**Linear-time Median**

Def: **Median** of elements $A = a_1, a_2, ..., a_n$ 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? $O(n \log n)$
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 $A$
Linear-time Median

Idea: Suppose \( A = \)

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

6. if \( k = p \):
   return pivot

7. if \( k < p \):
   return select(\( A[1...p-1] \), \( k \))

8. if \( k > p \):
   return select(\( A[p+1...n] \), \( k-p \))

Recall with \( k = 6 \) (=20-14)

**func select(\( A, k \))**

1. split the input into groups of 5
2. find the median of each group
3. find the median of the medians, \( p \), pivot
4. rearrange \( A \) to have numbers \( <p \) on the left, \( >p \) on the right
5. let \( p \) be the position of the pivot
**Linear-time Median**

**SELECT** \((A, k)\) \(T(n)\)

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

Running the algorithm:

\[ A = [a_1, \ldots, a_n] \]

Smaller (\( \leq \)) pivot

\# elements < pivot \( \leq n - \text{the size of } \leq \frac{3}{4}n \)

\# elements > pivot \( \leq n - \text{the size of } \leq \frac{3}{4}n \)
Linear-time Median

Running the algorithm:

\[ T(n) \leq T(\frac{n}{5}) + T(\frac{3}{4}n) + c \cdot n \]

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

Recurrence: \[ T(n) \leq T(n/5) + T(3n/4) + cn \quad \text{if } n > 5 \]
\[ T(n) \leq c \quad \text{if } n \leq 65 \]

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

\text{BASE CASE: } n \leq 5 : \quad T(n) \leq c \quad \text{we want to show } T(n) \leq d \cdot n \quad \text{for some } d
\text{we need: } d \geq c

\text{IND. CASE: } n > 5 :
\[ T(n) \leq T(n/5) + T(3n/4) + c \cdot n \]
by IH: \[ T(n/5) \leq d \cdot n/5 \]
\[ T(3n/4) \leq d \cdot 3n/4 \]
\[ \Rightarrow \leq d \cdot \frac{n}{5} + d \cdot \frac{3n}{4} + c \cdot n = \left( \frac{19}{20} \cdot d + c \right) \cdot n \quad \text{want: } T(n) \leq d \cdot n
\text{want: } \frac{19}{20} \cdot d + c \leq d \Rightarrow d \geq 20c
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,...,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

$$T(n) = a \cdot T(n/b) + f(n),$$

where $n/b$ is rounded either way.

Then,

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

  $$T(n) = \Theta(n).$$