# Towards Optimal Multiple Selection

Kanela Kaligosi[1], Kurt Mehlhorn[1], J. Ian Munro[2], and Peter Sanders[3]

[1] Max Planck Institut für Informatik
Saarbrücken, Germany, [kaligosi,mehlhorn]@mpi-sb.mpg.de
[2] University of Waterloo, Ontario, Canada, imunro@uwaterloo.ca
[3] Universität Karlsruhe, Germany, sanders@ira.uka.de

**Abstract.** The multiple selection problem asks for the elements of rank $r_1, r_2, \ldots, r_k$ from a linearly ordered set of $n$ elements. Let $B$ denote the information theoretic lower bound on the number of element comparisons needed for multiple selection. We first show that a variant of multiple quickselect — a well known, simple, and practical generalization of quicksort — solves this problem with $B + \mathcal{O}(n)$ expected comparisons. We then develop a deterministic divide-and-conquer algorithm that solves the problem in $\mathcal{O}(B)$ time and $B + o(B) + \mathcal{O}(n)$ element comparisons.

## 1 Introduction

We consider the problem of determining the elements of rank $r_1, r_2, \ldots, r_k$ in an ordered set $S$ with $n$ elements using as few comparisons as possible. The problem has been studied extensively, primarily at its extreme ends, when one either requires one or two order statistics (such as the median or the maximum and minimum) or when all ranks are to be determined, and so the set is to be sorted. The more general problem can be very helpful in statistical analyses, providing a clean summary of the data by giving elements of a chosen set of ranks. Multiple selection is also an ingredient of *sample sort* [1].

Let $\Delta_j := r_j - r_{j-1}$ where $r_0 = 0$ and $r_{k+1} = n$ and let[4]

$$B = n \lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j - \mathcal{O}(n) = \sum_{j=1}^{k+1} \Delta_j \lg \frac{n}{\Delta_j} - \mathcal{O}(n)\,.$$

A comparison-based multi-selection algorithm identifies the $\Delta_1$ smallest elements, the $\Delta_2$ next smallest elements, $\ldots$, and hence an additional $\sum_j \Delta_j \lg \Delta_j + \mathcal{O}(n)$ comparisons suffice to sort the entire set. Thus $B$ is a lower bound for the number of comparisons needed by any multiselection algorithm, even in the expected case [5]. Note that this lower bound takes both the input size $n$ and the structure of the required ranks $R$ into account.

[4] Throughout this paper, $\lg x$ stands for $\log_2 x$.

### 1.1 Background and Ancient History

The problem of sorting is one of the oldest and most heavily studied in computing. “Interesting” sorting algorithms date back to the 1940’s, or earlier. The information theoretic lower bound of $\lg(n!) = n \lg n - n \lg e + \mathcal{O}(\lg n)$ comparisons is missed by only about $n(\lg e - 1)$ comparisons by mergesort, when $n$ is a power of 2. Despite such an easy approach to optimality, even Ford-Johnson [9] merge insertion and other very complex methods still leave a $\Theta(n)$ gap (with an admittedly small constant).

It seems that Charles Dodgeson (Lewis Carroll) was the first to point to the unfairness of declaring the runner-up (the player losing to the last game to the winner in a single knockout competition) to be the “second best”. Hoare [8] proposed a simple $\mathcal{O}(n)$ expected case (median) selection algorithm as an addendum to his original work on Quicksort. Close to a decade later Blum, Floyd, Pratt, Rivest and Tarjan [2] provided the first median finding technique with $\mathcal{O}(n)$ worst case behavior. The result was certainly a major surprise at the time, although it is now often presented as a basic example of divide and conquer. At essentially the same time, two of those authors, Floyd and Rivest [7], gave a probabilistic method for selection using $n + \min(r_1, n - r_1) + o(n)$ comparisons in the expected case. This bound is optimal, at least up to $o(n)$ terms, as shown in [4]. We note that this lower bound exceeds that of the information theoretic argument, above, for selecting the median by $n/2$. The difference is even greater for selecting other ranks.

The contrast between the simple randomized techniques and the complex worst case approach is striking. The development of the $3n + o(n)$ worst case method of Schönhage et al. [14] and the $(2.94\cdots)n$ method of Dor and Zwick [6] further amplifies

Multiple selection problems have received much less attention. Chambers [3] proposed multiple quickselect — a natural generalization of quicksort: Pick a *pivot* element $m$ as an estimate for the median of $S$ and partition $S$ into two sets $S_{\leq} = \{s \in S : s \leq m\}$ and $S_{>} = \{s \in S : s > m\}$. Output $m$ as the element of rank $r_m := |S_{\leq}|$ if $r_m \in R$. Recurse on $(S_{\leq}, \{r \in R : r < r_m\})$ and $(S_{>}, \{r - r_m \in R : r > r_m\})$. The recursion can be broken when $R = \emptyset$.

Pohl [12] gave a slick deterministic algorithm for finding the maximum and minimum of a set in the optimal number of $\lceil 3n/2 \rceil - 1$ comparisons. Cunto and Munro [4] provided optimal and near optimal randomized techniques for finding elements of two or three ranks. The “full blown” deterministic multiselection problem that we address appears to have first been considered by Dobkin and Munro [5], who suggest a method based on median finding that solves the problem using $3B + \mathcal{O}(n)$ comparisons.

A quarter century after Chamber’s paper, there has been renewed interest in the analysis of multiple quickselect. Prodinger [13] has given an exact formula for the expected number of comparisons performed by multiple quickselect with random pivot which can be written as $2B/\lg(e) + \mathcal{O}(n)$. Panholzer [11] has derived very accurate bounds for the expected number of comparisons for the case that the pivot is the median of $2t+1$ random sample elements and where $R$

is a *random* $k$ element subset of $\{1, \ldots, n\}$. These bounds show that $n \lg k + \mathcal{O}(n)$ expected comparisons are sufficient if $t = \Omega(\lg n)$. Note that for general $R$ the lower bound $B$ might be much smaller. For example, consider the set of ranks $R := \{\lceil n/i \rceil : i \leq n\}$. Observe further that this set has $k = \Theta(\sqrt{n})$ elements, i.e., $n \lg k = \Theta(n \lg n)$. On the other hand, $B \approx n \sum_i \frac{\ln(i(i+1))}{i(i+1)} = \Theta(n)$, i.e., there is a logarithmic factor gap between Panholzer's upper bound and the lower bound $B$. It is a natural question whether multiple quicksort is still a near optimal algorithm for inputs with nonuniformly spread ranks in $R$. For example, it was initially not clear to us whether one should choose $m$ such that $S$ is approximately halved, such that $R$ is approximately halved, or some compromise between these two strategies.

### 1.2 The Rest of the Paper

Section 2 introduces our basic proof technique by analyzing an idealized variant of multiple quickselect that is given the exact median for free. This algorithm needs at most $n \lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 2n$ comparisons. In Section 3 we show that multiple quickselect with pivot selection based on random sampling can closely approximate the idealized algorithm with respect to the expected number of comparisons. The method is reasonably straightforward and so usable in practice: one uses about $n$ comparisons to split $S$ into elements above and below a pivot value which, with high probability, is quite close to the median. Some care is required in sampling, but as long as one does not insist on minimizing lower order terms too vigorously, this aspect of the analysis is fairly straightforward. Handling the details of the splits is a bit trickier.

Section 4 deals with the deterministic algorithm. As in the deterministic median algorithm of Blum et al. [2], we represent sets as collections of sorted sequences. Each sequence has length about $\ell = \lg(B/n)$. The number of sequences is about $n/\ell$. We determine the median of the median of the sequences, call it $m$, and partition each sequence at $m$ using binary search. This has cost $\mathcal{O}((n/\ell) \lg \ell)$ which is sublinear, if $B = \omega(n)$. After partitioning, we have two collections of sorted sequences on which we recurse. In each collection, we have short sequences (length less than $\ell$) and long sequences (length between $\ell$ and $2\ell$). As long as we have two short sequences of the same length, we merge them using an optimal number of comparisons. The key insight is that the amortized cost of merging is $nH(\alpha)$ where $n\alpha$ is the rank of the pivot element and $H(\alpha) = -\alpha \lg \alpha - (1 - \alpha) \lg(1 - \alpha)$ is the binary entropy function. A simple induction shows that the total cost of merging is $B + o(B) + \mathcal{O}(n)$ and that the total cost of partitioning is $o(B) + \mathcal{O}(n)$.

## 2 An Idealized Algorithm

Consider multiple quickselect where we pick the exact median of the current subproblem as a pivot. Let $T(n, R)$ denote the number of comparisons needed in the partitioning steps, i.e., we disregard the cost needed for obtaining the pivot.

**Theorem 1.** $T(n,R) \le n\lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 2n.$

**Proof:** The proof is by induction. For $n = 1$ the claim $0 = T(n,R) \le 0 - 0 + 1 - 1 + 2$ is true. Now let $r_i$ denote the largest element of $R$ with $r_i \le n/2$. For the induction step we distinguish two cases and substitute the induction hypothesis for the subproblems in each case.

**Case** $0 < i < k$**:**

$$
\begin{aligned}
T(n,R) =& n - 1 + T\left(\frac{n}{2}, \{r_1, \ldots, r_i\}\right) + T\left(\frac{n}{2}, \{r_{i+1} - \frac{n}{2}, \ldots, r_k - \frac{n}{2}\}\right) \\
\le& n - 1 + n\lg\frac{n}{2} - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_i - r_1 + r_k - r_{i+1} + 2n \\
&+ \Delta_{i+1}\lg\Delta_{i+1} - \left(\frac{n}{2} - r_i\right)\lg\left(\frac{n}{2} - r_i\right) - \left(r_{i+1} - \frac{n}{2}\right)\lg\left(r_{i+1} - \frac{n}{2}\right) \\
\le& n\lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j - r_k + r_1 + 2n
\end{aligned}
$$

For the last estimate we have used $n - 1 + n\lg(n/2) = n\lg n - 1 \le n\lg n$ and Lemma 1 below with $x = n/2 - r_i$ and $y = r_{i+1} - n/2$.
**Case** $i = k$**:** (the case $i = 0$ is symmetric)

$$
\begin{aligned}
T(n,R) =& n - 1 + T\left(\frac{n}{2}, R\right) \\
\le& n - 1 + \frac{n}{2}\lg\frac{n}{2} - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 2\frac{n}{2} \\
&+ (n - r_k)\lg(n - r_k) - \left(\frac{n}{2} - r_k\right)\lg\left(\frac{n}{2} - r_k\right)
\end{aligned}
$$

Now we use $n-1+2n/2 \le 2n$ and, by Lemma 1,
$(n - r_k)\lg(n - r_k) - \left(\frac{n}{2} - r_k\right)\lg\left(\frac{n}{2} - r_k\right) \le \frac{n}{2}\lg\frac{n}{2} + n - r_k$.

$$
T(n,R) \le n\lg\frac{n}{2} - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 2n + n - r_k \quad .
$$

Finally, observe that $n\lg\frac{n}{2} + n - r_k = n\lg n - r_k \le n\lg n$. $\square$

**Lemma 1.** *For* $x, y \ge 0$ *we have* $(x+y)\lg(x+y) - x\lg x - y\lg y - (x+y) \le 0$.

**Proof:** Let $s = x + y$ and $x = \gamma s$, with $0 \le \gamma \le 1$. Then
$s(\lg s - \gamma\lg(\gamma s) - (1-\gamma)\lg((1-\gamma)s) - 1)$
$= s(-\gamma\lg(\gamma) - (1-\gamma)\lg(1-\gamma) - 1) = s(H(\gamma) - 1) \le 0.$ $\square$

## 3 The Randomized Algorithm

We now analyze a variant of quickselect where the pivot $m$ is the median of $n^{3/4}$ sample elements chosen uniformly at random from $S$ with replacement. Let $T(n,R)$ denote the expected number of comparisons needed to execute this of multiple quickselect.

**Theorem 2.** $T(n,R) \le n \lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 2n + \mathcal{O}\left(k^{1/4} n^{3/4}\right)$.

The remainder of this section outlines a proof for the above theorem. Most probabilistic aspects of the problem are packaged in the following lemma:

**Lemma 2.** *The rank of the pivot $m$ is outside the range $[n/2 - n^{3/4}, n/2 + n^{3/4}]$ with probability at most $1/n$.*

The proof is in the full paper. It is based on Chernoff bounds and very similar to the proof used in the analysis of the Floyd Rivest algorithm [7] in [10].

Using Lemma 2 we can conclude that

$$\begin{aligned} T(n,R) \le & n - 1 + \mathcal{O}\left(n^{3/4}\right) + \frac{1}{n} T(n,R) \\ & + (1 - \frac{1}{n}) \max_{|\frac{n}{2} - n'| \le n^{3/4}} (T(n', R') + T(n'', R'')) \end{aligned}$$

where $R' = \{r \in R : r < n'\}$, $R'' = \{r - n' : r \in R, r > n'\}$ and $n'' = n - n'$. The term $n-1$ stems from the comparisons of elements in $S \setminus \{m\}$ with $m$. The term $\mathcal{O}(n^{3/4})$ accounts for the expected cost of finding $m$ in the sample using a (randomized) linear time selection algorithm. The next term is the probability $1/n$ that $m$ is an inaccurate estimate of the median times a conservative estimate of the cost in this case — partitioning can never make the subproblems more difficult than the original problem. Subtracting $T(n,R)/n$ and dividing by $1-1/n$ yields

$$T(n,R) \le n + \mathcal{O}\left(n^{3/4}\right) + \max_{|\frac{n}{2} - n'| \le n^{3/4}} (T(n', R') + T(n'', R'')) \tag{1}$$

We now use induction over $n$ using the induction hypothesis

$$T(n,R) \le n \lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 2n + (ak^{1/4} - b) n^{3/4}$$

where $a$ and $b$ are constants whose value will be implied by constraints showing up in the proof. First note that for any constant $n$ we can make this work by choosing a value for $a$ that is sufficiently much larger than $b$.

For the induction step we will use the following two lemmata which encapsulate a number of calculations that we defer to the full paper.

**Lemma 3.** *$f(\delta) := (1+\delta)\lg(1+\delta) + (1-\delta)\lg(1-\delta) \le 2\delta^2$ for $\delta \in [-1,1]$.*

**Lemma 4.** *For $0 \le i \le k$ and $-1 \le \delta \le 1$ we have*

$$i^{1/4}(1+\delta)^{3/4} + (k-i)^{1/4}(1-\delta)^{3/4} \le 2(k/2)^{1/4}$$

To simplify the max term in Eq. (1), consider a fixed value $0 \le \delta \le 2n^{-1/4}$ such that $n' = (1+\delta)\frac{n}{2}$ (the case $\delta < 0$ is symmetric). Let $r_i$ denote the largest element of $R$ with $r_i \le n'$. We get three cases:
**Case $0 < i < k$:** We use the abbreviation $I(x) := x\lg x$.

$$\begin{aligned}
T(n',R') + T(n'',R'') \le\, & I((1+\delta)\frac{n}{2}) + I((1-\delta)\frac{n}{2}) - \sum_{j=1}^{k+1} I(\Delta_j) \\
& + I(\Delta_{i+1}) - I(n'-r_i) - I(r_{i+1}-n') \\
& + r_i - r_1 + 2n' + (ai^{1/4}-b)n'^{3/4} \\
& + r_k - r_{i+1} + 2n'' + (a(k-i)^{1/4}-b)n''^{3/4} \\
=& n\lg n - n + \frac{n}{2}(I(1+\delta) + I(1-\delta)) - \sum_{j=1}^{k+1} I(\Delta_j) \\
& + I(\Delta_{i+1}) - I(n'-r_i) - I(r_{i+1}-n') - (r_{i+1}-r_i) \\
& + 2^{-3/4}a(i^{1/4}(1+\delta)^{3/4} + (k-i)^{1/4}(1-\delta)^{3/4})n^{3/4} \\
& - 2^{-3/4}((1+\delta)^{3/4} + (1-\delta)^{3/4})bn^{3/4} + r_k - r_1 + 2n
\end{aligned}$$

Now we can apply a number of estimates:
$I(1+\delta) + I(1-\delta) \le 2\delta^2 \le 2(2n^{-1/4})^2 = \mathcal{O}\big(n^{-1/2}\big)$ (using Lemma 3). As in the idealized case, Lemma 1 can be used to eliminate the $I(x)$ terms not in $\sum_{j=1}^{k+1} I(\Delta_j)$. Now it follows (using Lemma 4) that
$i^{1/4}(1+\delta)^{3/4} + (k-i)^{1/4}(1-\delta)^{3/4} \le 2(k/2)^{1/4} = 2^{3/4}k^{1/4}$. Finally,
$-(1+\delta)^{3/4} - (1-\delta)^{3/4} \le -2 + \mathcal{O}\big(n^{-1/2}\big)$ using calculus. So

$$\le n\lg n - n - \sum_{j=1}^{k+1} I(\Delta_j) + \mathcal{O}\big(\sqrt{n}\big) + ak^{1/4}n^{3/4} - 2^{1/4}bn^{3/4} + r_k - r_1 + 2n.$$

Substituting this back into Eq. (1) we obtain

$$T(n,R) \le n\lg n - \sum_{j=1}^{k+1} I(\Delta_j) + \mathcal{O}\Big(n^{3/4}\Big) + ak^{1/4}n^{3/4} - 2^{1/4}bn^{3/4} + r_k - r_1 + 2n \ .$$

Since $2^{1/4} > 1$ we can use the term $b(2^{1/4}-1)n^{3/4}$ to cancel the $\mathcal{O}\big(n^{3/4}\big)$ term by choosing a sufficiently large value for $b$.

**Case $i = k$:** This case is a slight generalization of the reasoning from the idealized algorithm and from the case $0 < i < k$. Compared to the idealized case, we get various $\mathcal{O}(n^{3/4})$ terms that we need to cancel. For this purpose we use the term $ak^{1/4}((1+\delta)\frac{n}{2})^{3/4} = 2^{-3/4}ak^{1/4}n^{3/4} + \mathcal{O}(k^{1/4}n^{1/2})$. Since $2^{-3/4} < 1$ we can choose $a$ sufficiently large to cancel all the unwanted lower order terms.

**Case $i = 0$:** This case is similar to the case $i = k$ but "easier" because the non-trivial recursion is on a smaller input. We therefore omit the detailed proof. ■

## 4 The Deterministic Algorithm

In this section we present a deterministic algorithm for the multiple selection problem. It performs $B + o(B) + \mathcal{O}(n)$ comparisons.

The algorithms sets $\ell = \max(1, \lceil \lg(B/n) \rceil)$ and is based on the recursive procedure *multiselect*. The input to *multiselect* is a collection $C$ of non-empty sorted sequences and a non-empty set $R = \{r_1, \ldots, r_k\}$ of ranks. Each sequence in the collection has length at most $2\ell$. The procedure returns the elements of rank $r_1$, $r_2$, $\ldots$, $r_k$ in the input set. We solve the multiselection problem by calling *multiselect* with a collection of $n$ sequences of length one each and our set $R$ of desired ranks.

We call a sequence *short* if it has length less than $\ell$. A collection $C$ satisfies the length invariant if it contains no two short sequences of the same length.

Procedure *multiselect* works as follows: If the total number of elements in the sequences of $C$ is less than $4\ell^2$, then solve the problem by sorting all the elements and if $R$ is empty, simply return. Otherwise, we first establish the length invariant. As long as there are two short sequences of the same length, we merge them into a single sequence. Recall that two sorted sequences of length $s$ can be merged using $2s - 1$ comparisons. After the merge step, we have a collection $D$ of $q$ sequences satisfying the length invariant. We determine the median of the medians of the sequences in $D$, call it $m$, and split every sequence in $D$ at $m$, i.e., into the elements smaller than or equal to $m$ and into the elements larger than $m$. This gives us collections $C'$ and $C''$ of size $p'$ and $p''$, respectively. We have $p' \le q$, $p'' \le q$ and $p' + p'' \ge q$. We also split the ranks and make one or two recursive calls.

This ends the description of the algorithm. We turn to the analysis. For a sequence $c$ we use $|c|$ to denote its length. For a collection $C$ of sorted sequences, let $I(C) = \sum_{c \in C} |c| \lg |c|$ be the *information content* of the collection. Multiselect generates a recursion tree $T$. We use $v$ to denote an arbitrary node of the recursion tree and introduce the following notation: $I_v = I(C)$ is the information content of the collection entering node $v$, $p_v = |C|$ is the number of sequences in the collection and $n_v = \sum_{c \in C} |c|$ is the total length of the sequences in the collection. $J_v = I(D)$ is the information content of the collection after the merging step and $q_v$ is the number of sequences in the collection. $I'_v = I(C')$ and $I''_v = I(C'')$ are the information content of the sequences resulting from the partitioning step and $p'_v$ and $p''_v$ are the number of sequences in these collections, respectively. Also, the total number of elements in $C'$ is $\alpha_v n_v$. Finally, we use $C^m_v$ for the number of comparisons spent on merging in node $v$ and $C^p_v$ for the number of comparisons spent on computing the median and partitioning in node

$v$. The total number of comparisons in the calls with $n_v < 4\ell^2$ is $\mathcal{O}(n \lg \ell)$, since the calls are applied to disjoint sets of elements.

**Lemma 5.** $C_v^m \le J_v - I_v + q_v - p_v$.

**Proof:** It takes at most $2s - 1$ comparisons to merge two sequence of length $s$. Also $2s\lg(2s) - 2(s \lg s) + 1 - 2 = 2s - 1$. $\square$

**Lemma 6.** $J_v \le I_v' + I_v'' + n_v H(\alpha_v)$.

**Proof:** Let $D = \{d_1, \ldots, d_q\}$ be the collection after the merging step and let $n = n_v$ and $\alpha = \alpha_v$. Each $d_i$ is split into two sequences $d_i'$ and $d_i''$ and $C'$ is formed by the non-empty sequences in $\{d_1', \ldots, d_q'\}$. $C''$ is defined analogously. Define $\alpha_i$ as $\alpha_i = d_i'/d_i$. We have (using $0 \lg 0 = 0$)

$$\begin{aligned} J_v - I_v' - I_v'' &= \sum_{1 \le i \le q} (d_i \lg d_i - d_i' \lg d_i' - d_i'' \lg d_i'') \\ &= \sum_{1 \le i \le q} d_i(\lg d_i - \alpha_i \lg(\alpha_i d_i) - (1 - \alpha_i)\lg((1-\alpha_i)d_i) \\ &= \sum_{1 \le i \le q} d_i H(\alpha_i). \end{aligned}$$

Next observe that $\sum_i d_i/n = 1$, $\sum_i \alpha_i d_i/n = \sum_i d_i'/n = \alpha$. Thus $\sum_i (d_i/n)H(\alpha_i)$ is a convex combination of the values of $H$ at the arguments $\alpha_i$. Since $H$ is convex this is bounded by the value of $H$ at the corresponding combination of the arguments, i.e., by the value at $\sum_i (d_i/n)\alpha_i$. Thus $\sum_i (d_i/n)H(\alpha_i) \le H(\sum_i (d_i/n)\alpha_i) = H(\alpha)$. $\square$

**Lemma 7.** $C_v^m \le n_v H(\alpha_v) + I_v' + I_v'' - I_v + p_v' + p_v'' - p_v$.

**Proof:** This follows immediately from the two lemmas above and the inequality $q_v \le p_v' + p_v''$. $\square$

The next Lemma shows that the splitting process is reasonably balanced. A similar Lemma is used in the analysis of the deterministic median algorithm of Blum et al.

**Lemma 8.** *If* $n_v \ge 4\ell^2$ *then* $1/16 \le \alpha_v \le 15/16$ *and* $H(\alpha_v) \ge 1/4$.

**Proof:** We count the number of elements that are smaller than or equal to the median of the medians $m$ after partitioning.

After the merge step, we have a collection of $q_v$ sequences, each of length at most $2\ell$. Therefore, $q_v \ge \lceil n_v/2\ell \rceil$. Recall that by the invariant of the algorithm there are no two sequences of the same length less than $\ell$. Or, in other words, there can be at most one sequence for each of the lengths $1, \ldots, \ell - 1$. Consider the sequences whose median is less than or equal to $m$. Their number is $\lceil q_v/2 \rceil \ge n_v/4\ell$. In the worst case this collection contains one sequence from each of the lengths $1, \ldots, \ell - 1$. Thus, the number of non-short sequences is at least $n_v/4\ell - (\ell - 1)$ and each of them has at least $\lceil \ell/2 \rceil$ elements that are less than or equal to $m$. Thus, in total they have at least $(n_v/4\ell - (\ell-1))\lceil \ell/2 \rceil \geq n_v/8 - \ell(\ell-1)/2$ elements. Moreover, the number of elements contained in the short sequences that are smaller than or equal to $m$ are $\sum_{i=1}^{\ell-1} \lceil i/2 \rceil \geq 1/2 \sum_{i=1}^{\ell-1} i = \ell(\ell-1)/4$. Therefore, over all $\alpha_v n_v \geq n_v/8 - \ell(\ell-1)/4$. Similarly, for the number of elements greater than $m$ we can show that $(1-\alpha_v)n_v \geq n_v/8 - \ell(\ell-1)/4$ and the claim follows. $\square$

Consider the recursion tree $T$ generated by the algorithm. This is a binary tree. The leaves correspond to calls with either $n_v < 4\ell^2$ or empty $R_v$. Interior nodes have $n_v > 4\ell^2$ and non-empty $R_v$. The total cost of merging is

$$\sum_{v \in T} C_v^m \leq \sum_{v \in T} (n_v H(\alpha_v) + I'_v + I''_v - I_v + p'_v + p''_v - p_v).$$

We first bound $\sum_{v \in T} n_v H(\alpha_v)$.

**Lemma 9.** *Let $T(n, R) = \sum_{v \in T} n_v H(\alpha_v)$. Then*

$$T(n,R) \leq \sum_{j=1}^{k+1} \Delta_j \lg \frac{n}{\Delta_j} + r_k - r_1 + 16n = n \lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 16n$$

*whenever $R$ is non-empty*[5].

**Proof:** We use induction. If $n < 4\ell^2$ and $R$ is non-empty, $T(n,R) = 0$ and the right hand side is at least zero. If $R$ is non-empty, we split into subproblems $(n', R')$ and $(n'', R'')$, with $n' = \alpha n$, $n'' = (1-\alpha)n$, $R' = \{r : r \in R, r < n'\}$ and $R'' = \{r - n' : r \in R, r > n'\}$. Then $nH(\alpha) = n \lg n - n' \lg n' - n'' \lg n''$. Let $k' = |R'|$. By symmetry, we may assume $k' > 0$. Let $\Delta'_1, \ldots, \Delta'_{k'+1}$ and $\Delta''_1, \ldots, \Delta''_{k''+1}$ be the sequence of $\Delta$'s for the two subproblems. Then $\Delta'_1, \ldots, \Delta'_{k'}, \Delta'_{k'+1} + \Delta''_1, \Delta''_2 \ldots, \Delta''_{k''+1}$ is the sequence of $\Delta$'s in the original problem. We need to distinguish cases.

Assume first $k'' > 0$. Then we can apply the induction hypothesis to both subproblems and obtain (writing $x$ for $\Delta'_{k'+1}$ and $y$ for $\Delta''_1$)

$$\begin{aligned}
T(n,R) &\leq nH(\alpha) + T(n', R') + T(n'', R'') \\
&\leq n \lg n - n' \lg n' - n'' \lg n'' + n' \lg n' - \sum_{j=1}^{k'+1} \Delta'_j \lg \Delta'_j + r'_k - r_1 + 16n' \\
&\qquad\qquad + n'' \lg n'' - \sum_{j=1}^{k''+1} \Delta''_j \lg \Delta''_j + r_k - r_{k'+1} + 16n'' \\
&= n \lg n - \sum_{j=1}^{k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 16n \\
&\quad + (x+y)\lg(x+y) - x \lg x - y \lg y - (r_{k'+1} - r_{k'})
\end{aligned}$$

[5] If $R$ is empty, $k = 0$ and $r_k = 0$ and $r_1 = n$. This will not make sense in the inductive proof.

and we have established the induction step using Lemma 1 and $x+y = r_{k'+1} - r_{k'}$.

Assume next $k'' = 0$. We apply the induction hypothesis to the first subproblem and obtain

$$\begin{aligned}
T(n,R) &\le nH(\alpha) + T(n',R') \\
&\le n\lg n - n'\lg n' - n''\lg n'' + n'\lg n' - \sum_{1\le j\le k+1} \Delta'_j \lg \Delta'_j + r_k - r_1 + 16n' \\
&= n\lg n - \sum_{1\le j\le k+1} \Delta_j \lg \Delta_j + r_k - r_1 + 16n \\
&\quad + (\Delta'_{k+1} + n'')\lg(\Delta'_{k+1} + n'') - \Delta'_{k+1}\lg \Delta'_{k+1} - n''\lg n'' - 16n''
\end{aligned}$$

We have $\Delta'_{k+1} + n'' \le n \le 16n''$ by Lemma 8. An application of Lemma 1 completes the induction step. $\square$

**Lemma 10.** $\sum_{v\in T}(I'_v + I''_v - I_v + p'_v + p''_v - p_v) \le n\lg(2\ell)$.

**Proof:** The sum telescopes to

$$-I_{\text{root}} - p_{\text{root}} + \sum_{\substack{v\,\in\, T \text{ and there is} \\ \text{no recursive call for } C'}} (I'_v + p'_v) + \sum_{\substack{v\,\in\, T \text{ and there is} \\ \text{no recursive call for } C''}} (I''_v + p''_v).$$

The collection entering the root consists of sequences of length 1 and hence $I_{\text{root}} = 0$ and $p_{\text{root}} = n$. The collections for which there is no recursive call are disjoint and hence their total length is $n$. Also no sequence is longer than $2\ell$ and every sequence is non-empty. Thus their contribution is bounded by $n\lg(2\ell) + n$. $\square$

We next turn to the cost of computing the median of the medians and partitioning our sequences at the median of medians. We let $\beta$ be a constant such that the median of $z$ elements can be computed with $\beta z$ comparisons.

**Lemma 11.** *The total cost $C^p$ of computing the median of the medians and partitioning is bounded by*

$$\frac{4(\lg 2\ell + \beta)}{\ell}\sum_{j=1}^{k+1} \Delta_j \lg\frac{n}{\Delta_j} + \frac{4(\lg 2\ell + \beta)}{\ell}(r_k - r_1) + \mathcal{O}(n) + \mathcal{O}(\beta\ell n) + \mathcal{O}(n\ell\lg\ell)\,.$$

**Proof:** We split the cost for computing medians and partitioning into two parts: the cost arising from non-short sequences and the cost arising from short sequences.

The first three terms refer to the cost of finding the median of medians and partitioning arising from the non-short sequences. The median of each sequence is known since they are sorted. At each node $v$ the number of non-short sequences is at most $\lfloor n_v/\ell \rfloor \le n_v/\ell$ and hence the share of the non-short sequences is $\beta n_v/\ell$ comparisons in node $v$. In order to partition at $m$ we perform binary search on each sequence with cost at most $\lg 2\ell$, thus, the cost over all sequences is at most $n_v \lg 2\ell/\ell$. Therefore, the comparisons spent in node $v$ for non-short sequences is $(\lg 2\ell + \beta)n_v/\ell$. Next observe that $1 \leq 4H(\alpha_v)$ for any vertex $v$ and hence $(\lg 2\ell + \beta)n_v/\ell \leq 4(\lg 2\ell + \beta)H(\alpha_v)n_v/\ell$. The bound now follows from Lemma 9.

The fourth term refers to the cost contributed by the short sequences for computing the median of medians. Since we have at most $\ell - 1$ short sequences in each node, they contribute $\beta(\ell - 1)$ to the number of comparisons at each node of the recursion tree. Hence, they contribute an $\mathcal{O}(\beta\ell n)$ over all.

The last term refers to the cost of partitioning the short sequences at $m$. At each node this cost is at most $\sum_{i=1}^{\ell-1} \lg i \leq \ell \lg \ell$ and the bound follows. □

**Lemma 12.** *The total cost for the base case of the recursion is bounded by $\beta n$.*

**Proof:** The collections for which there is no recursive call are disjoint and the cost of selecting the median of $s$ elements is $\beta s$. □

**Theorem 3.** *The deterministic multi-selection algorithm uses $B + o(B) + \mathcal{O}(n)$ comparisons.*

**Proof:** Summing the bounds in Lemmas 9, 10, 11,and 12 we obtain the following upper bound:

$$
\begin{aligned}
&B + \mathcal{O}(n) + \mathcal{O}(n \lg \ell) + \mathcal{O}((1 + \lg \ell)/\ell) \cdot B + \mathcal{O}(n) + \mathcal{O}(n\ell \lg \ell) \\
&\qquad = B + \mathcal{O}(n) + \mathcal{O}\left( \frac{1 + \lg \lg(B/n)}{\lg(B/n)} + \frac{\lg(B/n) \lg \lg(B/n)}{B/n} \right) \cdot B.
\end{aligned}
$$

This is $B + o(B) + \mathcal{O}(n)$. □

## 5 Conclusion

We have shown that multiple quickselect performs an optimal number of comparisons up to a linear term. There are several ways to improve this linear term which is particularly interesting when the lower bound is linear itself. For example, when $r_k < n/2$ (or $r_1 > n/2$) it pays to choose the pivot such that its rank is just a bit larger than $r_k$ (or a bit smaller than $r_1$). It can then be shown that the linear term is reduced from $2n$ to $3n/2$. Also note that for $k = 1$ the algorithm then becomes a special case of the optimal algorithm by Floyd and Rivest [7]. Likewise, the lower bound could be further refined. However, since there is even a remaining linear size gap between upper and lower bound for sorting, completely closing the gap remains a distant possibility.

There is a long standing constant factor gap between the upper bounds for the best deterministic selection algorithms and the lower bound. Our deterministic algorithm shows that this gap becomes a lower order term for multiple selection.

Multiple quickselect is obviously highly practical. It might be quite complicated to implement our *deterministic* algorithm efficiently but at the end it might work quite well: Its cost is dominated by merging operations that are known to be quite fast. Moreover, since the algorithm executes batches of many small binary merging operations, one might execute them in parallel using multithreading or instruction parallelism. This approach might mitigate the impact of data dependencies and branch mispredictions.

## References


1. G. E. Blelloch, C. E. Leiserson, B. M. Maggs, C. G. Plaxton, S. J. Smith, and M. Zagha. A comparison of sorting algorithms for the connection machine CM-2. In *3rd ACM Symposium on Parallel Algorithms and Architectures*, pages 3–16, 1991.
2. M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.
3. J. Chambers. Partial sorting (algorithm 410). *Communications of the ACM*, 14:357–358, 1971.
4. W. Cunto and J. I. Munro. Average case selection. *J. ACM*, 36(2):270–279, 1989.
5. D. P. Dobkin and J. I. Munro. Optimal time minimal space selection algorithms. *Journal of the ACM*, 28(3):454–461, 1981.
6. D. Dor and U. Zwick. Selecting the median. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1995.
7. R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *Commun. ACM*, 18(3):165–172, 1975.
8. C.A.R. Hoare. Find (algorithm 65). *Communications of the ACM*, 4(7):321–322, 1961.
9. L. R. Ford Jr. and S. B. Johnson. A tournament problem. *AMM*, 66(5):387–389, 1959.
10. R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
11. A. Panholzer. Analysis of multiple quickselect variants. *Theor. Comput. Sci.*, 302(1-3):45–91, 2003.
12. I. Pohl. A sorting problem and its complexity. *Commun. ACM*, 15(6):462–464, 1972.
13. H. Prodinger. Multiple quickselect - Hoare's find algorithm for several elements. *Information Processing Letters*, 56:123–129, 1995.
14. A. Schönhage, M. Paterson, and N. Pippenger. Finding the median. *J. Comput. Syst. Sci.*, 13:184–199, 1976.