Median:

- Given an unsorted list $A=[a_1,\dots,a_n]$ of n numbers, find the median of A.

For concreteness, say the median is the $\lceil \frac{n}{2} \rceil$-th smallest.

We'll find it useful to solve a more general problem:

- Given unsorted A & an integer k where $1 \le k \le n$, find the k^{th} smallest of A.

Easy algorithm: sort A & then output the k^{th} element of the sorted array.

Recall MergeSort takes $O(n \log n)$ time.

Today: $O(n)$ time algorithm.
(Due to Blum, Floyd, Pratt, Rivest & Tarjan 1973)

The approach is divide-and-conquer.

The basic approach is reminiscent of QuickSort:

- Choose a pivot p
- Partition A into $A_{<p}, A_{=p}, A_{>p}$
- Recursively sort the subgroups.

In our case, we only have to recurse on one of the subgroups. Consider the following example.

Example: $A = [5, 2, 20, 17, 11, 13, 8, 9, 11]$

Say $p = 11$ so: $A_{<p} = [5, 2, 8, 9]$, $A_{=p} = [11, 11]$, $A_{>p} = [20, 17, 13]$

if $k \leq 4$, then we want the k^{th} smallest in $A_{<p}$.

if $k = 5$ or 6, then we output 11.

if $k > 6$, then we want the $(k-6)^{th}$ smallest in $A_{>p}$.

Here is the pseudocode:

Select (A, k):

1) Choose a pivot p (How? This is the challenging step.)
2) Partition A into $A_{<p}$, $A_{=p}$ & $A_{>p}$
3) - If $k \leq |A_{<p}|$ then return$(\text{Select}(A_{<p}, k))$
   - If $|A_{<p}| < k \leq |A_{<p}| + |A_{=p}|$ then return(p)
   - If $k > |A_{<p}| + |A_{=p}|$ then

     return$(A_{>p}, k - |A_{<p}| - |A_{=p}|)$

We're aiming for an $O(n)$ running time.

So we want a recurrence like:

$$T(n) = T\left(\frac{n}{2}\right) + O(n)$$

which solves to: $T(n) = O(n)$.

Notice that $T(n) = T(.99n) + O(n)$

also solves to $T(n) = O(n)$

In fact, for any constant a where $0 \leq a < 1$,

$$T(n) = T(an) + O(n) = O(n)$$

Thus we say a pivot p is <u>good</u> if:

$$|A_{<p}| \leq \frac{3}{4}n \quad \& \quad |A_{>p}| \leq \frac{3}{4}n$$

For a good pivot our 1 recursive subproblem will be of size $\leq \frac{3}{4}n$.

Hence if we can find a good pivot in $O(n)$ time, then our running time satisfies:

$$T(n) \leq T\left(\frac{3}{4}n\right) + O(n) = O(n).$$

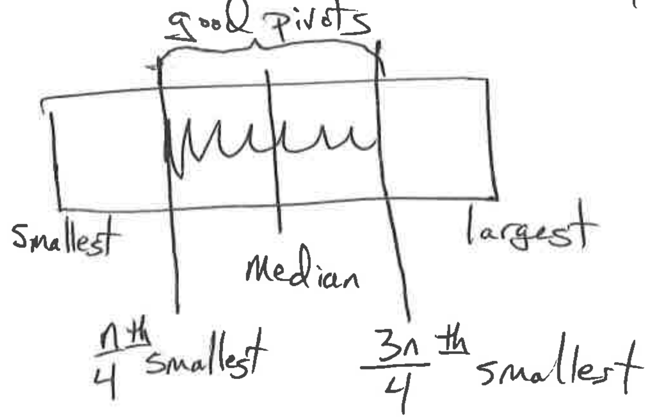
How do we find a good pivot?

When in doubt, just act randomly!

Suppose we pick a random element of A.
What's the chance that it is a good pivot?

Think of sorted A (this is just in this proof, we don't actually sort A)

Sorted A:



There are exactly $\frac{n}{2}$ good pivots
(maybe more if there are repeats)

$$\Pr\left(\begin{array}{c}\text{random element of } A \\ \text{is a good pivot}\end{array}\right) = \frac{\#\text{ of good pivots of } A}{\#\text{ of elements of } A} = \frac{\frac{n}{2}}{n} = \frac{1}{2}$$

So with prob. $\frac{1}{2}$ we get a good pivot.
In fact, in $O(n)$ time we can check if it is a good pivot or not.

Here is our approach to find a good pivot:

- Choose a random element of A to be p
- Check if p is a good pivot or not
  - if it's good, use it in the Select alg.
  - if it's not, repeat

How many times do we have to repeat?

— Twice in expectation.

Yields $O(n)$ expected run time for the entire algorithm.

But we want an algorithm that's guaranteed to finish in $O(n)$ time, so more work to do!

Let's reverse-engineer a bit more:

Recall we're aiming for a recurrence:

$$T(n) = T(\tfrac{3}{4}n) + O(n)$$

Because for $a<1$, $T(n) = T(an) + O(n) = O(n)$

But we have some slack, our subproblem can be of size $.999n$, instead of $.75n$.

What if we spend $T(.2n)$ time to find p?

Then our running time satisfies:

$$T(n) \leq T(.2n) + T(.75n) + O(n)$$

this solves to: $T(n) = O(n)$

In general, for a & b where $a+b<1$,

$$T(n) = T(an) + T(bn) + O(n)$$

solves to: $T(n) = O(n)$.

So we have $O(n) + T(.2n)$ time to find a good pivot.

$T(.2n)$ means we recursively find the median of a subset S of A where $|S| = .2n = \frac{n}{5}$

How to choose S?

Easy scheme: Let $S = [a_1, \ldots, a_{\frac{n}{5}}] = 1^{st}$ $\frac{n}{5}$ elements of A.

Then worst case, S are the $\frac{n}{5}$ smallest elements of A.

Hence, $p = \text{median}(S) = \frac{n}{10}^{th}$ smallest of A.

Therefore, $|A_{>p}| \le \frac{9n}{10}$,

So our running time satisfies:

$$T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{9n}{10}\right) + O(n)$$

but this doesn't solve to $O(n)$

because $\frac{1}{5} + \frac{9}{10} > 1$.

Need to choose S in a more clever manner.

We want a subset S that is representative of A.

For each $x \in S$, we want that:

a few elements of A are $\geq x$

& a few " are $\leq x$

How to achieve this?

Look at a group of 5 elements of A,

Call them x_1, x_2, x_3, x_4, x_5

Sort them, so say: $x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5$.

Let $m = x_3 = \text{Median}(x_1, x_2, x_3, x_4, x_5)$

Note: $x_1, x_2 \leq m$

& $x_4, x_5 \geq m$.

That gives the idea for our algorithm.

Break A into $\frac{n}{5}$ groups of 5 elements each.

For simplicity assume n is a power of 5.

Call the groups $G_1, G_2, \ldots, G_{\frac{n}{5}}$.

Since each group has $O(1)$ elements,
we can sort 1 group in $O(1)$ time
& we can sort all the groups in $O(n)$ time.

Let $m_i = \text{median}(G_i)$

Let $S = \{m_1, m_2, \ldots, m_{\frac{n}{5}}\}$ be the medians of all $\frac{n}{5}$ groups.

This is our representative S.

Then we set $p = \text{median}(S)$.

Here is the Pseudocode:

FastSelect(A, k):

input: unsorted $A = [a_1, \ldots, a_n]$ where n is a power of 5
& integer k where $1 \leq k \leq n$

output: k^{th} smallest of A

1) Break A into $\frac{n}{5}$ groups of 5 elements each.
Call these groups $G_1, G_2, \ldots, G_{\frac{n}{5}}$
2) For $i = 1 \to \frac{n}{5}$, sort G_i & let $m_i = \text{median}(G_i)$
3) Let $S = \{m_1, m_2, \ldots, m_{\frac{n}{5}}\}$
4) $p = \text{FastSelect}(S, \frac{n}{10})$ (Hence, p is the median of S)
5) Partition A into $A_{<p}$, $A_{=p}$, & $A_{>p}$.
6) - If $k \leq |A_{<p}|$ then:
return$(\text{FastSelect}(A_{<p}, k))$
- If $|A_{<p}| < k \leq |A_{<p}| + |A_{=p}|$ then:
output p
- If $k > |A_{<p}| + |A_{=p}|$ then:
return$(\text{FastSelect}(A_{>p}, k - |A_{<p}| - |A_{=p}|))$

Claim: P is a good pivot.

Therefore the running time of the algorithm satisfies:

$$T(n) \leq \underbrace{T\left(\frac{n}{5}\right)}_{\text{step 4}} + \underbrace{T\left(\frac{3n}{4}\right)}_{\text{step 6}} + O(n)$$

Since $\frac{1}{5} + \frac{3}{4} < 1$

we have: $T(n) = O(n)$.

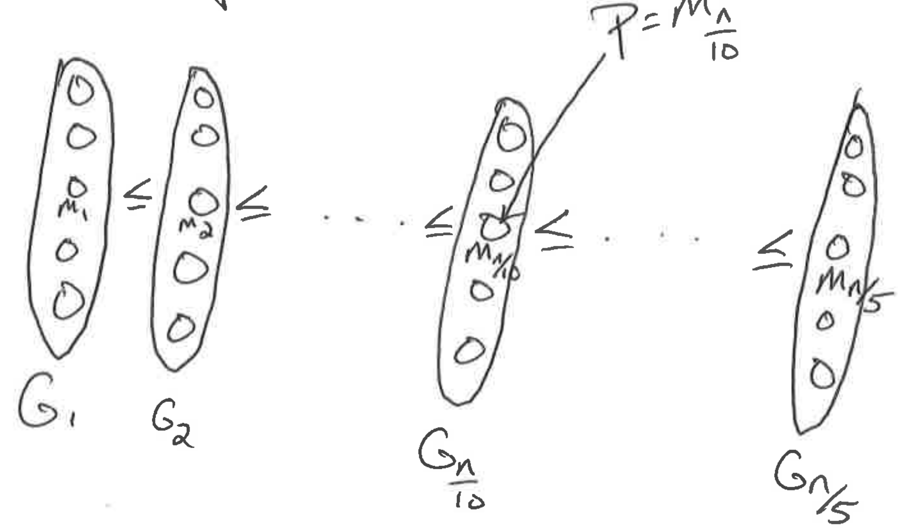
To prove the claim:

Sort the groups by their medians so that:

$$m_1 \leq m_2 \leq \cdots \leq m_{\frac{n}{5}}$$

This is just a relabelling of the groups in the proof, we don't do this step in the algorithm.

Here's the picture:



Which elements of S are guaranteed to be $\leq p$?

$m_1, m_2, \ldots, m_{\frac{n}{10}}$ are $\leq m_{\frac{n}{10}} = p$.

& for each of these,

3 elements in its group are $\leq$ it.

Hence, $\geq \frac{n}{10} \times 3 = \frac{3n}{10}$ are $\leq p$

So, $|A_{>p}| \leq n - \frac{3n}{10} = \frac{7n}{10} < \frac{3n}{4}$.

Similarly, $m_{\frac{n}{10}}, m_{\frac{n}{10}+1}, \ldots, m_{\frac{n}{5}} \geq p$

So, $\geq \frac{3n}{10}$ are $\geq p$ & $|A_{<p}| \leq \frac{3n}{4}$

Thus, $p = m_{\frac{n}{10}}$ is a good pivot. $\blacksquare$