Algorithmica manuscript No.
(will be inserted by the editor)

Analysis of Quickselect under Yaroslavskiy's Dual-Pivoting Algorithm

Sebastian Wild · **Markus E. Nebel** · **Hosam Mahmoud**

Received: date / Accepted: date

Abstract There is excitement within the algorithms community about a new partitioning method introduced by Yaroslavskiy. This algorithm renders Quicksort slightly faster than the case when it runs under classic partitioning methods. We show that this improved performance in Quicksort is *not* sustained in Quickselect; a variant of Quicksort for finding order statistics.

We investigate the number of comparisons made by Quickselect to find a key with a randomly selected rank under Yaroslavskiy's algorithm. This *grand averaging* is a smoothing operator over all individual distributions for specific fixed order statistics. We give the exact grand average. The grand distribution of the number of comparison (when suitably scaled) is given as the fixed-point solution of a distributional equation of a contraction in the Zolotarev metric space. Our investigation shows that Quickselect under older partitioning methods slightly outperforms Quickselect under Yaroslavskiy's algorithm, for an order statistic of a random rank. Similar results are obtained for extremal order statistics, where again we find the exact average, and the distribution for the number of comparisons (when suitably scaled). Both limiting distributions are of perpetuities (a sum of products of independent mixed continuous random variables).

Mathematics Subject Classification (2000) Primary: 60C05; secondary: 68P10, 68P20.

Keywords Quicksort, Quickselect, combinatorial probability, algorithm, recurrence, average-case analysis, grand average, perpetuity, fixed point, metric, contraction.

An extended version with detailed technical proofs in the appendix is available in the arXiv: `http://arxiv.org/abs/1306.3819`.

Sebastian Wild
Computer Science Department, University of Kaiserslautern, Germany, E-mail: wild@cs.uni-kl.de

Markus E. Nebel
Computer Science Department, University of Kaiserslautern, Germany; and
Department of Mathematic and Computer Science, University of Southern Denmark, Denmark
E-mail: nebel@cs.uni-kl.de

Hosam Mahmoud
Department of Statistics, The George Washington University, Washington, D.C. 20052, U.S.A.

1 Quicksort, What Is New?

Quicksort is a classic fast sorting algorithm. It was originally published by Hoare [16]; see also [6,19,21,39]. Quicksort is the method of choice to implement a sorting function in many widely used program libraries, e. g. in the C/C++ standard library and in the Java runtime environment. The algorithm is recursive, and at each level of recursion it uses a partitioning algorithm. Classic implementations of Quicksort use a variety of partitioning methods derived from fundamental versions invented by Hoare [15,16] and refined and popularized by Sedgewick [36,37,38,40].

Very recently, a partitioning algorithm proposed by Yaroslavskiy created some sensation in the algorithms community. The excitement arises from various indications, theoretical and experimental, that Quicksort on average runs faster under Yaroslavskiy's dual pivoting (see [43]). Indeed, after extensive experimentation Oracle adopted Yaroslavskiy's dual-pivot Quicksort as the default sorting method for their Java 7 runtime environment, a software platform used on many computers worldwide.

2 Quicksort and Quickselect

Quicksort is a two-sided algorithm for sorting data (also called *keys*). In a classic implementation, it puts a pivot key in its correct position, and arranges the data in two groups relative to that pivot. Keys smaller than the pivot are put in one group, the rest are placed in the other group. The two groups are then sorted recursively.

The one-sided version (Quickselect) of the algorithm can be used to find order statistics. It is also known as Hoare's "Find" algorithm, which was first given in [15]. To find a certain order statistic, such as the first quartile, Quickselect goes through the partitioning stage, just as in Quicksort, then the algorithm decides whether the pivot is the sought order statistic or not. If it is, the algorithm terminates (announcing the pivot to be the sought element); if not, it recursively pursues only the group on the side (left or right of the pivot) where the order statistic resides. We know which side to choose, as the rank of the pivot becomes known after partitioning.

There are algorithms for specific order statistics like smallest, second smallest, largest, median, etc. However, adapting one of them to work for a different order statistic is not an easy task. Take for example the algorithm for the second largest, and suppose we want to tinker with it to find the median. It cannot be done without entirely rewriting the algorithm. On the other hand, Quickselect is one algorithm that is versatile enough to deal with *any* rank without changing any statements in it, a feature that may be appealing in practice, particularly for a library function that cannot predict which order statistic will be sought.

A standard measure for the analysis of a comparison-based sorting algorithm is the number of *data* comparisons it makes while sorting; see for example [18,19]. Other types of comparison take place while sorting, such as index or pointer comparisons. However, they are negligible in view of the fact that they mostly occur at lower asymptotic orders, and any individual one of them typically costs considerably less than an individual data comparison. For instance, comparing two indices is a comparison of two short integers, while two keys can be rather long such as business records, polynomials, or DNA strands, typically each comprising thousands of nucleotides. Hence, these additional index and pointer comparisons are often ignored in the analysis. We shall follow this tradition.

Unless further information on the input data is available (e. g., in some specialized application), one strives for an analysis that remains as generic as possible while still providing sensible predictive quality for use cases in practice. Therefore, we consider the *expected* costs when operating on an input chosen uniformly at random, which for selection means that all $n!$ orderings of n keys are equally likely. (This assumption can be enforced for any input if pivots are chosen randomly.)

We furthermore assume that the sought order statistic is also chosen uniformly at random among all n possible ranks. This models the scenario of a generic library function intended to cope with any possible selection task. We note that parts of the analysis remain feasible when the sought rank r is kept fixed (becoming a second parameter of the analysis) [18,19]. However, for *comparing* different versions of Quickselect, a one-dimensional measure is much easier to interpret and in light of our clear (and negative) results, the parametric analysis is unlikely to provide additional algorithmic insights.[1]

We do, however, think that a *distributional* analysis is important. Unlike for Quicksort, whose costs become more and more concentrated around their expectation as n increases, the standard deviation of (classic) Quickselect is of the same order as the expectation, (both are linear) [18]. This means that even for $n \to \infty$, substantial deviations from the mean are to be expected. As the use of more pivots tends to give more balanced subproblem sizes, it seems plausible that these deviations can be reduced by switching to the dual-pivot scheme described below. Therefore, we also compute the variance and a limit distribution for the number of comparisons.

It is worth mentioning that a fair contrast between comparison-based sorting algorithms and sorting algorithms based on other techniques (such as radix selection, which uses comparisons of bits) should resort to the use of one basis, such as how many bits are compared in both. Indeed, in comparing two very long bit strings, we can decide almost immediately that the two strings are different, if they differ in the first bit. In other instances, where the two strings are "similar," we may run a very long sequence of bit comparisons till we discover the difference. Attention to this type of contrast is taken up in [4,8,9,10,41].

Other associated cost measures include the number of swaps or data moves [22, 25]. We do not discuss those in detail in this paper, but we note that swaps can be analyzed in a very similar way (reusing the number of swaps in one partitioning step from our previous work on Quicksort [44]). The resulting expected values are reported in Table 1.

[1] To shed some light on that, we computed the expected comparison counts for $n = 200$ and $n = 300$ with fixed r for all ranks $1 \leq r \leq n$. We find that Yaroslavskiy's algorithm needs more comparisons in *all* these cases. We see no reason to believe that this will change for larger n.

3 Dual Pivoting

The idea of using two pivots (*dual-pivoting*) had been suggested before, see Sedgewick's and Hennequin's Ph. D. dissertations [14,36]. Nonetheless, the implementations considered at the time did not show any promise. Analysis reveals that Sedgewick's dual-pivot Quicksort variant performs an asymptotic average of $\frac{32}{15}n\ln n + \mathcal{O}(n)$ data comparisons, while the classic (single-pivot) version uses only an asymptotic average of $2n\ln n + \mathcal{O}(n)$ comparisons [36,43]. Hennequin's variant performs $2n\ln n + \mathcal{O}(n)$ comparisons [14] — asymptotically the same as classic Quicksort. However, the inherently more complicated dual-pivot partitioning process is presumed to render it less efficient in practice.

These discoveries were perhaps a reason to discourage research on sorting with multiple-pivot partitioning, till Yaroslavskiy carefully heeded implementation details. His dual-partitioning algorithm improvement broke a psychological barrier. Would such improvements be sustained in Quickselect? It is our aim in this paper to answer this question. We find out that surprisingly there is no improvement in the number of comparisons: Quickselect under Yaroslavskiy's dual-pivot partitioning algorithm (simply Yaroslavskiy's algorithm, henceforth) is slightly *worse* than classic single-pivot Quickselect.

Suppose we intend to sort n distinct keys stored in the array $A[1..n]$. Dual partitioning uses *two* pivots, as opposed to the single pivot used in classic Quicksort. Let us assume the two pivots are initially $A[1]$ and $A[n]$, and suppose their ranks are p and q. If $p > q$, we swap the pivots. While seeking two positions for the two pivots, the rest of the data is categorized in three groups: small, medium and large. Small keys are those with ranks less than p, medium keys have ranks at least p and less than q, and large keys are those with ranks at least q. Small keys are moved to positions lower than p, large keys are moved to positions higher than q, medium keys are kept in positions between $p+1$ and $q-1$. So, the two keys with ranks p and q can be moved to their correct and final positions.

After this partitioning stage, dual-pivot Quicksort then invokes itself recursively (thrice) on $A[1..p-1]$, $A[p+1..q-1]$ and $A[q+1..n]$. The boundary conditions are the very small arrays of size 0 (no keys to sort), arrays of size 1 (such an array is already sorted), and arrays of size 2 (these arrays need only one comparison between the two keys in them); in these cases no further recursion is invoked.

This is the general paradigm for dual pivoting. However, it can be implemented in many different ways. Yaroslavskiy's algorithm keeps track of three pointers:

- ℓ, moving up from lower to higher indices, and below which all the keys have ranks less than p.
- g, moving down from higher to lower indices, and above which all the keys have ranks at least q.
- k, moving up beyond ℓ and not past g. During the execution, all the keys at or below position k and above ℓ are medium, with ranks lying between p and q (both inclusively).

Algorithm 1 Dual-pivot Quickselect algorithm for finding the rth order statistic.

```
QUICKSELECT $(A, left, right, r)$
    // Assumes $left \le r \le right$.
    // Returns the element that would reside in $A[r]$ after sorting $A[left..right]$.
    if $right \le left$
        return $A[left]$
    else
        $(i_p, i_q) :=$ PARTITIONYAROSLAVSKIY$(A, left, right)$
        $c := \operatorname{sgn}(r - i_p) + \operatorname{sgn}(r - i_q)$        // Here sgn denotes the signum function.
        case distinction on the value of $c$
            in case $-2$ do return QUICKSELECT$(A, left, \ell - 1, r)$
            in case $-1$ do return $A[i_p]$
            in case $0$ do return QUICKSELECT$(A, \ell + 1, g - 1, r)$
            in case $+1$ do return $A[i_q]$
            in case $+2$ do return QUICKSELECT$(A, g + 1, right, r)$
        end cases
    end if
```

Algorithm 2 Yaroslavskiy's dual-pivot partitioning algorithm.

```
PARTITIONYAROSLAVSKIY $(A, left, right)$
    // Assumes $left \le right$.
    // Rearranges $A$ such that with $(i_p, i_q)$ the return value holds $\begin{cases} \forall\, left \le j \le i_p & A[j] \le p \\ \forall\, i_p \le j \le i_q & p \le A[j] \le q \\ \forall\, i_q \le j \le right & A[j] \ge q \end{cases}$.
    if $A[left] > A[right]$
        $p := A[right]$;   $q := A[left]$
    else
        $p := A[left]$;   $q := A[right]$
    end if
    $\ell := left + 1$;   $g := right - 1$;   $k := \ell$
    while $k \le g$
        if $A[k] < p$
            Swap $A[k]$ and $A[\ell]$
            $\ell := \ell + 1$
        else
            if $A[k] \ge q$
                while $A[g] > q$ and $k < g$ do $g := g - 1$ end while
                if $A[g] \ge p$
                    Swap $A[k]$ and $A[g]$
                else
                    Swap $A[k]$ and $A[g]$;   Swap $A[k]$ and $A[\ell]$
                    $\ell := \ell + 1$
                end if
                $g := g - 1$
            end if
        end if
        $k := k + 1$
    end while
    $\ell := \ell - 1$;   $g := g + 1$
    $A[left] := A[\ell]$;   $A[\ell] := p$      // Swap pivots to final positions
    $A[right] := A[g]$;   $A[g] := q$
    return $(\ell, g)$
```

Hence, the three pointers ℓ, k and g divide the array into four ranges, where we keep the relation of elements invariantly as given above. Graphically, this reads as follows:

$$\begin{array}{|c|c|c|c|c|c|} \hline p & \quad < p \quad & \quad p \le \circ \le q \quad & \qquad ? \qquad & \quad \ge q \quad & q \\ \hline \end{array}$$

$$\qquad \overset{\ell}{\rightarrow} \qquad\qquad \overset{k}{\rightarrow} \qquad\qquad \overset{g}{\leftarrow}$$

The adaptation of Quicksort to deliver a certain order statistic r (the rth smallest key) is straightforward. Once the positions for the two pivots are determined, we know whether $r < p$, $r = p$, $p < r < q$, $r = q$, or $r > q$. If $r = p$ or $r = q$, the algorithm declares one of the two pivots (now residing at position p or q) as the required rth order statistic, and terminates. If $r \neq p$ and $r \neq q$, the algorithm chooses one of three subarrays: If $r < p$ the algorithm recursively seeks the rth order statistic in $A[1\,..\,p-1]$, if $p < r < q$, the algorithm seeks the $(r-p)$th order statistic among the keys of $A[p+1\,..\,q-1]$; this $(r-p)$th key is, of course, ranked rth in the entire data set. If $r > q$, the algorithm seeks the $(r-q)$th order statistics in $A[q+1\,..\,n]$; this $(r-q)$th element is then ranked rth in the entire data set. Thus, only the subarray containing the desired order statistic is searched and the others are ignored.

Algorithm 1 is the formal algorithm in pseudo code. The code calls Yaroslavskiy's dual-pivot partitioning procedure (given as Algorithm 2). The code is written to work on the general subarray $A[\mathit{left}\,..\,\mathit{right}]$ in later stages of the recursion, and the initial call is $\textsc{Quickselect}(A,1,n,r)$.

Note that in Algorithm 2, variables p and q are used to denote the data elements used as pivots, whereas in the main text, p and q always refer to the *ranks* of these pivot elements relative to the current subarray. We kept the variables names in the algorithm to stay consistent with the literature.

A few words are in order to address the case of *equal elements*. If the input array contains equal keys, several competing notions of ranks exist. We choose an *ordinal ranking* that fits our situation best: The rank of an element is defined as its *index* in the array after sorting it with (a corresponding variant of) Quicksort. With this definition of ranks, Algorithm 1 correctly handles arrays with equal elements.

4 Randomness Preservation

We recall from above that our probability model on data assumes the keys to be in random order (*random permutation model*). Since only the relative ranking is important, we assume w. l. o. g. (see [19]) that the n keys are real numbers independently sampled from a common *continuous* probability distribution.

Several partitioning algorithms can be employed; a good one produces subarrays again following the random permutation model in subsequent recursive steps: *If the whole array is a (uniformly chosen) random permutation of its elements, so are the subarrays produced by partitioning.*

For instance, in classic single-pivot Quicksort, if p is the final position of the pivot, then right after the first partitioning stage the relative ranks of $A[1],\ldots,A[p-1]$ are a random permutation of $\{1,\ldots,p-1\}$ and the relative ranks of $A[p+1],\ldots,A[n]$ are a random permutation of $\{1,\ldots,n-p\}$, see [13] or [19].

Randomness preservation enhances performance on random data, and is instrumental in formulating recurrence equations for the analysis. Hoare's [16] and Lomuto's [3] single-pivot partitioning algorithms are known to enjoy this important and desirable property.

Lemma 1 *Yaroslavskiy's algorithm (Algorithm 2) is randomness preserving.*

Proof Obviously, every key comparison in Yaroslavskiy's algorithm involves (at least) one pivot element; see lines 1, 8, 12, 13 and 14 of Algorithm 2. Hennequin shows that this is a sufficient criterion for randomness preservation [13], so Yaroslavskiy's algorithm indeed creates random subarrays. □

5 Main Results

We investigate the performance of Quickselect's number of data comparisons, when it seeks a key of a randomly selected rank, while employing Yaroslavskiy's algorithm. The exact grand average number of data comparisons is given in the following statement, in which H_n is the nth harmonic number $\sum_{k=1}^{n} 1/k$.

Proposition 1 *Let C_n be the number of data comparisons exercised while Quickselect is searching under Yaroslavskiy's algorithm for an order statistic chosen uniformly at random from all possible ranks. For $n \geq 4$,*

$$\mathbf{E}[C_n] \;=\; \tfrac{19}{6}n - \tfrac{37}{5}H_n + \tfrac{1183}{100} - \tfrac{37}{5}H_n n^{-1} - \tfrac{71}{300}n^{-1} \;\sim\; \tfrac{19}{6}n.$$

We use the notation $\overset{\mathscr{D}}{=}$ to mean (exact) equality in distribution, and $\xrightarrow{\mathscr{D}}$ to mean weak convergence in distribution.

Theorem 1 *Let C_n be the number of comparisons made by Quickselect with Yaroslavskiy's algorithm while searching for an order statistic chosen uniformly at random from all possible ranks. The random variables $C_n^* := C_n/n$ converge in distribution and in second moments to a limiting random variable C^* that satisfies the distributional equations*

$$\begin{aligned} C^* \;&\overset{\mathscr{D}}{=}\; U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}C^* \;+\; \big(U_{(2)}-U_{(1)}\big)\mathbf{1}_{\{U_{(1)}<V<U_{(2)}\}}C^{*\prime} \qquad (1)\\ &\qquad +\; \big(1-U_{(2)}\big)\mathbf{1}_{\{V>U_{(2)}\}}C^{*\prime\prime} \;+\; 1+U_{(2)}\big(2-U_{(1)}-U_{(2)}\big)\,.\\ &\overset{\mathscr{D}}{=}\; X^*C^* + g(X^*,W^*)\,, \qquad (2)\end{aligned}$$

where $C^{*\prime}$ and $C^{*\prime\prime}$ are independent copies of C^*, which are also independent of $(U_{(1)},U_{(2)},V)$ and (X^*,W^*); $(U_{(1)},U_{(2)})$ are the order statistics of two independent Uniform$(0,1)$ *random variables, V is a* Uniform$(0,1)$ *random variable independent of all else, and (X^*,W^*) have a bivariate density*

$$f(x,w) \;=\; \begin{cases} 6x, & \text{for } 0<x<w<1;\\ 0, & \text{elsewhere},\end{cases}$$

and $g(X^*, W^*)$ is a fair mixture[2] *of the three random variables*

$$1+W^*(2-X^*-W^*), \quad 1+(1+X^*-W^*)(2W^*-X^*), \quad 1+(1-X^*)(X^*+W^*)\,.$$

As a corollary of Theorem 1, we find that $\mathbf{Var}[C_n^*] \sim \frac{25}{36}n^2 = 0.69\overline{4}n^2$, as $n \to \infty$. Another corollary is to write C^* explicitly as a sum of products of independent random variables:

$$C^* \stackrel{\mathscr{D}}{=} \sum_{j=1}^{\infty} g(X_j, W_j)\Big(\prod_{k=1}^{j-1} X_j\Big), \qquad \text{where} \qquad X_j \stackrel{\mathscr{D}}{=} X^*, \qquad W_j \stackrel{\mathscr{D}}{=} W^*,$$

and $\{X_j\}_{j=1}^{\infty}$ is a family of totally independent random variables,[3] and so is $\{W_j\}_{j=1}^{\infty}$.

Remark *Let $\{V_j\}_{j=1}^{\infty}$ and $\{Z_k\}_{k=1}^{\infty}$ be two families of totally independent random variables, and assume V_j is independent of Z_k for each $j,k \geq 1$. Sums of products of independent random variables of the form*

$$V_1 + V_2 Z_1 + V_3 Z_1 Z_2 + V_4 Z_1 Z_2 Z_3 + \cdots$$

are called perpetuities. They appear in financial mathematics [7], in stochastic recursive algorithms [1], and in many other areas.

Proposition 2 *Let $\hat{C}_n$ be the number of comparisons made by Quickselect with Yaroslavskiy's algorithm to find the smallest key of a random input of size n. We then have*

$$\begin{aligned}\mathbf{E}\big[\hat{C}_n\big] &= \frac{1}{24n(n-1)(n-2)}\Big(57n^4 - 48n^3 H_n - 178n^3 + 144n^2 H_n \\ &\qquad + 135n^2 - 96nH_n - 14n + 24\Big), \qquad \textit{for } n \geq 4\,, \\ &\sim \tfrac{19}{8}n\,.\end{aligned}$$

Theorem 2 *Let $\hat{C}_n$ be the number of comparisons made by Quickselect under Yaroslavskiy's algorithm on a random input of size n to find the smallest order statistic. The random variables $\hat{C}^* = \hat{C}_n/n$ converge in distribution and in second moments to a limiting random variable $\hat{C}^*$ satisfying the distributional equation*

$$\hat{C}^* \stackrel{\mathscr{D}}{=} U_{(1)}\hat{C}^* \;+\; 1 + U_{(2)}\big(2 - U_{(1)} - U_{(2)}\big), \tag{3}$$

where $U_{(1)}$ and $U_{(2)}$ are respectively the minimum and maximum of two independent random variables, both distributed uniformly on $(0,1)$.

[2] A fair mixture of three random variables is obtained by first choosing one of the three distributions at random, all three being equally likely, then generating a random variable from that distribution.

[3] For the usual definition of total independence see any classic book on probability, such as [5, p. 53], for example.

As a corollary, we find for $\hat{C}_n$, as $n \to \infty$,

$$\mathbf{E}[\hat{C}_n] \sim \tfrac{19}{6}n, \qquad \text{and} \qquad \mathbf{Var}[\hat{C}_n] \sim \tfrac{1261}{4800}n^2 = 0.262708\overline{3}n^2 .$$

Another corollary is that $\hat{C}^*$ can be written explicitly as a sum of products of independent random variables:

$$\hat{C}^* \overset{\mathcal{D}}{=} \sum_{j=1}^{\infty} Y_j \Big(\prod_{k=1}^{j-1} \hat{X}_j\Big), \quad \text{where} \quad \hat{X}_j \overset{\mathcal{D}}{=} U_{(1)}, \quad Y_j \overset{\mathcal{D}}{=} 1 + U_{(2)}\big(2 - U_{(1)} - U_{(2)}\big),$$

here $\{\hat{X}_j\}_{j=1}^{\infty}$ is a family of totally independent random variables, and so is $\{Y_j\}_{j=1}^{\infty}$. Similarly, we have

Similar results can be developed for $\check{C}_n = C_n^{(n)}$, the maximal order statistics. We note that it is not exactly symmetrical with $\hat{C}_n$. For instance, while $\check{C}_n$ has the same asymptotic mean as $\hat{C}_n$, it has a different asymptotic variance, which is namely

$$\mathbf{Var}[\check{C}_n] \sim \tfrac{1717}{4800}n^2 = 0.357708\overline{3}n^2 .$$

In fact, similar results can be developed for the number of comparisons needed for any extremal order statistic (very small or very large), that is when the rank r is $o(n)$ or $n - o(n)$.

6 Organization

The rest of the paper is devoted to the proof and is organized as follows. Section 7 sets up fundamental components of the analysis, and working notation that will be used throughout. In Section 8, we present a probabilistic analysis of Yaroslavskiy's algorithm. The analysis under rank smoothing is carried out in Section 9, which has two subsections: Subsection 9.1 is for the exact grand average, and Subsection 9.2 is for the limiting grand distribution via the contraction method. We say a few words in that subsection on the origin and recent developments of the method and its success in analyzing divide-and-conquer algorithms.

In Section 10, we import the methodology to obtain results for extremal order statistics. Again, Section 10 has two subsections: Subsection 10.1 is for the exact average, and Subsection 10.2 is for the limiting distribution. We conclude the paper in Section 11 with remarks on the overall perspective of the use of Yaroslavskiy's algorithm in Quicksort and Quickselect.

The extended version of this article contains detailed proofs in the appendix (available online at `http://arxiv.org/abs/1306.3819`).

7 Preliminaries and Notation

We shall use the following standard notation: $\mathbf{1}_{\mathscr{E}}$ is the *indicator random variable* of the event $\mathscr{E}$ that assumes the value 1, when $\mathscr{E}$ occurs, and assumes the value 0, otherwise. The notation $\xrightarrow{a.s.}$ stands for *convergence almost surely*. By $\|X\|_p := \mathbf{E}[|X|^p]^{1/p}$, $1 \le p < \infty$, we denote the L_p*-norm* of random variable X, and we say random variables $X_1, X_2, \ldots$ *converge in L_p to X*, shortly written as $X_n \xrightarrow{L_p} X$, when $\lim_{n\to\infty} \|X_n - X\|_p = 0$. Unless otherwise stated, all asymptotic equivalents and bounds concern the limit as $n \to \infty$.

Let $\mathrm{Hypergeo}(n,s,w)$ be a *hypergeometric* random variable; that is the number of white balls in a size s sample of balls taken at random without replacement (all subsets of size s being equally likely) from an urn containing a total of n white and black balls, of which w are white.

Let P_n be the (random) *rank* of the smaller of the two pivots, and Q_n be the (random) rank of the larger of the two. For a random permutation, P_n and Q_n have a joint distribution uniform over all possible choices of a distinct pair of numbers from $\{1,\ldots,n\}$. That is to say,

$$\mathbf{Prob}(P_n = p, Q_n = q) \;=\; \frac{1}{\binom{n}{2}}, \qquad \text{for } 1 \le p < q \le n.$$

It then follows that P_n and Q_n have the marginal distributions

$$\begin{aligned}\mathbf{Prob}(P_n = p) \;&=\; \frac{n-p}{\binom{n}{2}}, \qquad \text{for } p = 1,\ldots,n-1\,;\\ \mathbf{Prob}(Q_n = q) \;&=\; \frac{q-1}{\binom{n}{2}}, \qquad \text{for } q = 2,\ldots,n\,.\end{aligned}$$

Let $U_{(1)}$, and $U_{(2)}$ be the order statistics of U_1 and U_2, two independent continuous $\mathrm{Uniform}(0,1)$ random variables, i. e.

$$U_{(1)} \;=\; \min\{U_1,U_2\}, \qquad \text{and} \qquad U_{(2)} \;=\; \max\{U_1,U_2\}\,.$$

The two order statistics have the joint density

$$f_{(U_{(1)},U_{(2)})}(x,y) \;=\; \begin{cases} 2, & \text{for } 0 < x < y < 1;\\ 0, & \text{elsewhere,}\end{cases} \tag{4}$$

and consequently have the marginal densities

$$f_{U_{(1)}}(x) \;=\; \begin{cases} 2(1-x), & \text{for } 0 < x < 1;\\ 0, & \text{elsewhere,}\end{cases} \quad \text{and} \quad f_{U_{(2)}}(y) \;=\; \begin{cases} 2y, & \text{for } 0 < y < 1;\\ 0, & \text{elsewhere.}\end{cases} \tag{5}$$

The ensuing technical work requires all the variables to be defined on the same probability space. Let (U_1,U_2,V) be three independent $\mathrm{Uniform}(0,1)$ random variables defined on the probability space $([0,1], \mathbb{B}_{[0,1]}, \lambda)$, where $\mathbb{B}_{[0,1]} = B \cap [0,1]$, for B the usual *Borel sigma field* on the real line, and λ is the *Lebesgue measure*. We have

$$R_n \;\overset{\mathscr{D}}{=}\; \lceil nV \rceil \;\overset{\mathscr{D}}{=}\; \mathrm{Uniform}[1..n]\,. \tag{6}$$

8 Analysis of Yaroslavskiy's Dual-Partitioning

An analysis of Quickselect using Yaroslavskiy's algorithm (Algorithm 2) requires a careful examination of this algorithm. In addition, being a novel partitioning method, it is a goal to analyze the method in its own right.

Let T_n be the number of comparisons exercised in the first call to Yaroslavskiy's algorithm. This quantity serves as a *toll function* for the recurrence relation underlying Quickselect: For unfolding the recurrence at size n, we have to "pay" a toll of T_n comparisons. The distribution of T_n below is given implicitly in the arguments of [43] and later used explicitly in [42,44].

Lemma 2 *The number of comparisons T_n of Yaroslavskiy's partitioning method satisfies the following distributional equation conditional on (P_n, Q_n):*

$$\begin{aligned} T_n \;\overset{\mathcal{D}}{=}\; & n-1 \;+\; \mathrm{Hypergeo}(n-2, n-P_n-1, Q_n-2) \\ & \qquad +\; \mathrm{Hypergeo}(n-2,\;\; Q_n-2\;\; , n-Q_n) \;+\; 3\cdot\mathbf{1}_{\{A[Q_n]>\max\{A[1],A[n]\}\}}\,. \end{aligned}$$

Corollary 1 [43] *The expectation of T_n is given by*

$$\mathbf{E}[T_n] \;=\; \tfrac{19}{12}(n+1)-3.$$

Corollary 2 [44] *The normalized number of comparisons $T_n^* := T_n \,/\, n$ converges to a limit T^* in L_2:*

$$T_n^* \;\xrightarrow{L_2}\; T^*\,, \qquad \textit{with} \qquad T^* \;\overset{\mathcal{D}}{=}\; 1+U_{(2)}\bigl(2-U_{(1)}-U_{(2)}\bigr)\,.$$

Remark 1 We re-obtain the leading term coefficient of $\mathbf{E}[T_n]$ as the mean of the limit distribution of T_n^*,

$$\mathbf{E}[T^*] \;=\; 1+\int_0^1\!\!\int_0^y y(2-x-y)f_{(U_{(1)},U_{(2)})}(x,y)\,dx\,dy \;=\; \tfrac{19}{12}\,.$$

The bivariate density $f_{(U_{(1)},U_{(2)})}(x,y)$ is given in equation (4).

9 Analysis of Yaroslavskiy's Algorithm for Random Rank

Let $C_n^{(r)}$ be the *number of comparisons* made by Quickselect under Yaroslavskiy's algorithm on a random input of size n to seek the rth order statistic. While this variable is easy to analyze for extremal values r (nearly smallest and nearly largest), it is harder to analyze for intermediate values of r, such as when $r = \lfloor 0.17n \rfloor$.

For a library implementation of Quickselect, though, it is quite natural to consider r to be part of the input (so that the only fixed parameter is n). Analyzing $C_n^{(R)}$ when R itself is random provides smoothing over all possible values of $C_n^{(r)}$. We let $R = R_n$ be a random variable distributed like Uniform$[1\,..\,n]$, i. e., every possible rank is requested with the same probability. This rank randomization averages over the easy and hard cases, which makes the problem of moderate complexity and amenable to analysis.

In this case we can use the simplified notation $C_n := C_n^{(R_n)}$. One seeks a grand average of all averages, a grand variance of all variances and a grand (average) distribution of all distributions in the specific cases of r as a global measure over all possible order statistics. This smoothing technique was introduced in [23], and was used successfully in [20,30]. Panholzer and Prodinger give a generating function formulation for grand averaging [29].

Right after the first round of partitioning, the two pivots (now moved to positions P_n and Q_n) split the data array into three subarrays: $A[1..P_n-1]$ containing keys with ranks smaller than P_n, $A[P_n+1..Q_n-1]$ containing keys with ranks between P_n and Q_n (both inclusively), and $A[Q_n+1..n]$ containing keys with ranks that are at least as large as Q_n. Quickselect is then invoked recursively on one of the three subarrays, depending on the desired order statistic.

As we have pairwise distinct elements almost surely, ranks are in one-to-one correspondence with key values and the three subarrays contain ranks *strictly* smaller, between and larger than the pivots. Therefore, we have the stochastic recurrence

$$C_n \overset{\mathscr{D}}{=} T_n + C_{P_n-1}\mathbf{1}_{\{R_n<P_n\}} + C'_{Q_n-P_n-1}\mathbf{1}_{\{P_n<R_n<Q_n\}} + C''_{n-Q_n}\mathbf{1}_{\{R_n>Q_n\}}, \qquad (7)$$

where, for each $i \geq 0$, $C'_i \overset{\mathscr{D}}{=} C''_i \overset{\mathscr{D}}{=} C_i$, and $(C_{P_n}, C'_{Q_n-P_n-1}, C''_{n-Q_n})$ are conditionally independent (in the sense that, given $P_n = p$, and $Q_n = q$, C_{p-1}, C'_{q-p-1}, and C''_{n-p} are independent).

9.1 Exact Grand Average

The distributional equation (7) yields a recurrence for the average:

$$\mathbf{E}[C_n] = \mathbf{E}[T_n] + 3\,\mathbf{E}\big[C_{P_n-1}\mathbf{1}_{\{R_n<P_n\}}\big], \qquad (8)$$

where symmetry is used to triple the term containing the first indicator. By conditioning on (P_n, Q_n) and the independent R_n, using Corollary 1 we get

$$\begin{aligned}\mathbf{E}[C_n] &= \mathbf{E}[T_n] + 3\sum_{1\le p<q\le n}\sum_{r=1}^{n}\mathbf{E}\big[C_{P_n-1}\mathbf{1}_{\{R_n<P_n\}}\,|\,P_n=p, Q_n=q, R_n=r\big]\\ &\qquad\qquad \times \mathbf{Prob}(P_n=p, Q_n=q, R_n=r)\\ &= \tfrac{19}{12}(n+1)-3 + 3\sum_{p=1}^{n}\sum_{r=1}^{n}\mathbf{E}\big[C_{p-1}\mathbf{1}_{\{r<p\}}\big]\times\mathbf{Prob}(P_n=p)\,\mathbf{Prob}(R_n=r)\\ &= \tfrac{19}{12}(n+1)-3 + \frac{6}{n^2(n-1)}\sum_{p=1}^{n}(p-1)(n-p)\,\mathbf{E}[C_{p-1}]. \qquad (9)\end{aligned}$$

This recurrence equation can be solved via generating functions. Let

$$A(z) := \sum_{n=0}^{\infty} n\,\mathbf{E}[C_n]\,z^n.$$

be the (ordinary) generating function for $n\,\mathbf{E}[C_n]$.

First, organize the recurrence (9) in the form

$$n^2(n-1)\,\mathbf{E}[C_n] \;=\; n^2(n-1)\big(\tfrac{19}{12}(n+1)-3\big) \;+\; 6\sum_{p=1}^{n}(p-1)(n-p)\,\mathbf{E}[C_{p-1}].$$

Next, multiply both sides of the equation by z^n (for $|z|<1$), and sum over $n \ge 3$, the range of validity of the recurrence, to get

$$z^2\sum_{n=3}^{\infty}\big(n^2(n-1)\,\mathbf{E}[C_n]\big)z^{n-2} \;=\; 6\sum_{n=3}^{\infty}\sum_{k=1}^{n}(n-k)(k-1)\,\mathbf{E}[C_{k-1}]z^n \;+\; g(z)\,,$$

where

$$g(z) \;=\; \sum_{n=3}^{\infty} n^2(n-1)\big(\tfrac{19}{12}(n+1)-3\big)\,z^n \;=\; \frac{z^3}{(1-z)^5}\big(7z^4-35z^3+70z^2-64z+60\big)\,.$$

Shifting summation indices and using the boundary conditions $C_0 = C_1 = 0$, and $C_2 = 1$, we extend the series to start at $n = 0$, and get

$$z^2\big(A''(z) - 2^2(2-1)\cdot 1z^0\big) \;=\; 6\sum_{n=0}^{\infty}\sum_{k=0}^{n}(n-k)z^{n-k}\times\big(k\,\mathbf{E}[C_k]\big)z^k \;+\; g(z).$$

Finally, we get an Euler differential equation

$$z^2A''(z) \;=\; 6\frac{z^2}{(1-z)^2}A(z) + 4z^2 + g(z),$$

to be solved under the boundary conditions $A(0) = 0$, and $A'(0) = 0$. The solution to this differential equation is

$$A(z) \;=\; \frac{1}{300(1-z)^3}\Big(2220z - 510z^2 + 830z^3 - 1185z^4 + 699z^5 - 154z^6 \\ + 2220(1-z)\ln(1-z)\Big).$$

Extracting coefficients of z^n, we find for $n \ge 4$,

$$\mathbf{E}[C_n] \;=\; \tfrac{19}{6}n - \tfrac{37}{5}H_n + \tfrac{1183}{100} - \tfrac{37}{5n}H_n - \tfrac{71}{300n} \;\sim\; \tfrac{19}{6}n, \qquad \text{as } n\to\infty.$$

Proposition 1 is proved. □

9.2 Limit Distribution

Higher moments are harder to compute by direct recurrence as was done for the mean. For instance, exact variance computation involves rather complicated dependencies. We need a shortcut to determine the asymptotic distribution (i. e. all asymptotic moments), without resorting to exact calculation of each moment. A tool suitable for this task is the *contraction method*.

The contraction method was introduced by Rösler [32] in the analysis of the Quicksort algorithm, and it soon became a popular method because of the transparency it provides in the limit. Rachev and Rüschendorf added several useful extensions [31] and general contraction theorems, and multivariate extensions are available [27,28,33]. A valuable survey of the method was given by Rösler [35]. Neininger gives a variety of applications to random combinatorial structures and algorithms such as random search trees, random recursive trees, random digital trees and Mergesort [28]. The contraction method has also been used in the context of classic Quickselect [12,34]. Other methods for the analysis of Quickselect have been used; for example, Grübel uses Markov chains [11].

We shall use the contraction method to find the grand distribution of Quickselect's number of comparisons under rank smoothing.

By dividing (7) by n and rewriting the fractions, we find

$$\begin{aligned}\frac{C_n}{n} \overset{\mathscr{D}}{=} \; & \frac{C_{P_n-1}}{P_n-1}\cdot\frac{P_n-1}{n}\mathbf{1}_{\{R_n<P_n\}} \\ & + \frac{C'_{Q_n-P_n-1}}{Q_n-P_n-1}\cdot\frac{Q_n-P_n-1}{n}\mathbf{1}_{\{P_n<R_n<Q_n\}} \\ & + \frac{C''_{n-Q_n}}{n-Q_n}\cdot\frac{n-Q_n}{n}\mathbf{1}_{\{R_n>Q_n\}} \\ & + \frac{T_n}{n}\,. \end{aligned} \tag{10}$$

This equation is conveniently expressed in terms of the *normalized* random variables $C_n^* := C_n/n$, that is

$$\begin{aligned} C_n^* \overset{\mathscr{D}}{=} \; & C^*_{P_n-1}\frac{P_n-1}{n}\mathbf{1}_{\{R_n<P_n\}} + C^{*\prime}_{Q_n-P_n-1}\frac{Q_n-P_n-1}{n}\mathbf{1}_{\{P_n<R_n<Q_n\}} \\ & + C^{*\prime\prime}_{n-Q_n}\frac{n-Q_n}{n}\mathbf{1}_{\{R_n>Q_n\}} + T_n^*, \end{aligned}$$

where for each $j \ge 0$, $C_j^{*\prime} \overset{\mathscr{D}}{=} C_j^{*\prime\prime} \overset{\mathscr{D}}{=} C_j^*$ and each of the families $\{C_j^*\}$, $\{C_j^{*\prime}\}$, $\{C_j^{*\prime\prime}\}$, $\{T_j\}$, and $\{R_j\}$ is comprised of totally independent random variables. This representation suggests a limiting functional equation as follows. If C_n^* converges to a limit C^*, so will $C^*_{P_n-1}$ because $P_n \to \infty$ almost surely, and it is plausible to guess that the combination $C^*_{P_n-1}\frac{P_n-1}{n}\mathbf{1}_{\{R_n<P_n\}}$ converges in distribution to $C^*U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}$. Likewise, it is plausible to guess that

$$C^{*\prime}_{Q_n-P_n-1}\frac{Q_n-P_n-1}{n}\mathbf{1}_{\{P_n<R_n<Q_n\}} \xrightarrow{\mathscr{D}} C^{*\prime}\big(U_{(2)}-U_{(1)}\big)\mathbf{1}_{\{U_{(1)}<V<U_{(2)}\}}\,,$$

and

$$C^{*\prime\prime}_{n-Q_n} \frac{n-Q_n}{n} \mathbf{1}_{\{R_n > Q_n\}} \ \xrightarrow{\mathcal{D}}\ C^{*\prime\prime}\bigl(1-U_{(2)}\bigr)\mathbf{1}_{\{V>U_{(2)}\}}\,,$$

where $C^{*\prime} \overset{\mathcal{D}}{=} C^{*\prime\prime} \overset{\mathcal{D}}{=} C^*$, and $\bigl(C^*, C^{*\prime}, C^{*\prime\prime}\bigr)$ are totally independent.

To summarize, if C^*_n converges in distribution to a limiting random variable C^*, one can guess that the limit satisfies the following distributional equation:

$$\begin{aligned} C^* \ \overset{\mathcal{D}}{=}\ & U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}C^* \ +\ \bigl(U_{(2)}-U_{(1)}\bigr)\mathbf{1}_{\{U_{(1)}<V<U_{(2)}\}}C^{*\prime} \\ & +\ \bigl(1-U_{(2)}\bigr)\mathbf{1}_{\{V>U_{(2)}\}}C^{*\prime\prime} \ +\ T^*, \end{aligned} \tag{11}$$

with $\bigl(C^*, C^{*\prime}, C^{*\prime\prime}, (U_{(1)}, U_{(2)}), V\bigr)$ being totally independent, and T^* as given in Corollary 2.

The formal proof of convergence is done by coupling the random variables C_n, P_n, Q_n, and R_n to be defined on the same probability space, and showing that the *distance* between the distributions of C^*_n and C^* converges to 0 in some suitable metric space of distribution functions. Here, we use the *Zolotarev metric* ζ_2, for which Neininger and Rüschendorf give convenient contraction theorems [28]. Convergence in ζ_2 then implies the claimed convergence in distribution and in second moments. The technical details are provided in Appendix B of the extended version of this article.

The representation in (11) admits direct calculation of asymptotic mean and variance. Taking expectations on both sides and exploiting symmetries gives

$$\begin{aligned} \mathbf{E}[C^*] \ &=\ \mathbf{E}\bigl[U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}\bigr]\,\mathbf{E}[C^*] \ +\ \mathbf{E}\bigl[\bigl(U_{(2)}-U_{(1)}\bigr)\mathbf{1}_{\{U_{(1)}<V<U_{(2)}\}}\bigr]\,\mathbf{E}[C^{*\prime}] \\ &\qquad +\ \mathbf{E}\bigl[\bigl(1-U_{(1)}\bigr)\mathbf{1}_{\{V>U_{(2)}\}}\bigr]\,\mathbf{E}[C^{*\prime\prime}] \ +\ \mathbf{E}[T^*] \\ &=\ 3\,\mathbf{E}\bigl[U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}\bigr]\,\mathbf{E}[C^*] \ +\ \tfrac{19}{12}\,. \end{aligned}$$

So, we compute

$$\begin{aligned} \mathbf{E}\bigl[U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}\bigr] \ &=\ \int_{x=0}^{1}\int_{v=0}^{1} x\,\mathbf{1}_{\{v<x\}}\, f_{U_{(1)}}(x)\, f_V(v)\, dv\, dx \\ &=\ 2\int_{x=0}^{1}\int_{v=0}^{x} x(1-x)\, dv\, dx \ =\ \tfrac{1}{6}\,. \end{aligned}$$

It follows that

$$\mathbf{E}[C^*] \ =\ \tfrac{19}{6}\,, \qquad \text{and, as } n\to\infty, \qquad \mathbf{E}[C_n] \ \sim\ \tfrac{19}{6}n\,.$$

Similarly, we can get the asymptotic variance of C_n. We only sketch this calculation. First, square the distributional equation (11), then take expectations. There will appear ten terms on the right-hand side. The three terms involving $(C^*)^2$ are symmetrical, and the three terms involving cross-products of indicators are 0 (the indicators are for mutually exclusive events). By independence, we have

$$\begin{aligned}\mathbf{E}\big[(C^*)^2\big] &= 3\,\mathbf{E}\big[U_{(1)}^2\mathbf{1}_{\{V<U_{(1)}\}}\big]\,\mathbf{E}\big[(C^*)^2\big]\\ &\quad+ 2\,\mathbf{E}\big[T^*U_{(1)}\mathbf{1}_{\{V<U_{(1)}\}}\big]\,\mathbf{E}[C^*]\\ &\quad+ 2\,\mathbf{E}\big[T^*\big(U_{(2)}-U_{(1)}\big)\mathbf{1}_{\{U_{(1)}<V<U_{(2)}\}}\big]\,\mathbf{E}[C^{*\prime}]\\ &\quad+ 2\,\mathbf{E}\big[T^*\big(1-U_{(2)}\big)\mathbf{1}_{\{V>U_{(2)}\}}\big]\,\mathbf{E}[C^{*\prime\prime}]\\ &\quad+ \mathbf{E}\big[(T^*)^2\big].\end{aligned}$$

We show the computation for one of these ingredients:

$$\mathbf{E}\big[U_{(1)}^2\mathbf{1}_{\{V<U_{(1)}\}}\big] \;=\; 2\int_{y=0}^{1}\int_{x=0}^{y}\int_{v=0}^{1} x^2\mathbf{1}_{\{v<x\}}\,dv\,dx\,dy \;=\; \tfrac{1}{10}\,.$$

After carrying out similar calculations and using Corollary 2, we obtain

$$\mathbf{E}\big[(C^*)^2\big] \;=\; \tfrac{3}{10}\,\mathbf{E}\big[(C^*)^2\big]+2\Big(\tfrac{43}{180}\,\mathbf{E}[C^*]+\tfrac{53}{180}\,\mathbf{E}[C^{*\prime}]+\tfrac{1}{4}\,\mathbf{E}[C^{*\prime\prime}]\Big)+\tfrac{229}{90}\,.$$

We can solve for $\mathbf{E}\big[(C^*)^2\big]$ and by inserting $\mathbf{E}[C^*]=\mathbf{E}[C^{*\prime}]=\mathbf{E}[C^{*\prime\prime}]=\frac{19}{6}$ get

$$\mathbf{E}\big[(C^*)^2\big] \;=\; \tfrac{10}{7}\Big(2\big(\tfrac{43}{180}+\tfrac{53}{180}+\tfrac{1}{4}\big)\tfrac{19}{6}+\tfrac{229}{90}\Big) \;=\; \tfrac{193}{18}\,.$$

The variance follows:

$$\begin{aligned}\mathbf{Var}[C_n] &= \mathbf{E}[C_n^2]-\big(\mathbf{E}[C_n]\big)^2 \;\sim\; \Big(\mathbf{E}[(C^*)^2]-\big(\mathbf{E}[C^*]\big)^2\Big)n^2\\ &= \big(\tfrac{193}{18}-\tfrac{361}{36}\big)n^2 \;=\; \tfrac{25}{36}n^2, \qquad \text{as } n\to\infty\,.\end{aligned}$$

We next present an explicit (unique) solution to the distributional equation (11). A random variable with this distribution takes the form of a perpetuity.

Lemma 3 *Every solution C^* of* (11) *also satisfies the distributional equation* (2).

The proof is by showing that the characteristic functions for the solutions of both equations coincide. Details are given Appendix D of the extended version of this article.

The representation in Lemma 3 allows us to obtain an expression for C^* as a sum of products of independent random variables. Toward this end, let $X_1, X_2, \ldots$ be independent copies of X^*, and let $Y_1, Y_2, \ldots$ be independent copies of $g(X^*, W^*)$, then

$$C^* \;\overset{\mathscr{D}}{=}\; Y_1+X_1C^* \;\overset{\mathscr{D}}{=}\; Y_1+X_1(Y_2+X_2C^*).$$

Note that because C^* is independent of both X_1 and Y_1, the X and Y introduced in the iteration must be independent copies of X_1 and Y_1. Continuing the iterations (always introducing new independent random variables), we arrive at

$$\begin{aligned} C^* &\overset{\mathcal{D}}{=} Y_1 + X_1 Y_2 + X_1 X_2 (Y_3 + X_3 C^*) \\ &\vdots \\ &\overset{\mathcal{D}}{=} \sum_{j=1}^{M} \Big(Y_j \prod_{k=1}^{j-1} X_k \Big) + X_1 X_2 \cdots X_M C^*, \end{aligned} \tag{12}$$

for any positive integer M. However, by the strong law of large numbers,

$$\frac{1}{M} \ln(X_1 X_2 \dots X_M) \xrightarrow{a.s.} \mathbf{E}[\ln X^*] = -\tfrac{5}{6}, \qquad \text{as } M \to \infty\,,$$

and

$$X_1 X_2 \dots X_M \xrightarrow{a.s.} 0, \qquad \text{as } M \to \infty\,.$$

Hence, we can proceed with the limit of (12) and write

$$C^* \overset{\mathcal{D}}{=} \sum_{j=1}^{\infty} \Big(Y_j \prod_{k=1}^{j-1} X_k \Big)\,.$$

10 Analysis of Yaroslavskiy's Algorithm for Extremal Ranks

The methods used for deriving results for dual-pivot Quickselect to locate a key of random rank carry over to the case of a relatively small or relatively large extremal order statistic We shall only sketch the arguments and results for the case $r = 1$, as they closely mimic what has been done for a random rank.

Let $\hat{C}_n := C_n^{(1)}$ be the number of key comparisons required by Quickselect running with Yaroslavskiy's algorithm to find the smallest element in an array $A[1\,..\,n]$ of random data. If the smaller of the two pivots (of random rank P_n) in the first round is not the smallest key, the algorithm always pursues the leftmost subarray $A[1\,..\,P_n - 1]$. We thus have the recurrence

$$\hat{C}_n \overset{\mathcal{D}}{=} \hat{C}_{P_n - 1} + T_n\,. \tag{13}$$

10.1 Exact Mean

Equation (13) yields a recurrence for the average

$$\mathbf{E}[\hat{C}_n] = \mathbf{E}[T_n] + \mathbf{E}\big[\hat{C}_{P_n - 1}\big].$$

Conditioning on P_n, we find

$$\begin{aligned} \mathbf{E}[C_n] &= \mathbf{E}[T_n] + \sum_{p=1}^{n} \mathbf{E}\big[C_{P_n - 1} \,|\, P_n = p\big]\, \mathbf{Prob}(P_n = p) \\ &= \tfrac{19}{12}(n+1) - 3 + \frac{1}{\binom{n}{2}} \sum_{p=1}^{n} (n-p)\, \mathbf{E}[C_{p-1}]. \end{aligned} \tag{14}$$

This recurrence equation can be solved via generating functions, by steps very similar to what we did in Subsection 9.1, and we only give an outline of intermediate steps. If we let

$$\hat{A}(z) \;:=\; \sum_{n=0}^{\infty} \mathbf{E}[\hat{C}_n] z^n ,$$

multiply (14) by $n(n-1)z^n$ and sum over $n \ge 3$, we get an Euler differential equation

$$z^2\big(\hat{A}''(z) - 2\big) \;=\; \frac{2z^2\hat{A}(z)}{(1-z)^2} + h(z) ,$$

with

$$h(z) \;:=\; \sum_{n=3}^{\infty} \big(\tfrac{19}{12}(n+1) - 3\big) n(n-1) z^n \;=\; \frac{z^3}{2(1-z)^4}\big(-7z^3 + 28z^2 - 42z + 40\big) .$$

This differential equation is to be solved under the boundary conditions $\hat{A}(0) = 0$, and $\hat{A}'(0) = 0$. The solution is

$$\hat{A}(z) \;=\; \frac{1}{24(1-z)^2}\big(36z + 42z^2 - 28z^3 + 7z^4 + 12\big(3 - 6z^2 + 4z^3 - z^4\big)\ln(1-z)\big) .$$

Extracting coefficients of z^n, we find, for $n \ge 4$,

$$\begin{aligned}\mathbf{E}\big[C_n^{(1)}\big] \;&=\; \frac{1}{24n(n-1)(n-2)}\Big(57n^4 - 48n^3H_n - 178n^3 + 144n^2H_n \\ &\qquad\qquad + 135n^2 - 96nH_n - 14n + 24\Big) \\ &\sim\; \tfrac{19}{8}n, \qquad \text{as } n \to \infty .\end{aligned}$$

10.2 Limit Distribution

By similar arguments as in case of random ranks we see that $\hat{C}_n^* := \hat{C}_n/n$ approaches $\hat{C}^*$, a random variable satisfying the distributions equation

$$\hat{C}^* \;\overset{\mathscr{D}}{=}\; U_{(1)}\hat{C}^* + T^* , \tag{15}$$

where $(U_{(1)}, T^*)$ is independent of $\hat{C}^*$. We formally establish convergence in law and second moments by showing that the distance of the distributions of $\hat{C}_n^*$ and $\hat{C}^*$ diminishes to 0 in the Zolotarev metric ζ_2. We go through the technical work using a handy theorem of Neininger and Rüschendorf [28], which standardized the approach. By now, these techniques are popular, and we do not show the details here. However, they are given in Appendix C of the extended version of this article.

The representation in equation (15) allows us to obtain an expression for $\hat{C}^*$ by an unwinding process like that we used for C^*; one gets

$$\hat{C}^* \;\overset{\mathscr{D}}{=}\; \sum_{j=1}^{\infty}\Big(Y_j \prod_{k=1}^{j-1} X_k\Big) ,$$

Cost Measure		error	**Quickselect with Yaroslavskiy's Algorithm**	**Classic Quickselect with Hoare's Algorithm**
			when selecting a uniformly chosen order statistic	
Comparisons	expectation	$\mathcal{O}(\log n)$	$3.1\overline{6}n$	$3n$ †
	std. dev.	$o(n)$	$0.8\overline{3}n$	$1n$ †
Swaps	expectation	$\mathcal{O}(\log n)$	$1n$ ‡	$0.5n$ *
			when selecting an extremal order statistic	
Comparisons	expectation	$\mathcal{O}(\log n)$	$2.375n$	$2n$ †
	std. dev.	$o(n)$	small: $0.512551n$ large: $0.598087n$	$0.707107n$ †
Swaps	expectation	$\mathcal{O}(\log n)$	$0.75n$ ‡	$0.\overline{3}n$ *

† see [23]; Theorems 1 and 2.
‡ by the linear relation of expected swaps and comparisons, we can insert the toll function for swaps from [44].
* see [17]; integrating over $\alpha \in [0,1]$ resp. taking $\alpha \to 0$ in eq (12).

Table 1 Main results of this paper and the corresponding results for classic Quickselect from the literature.

with $\{X_j\}_{j=1}^{\infty}$ and $\{Y_j\}_{j=1}^{\infty}$ being two families of totally independent random variables whose members are all distributed like $U_{(1)}$ respectively T^*. This completes a sketch of the proof of Theorem 2. □

11 Conclusion

In this paper, we discussed the prospect of running Quickselect making use of a dual-pivot partitioning strategy by Yaroslavskiy, which recently provided a speedup for Quicksort and is used today in the library sort of Oracle's Java 7. It has been proven that, for sorting, the total number of comparisons becomes smaller on average, upon using Yaroslavskiy's algorithm compared to the classic single-pivot variant [43]. Even if a single partitioning phase may need more comparisons than in the classic case, the reduced sizes of the subproblems to be processed recursively — the input to be sorted is partitioned into three instead of two parts — lead to an overall saving.

The speedup in Quicksort by Yaroslavskiy's partitioning algorithm raises the hope for similar improvements in Quickselect. However, our detailed analysis, presented in this paper and summarized in Table 1, proves the opposite: When searching for a (uniformly) random rank, we find an expected number of comparisons of asymptotically $\frac{19}{6}n = 3.1\overline{6}n$ for a Quickselect variant running under Yaroslavskiy's algorithm, as opposed to $3n$ for classic Quickselect. For extremal cases, i. e. for ranks close to the minimum or maximum, an asymptotic average of $\frac{19}{8}n = 2.375n$ comparisons is needed, whereas the classic algorithm only uses $2n$. Though not considered here in detail, similar trends are observed for the number of swaps: In expectation, Yaroslavskiy's algorithm, also needs more swaps than the classic variant.

The observed inferiority of dual-pivoting in Quickselect goes beyond the case of Yaroslavskiy's algorithm. A simple argument shows that *any* dual-pivoting method must use at least $\frac{3}{2}n$ comparisons on average for a single partitioning step [2, Theorem 1]. Even with this optimal method, Quickselect needs $3n + o(n)$ comparisons on average to select a random order statistic and $2.25n + o(n)$ comparisons when searching for extremal elements; no improvement over classic Quickselect in both measures.

Our analysis provides deeper insight than just the average case — we derived variances and fixed-point equations for the distribution of the number of comparisons. Even though of less practical interest than the average considerations, it is worth noting that the variance of the number of comparisons made by Quickselect under Yaroslavskiy's algorithm is significantly smaller than for the classic one, making actual costs more predictable.

We can give some intuition based on our analysis on why dual pivoting does not improve Quickselect. As already pointed out, the new algorithm may need more comparisons for a single partitioning round than the classic strategy, but leads to smaller subproblems to be handled recursively. In classic Quickselect, using pivot p and searching for a random rank r, we use n comparisons to exclude from recursive calls either $n - p$ elements, if $r < p$, or p elements, if $r > p$, or all n elements, if $r = p$. Averaging over all p and r, this implies that a single comparison helps to exclude $\frac{1}{3} + o(1)$ elements from further computations, as $n \to \infty$. When interpreting our findings accordingly, Quickselect under Yaroslavskiy's algorithm excludes asymptotically only $\frac{6}{19} \approx 0.3125$ elements per comparison on average. We have to conclude that the reduction in subproblem sizes is not sufficient to compensate for the higher partitioning costs.

Nevertheless, the attempts to improve the Quickselect algorithm are not a complete failure. Preliminary experimental studies give some hope for a faster algorithm in connection with cleverly chosen pivots (comparable to the median-of-three strategy well-known in the classic context), especially for presorted data. Future research may focus on this scenario, trying to identify an optimal choice for the pivots. Related results are known for classic Quickselect [24,26] and Yaroslavskiy's algorithm in Quicksorting [45].

Furthermore, it would be interesting to extend our analysis to the number of bit comparisons instead of atomic key comparisons. This is especially of interest in connection with nonprimitive data types like strings. However, in this context one typically has to deal with much more complicated analysis for the resulting subproblems no longer preserve randomness in the subarrays (see [10] for corresponding results for classic Quickselect). As a consequence, the methods used in this paper are no longer applicable.

References

1. Alsmeyer, G., Iksanov, A., Rösler, U.: On distributional properties of perpetuities. Journal of Theoretical Probability **22**, 666–682 (2009)
2. Aumüller, M., Dietzfelbinger, M.: Optimal Partitioning for Dual Pivot Quicksort (2013). URL `http://arxiv.org/abs/1303.5217`

3. Bentley, J.: Programming pearls: how to sort. Communications of the ACM **27**(4), 287–291 (1984)
4. Bindjeme, P., Fill, J.: The limiting distribution for the number of symbol comparisons used by quicksort is nondegenerate (extended abstract). DMTCS Proceedings **0**(01) (2012)
5. Chung, K.L.: A Course in Probability Theory, 3rd edn. Academic Press (2001)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
7. Embrechts, P., Klüppelberg, C., Mikosch, T.: Modelling Extremal Events. Springer Verlag, Berlin, Heidelberg (1997)
8. Fill, J., Janson, S.: The number of bit comparisons used by Quicksort: an average-case analysis. Electronic Journal of Probability **17**, 1–22 (2012)
9. Fill, J.A.: Distributional convergence for the number of symbol comparisons used by QuickSort. The Annals of Applied Probability **23**(3), 1129–1147 (2013)
10. Fill, J.A., Nakama, T.: Analysis of the Expected Number of Bit Comparisons Required by Quickselect. Algorithmica **58**(3), 730–769 (2009)
11. Grübel, R.: Hoare's Selection Algorithm: A Markov Chain Approach. Journal of Applied Probability **35**(1), 36–45 (1998)
12. Grübel, R., Rösler, U.: Asymptotic Distribution Theory for Hoare's Selection Algorithm. Advances in Applied Probability **28**(1), 252–269 (1996)
13. Hennequin, P.: Combinatorial analysis of Quicksort algorithm. Informatique théorique et applications **23**(3), 317–333 (1989)
14. Hennequin, P.: Analyse en moyenne d'algorithmes : tri rapide et arbres de recherche. PhD Thesis, Ecole Politechnique, Palaiseau (1991)
15. Hoare, C.A.R.: Algorithm 65: Find. Communications of the ACM **4**(7), 321–322 (1961)
16. Hoare, C.A.R.: Quicksort. The Computer Journal **5**(1), 10–16 (1962)
17. Hwang, H.K., Tsai, T.H.: Quickselect and Dickman function. Combinatorics, Probability and Computing **11** (2000)
18. Kirschenhofer, P., Prodinger, H.: Comparisons in Hoare's Find Algorithm. Combinatorics, Probability and Computing **7**(01), 111–120 (1998)
19. Knuth, D.E.: The Art Of Computer Programming: Searching and Sorting, 2nd edn. Addison Wesley (1998)
20. Lent, J., Mahmoud, H.M.: Average-case analysis of multiple Quickselect: An algorithm for finding order statistics. Statistics & Probability Letters **28**(4), 299–310 (1996)
21. Mahmoud, H.M.: Sorting: A distribution theory. John Wiley & Sons, Hoboken, NJ, USA (2000)
22. Mahmoud, H.M.: Distributional analysis of swaps in Quick Select. Theoretical Computer Science **411**, 1763–1769 (2010)
23. Mahmoud, H.M., Modarres, R., Smythe, R.T.: Analysis of quickselect : an algorithm for order statistics. Informatique théorique et applications **29**(4), 255–276 (1995)
24. Martínez, C., Panario, D., Viola, A.: Adaptive sampling strategies for quickselects. ACM Transactions on Algorithms **6**(3), 1–45 (2010)
25. Martínez, C., Prodinger, H.: Moves and displacements of particular elements in

Quicksort. Theoretical Computer Science **410**(21–23), 2279–2284 (2009)

26. Martínez, C., Roura, S.: Optimal Sampling Strategies in Quicksort and Quickselect. SIAM Journal on Computing **31**(3), 683 (2001)
27. Neininger, R.: On a multivariate contraction method for random recursive structures with applications to Quicksort. Random Structures & Algorithms **19**(3-4), 498–524 (2001)
28. Neininger, R., Rüschendorf, L.: A General Limit Theorem for Recursive Algorithms and Combinatorial Structures. The Annals of Applied Probability **14**(1), 378–418 (2004)
29. Panholzer, A., Prodinger, H.: A generating functions approach for the analysis of grand averages for multiple QUICKSELECT. Random Structures & Algorithms **13**(3-4), 189–209 (1998)
30. Prodinger, H.: Multiple Quickselect—Hoare's Find algorithm for several elements. Information Processing Letters **56**(3), 123–129 (1995)
31. Rachev, S.T., Rüschendorf, L.: Probability Metrics and Recursive Algorithms. Advances in Applied Probability **27**(3), 770–799 (1995)
32. Rösler, U.: A limit theorem for "quicksort". Informatique théorique et applications **25**(1), 85–100 (1991)
33. Rösler, U.: On the analysis of stochastic divide and conquer algorithms. Algorithmica **29**(1), 238–261 (2001)
34. Rösler, U.: QUICKSELECT revisited. Journal of the Iranian Statistical Institute **3**, 271–296 (2004)
35. Rösler, U., Rüschendorf, L.: The contraction method for recursive algorithms. Algorithmica **29**(1), 3–33 (2001)
36. Sedgewick, R.: Quicksort. PhD Thesis, Stanford University (1975)
37. Sedgewick, R.: The analysis of Quicksort programs. Acta Informatica **7**(4), 327–355 (1977)
38. Sedgewick, R.: Implementing Quicksort programs. Communications of the ACM **21**(10), 847–857 (1978)
39. Sedgewick, R., Flajolet, P.: An Introduction to the Analysis of Algorithms. Addison-Wesley-Longman (1996)
40. Sedgewick, R., Wayne, K.: Algorithms, 4th edn. Addison-Wesley (2011)
41. Vallée, B., Clément, J., Fill, J.A., Flajolet, P.: The Number of Symbol Comparisons in QuickSort and QuickSelect. In: S. Albers, et al. (eds.) ICALP 2009, *LNCS*, vol. 5555, pp. 750–763. Springer (2009)
42. Wild, S.: Java 7's Dual Pivot Quicksort. Master thesis, University of Kaiserslautern (2012)
43. Wild, S., Nebel, M.E.: Average Case Analysis of Java 7's Dual Pivot Quicksort. In: L. Epstein, P. Ferragina (eds.) ESA 2012, *LNCS*, vol. 7501, pp. 825–836. Springer (2012)
44. Wild, S., Nebel, M.E., Neininger, R.: Average Case and Distributional Analysis of Java 7's Dual Pivot Quicksort. ACM Transactions on Algorithms (accepted for publication). URL `http://arxiv.org/abs/1304.0988`
45. Wild, S., Nebel, M.E., Reitzig, R., Laube, U.: Engineering Java 7's Dual Pivot Quicksort Using MaLiJAn. In: P. Sanders, N. Zeh (eds.) ALENEX 2013, pp. 55–69. SIAM (2013)