Median finding

Vera Sacristán

Computational Geometry
Departament de Matemàtica Aplicada II
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Definition 1 *The median value of a finite set of real numbers* $X = \{x_1, \dots, x_n\}$*, is the number* $m = x_j \in X$ *such that:*

$$\#\{i \mid x_i < m\} < \tfrac{n}{2}$$
$$\#\{i \mid x_i > m\} \leq \tfrac{n}{2}$$

The median value of such a set is its $\frac{n}{2}$-th statistic:

Definition 2 *The* k*-th statistic of a finite set of real numbers* $X = \{x_1, \dots, x_n\}$ *is the number* $m = x_j \in X$ *such that:*

$$\#\{i \mid x_i < m\} < k$$
$$\#\{i \mid x_i > m\} \leq n - k$$

Proposition 3 *The* k*-th statistic and, particularly, the median value of a set of* n *real numbers can be computed in* $O(n \log n)$ *time.*

The most obvious solution consists in sorting the n numbers and then finding out the value throughout the sorted numbers.

Proposition 4 *The* k*-th statistic and, particularly, the median value of a set of* n *real numbers can be computed in* $O(n)$ *time.*

The solution algorithm follows a prune-and-search strategy:

Algorithm 1 $\text{SELECT}(\{x_1, \dots, x_n\}, k)$

1. If n is small, compute the statistic by sorting the set.
2. Else, choose one $p \in \{x_1, \dots, x_n\}$ (how to choose it will be explained later on) and do:
   2.1 Partition:
      2.1.1 Test all x_i and classify them as smaller, equal or bigger than p.
   2.2 Recursion:
      2.2.1 If the number of $x_i < p$ is $< k$ and the number of $x_i > p$ is $\leq n - k$, return p.
      2.2.2 Else, if the number of $x_i < p$ is $\geq k$, return $\text{SELECT}(\{x_i \mid x_i < p\}, k)$.
      2.2.3 Else, return $\text{SELECT}(\{x_i \mid x_i > p\}, k - j)$, where j is the number of $x_i \leq p$.

The partition phase takes $\Theta(n)$ time. On the other hand, the recursion phase depends on the value of the chosen p. A bad choice of p may lead to a $T(n) = T(n-1) + O(n)$ running time, and the algorithm will have complexity $T(n) = O(n^2)$. Therefore, it is convenient to cleverly choose p. The following algorithm (to be inserted in step 2 of Algorithm 1) is a convenient solution:

Algorithm 2 CHOOSE p

1. Divide $x_1, \ldots, x_n$ into subsets of 5 elements.
2. Compute the median value m_i of each subset $x_{5i+1}, x_{5i+2}, x_{5i+3}, x_{5i+4}, x_{5i+5}$, by sorting.
3. Return SELECT($\{m_1, \ldots, m_r\}, \lceil r/2 \rceil$), where $r = \lfloor n/5 \rfloor$.

This way of computing p guarantees that at least 1/4 of all x_i are smaller than p, and at least another 1/4 of all x_i are greater than p. As a consequence, the running time of SELECT is

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + O(n) \leq T\left(\frac{19n}{20}\right) + O(n) = O(n),$$

where the factor $T(n/5)$ corresponds to the recursive call SELECT($\{m_1, \ldots, m_r\}, \lceil r/2 \rceil$), the factor $T(3n/4)$ corresponds to the recursive call SELECT($\{x_i \mid x_i < p\}, k$) or SELECT($\{x_i \mid x_i > p\}, k - j$), and the factor $O(n)$ is the running time of the partition, the division into subsets of five elements, and the computation of the median value, m_i, of the subsets.

Notice that the choice of making subsets of 5 elements is intended to guarantee that $\frac{3}{4} + \frac{1}{5} = \frac{19}{20} < 1$. Therefore, any other number grater than 5 could have been suitable.

Proposition 5 *The k-th statistic and, particularly, the median value of a set of n real numbers can be computed in $O(n)$ expected time.*

The algorithm is the same as Algorithm 1, but now p is randomly chosen:

Algorithm 3 CHOOSE p

1. Randomly choose p among $x_1, \ldots, x_n$.

This way of choosing p makes the algorithm run in $O(n)$ *expected* time, let us see why. First notice that if p is randomly chosen, the probability of p matching each x_i is $\frac{1}{n}$. When $p = x_i$, the recursion step of the algorithm runs in $T(i-1)$ or $T(n-i)$ time, i.e., in $T(\max(i-1, n-i))$ time. Therefore, the algorithm running time is:

$$\begin{aligned}
T(n) &\leq an + \frac{1}{n}\sum_{i=1}^{n} T(\max(i-1, n-i)) \\
&= an + \frac{1}{n}\sum_{i=0}^{n-1} T(\max(i, n-i-1)) \\
&= an + \frac{2}{n}\sum_{i=n/2}^{n-1} T(i) \\
&\overset{*}{\leq} cn \\
&= O(n)
\end{aligned}$$

The factor an corresponds to the partition step running time. The inequality marked with an asterisk can be proved by induction. The base case is $T(1) \leq c$, which is true if we choose $c \geq a$. The induction step is proved as follows. Assume that $T(i) \leq ci$ for all $i < n$, then prove that

$T(n) \leq cn$:

$$\begin{aligned}
T(n) &\leq an + \frac{2}{n}\sum_{i=n/2}^{n-1} T(i) \\
&\leq an + \frac{2c}{n}\sum_{i=n/2}^{n-1} i \\
&= an + \frac{2c}{n}\left(\frac{n}{2} + (n-1)\right)\frac{1}{2}\left((n-1) - (\frac{n}{2} - 1)\right) \\
&= an + \frac{2c}{n}\left(\frac{3n}{2} - 1\right)\frac{1}{2}\frac{n}{2} \\
&= an + \frac{3}{4}cn - \frac{c}{2} \\
&= \left(\frac{3}{4} + \frac{a}{c}\right)cn - \frac{c}{2} \\
&\leq \left(\frac{3}{4} + \frac{a}{c}\right)cn \\
&\stackrel{*}{\leq} cn.
\end{aligned}$$

In order for the inequality marked with an asterisk to be true, c must be chosen such that $\frac{3}{4} + \frac{a}{c} \leq 1$, i.e., $c \geq 4a$.