Linear-time Median

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

$\{2, 7, 2, 1, 5, 8, 4\}$

How to find median?

- sort the elements, output the elem. at $(n/2)$-th position

$\{1, 2, 2, 5, 7, 8\}$

- 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:

  \[
  \text{SELECT} \ (A, k) : \text{ returns the } k\text{-th smallest element in } A
  \]

  to get the median: \[\text{SELECT}(A, \frac{n}{2})\]
Linear-time Median

Idea: Suppose $A =$

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

$B =$ \[0, 8, 20, 14, 19, 13\]

$\text{median of } A =$ \[13\] (bec. $O(1)$ steps per group of $5$, we have $\frac{n}{5}$ groups)

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

$O(1)$

1. split $A$ into groups of $5$ elements

$O(n)$

2. find the median in each group

3. create array $B$ of all the medians

$T(\frac{n}{5})$

4. find the median of $B$ (i.e. $\text{median of } B = \text{SELECT} (B, \frac{\text{length}}{2})$)

5. rearrange $A$ so that $\text{elem. < median of } B$ on the left, $\text{elem = median of } B$ follow, then $\text{elem > median of } B$ on the right

$O(n)$

6. $\text{count left} =$ #elem. $< \text{median of } B$, $\text{count right} =$ #elem. $> \text{median of } B$

$T(?)$

7. if $k \leq \text{count left}$: return $\text{SELECT} (\text{A rearranged [1... count left]}, k)$

$O(1)$

8. else if $k \leq \text{A.length - count right}$: return median of $B$

$T(?)$

9. else: return $\text{SELECT} (\text{A rearranged [n - count right + 1... n]}, k - (n - \text{count right}))$
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\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:

- **Every dot**: element in A

- **B**: elements arranged in some order
  - 1st group of \( k \)
  - 2nd
  - 3rd

- Pretend B is sorted (the algo does not sort it)
- Pink: only for the analysis, does not happen in algo

- # elements \( \leq \) median B: at least \( \frac{n}{4} \)
- # elements \( \geq \) median B: at least \( \frac{n}{4} \)

\[ \Rightarrow \] # elements in the left remaining cell (line 7): \( \leq \frac{3}{4} n \)
Linear-time Median

Running the algorithm:

Rearrange columns so that medianB in the “middle.”

Recurrence:
\[
T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{4}{5}n\right) + cn \quad n > 5 \\
T(n) \leq c \quad n \leq 5
\]
Linear-time Median

Recurrence: $T(n) \leq T(n/5) + T(3n/4) + cn$ if $n > 5$ \hspace{1cm} (a)

$T(n) \leq c$ if $n < 6$ \hspace{1cm} (x)

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

By induction:

**Base Case:** $n \leq 5$:

- know (x): $T(n) \leq c$
- want to show: $T(n) \leq dn$

**Inductive Case:** $n > 5$:

- know (a): $T(n) \leq T(n/5) + T(3n/4) + cn$
- want to show: $T(n) \leq dn$

IH: $T(m) \leq d \cdot m$ holds for $m < n$

\[ T(n) \leq \underbrace{T(n/5)}_{(a)} + \underbrace{T(3n/4)}_{(a)} + cn \leq \frac{d \cdot n}{5} + \frac{3d \cdot n}{4} + cn = \]

\[ = \left( \frac{d}{5} + \frac{3d}{4} + c \right) \cdot n = \left( \frac{13}{20} d + c \right) \cdot n \leq dn \]

\[ \frac{13}{20} d + c \leq \frac{d}{20} \Rightarrow \frac{13}{20} c \leq \frac{d}{20} \Rightarrow d \geq 20c \]
Randomized Linear-time Median

Idea:
Instead of finding median\(B\), take a random element from \(A\).

\[\text{SELECT-RAND}\ (A, \ k)\]
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 \(\text{SELECT-RAND}\ (A[1\ldots j-1], \ k)\)
5. if \((k = j)\) return median\(B\)
6. if \((k > j)\) return \(\text{SELECT-RAND}\ (A[j+1\ldots n], \ k-j)\)
Randomized Linear-time Median

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

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

1. $x = a_i$ where $i$ 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. $j = \text{position of } x\text{ in rearranged } A$ (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 $O(n)$.  