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? $O(n \log n)$

**goal:** $O(n)$ algorithm
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: $\Theta(n \log n)$
• we will see a faster algorithm
  - will solve a more general problem:
    SELECT ( $A$, $k$ ): returns the $k$-th smallest element in $A$
Linear-time Median

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

\[ h=30 \]

- \( \text{SELECT}(A, k) \)
  - 0. if \( A, \text{length} \leq 5 \): find the \( k \)-th in constant steps and return it
  - 1. split \( A \) into groups of 5 elements
  - 2. for every group of 5, find its median (4 groups, constant steps per group)
  - 3. let \( B \) be an array of all these medians
  - 4. find the median of \( B \) (simply call \( \text{SELECT}(B, \frac{B, \text{length}}{2}) \))
    - let it be \( \text{medianB} \)
  - 5. rearrange \( A \) so that elements \( < \text{medianB} \) come first, then elements \( = \text{medianB} \) hollow, then elements \( > \text{medianB} \)
    - \( 5, 10, 11, 9, 8, 2, 0, 4, 1, 3, 12, 6, 7, 13, 25, 16, 18, 21, 17, 19, 27, 14, 28, 24, 29, 25, 20, 15, 23, 22 \)
    - smaller than \( \text{medianB} \)
    - \( 13 \)
    - \( \text{medianB} \)
    - \( > \text{medianB} \)
  - 6. let \( \text{posmedB1} = \text{pos} \) position of \( \text{medianB} \) in the rearranged \( A \) (\( \text{Arrangedsmall} \))
  - \( \text{posmedB2} = \text{last} \)
  - 7. if \( k < \text{posmedB1} \): return \( \text{SELECT}(\text{Arrangedsmall}, k) \)
  - 8. if \( k > \text{posmedB2} \): return \( \text{medianB} \)
  - 9. if \( k \leq \text{posmedB1} \) and \( k \geq \frac{\text{posmedB2}}{2} \): return \( \text{SELECT}(\text{Arrangedlarge}, k - \text{posmedB1}) \)

\[ T(n) = T\left(\frac{n}{5}\right) + T(\frac{n}{2}) + cn \]

\[ n \geq 5 \]

\[ T(n) = \Theta(n) \]
**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 $B = [b_1, b_2, ..., b_{n/5}]$
4. $\text{medianB} = \text{SELECT} (B, B.\text{length}/2)$
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. $j = \text{position of medianB in rearranged A}$ (if more medianB’s, then take the closest position to n/2)
7. if ($k < j$) return $\text{SELECT} (A[1...j-1], k)$
8. if ($k = j$) return medianB
9. if ($k > j$) return $\text{SELECT} (A[j+1...n], k-j)$
Linear-time Median

Running the algorithm:

\[ B = \begin{array}{c}
2, 5, 10, 11, 23
\end{array} \]

SELECT does not sort!

In sorted B is not necessarily first group

Pink: pretend B is sorted
(algo doesn't do this)

\[
\begin{align*}
\# \text{ elems} \leq \text{median } B & : \text{ at least } \frac{n}{4} \text{ such elem. } \Rightarrow \# \text{ elem. } > \text{median } B \text{ is at most } \frac{3}{4} n \\
\# \text{ elems} \geq 2 & : \quad \frac{n}{4} \quad < \quad \frac{2}{3} n
\end{align*}
\]
Linear-time Median

Running the algorithm:

Rearrange columns so that medianB in the “middle.”

Recurrence:

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

\[ T(n) \leq c \quad n \leq 5 \]
Linear-time Median

Recurrence: \[ T(n) < T(n/5) + T(3n/4) + cn \quad \text{if } n > 5 \]
\[ T(n) < c \quad \text{if } n < 6 \]

Claim: There exists a constant \( d \) such that \( T(n) < dn \).
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 \( i \) = 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 median_B
6. if \( k > j \) return **SELECT-RAND** ( \( A[j+1...n] \), \( k-j \) )
Randomized Linear-time Median

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

**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 medianB
6. if $(k > j)$ return **SELECT-RAND** $(A[j+1 \ldots n], k-j)$
Randomized Linear-time Median

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

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