## Problem:

Given an array A of $n$ numbers and an index $i$ $(1 \leq i \leq n)$, find the $i^{th}$ smallest element of A.

E.g: If $i=1$ then the problem is equivalent to finding the minimum element.

If $i=n$ then it is a maximum element.

# Finding Median

$A = \{a_1, a_2, \dots a_n\}$ be set of $n$ elements.

Informally, median is the middle element in sorted set $A$.

if $n$ is odd median is the $i^{th}$ smallest element, where $i = \frac{n+1}{2}$.

if $n$ is even, there are two medians at $i^{th}$ and $(i+1)^{th}$ smallest elements,

where $i = \frac{n}{2}$ and $i = \frac{n}{2} + 1$

Lower median    Upper median

For simplicity, we use "the median" to refer to the lower median.

## First Approach:

Sort the entire array A, then o/p either $\left(\frac{n+1}{2}\right)^{th}$ element or $\left(\frac{n}{2}\right)^{th}$ element based on the Parity of n.

### Running time :

Sorting : $O(n\log n)$

o/p median : $O(1)$

$O(n\log n)$

**Q** Can we do better than this first approach ?

Ans: YES, $O(n)$ time

1973 : Median of median by Blum, Floyd, Pratt, Rivest and Tarjan.

**Def:** For a set $S$ of distinct numbers, we define the rank of an element $x \in S$ to be the number $k$ such that $x$ is the $k^{th}$ smallest element of $S$.

**E.g:** $S = \{5, 8, 2, 3\}$

$\text{rank}(5) = 3$.

## Algorithm to find the $i^{th}$ smallest element of an input array of $n$ distinct elements.

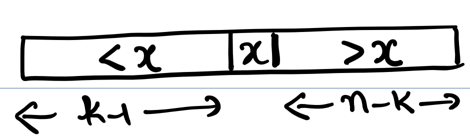
SELECT$(A, i)$

① Divide the $n$ elements of the input array into $\lfloor \frac{n}{5} \rfloor$ groups of 5 elements each and one group with $n \bmod 5$ elements

② Find the median of each of $\lceil \frac{n}{5} \rceil$ groups.

③ Use SELECT recursively to find the median (call it $x$) of these $\lceil \frac{n}{5} \rceil$ medians.

④ Partition the input array around the median of medians $x$. Say $\text{rank}(x) = k$

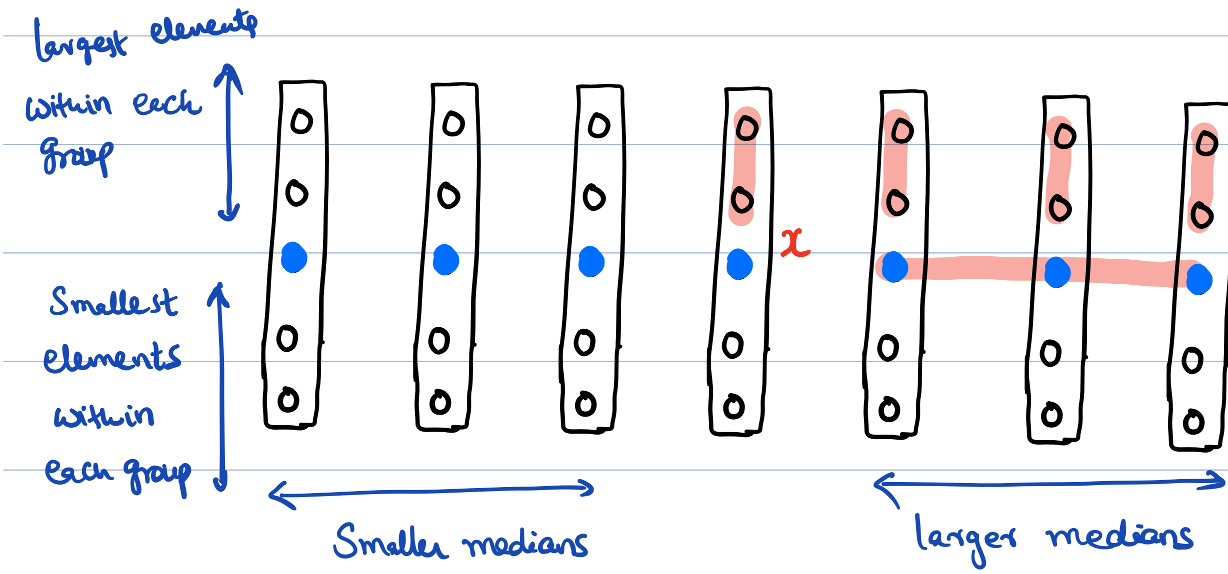


(5)

- If $i = k$ then return $x$
- Else, if $i < k$, use SELECT recursively by calling SELECT$(A[1, .. k-1], i)$
- Else, if $i > k$ use SELECT recursively by calling SELECT$(A[k+1, .. n], i-k)$

Lower bound on the number of elements that are greater than $x$.

Order groups with to their medians.




At least half of the medians found in Step 2 are greater than $x$.

Thus at least half of the $\lceil \frac{n}{5} \rceil$ groups contribute at least 3 elements that are greater than $x$, except one group that has fewer than 5 elements (if $5 \nmid n$) and the one group containing $x$ itself.

# of elements greater than $x$ is atleast

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

Similarly at least $\frac{3n}{10} - 6$ elements are less than $x$.

Thus, in the worst case step 5 calls SELECT recursively on at most $\frac{7n}{10} + 6$

## Recurrence for the worst case running time

Steps 1, 2, and 4 take $O(n)$ time

Step 3 takes $T\left(\left\lceil \frac{n}{5} \right\rceil\right)$ time and

Step 4 Partitioning around $x$ can be done in $O(n)$ time [Same as QUICK SORT Partitioning]

Step 5 takes time at most $T\left(\frac{7n}{10}+6\right)$

$$T(n) \leq \begin{cases} O(1) & \text{if } n < 140 \\ T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10}+6\right) + O(n) & \text{if } n \geq 140 \end{cases}$$

## Solving the recurrence:

We use Substitution method.

Guess: $T(n) = O(n)$

ie, $T(n) \le cn$ for suitable const $c$

and all $n > 0$.

Assume $T(m) \le cm$, $\forall$ $m < n$,

a is const.

$$T(n) \le c\left\lceil \frac{n}{5} \right\rceil + c\left(\frac{7n}{10} + 6\right) + an$$

$$\le \frac{cn}{5} + c + \frac{7cn}{10} + 6c + an$$

$$\le \frac{9c}{10}n + 7c + an$$

$$\le cn + \left(7c + an - \frac{cn}{10}\right)$$

$T(n) \le cn$ if $7c + an - \frac{cn}{10} \le 0$ —(1)

ie, $c \geq \frac{10an}{n-70}$ when $n > 70$

We assumed that $n \geq 140$, $\frac{n}{n-70} \leq 2$

So choosing $c \geq 20a$ will satisfy ①

$\therefore$ $T(n) = O(n)$.

Q: In the SELECT algorithm, the input elements are divided into groups of 5.

Will the algorithm work in linear time if they are divided into (a) groups of 7? (b) groups of 3.

Ans group of 7.

$x$ = median of medians

then the # of elements greater than $x$ is at least

$$4\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{7} \right\rceil \right\rceil - 2\right) \geq \frac{2n}{7} - 8$$

Similary at least $\frac{2n}{7} - 8$ elements are less than $x$.

Thus, in the worst case step 5 calls SELECT recursively on at most $n - \left(\frac{2n}{7} - 8\right) = \frac{5n}{7} + 8$

we get $T(n) = T\left(\left\lceil \frac{n}{7} \right\rceil\right) + T\left(\frac{5n}{7} + 8\right) + O(n)$

$= O(n)$ (Skipped intermediate steps)

Ans group of 3

$x$ = median of medians

then the # of elements greater than $x$ is at least

$$2\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{3} \right\rceil \right\rceil - 2\right) \geq \frac{n}{3} - 4$$

Similary at least $\frac{n}{3} - 4$ elements are less than $x$.

Thus, in the worst case Step 5 Calls SELECT recursively on at most $n - \left(\frac{n}{3} - 4\right) = \frac{2n}{3} + 4$

we get $T(n) = T\left(\left\lceil \frac{n}{3} \right\rceil\right) + T\left(\frac{2n}{3} + 4\right) + O(n)$

The Solution for above recurrence does not Satisfy $T(n) = O(n)$.

$T(n) \geq cn\log n$

$T(n) \geq c\left(\frac{n}{3}\right)\log\left(\frac{n}{3}\right) + c\left(\frac{2n}{3}\right)\log\left(\frac{2n}{3}\right) + O(n)$

$= \frac{cn}{3}(\log n - \log 3) + \frac{2cn}{3}(\log 2n - \log 3) + O(n)$

$= \frac{cn}{3}\log n - \frac{cn}{3}\log 3 + \frac{2cn}{3}\log 2$

$+ \frac{2cn}{3}\log n - \frac{2cn}{3}\log 3 + O(n)$

$= cn\log n - cn\log 3 + \frac{2}{3}cn\log 2 + O(n)$

$\geq cn\log n$

$\therefore$ $T(n)$ grows more quickly than $O(n)$.