Def: **Median** of elements $A = a_1, a_2, ..., a_n$ 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?

    $$\Omega(n \log n) + O(1) = \Omega(n \log n)$$
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:
    
    $\text{SELECT}(A, k)$: returns the $k$-th smallest element in $A$
    
    $A$ indexed from 0
    $k=0$ means the min. element
Linear-time Median

Idea: Suppose $A =$

\[
\begin{array}{ccccccccccccccccccccccccccc}
22, & 5, & 10, & 11, & 23, & 15, & 9, & 8, & 2, & 0, & 4, & 20, & 25, & 1, & 29, & 24, & 3, & 12, & 28, & 14, & 27, & 19, & 17, & 21, & 18, & 6, & 7, & 13, & 16, & 26 \\
\end{array}
\]

$B =$

\[
\begin{array}{ccccccccccccccccccccccccccc}
11, & 8, & 20, & 14, & 19, & 13 \\
\end{array}
\]

\[\text{SELECT} \ (A, k) :\]

0(1) 1. let $n =$ the length of $A$

$O(n)$ 2. if $n \leq 5$ then sort $A$ using bubble sort and return the median

$O(n)$ 3. split $A$ into groups of 5 elements $\Rightarrow$ take the median of every group $\Rightarrow$ in an array $B$

$T(M)$ 4. let $x =$ be the median of $B$ , i.e. $x = \text{SELECT}(B, \lfloor \frac{n}{5} \rfloor)$

$O(n)$ 5. for us $x = 13$

$O(n)$ 6. rearrange $A$ so that elem. $< x$ come first, $= x$ come afterwards, and $> x$ come last

for us:

\[\text{Array} = 5, 10, 11, 9, 8, 2, 0, 4, 1, 3, 12, 6, 13, 26, 1, 18, 21, 17, 19, 23, 24, 29, 25, 20, 15, 23, 22 \]

\[\text{j} \ \text{index of} \ x \ \text{in} \ \text{Array}.\]

$O(2)$ 5. if $k = j$ then return $x$

$T(2)$ 6. if $k < j$ then return $\text{SELECT} \ (\text{Array}[0 \ldots j-1], k)$

7. if $k > j$ then return $\text{SELECT} \ (\text{Array}[j+1 \ldots n-1], k-j-1)$

\[\frac{3}{4} n\]
Linear-time Median

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…j-1], k)
8. if ($k = j$) return medianB
9. if ($k > j$) return SELECT (A[j+1…n], k-j)
Linear-time Median

Running the algorithm:

- Elements $\leq x$: Therefore, $\# \text{elements} > x$ is at most $\frac{3}{4} n$.
- Elements $\geq x$: Therefore, $\# \text{elements} < x$ is at most $\frac{3}{4} n$.

Yellow: algo does not do the yellow part!

(Sort B)
Linear-time Median

Running the algorithm:

Recurrence:

\[ T(n) \leq T\left( \frac{n}{5} \right) + T\left( \frac{3n}{4} \right) + cn \quad \forall n > 5 \]
\[ T(n) \leq c \quad \forall n \leq 5 \]

Rearrange columns so that medianB in the “middle.”
Linear-time Median

Recurrence: \( T(n) \leq T(n/5) + T(3n/4) + cn \) if \( n > 5 \)

\( T(n) \leq c \) if \( n < 6 \)

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

BASE CASE: \( n \leq 5 \)
known: \( T(n) \leq c \)
want to show: \( T(n) \leq dn \)
choose \( d \) s.t. \( c \leq dn \) holds for \( d \geq c \)

IND. CASE: \( n > 5 \)
known: \( T(n) \leq T(n/5) + T(3n/4) + cn \)
want to show: \( T(n) \leq dn \)
assume (inductive hypothesis - IH): \( \forall m < n: \ T(m) \leq d \cdot m \) (strong math. induction)

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

i.e. \( \frac{19}{20}d + c \leq d \Rightarrow \boxed{d \geq 20c} \)

want \( \leq dn \)
Randomized Linear-time Median

Idea:
Instead of finding medianB, take a random element from A.

SELECT-RAND (A, k)
1. \(x = a_i\) where \(i = \) 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 medianB
6. if \((k > j)\) return SELECT-RAND ( \(A[j+1\ldots n]\), \(k-j\) )
Randomized Linear-time Median

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

**SELECT-RAND** \((A, k)\)
1. \(x = a_i\) where \(i = \) 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: $O(n^2)$.

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

$$T(n) \leq aT(n/b) + f(n),$$

where $n/b$ is rounded either way.

Then,

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

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

- If $f(n) = \Omega(n^{\log a/\log b + e})$ for some constant $e > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c_2 < 1$ (and all sufficiently large $n$), then $T(n) = \Theta(f(n))$. 

\[ c^n = O(n^{2.01}) \]
\[ c^n = \Theta(n^{1.99}) \]
\[ c^n = \Omega(n^{2.02}) \]