|S| \geq 2$:

  choose an element p in S;   (pivot)

  divide S into $S_<$, $S_=$, $S_>$

$$\underbrace{\boxed{\quad < p \quad}}_{S_<}\underbrace{\boxed{\quad = p \quad}}_{S_=}\underbrace{\boxed{\quad > p \quad}}_{S_>}$$

  if $k \leq |S_<|$: run Select$(S_<, k)$

  else if $k > |S_<| + |S_=|$:

    run Select$(S_>, k - |S_<| - |S_=|)$

  else return p

Since $p \in S_=$: $|S_=| \geq 1$

$\therefore$ recursive call on a sequence of length $< |S|$

$\therefore$ algorithm terminates

Running time : depends on the pivot p.

worst case :
- S is sorted already
- $k = 1$
- in each recursive call : p = largest element
- $O(n^2)$ running time

good case : in each recursive call : p = median

this gives the recurrence $T(n) = n + T(n/2)$

$$\therefore T(n) \leq \sum_{i=0}^{\infty} n/2^i = 2n = O(n).$$

how to get the "good case" :

in each recursive call : choose p randomly.

intuition: on average, p will be close to the median

~~(for details : COMP 4804)~~

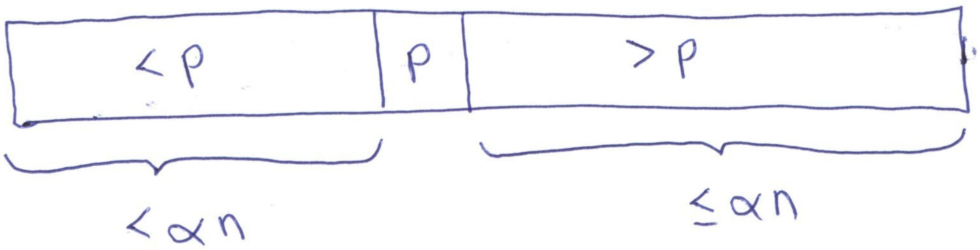
See my separate notes

Blum, Floyd, Pratt, Rivest, Tarjan (1973):
selection in $O(n)$ worst-case time

General approach:

input: sequence S of n numbers, $1 \le k \le n$.
(assume all numbers distinct)

Assume: in $O(n)$ time, we can find an element p in S, such that $|S_<| \le \alpha n$ and $|S_>| \le \alpha n$, where $0 < \alpha < 1$ is a constant.



Then the running time $T(n)$ satisfies

$$T(n) = n + T(\alpha n)$$

$$\therefore\ T(n) = n(1 + \alpha + \alpha^2 + \alpha^3 + \dots)$$

$$\le \frac{1}{1-\alpha} n = O(n)$$

Here is the algorithm:

Step 1: Divide the input sequence into $n/5$ groups, each of length 5.

Step 2: for $i = 1, 2, \dots, n/5$: compute the median of the i-th group; call this median m_i

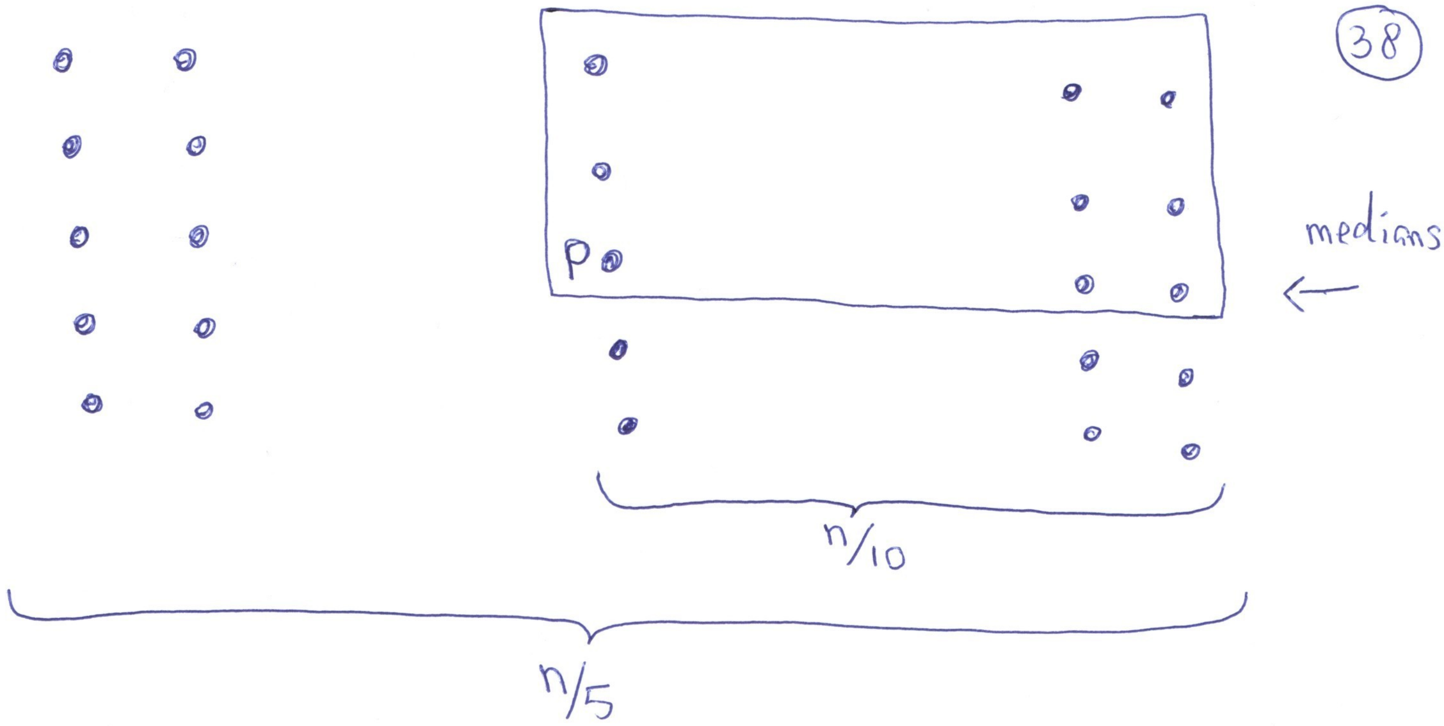
Step 3: Compute the median p of $m_1, m_2, \dots, m_{n/5}$.

Step 4: Use p as the pivot; proceed as on page 34.

Why is p a good pivot:

How many of the n elements are $\geq p$?

In the following figure, each column is a group of 5 elements; the middle row is the sequence of medians; for the purpose of the figure, assume each column is sorted, and the middle row is sorted from left to right.
↳ from bottom to top



all elements in the $3 \times \frac{n}{10}$ rectangle are $\geq p$

$\therefore \geq \frac{3}{10} n$ elements are $\geq p$

$\therefore \leq \frac{7}{10} n$ elements are $< p$

$\therefore |S_<| \leq \frac{7}{10} n$

By a symmetric argument: $|S_>| \leq \frac{7}{10} n$

$\therefore$ on page 36, we can take $\alpha = 7/10$

Define $T(n)$ = worst-case running time on an input of length n.

$$\begin{aligned} T(n) = \; & O(n) && \leftarrow \text{Step 1} \\ & + O(n) && \leftarrow \text{Step 2} \\ & + \; ? && \leftarrow \text{Step 3} \\ & + O(n) + T\left(\tfrac{7}{10}n\right) && \leftarrow \text{Step 4} \end{aligned}$$

How to do Step 3: Recursively compute the $\frac{n}{10}$-th smallest element of the sequence $m_1, \ldots, m_{n/5}$; this takes $T(n/5)$ time.

We obtain the recurrence

$$T(n) = n + T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right).$$

How to solve this:

- unfolding: becomes messy.
- Master Theorem: does not cover this recurrence.
- use induction to show that $T(n) = O(n)$.

<u>Claim:</u> $T(n) \leqslant cn$ for some constant c.

<u>Proof:</u> By choosing c sufficiently large, the claim is true for "small" n. (This is the base case of the induction.)

Let n be "large" and assume $T(m) \leq cm$ for all $1 \leqslant m < n$.

Then
$$
\begin{aligned}
T(n) &= n + T\left(\tfrac{n}{5}\right) + T\left(\tfrac{7}{10}n\right) \\
&\leqslant n + c \cdot \tfrac{n}{5} + c \cdot \tfrac{7}{10} n \\
&= n + \tfrac{9}{10} cn \\
&\leqslant cn \quad \text{if } n \leqslant \tfrac{1}{10} cn \text{ iff } c \geqslant 10.
\end{aligned}
$$

$\square$

<u>Conclusion:</u> The k-th smallest element in a sequence of n numbers can be computed in $O(n)$ time.