**Linear-time Median**

Def: *Median* of elements $A = a_1, a_2, \ldots, a_n$ is the $\lceil n/2 \rceil$-th smallest element in $A$.

**How to find median?**

- sort the elements, output the elem. at $(n/2)$-th position
- running time? $O(n \log n)$
Linear-time Median

Def: **Median** of elements \( A = a_1, a_2, \ldots, a_n \) is the \((n/2)\)-th smallest element in \( A \).

How to find median?

- sort the elements, output the elem. at \((n/2)\)-th position
  - running time: \( \Theta(n \log n) \)
- we will see a faster algorithm
  - will solve a more general problem:
    \[
    \text{SELECT} \ (A, k) : \text{returns the } k\text{-th smallest element in } A
    \]
Linear-time Median

Idea: Suppose $A =$ $22, 5, 10, 11, 23, 15, 9, 8, 2, 0, 4, 20, 25, 1, 29, 24, 3, 12, 28, 14, 27, 19, 17, 21, 18, 6, 7, 13, 16, 26$

Looking for the k-th smallest elem.

$B = 11, 8, 20, 14, 19, 13$

$T(3)$

$\text{SELECT}(A, k)$:

0(1) { 0. if $n < 5$, then compute k-th smallest brute force (bubble sort) median $B = 13$ and return it $\in O(2)$
1. split $A$ into groups of 5
2. for each group, find its median $\leftarrow$ takes $O(1)$ steps per group (e.g. bubble sort 5 elms.)
3. create $B$, a list/ary of these medians
4. find the median of $B$ $\leftarrow$ let it be median $B = \text{SELECT}(B, \frac{n}{5})$
5. rearrange $A$ so that elem. < median $B$ come first, then elem. = median $B$, then elem. > median $B$

$T(3)$

$\text{SELECT}(A[1..i] \cup \ldots \cup A[j], k)$

0(1) { 6. let $j_1$, $j_2$ be the first and last index of median $B$ in the rearranged $A$
7. if $k < j_1$, then find the $k$-th smallest elem. among the list $j_1-1$ elms. of $A$
8. if $j_1 \leq k < j_2$ then return median $B$
9. if $k \geq j_2$ then find the $k$-th smallest elem. among the elms. in rearranged $A$ at indices $j_2+1 \ldots n$

$T(3)$
SELECT (A, k)
1. split A into \( n/5 \) groups of five elements
2. let \( b_i \) be the median of the \( i \)-th group
3. let \( B = [b_1, b_2, \ldots, b_{n/5}] \)
4. medianB = SELECT (B, B.length/2)
5. rearrange A so that all elements smaller than medianB come before medianB, all elements larger than medianB come after medianB, and elements equal to medianB are next to medianB
6. \( j = \) position of medianB in rearranged A (if more medianB’s, then take the closest position to \( n/2 \))
7. if \( k < j \) return SELECT ( A[1\ldots j-1], k )
8. if \( k = j \) return medianB
9. if \( k > j \) return SELECT ( A[j+1\ldots n], k-j )
Linear-time Median

Running the algorithm:

Orange: what the algo does
Blue: just me playing with the picture
Green: to be filled in later

White dots are not in A
Blue dots are in A
Black dots are in B
Blue square is what we want

Red dots are not relevant
Blue arrows are relevant
Yellow arrows are not relevant

The algorithm is:  

1. Group elements of A into groups of 5.  
2. For each group:  
   a. Take the median of the group.  
   b. Add the median to B.  
3. Sort B.  
4. Take the median of B.  

The algorithm runs in linear time because:  

- Grouping elements into groups of 5 takes linear time.  
- Finding the median of a group of 5 elements takes constant time.  
- Adding the median to B also takes constant time.  
- Sorting B takes linear time.  

Therefore, the total time taken by the algorithm is linear.
Linear-time Median

Running the algorithm:

Rearrange columns so that median B in the “middle.”

Recurrence: \[ T(n) = T\left(\frac{\sqrt{2}}{7}n\right) + T\left(\frac{3}{4}n\right) + cn \quad \forall n \geq 5 \]
\[ T(n) \leq c \quad \forall n < 5 \]
**Linear-time Median**

Recurrence: \[ T(n) \leq T(n/5) + T(3n/4) + cn \] if \( n \geq 5 \)

\[ T(n) \leq c \] if \( n < 5 \)

Claim: There exists a constant \( d \) such that \( T(n) \leq dn \).

This implies \( T(n) = O(n) \)

**Pf (by induction):**

**BASE CASE:** \( n < 5 \):

\[ T(n) \leq c \] we want to show that \( T(n) \leq d \cdot n \) for some \( d \)

we know \( T(n) \leq c \leq d \cdot n \) where \( 1 \leq n < 5 \)

**IND. CASE:** \( n \geq 5 \):

we want to show: \( T(n) \leq d \cdot n \)

we know: \( \forall m: \) (strong weak induction): \( T(m) \leq d \cdot m \) \( \forall m < n \)

we know: \( T(n) \leq T(n/5) + T(3n/4) + c \cdot n \)

\[ \leq d \cdot \frac{n}{5} + d \cdot \frac{3n}{4} + c \cdot n \]

\[ = \left( \frac{4d}{5} + \frac{3d}{4} + c \right) \cdot n = \left( \frac{19}{20} \cdot d + c \right) \cdot n \]

\[ \leq d \cdot n \]

want: \( \frac{19}{20} \cdot d + c \leq d \) \( \Rightarrow \) \( c \leq \frac{d}{20} \) \( \Rightarrow d \geq 20c \)
Randomized Linear-time Median

Idea:
Instead of finding median $B$, take a random element from $A$.

**SELECT-RAND** ($A$, $k$)

1. $x = a_i$ where $i$ is a random number from $\{1, \ldots, n\}$
2. rearrange $A$ so that all elements smaller than $x$ come before $x$, all elements larger than $x$ come after $x$, and elements equal to $x$ are next to $x$
3. $j$ = position of $x$ in rearranged $A$ (if more $x$’s, then take the closest position to $n/2$)
4. if ($k < j$) return SELECT-RAND ($A[1\ldots j-1]$, $k$)
5. if ($k = j$) return median $B$
6. if ($k > j$) return SELECT-RAND ($A[j+1\ldots n]$, $k-j$)
Randomized Linear-time Median

Worst case running time: $O(n^2)$. but $O(n)$ expected runtime if keep selecting $x = \max A$ and $k=1$

SELECT-RAND (A, k)
1. $x = a_i$ where $i = a$ random number from \{1,...,n\}
2. rearrange A so that all elements smaller than $x$ come before $x$, all elements larger than $x$ come after $x$, and elements equal to $x$ are next to $x$
3. $j =$ position of $x$ in rearranged A (if more $x$’s, then take the closest position to $n/2$)
4. if ($k < j$) return SELECT-RAND ( A[1...j-1], k )
5. if ($k = j$) return medianB 
6. if ($k > j$) return SELECT-RAND ( A[j+1...n], k-j )
Randomized Linear-time Median

Worst case running time: $O(n^2)$.

Claim: Expected running time is $O(n)$. 
Master Theorem

Let \( a \geq 1 \) and \( b > 1 \) be constants, \( f(n) \) be a function and for positive integers we have a recurrence for \( T \) of the form

\[
T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n),
\]

where \( n/b \) is rounded either way.

Then,

- If \( f(n) = O\left(n^{\log a/\log b - \varepsilon}\right) \) for some constant \( \varepsilon > 0 \), then
  \[
  T(n) = \Theta\left(n^{\log a/\log b}\right).
  \]

- If \( f(n) = \Theta\left(n^{\log a/\log b}\right) \), then
  \[
  T(n) = \Theta\left(n^{\log a/\log b \log n}\right).
  \]

- If \( f(n) = \Omega\left(n^{\log a/\log b + \varepsilon}\right) \) for some constant \( \varepsilon > 0 \), and if \( af(n/b) \leq cf(n) \) for some constant \( c < 1 \) (and all sufficiently large \( n \)), then
  \[
  T(n) = \Theta(f(n)).
  \]
Master Theorem

Let $a \geq 1$ and $b>1$ be constants, $f(n)$ be a function and for positive integers we have a recurrence for $T$ of the form

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n),$$

where $n/b$ is rounded either way.

Then,

- If $f(n) = O\left(n^{\log a / \log b - \epsilon}\right)$ for some constant $\epsilon > 0$, then
  $$T(n) = \Theta\left(n^{\log a / \log b}\right).$$

- If $f(n) = \Theta\left(n^{\log a / \log b}\right)$, then
  $$T(n) = \Omega\left(n^{\log a / \log b}\right).$$

- If $f(n) = \Omega\left(n^{\log a / \log b + \epsilon}\right)$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ (and all sufficiently large $n$), then
  $$T(n) = \Omega(f(n)).$$