# Quickselect with Median of Medians

BY NIKLAS HAMBÜCHEN

Based on http://en.wikipedia.org/wiki/Median_of_medians.

## The Algorithm

### Quickselect

Quickselect is supposed to find the $k^{\text{th}}$ largest element in a list of numbers.

To do this, we select a pivot, and partition the list so that all elements smaller than the pivot are on its left and all elements greater than the pivot are on its right (like in Quicksort). This is $O(n)$ and in-place.

We can then check at which position the pivot is.

- If it is at a position $k$, the pivot is the result and we can stop.
- If it is at a position $>k$, then it is bigger than the $k$ largest element, and we recurse on the left partition.
- If it is at a position $<k$, then the $k^{\text{th}}$ largest element must be on its right, and we recurse on the right side of the list. However, the $k^{\text{th}}$ largest element of the original list will not be the $k^{\text{th}}$ largest element of the right list, since we are taking away the small partition and the pivot. So we continue with $k \leftarrow k - \text{sizeOfLeftPartition} - 1$.

### Choice of the pivot

Like for Quicksort, Quickselect's complexity is dependent on the pivot. In the worst case, the pivot is selected badly such that only a constant number of elements are on the side that we discard, leading to $O(n^2)$.

If however we can make sure that in every recursion we work on a list that is some *fraction of n* smaller (instead of a constant smaller), then $O(n)$ run time is achieved. Picking the pivot as the true (as opposed to doing recursive grouping) median of a list of medians of groups of constant size ($\geq 5$), we can guarantee this fraction.

### The median of medians

We organize the list into groups of five, and calculate the median of each group. This gives us $n/5$ medians. We then compute the *true* median of the list of medians and pick that as the pivot for Quickselect. (We can find the true median using `quickselectMOM(medians, medians.length/2)`, since finding a median is a special case of finding a $k^{\text{th}}$ largest number – simply pick $k = \frac{\text{length of list}}{2}$.)

This median of medians (call it $m$) is not the true median of the whole list, but we know that it is an "approximate" median with $30\% \leq m \leq 70\%$ of the list. The 30% bound on either side comes from the following: $m$ is the median of $\frac{n}{5}$ medians, so of these $\frac{1}{2}\frac{n}{5}$ are $\leq$ than $m$. In addition to that, each of them is a median of 5, so each of them has 2 numbers $\leq$ than it. In total, we have $\underbrace{\frac{1}{2}\frac{n}{5}}_{\text{medians} \leq m} + \underbrace{\frac{1}{2}\frac{n}{5} \cdot 2}_{\text{elements} \leq \text{medians}} = \frac{3}{10} n$ numbers that are $\leq m$.

**Example:** Let's say we have 100 numbers, and so get 20 group medians. The median of those medians is $m$. 10 of these group medians are $\leq m$, and each of the 10 is a median of 5 so it has two numbers $\leq$ than it. That makes $10 + 10 \cdot 2 = 30$ numbers $\leq m$.

This 30%-result means that in the worst case our Quickselect has to recurse on a list of length $0.7n$.

### Summary

For Quickselect with Median of Medians (`quickselectMOM`),

- we create groups of five of the list, calculate the median of each group, and then calculate the true median of those using an independent `quickselectMOM(medians, medians.length/2)`
- we use that median of medians as the pivot for partitioning
- we recursively call our `quickselectMOM` on one of the two sides, depending on where the pivot ends up

### Complexity

Let's call $T(n)$ the time required to run this Quickselect on a list of size $n$.

Then we know that in the worst case,

$$T(n) \leq \underbrace{c \cdot n}_{\substack{\text{calculating} \\ \text{group medians}}} + \underbrace{T\left(\frac{n}{5}\right)}_{\substack{\text{finding true median} \\ \text{of group medians} \\ \text{with Quickselect}}} + \underbrace{c \cdot n}_{\substack{\text{partitioning} \\ \text{for Quickselect}}} + \underbrace{T(0.7n)}_{\substack{\text{running Quickslect} \\ \text{on the worst case side}}}$$

So

$$T(n) \leq T(0.2n) + T(0.7n) + c \cdot n$$

We can show with strong induction that $T(n) \in O(n)$; by definition we need to show $T(n) \leq c' \cdot n$ for that.

Assume inductively that $T(x) \leq c' \cdot x$ for all $x \leq n$. We can apply this to

$$T(n) \leq T(0.2n) + T(0.7n) + c \cdot n$$

since $0.2n$ and $0.7n$ are smaller than $n$, and get

$$T(n) \leq 0.2 \cdot c' \cdot n + 0.2 \cdot c' \cdot n + c \cdot n$$

$$T(n) \leq (0.9 \cdot c' + c) \cdot n$$

We desire $T(n) \leq c' \cdot n$, so $c' = 0.9 \cdot c' + c$ will give us that. Rearranging, we get $c' = 10c$.

So we have found the appropriate $c'$.

Assuming that our non-recursive base cases are $n \in \{1...10\}$ and that for these, $T(n) \leq 10$, $T(n) \leq c' \cdot n$ also holds for the base cases.

Thus our Quickselect is in $O(n)$.

# FAQ:

- Do we perform this approximate-median finding algorithm recursively?

  **Answer:** No, we take medians of groups of five, and then find the *true* median of this median list by calling `quickselectMOM(list, list.length/2)`.

- Why is it a median of 5? Why is 3 not enough?

  **Answer:** Because with the median of 5, we obtain the $30\% \leq m \leq 70\%$ result, which makes sure that in the worst case our Quickselect has to recurse on a list of length $0.7n$. For this, we "pay" only $0.2n$ to obtain the true means, so to get from one Quickselect recursion to the next, the work required is $0.9n$ each time – the fraction of $n$ we were looking for.

  If we picked a median of 3 instead, we would get $33.\bar{3}\% \leq m \leq 66.\bar{6}\%$ (since $\frac{1}{2}\frac{1}{3}$ of the group medians are $\leq m$ and each of them has 1 number smaller than it, giving $\frac{1}{2}\frac{1}{3} + \frac{1}{2}\frac{1}{3} \cdot 1 = \frac{1}{3}$), so in the worst case we would have to recurse on a list of size $\frac{2}{3}n$. At the same time, we would have to find the true median of $\frac{1}{3}n$ grouped medians. So the work required here is $\frac{2}{3}n + \frac{1}{3}n = n$, which does not help us to prove linearity as opposed to $0.9n$ from before.

  Odd numbers $\geq 5$ also work. Even numbers are avoided because their median is not defined discretely.