Sesquickselect: One and a half pivots for cache-efficient selection*

Conrado Martínez† Markus Nebel‡ Sebastian Wild§

October 30, 2018

Abstract

Because of unmatched improvements in CPU performance, memory transfers have become a bottleneck of program execution. As discovered in recent years, this also affects sorting in internal memory. Since partitioning around several pivots reduces overall memory transfers, we have seen renewed interest in multiway Quicksort. Here, we analyze in how far multiway partitioning helps in Quickselect.

We compute the expected number of comparisons and scanned elements (approximating memory transfers) for a generic class of (non-adaptive) multiway Quickselect and show that three or more pivots are not helpful, but two pivots are. Moreover, we consider "adaptive" variants which choose partitioning and pivot-selection methods in each recursive step from a finite set of alternatives depending on the current (relative) sought rank. We show that "Sesquickselect", a new Quickselect variant that uses either one or two pivots, makes better use of small samples w.r.t. memory transfers than other Quickselect variants.

1 Introduction

We consider the selection problem: finding the mth smallest element within an unsorted array of n distinct elements. Quickselect [15] is the earliest (published) algorithm for general selection that runs in linear time (in expectation), and it forms the basis of practical implementations and more advanced algorithms.

From a theoretical perspective, this problem might be considered solved: The randomized Floyd-Rivest algorithm [10, 20] uses $n + \min\{m, n-m\} + o(n)$ comparisons in expectation, and this is optimal up to lower order terms [8]. The Floyd-Rivest algorithm is a variant of Quickselect that uses a large random sample of $\Theta(n^{2/3}\log^{1/3} n)$ elements from which it (recursively) selects two pivots P_1 and P_2, $P_1 < P_2$, so that their ranks surround m with high probability. Partitioning the input into the elements $< P_1$, between P_1 and P_2, and $> P_2$, respectively, yields a subproblem of size $o(n)$ with high probability.

Standard libraries do not use the asymptotically optimal algorithms [36, 1], presumably because for moderate-size inputs, lower order terms and their large hidden constants are not negligible, making variants with less overhead desirable. For example, the GNU implementation of the C++ STL uses introspective median-of-3 Quickselect for `std::nth_element`.[1] Introspective sorting was suggested by Musser [29] and refined by Valois [36].

Given the similarity of Quicksort and Quickselect, it is natural and tempting to employ the same optimizations for selection that work well for sorting. Indeed, this is exactly what is done in the GNU STL; the Quickselect implementation uses the same partitioning method, the same pivot rule (median of three elements), the same protection mechanism against bad-case inputs (a recursion-depth limit of $2\lceil\lg(n+1)\rceil$ with a $\Theta(n\log n)$ worst-case method based on heapsort), and the same base case for the recursion (Insertionsort) as in the Quicksort implementation.

On second thought, some of these design decisions are quite questionable in the context of selection. First, a linear worst case can be achieved instead of the $\Theta(n\log n)$ one [6, 36]. Second, it is known that choosing the pivot *adaptively,* i.e., depending on the value of m (mimicking the asymptotically optimal Floyd-Rivest algorithm!) improves the average costs by a significant factor even for small sample sizes [27].

Finally, in light of the recent success of multi-pivot Quicksort [4, 25, 30, 37], a question programmers will face is whether and how multiway partitioning should

*This work has been partially supported by funds from the Spanish Ministery of Economy, Industry and Competitiviness (MINECO) and the European Union (FEDER) under grant GRAMM (TIN2017-86727-C2-1-R), and by funds from the Catalan Government (AGAUR) under grant 2017 SGR 786. The last author is supported by the Natural Sciences and Engineering Research Council of Canada and the Canada Research Chairs Programme.

†Department of Computer Science, Universitat Politècnica de Catalunya, Spain. Email: conrado@cs.upc.edu

‡Faculty of Technology, Technische Universität Bielefeld, Germany. Email: nebel@techfak.uni-bielefeld.de

§David R. Cheriton School of Computer Science, University of Waterloo, Canada. Email: wild@uwaterloo.ca

[1] The code can be browsed online: https://gcc.gnu.org/onlinedocs/gcc-8.1.0/libstdc++/api/a00527_source.html#l04748.

also be used for Quickselect. Using YBB-partitioning[2] – the dual-pivot method used in `Arrays.sort` of Oracle's Java runtime library [41, 40] – was shown to be of *no* advantage for Quickselect w.r.t. the number of comparisons (averaging over all possible ranks to be selected) [39]. YBB partitioning can reduce the comparison count in sorting, but the main advantage of multiway partitioning lies in saving memory transfers, and indeed, the latter *is* improved using dual-pivot Quickselect (see § 4). The purpose of this article is thus a comprehensive assessment of the potential of multiway partitioning in Quickselect.

To this end, we present an average-case analysis of both classical cost measures and memory transfers. The latter is formalized as the number of *scanned elements*: the accumulated range scanned by all index/pointer variables used in the partitioning strategy. This has been shown to be a good indicator for the number of cache misses that occur during partitioning [30, 4].

Our focus is on low-overhead algorithms suitable for library implementations, and hence on small fixed-size samples. Taking inspiration from Floyd-Rivest, we propose "Sesquickselect",[3] an adaptive Quickselect variant that uses either one or two pivots from a sample of k elements, and we give strong evidence for its optimality w.r.t. scanned elements subject to a given sample size.

Overview and Method. We first consider multiway partitioning and pivot sampling in full generality (partitioning into any constant number $s \geq 2$ of segments while choosing pivots from samples of any constant size k), under the assumption that a uniformly chosen random rank is searched. These so-called "grand averages" [26] can be computed using a distributional master theorem [37] derived from Roura's continuous master theorem [32]. We can conclude that *more than three segments are indeed not helpful in Quickselect.* This matches the intuition that a large s "should" not help since all but one segment will be discarded for good, making further subdivisions superfluous. The precise argument requires some care, though.

Unfortunately, the grand-average analysis does not extend to adaptive methods like Sesquickselect. For the second part, we hence consider selecting the α-quantile in a large array for a fixed $\alpha \in (0,1)$ (extending techniques from [27]). We give an elementary proof for the correctness of a resulting integral equation for the leading-term coefficient as a function of α (under reasonable assumptions fulfilled for our applications) that appears to be novel. The setting with two parameters makes computations appreciably more challenging. For Quickselect with YBB partitioning (without pivot sampling) and Sesquickselect with $k = 2$ we solve the integral equations analytically, and we obtain precise numerical solutions for more general cases. From these, we can derive promising candidates of cache-optimal Quickselect variants for all practical sample sizes.

Outline. We give an overview of previous work in the remainder of this section. § 2 introduces notation and preliminaries. In § 3, we state a general distributional recurrence of costs. § 4 discusses the analysis for random ranks. We then switch to fixed ranks and derive the integral equation in § 5. We solve it for Quickselect with YBB-partitioning (§ 6) and for the novel "Sesquickselect" algorithm (§ 7). Our paper concludes with a discussion of our findings (§ 8).

There is an extended online version of this paper available as arxiv preprint `1810.12322` (`https://arxiv.org/abs/1810.12322`) that contains some missing proofs and computations.

1.1 Previous Work.

The first published analysis of (classic) Quickselect by Knuth served as one of two illustrating examples in an invited address at the IFIP Congress 1971, with the goal to advertise the emerging area of analysis of algorithms [22, 23]. The expected number of comparisons in classic Quickselect is $\mathbb{E}[C_{n,m}] = 2\big((n+1)H_n - (n+3-m)H_{n+1-m} - (m+2)H_m + n + 3\big)$, where $H_n = \sum_{i=1}^{n} \frac{1}{i}$.

An asymptotic approximation for selecting the α-quantile, $\alpha \in (0,1)$ fixed, follows with $m = \alpha n$: $\mathbb{E}[C_{n,\alpha n}] = 2(h(\alpha)+1)\cdot n - 8\ln n \pm O(1)$, $n \to \infty$, where $h(x) = -x\ln x - (1-x)\ln(1-x)$; (here we set $0\ln 0 := 0$).

Quickselect has been extensively studied. The variance is quadratic and precisely known [31, 19]; large deviations from the mean are very unlikely [9, 12]. Stochastic limits laws have been established for the random costs divided by n for random ranks [26], small ranks [16] and fixed quantiles [14].

Like for Quicksort, better pivot selection methods are important in Quickselect. The widely used median-of-three version was analyzed in [2] and [18], and the generalization of using the median of any fixed size sample ("median-of-k") in [28] (expectation and variance for random ranks) and in [13] (for fixed ranks); refined limit laws for the case where k grows polynomially with n were recently derived in [35]. Choosing the order statistic of the sample depending on m/n was studied in [27] (w.r.t. expectation) and [21] (limit laws, including growing k).

If the objects to select from are strings, symbol comparisons rather than key comparisons are the measure of interest; Quickselect has linear expected cost

[2] Named after its inventors Vladimir Yaroslavskiy, Jon L. Bentley, and Joshua Bloch.

[3] After the Latin prefix *sesqui-* meaning "one and a half".

also in this model [7]. Quickselect with YBB partitioning was analyzed in [39] and Krenn [24] considered the comparison-optimal dual-pivot partitioning method of Aumüller and Dietzfelbinger [3].

2 Preliminaries

We start with some notation. O-terms are bounds on the absolute value; we write $g = f \pm O(e)$ to emphasize this. $f \sim g$ means $f/g \to 1$. Vector's are written in boldface, e.g., $\boldsymbol{x} = (x_1, x_2, x_3)$, and operations are understood componentwise: $\boldsymbol{x}+1 = (x_1+1, x_2+1, x_3+1)$. We use the notation $x^{\overline{n}}$ and $x^{\underline{n}}$ of [11] for rising resp. falling (factorial) powers. (The former is also known as Pochhammer function).

We use capital letters for random variables and $\mathbb{E}$ for expectation. $X \overset{\mathcal{D}}{=} Y$ denotes equality in distribution. $\mathcal{U}[1..n]$ is the discrete uniform distribution over $[1..n]$, $\mathcal{U}(0,1)$ the continuous uniform distribution over $(0,1)$. We next recall the beta distribution and some of its properties. It will play a pivotal role in our analysis.

2.1 The beta distribution and its relatives.

The *beta distribution* has two parameters $\alpha, \beta \in \mathbb{R}_{>0}$ and is written as $\mathrm{Beta}(\alpha, \beta)$. If $X \overset{\mathcal{D}}{=} \mathrm{Beta}(\alpha, \beta)$, we have $X \in (0,1)$ and X has the density

$$f(x) \;=\; \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}, \qquad x \in (0,1),$$

where $\mathrm{B}(\alpha,\beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$ is the beta function. The reason why the beta arise in our analysis is its connection to order statistics: If we assume the input consists of n i.i.d. (independent and identically distributed) $\mathcal{U}(0,1)$ random variables, the ℓth smallest element of a sample of k elements has a $\mathrm{Beta}(\ell, k+1-\ell)$ distribution.

For multi-pivot methods, we will encounter the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$, which is the multivariate version of the beta distribution. For $X \overset{\mathcal{D}}{=} \mathrm{Dir}(\boldsymbol{\alpha})$ with $\boldsymbol{\alpha} \in \mathbb{R}^d_{>0}$, we use the convention that $X_d = 1 - X_1 - \cdots - X_{d-1}$ and specify $\boldsymbol{X}$ as a d-dimensional vector. Then $\boldsymbol{X}$ has the density $f(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{\alpha}-1}/\mathrm{B}(\boldsymbol{\alpha})$, where $\mathrm{B}(\boldsymbol{\alpha}) = \Gamma(\alpha_1)\cdots\Gamma(\alpha_d)/\Gamma(\alpha_1+\cdots\alpha_d)$ is the d-dimensional beta function.

For subproblem sizes, we will furthermore find the Dirichlet-multinomial distribution $\mathrm{DirMult}(n, \boldsymbol{\alpha}) \overset{\mathcal{D}}{=} \mathrm{Mult}(n, \mathrm{Dir}(\boldsymbol{\alpha}))$, which is a mixed multinomial distribution with a Dirichlet-distributed parameter. For the 2d case, the distribution is called beta binomial distribution, written as $\mathrm{BetaBin}(n, \alpha, \beta)$.

Since the binomial distribution is sharply concentrated, one can use Chernoff bounds to show that $\mathrm{BetaBin}(n,\alpha,\beta)/n$ converges to $\mathrm{Beta}(\alpha,\beta)$ in a specific sense. We can obtain stronger error bounds by directly comparing the PDFs, and the argument generalizes to higher dimensions. The two-dimensional case appears in [37, Lemma 2.38]; here we extend it to general (fixed) $s \geq 2$. We following the notation used there, in particular we write $\Sigma\boldsymbol{x} = \sum_{i=1}^{s} x_i$.

Lemma 2.1 (Local Limit Law DirMult): *Let $(\boldsymbol{I}^{(n)})_{n\in\mathbb{N}_{\geq 1}}$ be a family of random variables with Dirichlet-multinomial distribution, $\boldsymbol{I}^{(n)} \overset{\mathcal{D}}{=} \mathrm{DirMult}(n, \boldsymbol{\alpha})$ where $\boldsymbol{\alpha} \in (\{1\} \cup \mathbb{R}_{\geq 2})^s$ is fixed, and let $f_D(\boldsymbol{z})$ be the density of the $\mathrm{Dir}(\boldsymbol{\alpha})$ distribution. Then we have uniformly in $\boldsymbol{z} \in (0,1)^s$ (with $\Sigma\boldsymbol{z} = 1$) that*

$$n^{s-1} \cdot \mathbb{P}\big[\boldsymbol{I}^{(n)} = \lfloor \boldsymbol{z}(n+1) \rfloor\big] \;=\; f_D(\boldsymbol{z}) \,\pm\, O(n^{-1}),$$

as $n \to \infty$. That is, $\boldsymbol{I}^{(n)}/n$ converges to $\mathrm{Dir}(\boldsymbol{\alpha})$ in distribution, and the probability weights converge uniformly to the limiting density at rate $O(n^{-1})$.

The proof is given in the extended online version.

Remark 2.2: Since f_D is a polynomial in $\boldsymbol{z}$, it is in particular bounded and Lipschitz-continuous in the closed domain $\boldsymbol{z} \in [0,1]^s$ with $\Sigma\boldsymbol{z} = 1$. Hence, the local limit law also holds for the random variables $\boldsymbol{I}^{(n)} + c$ for any constant c. We use this for subproblem sizes, which are of this form: $J_r = I_r + t_r$.

2.2 Hölder-Continuity.

A function $f : I \to \mathbb{R}$ defined on a bounded interval I is called Hölder-continuous with exponent $h \in (0,1]$ when

$$\exists C\; \forall x, y \in I \;:\; \big|f(x) - f(y)\big| \;\leq\; C|x-y|^h.$$

Hölder-continuity is a form of smoothness of functions that is stricter than (uniform) continuity, but slightly more liberal than Lipschitz-continuity (which corresponds to $h = 1$). It provides a useful requirement in some of our theorems; $f : [0,1] \to \mathbb{R}$ with $f(z) = z\ln(1/z)$ is a stereotypical function that is Hölder-continuous (for any $h \in (0,1)$), but not Lipschitz. We will also need Hölder-continuity for functions from $\mathbb{R}^n$ to $\mathbb{R}$; we extend the definition by requiring $\big|f(\boldsymbol{x}) - f(\boldsymbol{y})\big| \leq C\|\boldsymbol{x} - \boldsymbol{y}\|^h$, i.e., using the Euclidian norm.

The most useful consequence of Hölder-continuity is given by the following lemma: an error bound on the difference between an integral and the Riemann sum.

Lemma 2.3: *Let $f : [0,1]^d \to \mathbb{R}$ be Hölder-continuous (w.r.t. $\|\cdot\|_2$) with exponent h. Then*

$$\int_{\boldsymbol{x}\in[0,1]^d} f(\boldsymbol{x})\, d\boldsymbol{x} \;=\; \frac{1}{n^d} \sum_{\boldsymbol{i}\in[0..n-1]^d} f(\boldsymbol{i}/n) \;\pm\; O(n^{-h}),$$

as $n \to \infty$.

The proof is a simple computation; it is given in the extended online version.

We considered only the unit interval resp. unit hypercube as the domain of functions rather than a product of general compact intervals, but this is no restriction: Hölder-continuity (on bounded domains) is preserved by addition, subtraction, multiplication and composition (see, e.g., [34, § 4.6]). Since any linear function is Lipschitz, the above result holds for Hölder-continuous functions $f : [a_1, b_1] \times \cdots \times [a_d, b_d] \to \mathbb{R}$.

If our functions are defined on a bounded domain, Lipschitz-continuity implies Hölder-continuity and Hölder-continuity with exponent h implies Hölder-continuity with exponent $h' < h$. A real-valued function is Lipschitz if its derivative is bounded resp. if all its partial derivatives are bounded (in the multivariate case). The latter follows form the mean-value theorem (in several variables) and the Cauchy-Schwartz inequality.

2.3 The Distributional Master Theorem. To solve the recurrences in § 4, we use the "distributional master theorem" (DMT) [37, Thm. 2.76], reproduced below for convenience. It is based on Roura's continuous master theorem [32], but reformulated in terms of distributional recurrences in an attempt to give the technical conditions and occurring constants in Roura's original formulation a more intuitive, stochastic interpretation. We start with a bit of motivation for the latter.

The DMT is targeted at divide-and-conquer recurrences where the recursive parts have a *random* size like in Quickselect. Because of the random subproblem sizes, a traditional recurrence for expected costs has to sum over all possible subproblem sizes, weighted appropriately. That way, the direct correspondence between the recurrence and the algorithmic process is lost. A *distributional recurrence* avoids this. It describes the full distribution of costs, where the cost for larger problem sizes is described by a "toll term" (the partitioning costs in Quickselect) plus the contributions of recursive calls.

Such a distributional formulation requires the toll costs and subproblem sizes to be stochastically independent of the recursive costs when conditioned on the subproblem sizes. In typical applications, this is fulfilled when the studied algorithm guarantees that the subproblems on which it calls itself recursively are of the same nature as the original problem. Such a form of randomness preservation is also required for the analysis using traditional recurrences.

The DMT allows us to compute an asymptotic approximation of the expected costs directly from the distributional recurrence. Intuitively speaking, it is applicable whenever the *relative* subproblem sizes of recursive applications converge to a (non-degenerate) limit distribution as $n \to \infty$ (in a suitable sense; see Equation (2) below). The local limit law provided by Lem. 2.1 gives exactly such a limit distribution.

Theorem 2.4 (DMT [37, Thm. 2.76]):
Let $(C_n)_{n\in\mathbb{N}_0}$ be a family of random variables that satisfies the distributional recurrence

$$C_n \overset{\mathcal{D}}{=} T_n + \sum_{r=1}^{s} A_r^{(n)} \cdot C^{(r)}_{J_r^{(n)}}, \qquad (n \geq n_0), \tag{1}$$

where the families $(C_n^{(1)})_{n\in\mathbb{N}}, \dots, (C_n^{(s)})_{n\in\mathbb{N}}$ are independent copies of $(C_n)_{n\in\mathbb{N}}$, which are also independent of $(J_1^{(n)}, \dots, J_s^{(n)}) \in \{0, \dots, n-1\}^s$, $(A_1^{(n)}, \dots, A_s^{(n)}) \in \mathbb{R}^s_{\geq 0}$ and T_n. Define $Z_r^{(n)} = J_r^{(n)}/n$, $= 1, \dots, s$, and assume that they fulfill uniformly for $z \in (0, 1)$

$$n \cdot \mathbb{P}\big[Z_r^{(n)} \in (z - \tfrac{1}{n}, z]\big] \;=\; f_{Z_r^*}(z) \,\pm\, O(n^{-\delta}), \tag{2}$$

as $n \to \infty$ for a constant $\delta > 0$ and a Hölder-continuous function $f_{Z_r^*} : [0, 1] \to \mathbb{R}$. Then $f_{Z_r^*}$ is the density of a random variable Z_r^* and $Z_r^{(n)} \xrightarrow{\mathcal{D}} Z_r^*$.

Let further

$$\mathbb{E}\big[A_r^{(n)} \,\big|\, Z_r^{(n)} \in (z - \tfrac{1}{n}, z]\big] \;=\; a_r(z) \,\pm\, O(n^{-\delta}), \tag{3}$$

as $n \to \infty$ for a function $a_r : [0, 1] \to \mathbb{R}$ and require that $f_{Z_r^*}(z) \cdot a_r(z)$ is also Hölder continuous on $[0, 1]$. Moreover, assume $\mathbb{E}[T_n] \sim K n^\alpha \log^\beta(n)$, as $n \to \infty$, for constants $K \neq 0$, $\alpha \geq 0$ and $\beta > -1$. Then, with $H = 1 - \sum_{r=1}^{s} \mathbb{E}[(Z_r^*)^\alpha a_r(Z_r^*)]$, we have the following cases.

1. *If $H > 0$, then $\mathbb{E}[C_n] \sim \dfrac{\mathbb{E}[T_n]}{H}$.*
2. *If $H = 0$, then $\mathbb{E}[C_n] \sim \dfrac{\mathbb{E}[T_n] \ln n}{\tilde{H}}$ with $\tilde{H} = -(\beta + 1) \displaystyle\sum_{r=1}^{s} \mathbb{E}[(Z_r^*)^\alpha a_r(Z_r^*) \ln(Z_r^*)]$.*
3. *If $H < 0$, then $\mathbb{E}[C_n] = O(n^c)$ for the $c \in \mathbb{R}$ with $\displaystyle\sum_{r=1}^{s} \mathbb{E}[(Z_r^*)^c a_r(Z_r^*)] = 1$.* □

2.4 Adaptive Quickselect. We consider the following generic family of Quickselect variants: In each step, we partition the input into $s \geq 2$ segments, choosing the $s - 1$ pivots $P_1, \dots, P_{s-1}$ as order statistics from a random sample of the input. The pivot-selection process is described by two parameters: the sample size k and the quantiles vector $\boldsymbol{\tau} = (\tau_1, \dots, \tau_s) \in [0, 1]^s$. The ℓth pivot, $\ell = 1, \dots, s-1$, is the $(\tau_1 + \cdots + \tau_\ell)$-quantile of the sample. Alternatively, we use the vector $\boldsymbol{t} = (k + 1)\boldsymbol{\tau} - 1 \in \mathbb{N}^s_{\geq 0}$ to specify how many elements are *omitted* between the

pivots. As an example, median-of-3 (for classic $s = 2$) corresponds to $k = 3$, $\boldsymbol{\tau} = (\frac{1}{2}, \frac{1}{2})$ or $\boldsymbol{t} = (1, 1)$. Choosing the two largest elements in a sample of 5 corresponds to $k = 5$, $\boldsymbol{\tau} = (\frac{2}{3}, \frac{1}{6}, \frac{1}{6})$ or $\boldsymbol{t} = (3, 0, 0)$. Note that τ_ℓ is the *expected fraction* of elements in the ℓth segment, $\ell = 1, \dots, s$. See Fig. 1 for another example.

We assume the partitioning algorithm is an instance of the generic one-pass partitioning scheme analyzed in [37] which unifies practically relevant methods. (A similar such scheme is considered in [4]). The details of the partitioning method are mostly irrelevant for our present discussion, and we refer the reader to [37, §4.3] for details; important here is that partitioning preserves randomness for recursive calls and that the expected partitioning costs are $an \pm O(1)$ for a known constant a (depending only on the partitioning method and $\boldsymbol{t}$).

We consider *adaptive* Quickselect variants, which are formally given by specifying the partitioning method and parameters s and $\boldsymbol{t}$ as a *function* of $\alpha = \frac{m}{n}$. We assume a fixed, finite portfolio of methods to choose from and the choice consists in finding the interval containing α in a finite collection $I_1, \dots, I_d$ of intervals (with $I_1 \cup \dots \cup I_d = [0, 1]$). We treat the parameters as functions of α with the meaning $s(\alpha) = s_v$ for the $v \in [d]$ with $\alpha \in I_v$, and similarly for $\boldsymbol{t}(\alpha) = \boldsymbol{t}^{(v)}$ and $a_{\mathcal{F}}(\alpha) = a_{\mathcal{F}}^{(v)}$ (introduced in the next section).As a specific example for an adaptive method, consider Sesquickselect with a sample of size $k = 2$. There, we can choose s and $\boldsymbol{t}$ as follows:

$$s(\alpha) = \begin{cases} 1, & \text{if } \alpha < 0.266; \\ 2, & \text{if } 0.266 \le \alpha \le 0.734; \\ 1, & \text{if } \alpha > 0.734; \end{cases}$$

$$\boldsymbol{t}(\alpha) = \begin{cases} (0,1), & \text{if } \alpha < 0.266; \\ (0,0,0), & \text{if } 0.266 \le \alpha \le 0.734; \\ (1,0), & \text{if } \alpha > 0.734. \end{cases}$$

Other combinations are of course possible, but we will show in § 7 that this is indeed a good choice.

2.5 Cost Measures and Notation. We consider several measures of cost for Quickselect: $\mathcal{C}$, the number of key comparisons, $\mathcal{SE}$, the number of scanned elements (total distance traveled by pointers / scanning indices), and $\mathcal{WA}$, the number of write accesses to the array.[4] The analysis can mostly remain agnostic to this in which case we use $\mathcal{F}$ as placeholder for any of the above. We use the following naming conventions for the quantities arising in our analysis:

- $F_{n,m}$ for the random costs to select the mth smallest out of n elements.
- F_{n,M_n} is the random cost to select a (uniform) *random rank* $M_n \overset{\mathcal{D}}{=} \mathcal{U}[1..n]$ from n elements.
- $f(\alpha)$ is the leading-term coefficients of $\mathbb{E}[F_{n,m}]$ for $n \to \infty$ and $m/n \to \alpha$.
- $\bar{f}$ is the leading-term coefficient of the grand average: $\bar{f} = \lim_{n\to\infty} \mathbb{E}[F_{n,M_n}]/n$.
- $A_{\mathcal{F}}(n, m)$ are the random costs of the first partitioning round; they indirectly depend on m for adaptive methods.
- $a_{\mathcal{F}}(\alpha)$ is the leading-term coefficient of partitioning costs for $n \to \infty$ and $m/n \to \alpha$.

3 Distributional Recurrence

We will focus on the expected costs, $\mathbb{E}[F_{n,m}]$, but a concise description can be given for the full distribution. The family of random variables $(F_{n,m})_{n\in\mathbb{N},m\in[n]}$ fulfills the following distributional recurrence (notation explained below)

(4)
$$F_{n,m} \overset{\mathcal{D}}{=} A_{\mathcal{F}}(n, m) + \sum_{\ell=1}^{s} \mathbb{1}_{\{R_{\ell-1}<m<R_\ell\}} F^{(\ell)}_{J_\ell,\, m-R_{\ell-1}}$$

for $n \ge n_0$; for small $n < n_0$ costs are given by some base-case method that contributes only $O(1)$ to overall costs. In the general setting, we partition the input into $s \ge 2$ segments around $s - 1$ pivot elements. We denote by $\boldsymbol{J}^{(n)} = (J_1^{(n)}, \dots, J_s^{(n)})$ the (vector of) resulting subproblem sizes (for recursive calls). $R_1^{(n)} \le \dots \le R_{s-1}^{(n)}$ are the (random) *ranks* of the pivot elements; we set $R_0^{(n)} = 0$ and $R_s^{(n)} = n + 1$ to unify notation for the outermost segments. As in (4), we usually suppress the dependence on n in our notation for better legibility. Note that pivot ranks and subproblem sizes are related via $J_\ell = R_\ell - R_{\ell-1} - 1$ for $\ell = 1, \dots, s$ (cf. Fig. 1). For $\ell = 1, \dots, s$, $(F^{(\ell)}_{n,m})_{n\in\mathbb{N},m\in[n]}$ are independent copies of $(F_{n,m})_{n\in\mathbb{N},m\in[n]}$, which are also independent of $\boldsymbol{J}^{(n)}$ (and hence $R_1^{(n)}, \dots, R_{s-1}^{(n)}$) and $A_{\mathcal{F}}(n, m)$.

The distribution of the subproblem sizes $\boldsymbol{J}$ is discussed in detail below for the standard non-adaptive case (§ 4). We remark that Equation (4) remains valid for *adaptive* variants, i.e., where the employed pivot sampling scheme and partitioning method are chosen in each step depending on $\alpha = \frac{m}{n}$, the relative rank of the sought element. (The distributions of $\boldsymbol{J}$ and $\boldsymbol{R}$ are then functions of n *and* m.)

[4] Write accesses are more appropriate than counting swaps for methods that move several elements at a time.

4 Random ranks

In this section we consider $\mathbb{E}[F_{n,M_n}]$, where $M_n \stackrel{\mathcal{D}}{=} \mathcal{U}[1..n]$. This "grand average" [26] is a reasonable measure for a rough comparison of different selection methods, and its analysis is feasible for a large class of algorithms. Indeed, an asymptotic approximation of the costs will follow from a distributional master theorem (DMT, Thm. 2.4). We consider only non-adaptive methods in this section, i.e., the splitting probabilities do not depend on the rank of the sought element.

We will derive an asymptotic approximation for the grand average for a whole class of Quickselect variants that cover the above special cases as well as many further hypothetical versions. Our partitioning method splits the input into $s \geq 2$ segments using $s-1$ pivot elements and the pivot elements are selected as order statistics from a fixed-size random sample of the input.

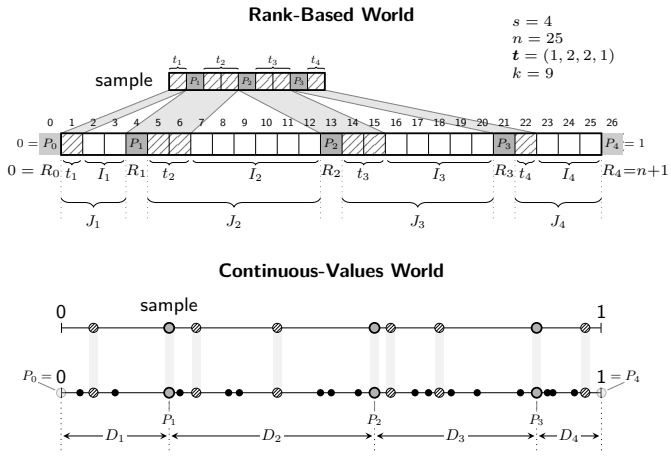

Figure 1: Illustration of the notations used in the analysis; **top:** quantities that refer to counts (sizes of segments and ranks of elements), **bottom:** quantities referring to numerical values of certain elements in the model of sorting n i.i.d. $\mathcal{U}(0,1)$ distributed numbers.

We can write the costs with $F_n := F_{n,M_n}$ as

$$F_n \stackrel{\mathcal{D}}{=} A_{\mathcal{F}}(n) + \sum_{\ell=1}^{s} \mathbb{1}_{\{R_{\ell-1}^{(n)} < M_n < R_\ell^{(n)}\}} \cdot F_{J_\ell^{(n)}}^{(\ell)}, \tag{5}$$

where $(F_j^{(\ell)})_{j\in\mathbb{N}}$ are independent copies of $(F_n)_{n\in\mathbb{N}}$ for $\ell = 1, \dots, s$, which are also independent of $\boldsymbol{J}^{(n)}$, $A_{\mathcal{F}}(n)$ and M_n.

This distributional recurrence is of the shape required for the distributional master theorem (DMT) to compute asymptotic approximations for the expected values. We next check the technical conditions for the DMT.

Distribution of Subproblem Sizes. For a single pivot chosen randomly (without pivot sampling) we have $J_\ell \stackrel{\mathcal{D}}{=} \mathcal{U}[0..n-1]$, a discrete uniform distribution. In general, we have two summands: $J_\ell = t_\ell + I_\ell$. The first one accounts for the part of the sample that belongs to the ℓth subproblem, which is a deterministic contribution dictated by the sampling scheme. I_ℓ is the number of elements that were not part of the sample and were found to belong to the ℓth segment in the partitioning step. I_ℓ is a random variable, and its distribution is $I_\ell \stackrel{\mathcal{D}}{=} \mathrm{BetaBin}(n-k, t_\ell+1, k-t_\ell)$, a so-called *beta-binomial distribution*.

The connection to the beta distribution is best seen by assuming n independent and uniformly in $(0,1)$ distributed reals as input. They are almost surely pairwise distinct and their relative ranking is a random permutation of $[n]$, so this assumption is w.l.o.g. for our analysis. Then, the ℓth subproblem contains all elements between $P_{\ell-1}$ and P_ℓ, $\ell = 1, \dots, s$. The *spacing* $D_\ell := P_\ell - P_{\ell-1}$ has a $\mathrm{Beta}(t_\ell+1, k-t_\ell)$ distribution (by definition!), and *conditional* on that *spacing* $I_\ell \stackrel{\mathcal{D}}{=} \mathrm{Bin}(n-k, D_\ell)$ has a binomial distribution: Once the pivot values $P_{\ell-1}$ and P_ℓ are fixed, any element falls between them with probability D_ℓ. The resulting mixture is the so-called beta-binomial distribution. Note that for $\boldsymbol{t} = 0$, $t_\ell + \mathrm{BetaBin}(n-k, t_\ell+1, k-t_\ell) = \mathrm{BetaBin}(n-1, 1, s-1)$ which coincides with $\mathcal{U}[0..n-1]$ for $s = 2$.

Convergence of Relative Subproblem Sizes. In light of the stochastic representation of the beta-binomial distribution, we know that conditional on D_ℓ, $Z_\ell = I_\ell^{(n)}/n$ is concentrated around D_ℓ. Bounding the errors carefully yields the required local limit law for the relative subproblem size Z_ℓ: By Lem. 2.1, we find that Equation (2) is satisfied with $\delta = 1$ and the limiting density $f_{Z_\ell^*}(z) = z^{t_\ell}(1-z)^{k-t_\ell-1}/\mathrm{B}(t_\ell+1, k-t_\ell)$, which is the density of the $\mathrm{Beta}(t_\ell+1, k-t_\ell)$ distribution. $f_{Z_\ell^*}$ is clearly Lipschitz (and hence Hölder) continuous on $[0,1]$ since its derivative is bounded in $[0,1]$, so the conditions of the DMT are fulfilled.

Conditional Convergence of Coefficients. For the second condition, Equation (3), we have to consider the distribution of $[R_{\ell-1} < M_n < R_\ell]$ conditional on the relative subproblem size $J_\ell^{(n)}/n$ for the ℓth recursive call. Since M_n is uniformly distributed, only the number of choices for M_n in the considered range $[R_{\ell-1} < M_n < R_\ell]$ is important: $R_\ell - R_{\ell-1} - 1 = J_\ell$. So we have $\mathbb{P}\big[R_{\ell-1}^{(n)} < M_n < R_\ell^{(n)} \;\big|\; J_\ell^{(n)}\big] = \frac{J_\ell^{(n)}}{n}$, so Equation (3) is fulfilled with $a_\ell(z) = z$ and $\delta = 1$. $f_{Z_\ell^*}(z) \cdot a_\ell(z)$ is Lipschitz and hence Hölder-continuous as required.

Solution for linear toll functions. We can hence apply the master theorem. The Z_ℓ^* from Thm. 2.4 correspond exactly to our spacings $D_\ell \stackrel{\mathcal{D}}{=} \mathrm{Beta}(t_\ell+1, k-t_\ell)$. We have $\mathbb{E}[A_{\mathcal{F}}(n)] \sim a_{\mathcal{F}} \cdot n$ with $a_{\mathcal{F}}$ a constant

depending on the method and $\mathcal{F}$. So $\alpha = 1$ and $\beta = 0$ and we compute

$$H = 1 - \sum_{\ell=1}^{s} \mathbb{E}\big[D_\ell^\alpha\, a_\ell(D_\ell)\big] = 1 - \sum_{\ell=1}^{s} \mathbb{E}\big[D_\ell^2\big] = 1 - \sum_{\ell=1}^{s} \frac{(t_\ell+1)^{\overline{2}}}{(k+1)^{\overline{2}}}.$$

Since $\frac{(t_\ell+1)^{\overline{2}}}{(k+1)^{\overline{2}}} < \frac{t_\ell+1}{k+1}$ and $\sum_{\ell=1}^{s} \frac{t_\ell+1}{k+1} = 1$, we have $H > 0$. So by Case 1, we find $\mathbb{E}[F_n] \sim \frac{a_{\mathcal{F}}}{H} \cdot n$.

4.1 Generic Multiway Partitioning. The partitioning methods of practical relevance – classic Hoare-Sedgewick partitioning ($s = 2$) [33], Lomuto partitioning [5] ($s = 2$), YBB partitioning ($s = 3$) [17] and "Waterloo partitioning" ($s = 4$) [25] – have been generalized to arbitrary s and analyzed in [37, Chapter 5] with respect to the expected number of comparisons, scanned elements and write accesses. By using the respective values for $a_{\mathcal{F}}$ given in [37, Thm. 7.1] in the asymptotic expression for $\mathbb{E}[F_n]$, we obtain the overall costs for selecting random ranks; (see Tab. 1 for some results).

4.2 Discussion. Although the results for generic s-way partitioning are readily available we refrain from stating them in full generality since the expressions are lengthy and many variants are not promising for selection. Intuitively, a large s can hardly be useful when we always recurse into only a single subproblem.

s	Name	$\mathbb{E}[C_n]/n$	$\mathbb{E}[SE_n]/n$	$\mathbb{E}[WA_n]/n$
2	classic	3	3	1
3	YBB	$3.1\overline{6}$	$2.\overline{6}$	$1.8\overline{3}$
4	Waterloo	$3.\overline{3}$	2.5	2
5		3.5	2.7	2.35
6		$3.\overline{6}$	2.8	$2.5\overline{3}$
7		$3.7\overline{857142}$	$3.\overline{047619}$	$2.8\overline{3}$
8		$3.\overline{857142}$	$3.2\overline{142857}$	$3.03\overline{571428}$

Table 1: The coefficient of the linear term of the expected number of comparisons, scanned elements and write accesses to the array for Quickselect with s-way partitioning without pivot sampling ($\boldsymbol{t} = 0$) when searching a random rank ("grand averages").

The optimal number of pivots Tab. 1 confirms this intuition; indeed for the classical cost measures of key comparisons (C_n) and write access (WA_n, related to key exchanges) there is no improvement whatsoever from multiway partitioning in Quickselect (as pointed out before [39]). In terms of scanned elements (SE_n), however, significant savings are observed. Here, Tab. 1 contains a surprise: the minimum for scanned elements is attained for $s = 4$! This is against the intuition since there will always be (at least) two adjacent segments whose subdivision was fruitless. How can this possibly be better than avoiding the extra work to produce a fourth segment?

The answer is that $s = 4$ is indeed suboptimal; but our comparison in Tab. 1 is not quite fair. We do not select pivots from a sample, but the multiway methods do have to sort their $s - 1$ pivots to operate correctly. We therefore allow multiway methods to enjoy pivots of better quality compared to methods with smaller s, thus giving the former an undue advantage. This unfairness is inherent in any such comparison (as previously noticed in the context of sorting [37, 38]).

Simulation by binary partitioning. A fair evaluation of the usefulness of multiway partitioning is nevertheless possible by considering the following (hypothetical) Quickselect variant: We select pivots as we would for the s-way method, but then use *several rounds of classic single-pivot partitioning* to obtain the same segments as with one round of the s-way method. For example, the four segments produced by Waterloo partitioning could also be obtained by first partitioning around the middle pivot and then the resulting left resp. right segment around the small resp. large pivot.

Note that the first round uses the median of three elements (the middle pivot), whereas the second round effectively runs with pivots selected uniformly from their subrange. By comparing the cost of both variants, we truly evaluate the quality of the partitioning methods since they use the same pivot values.

Comparing Waterloo partitioning with its simulation, we observe that both execute exactly the same set of comparisons, but w.r.t. scanned elements, the simulation scans each element twice. Waterloo partitioning scans all elements once and *only the elements in the outer two segments a second time* (an average of $1.5n$ vs. $2n$ scanned elements). This clearly exposes the superiority of multiway partitioning in terms of cache behavior and explains its advantage for sorting.

In Quickselect, we will only pursue one subproblem recursively. The simulation of Waterloo partitioning subdivides *both* segments resulting from the first split, even though one will be knowingly useless! We should therefore compare Waterloo select to a binary simulation without the useless second subdivision. The number of scanned elements then is n for the first round, plus the size of the segment on which we apply the second subdivision. The probability to subdivide the left resp. right resulting segment is the relative size of that segment (the probability that the random rank lies there). The partitioning costs are hence $n + \mathbb{E}[(J_1 + J_2)^2/n] + \mathbb{E}[(J_3 + J_4)^2/n] \sim 1.6n$ (for $\boldsymbol{t} = (0, 0, 0, 0)$), and the total cost

are given by $SE_n \sim 1.6n/H(0,0,0,0) = 2.\overline{6}$. This is still higher than $2.5n$, but much closer than $3n$. That Waterloo-select performs so much better than classic Quickselect according to Tab. 1 is thus mostly due to the use of a median-of-3 pivot for the first partitioning round, and only to a smaller extent due to its inherent advantage in terms of scanned elements.

We next consider YBB partitioning. Its simulation first partitions around the larger pivot and then subdivide the left segment around the smaller pivot. This "atomic" version would incur $3.\overline{3}n$ scanned elements, much more than the $2.\overline{6}n$ of YBB-Select and indeed more than the $3n$ for classic Quickselect. But for the lucky case that the sought pivot falls into the rightmost segment, the second subdivision is not needed and should be skipped; this lazy version incurs on average $n + \mathbb{E}[(J_1 + J_2)^2/n] \sim 1.5n$ scanned elements per partitioning step and thus still $1.5n/H(0,0,0) = 3n$ scanned elements in total.

Two pivots are optimal! But how does YBB-select compare to Waterloo-select? A simulation of one by the other does not seem sensible, but we can use the pivots for Waterloo-select (three random elements in order, $\boldsymbol{t} = (0,0,0,0)$) in YBB-select. Ignoring the largest pivot and doing the three-way split using YBB-partitioning corresponds to YBB-select with $\boldsymbol{t} = (0,0,1)$, which needs $\sim 2.5n$ scanned elements, the *same* as Waterloo-select.

This statement is also true when we let Waterloo-select choose pivots equidistantly from a sample: If $\boldsymbol{t} = (t,t,t,t)$ (for any $t \in \mathbb{N}_0$), the expected number of scanned elements is $\sim \frac{4t+5}{2t+2}n$. Selecting pivots the same way, discarding the largest and using YBB-partitioning with the two smaller pivots yields the same asymptotic result. Of course, Waterloo-select performs more comparisons and array accesses to achieve this, so we can conclude that when scanned elements dominate costs, *dual-pivot partitioning is the unique optimum choice for Quickselect!*

Summary. Splitting the input into several segments at the same time saves memory transfers. While this unconditionally helps in sorting, the game is different in selection where only one subproblem is considered recursively. The flexibility to postpone the decision which part of the input should be further partitioned (and hence the possibility to avoid the splitting of any discarded segments) outperforms the savings in scanned elements from multiway partitioning. Dual-pivot partitioning is an exception, though, since all splits were useful when we recurse into the middle segment.

4.3 Adaptive Methods.

All the methods discussed above are non-adaptive: they only take the value of m into account when they decide which subproblem to recurse into. Unfortunately, this is an inherent limitation of the single-parameter recurrence that we use. The validity of the recurrence relies on randomness preservation for m: apart from the which subproblem contains the mth smallest element, nothing has been learned about the rank of this element *within* the subproblem. Conditioned on the event that the sought rank is found in the given subproblem, its rank is still uniformly distributed within the subproblem.

For adaptive methods, this is different. Since partitioning costs and subproblem size distribution depend on the v for which $\alpha \in I_v$, we inevitably learn which interval α lies in, in addition to the index of subproblem on which we recurse. So even if m is originally uniformly distributed in $[n]$, for recursive calls it is known to lie in a smaller range. The grand average costs of adaptive Quickselect hence do not follow a simple one-parameter recurrence.

5 Asymptotic Approximation for Linear Ranks

We now consider selecting a fixed α-quantile, where $\alpha \in (0,1)$ is a parameter of the analysis. We start with the distributional equation (4) and take expectations on both sides. Since we expect the overall costs to be asymptotically linear, we divide by n:

$$\frac{\mathbb{E}[F_{n,m}]}{n} = \frac{\mathbb{E}[A_{\mathcal{F}}(n,m)]}{n} + \sum_{1 \le \underline{r} < \overline{r} \le n} \frac{\overline{r} - \underline{r} - 1}{n} \times \left(\sum_{\ell=1}^{s} \mathbb{P}\big[(R_{\ell-1}, R_\ell) = (\underline{r}, \overline{r})\big] \right) \cdot \frac{\mathbb{E}[F_{\overline{r}-\underline{r}-1, m-\underline{r}}]}{\overline{r} - \underline{r} - 1}.$$

Thm. 5.1 below confirms (under very general conditions) that passing to the limit in this recurrence yields the desired asymptotic approximation.

This has been proven for single-pivot Quickselect even in a stochastic sense [13, 14]; the used techniques can be extended to adaptive methods as outlined in [27]. We give an elementary proof that covers generic s-way Quickselect in the extended online version. Interestingly does not seem to appear in the literature. We point out that the computations are a bit lengthy, but do not need any sophisticated machinery: We simply use the *ansatz* $\mathbb{E}[F_{n,m}] \sim f(\frac{m}{n})n$ to obtain an educated guess for f and bound the error $\big|\mathbb{E}[F_{n,m}] - f(\frac{m}{n})n\big|$. The latter fulfills a similar recurrence as $\mathbb{E}[F_{n,m}]$, but with a much smaller toll function. A crude bound suffices for the claim.

Theorem 5.1 (Convergence Linear Ranks): *Consider generic (adaptive) Quickselect (as defined in § 2.4) and assume* $\mathbb{E}[A_{\mathcal{F}}(n,m)] = a_{\mathcal{F}}(\frac{m}{n})n \pm O(1)$. *Let* $f : [0,1] \to \mathbb{R}_{\geq}0$ *be a function that fulfills the following integral equation:*

(6)

$$\begin{aligned} f(\alpha) \;&=\; a_{\mathcal{F}}(\alpha) \\ &+\; \frac{1}{\mathrm{B}\big((t_1, \underset{\rightarrow}{t_1})+1\big)} \int_{u=\alpha}^{1} u^{t_1+1}(1-u)^{\overset{t_1}{\rightarrow}} f\Big(\frac{\alpha}{u}\Big)\, du \\ &+\; \frac{1}{\mathrm{B}\big((t_s, \underset{\leftarrow}{t_s})+1\big)} \int_{v=0}^{\alpha} v^{\overset{t_s}{\leftarrow}}(1-v)^{t_s+1} f\Big(\frac{\alpha-v}{1-v}\Big)\, dv \\ &+\; \sum_{\ell=2}^{s-1} \frac{1}{\mathrm{B}\big((\underset{\leftarrow}{t_\ell}, t_\ell, \underset{\rightarrow}{t_\ell})+1\big)} \times \\ &\int_{u=0}^{\alpha} \int_{v=\alpha}^{1} u^{\overset{t_\ell}{\leftarrow}}(v-u)^{t_\ell+1}(1-v)^{\overset{t_\ell}{\rightarrow}} f\Big(\frac{\alpha-u}{v-u}\Big)\, dv\, du, \end{aligned}$$

where we abbreviate $\underset{\leftarrow}{t_\ell} = \sum_{r=1}^{\ell-1}(t_r+1)-1$ *and* $\underset{\rightarrow}{t_\ell} = \sum_{r=\ell+1}^{s}(t_r+1)-1$. *For adaptive methods,* s *and* $\boldsymbol{t}$ *are functions of* α, *which is suppressed for legibility. Then* (6) *is required piecewise for* $\alpha \in I_v$, $v \in [d]$.

Assume that f *is "(piecewise) smooth", i.e.,* f *(restricted to* I_v*) is Hölder-continuous with exponent* $h \in (0,1]$ *(for all* $v \in [d]$*). Then the limit* $\lim_{n\to\infty;\frac{m}{n}\to\alpha} \mathbb{E}[F_{n,m}]/n$ *exists for* $\alpha \in (0,1) \setminus \mathcal{A}$, *where* $\mathcal{A}$ *is the set of boundaries of* $I_1, \ldots, I_d$.

Moreover, with $m = \lceil \alpha n \rceil$ *holds*

$$\mathbb{E}[F_{n,m}] \;=\; f(\tfrac{m}{n})n \;\pm\; O(n^{1-2h/3}), \quad (n \to \infty).$$

Our continuity requirements for f may appear restrictive, but they are fulfilled in all examples we studied. They might indeed follow from (6) in general, but we do not attempt to prove this conjecture.

How to obtain f. Equation (6) determines f only implicitly. Our route to an explicit expression consists of the following steps. 1) Use substitutions to obtain integrals that only involve $f(x)$ instead of the shifted and scaled arguments. 2) Take successive derivatives on both sides until all integrals vanish. This will result in a higher-order differential equation for f that we aim to solve. 3) Compute f by determining constants of integration from boundary conditions and known results (e.g., symmetry and results for $\alpha \to 0$ and random ranks).

Separable equations for adaptive sampling. Since taking derivatives does not change the argument of f, the differential equation will only relate different derivatives of f evaluated *at the same point* x. This is vital for adaptive sampling since it means that we can solve the differential equation for each I_v separately. Only step 3) involves the interactions of the regimes.

We remark that we can obtain the leading-term coefficient of the grand average by integrating: $\bar{f} = \int_0^1 f(\alpha)\, d\alpha$. This also works for adaptive methods.

The discussion in § 4 justifies a restriction to $s \leq 3$ segments, but an explicit solution for the differential equation seems out of reach for the general case. We therefore focus on the simplest special cases first.

6 YBB-Select with Linear Ranks

As a warm-up, and part of our main result on Sesquickselect, we consider YBB-Select (YQS) without sampling (as studied in [39]). We start with (6) and substitute $x \mapsto \alpha/u$, $x \mapsto \frac{\alpha-v}{1-v}$ and $x \mapsto \frac{\alpha-u}{v-u}$ in the first, second and third integral, respectively. Simplifying the integrals is fairly standard; we show details for the most interesting one:

$$\begin{aligned} &\int_{v=0}^{\alpha} \int_{v=\alpha}^{1} (v-u) f\Big(\frac{\alpha-u}{v-u}\Big)\, dv\, du \\ &=\; \int_{v=0}^{\alpha} (v-\alpha)^2 \int_{x=0}^{\alpha/v} \frac{f(x)}{(1-x)^3}\, dx\, dv \\ &=\; \int_{x=0}^{1} \frac{f(x)}{(1-x)^3} \int_{v=\alpha}^{\min\{1,\frac{\alpha}{x}\}} (v-\alpha)^2\, dv\, dx \\ &=\; \frac{(1-\alpha)^3}{3} \int_{x=0}^{\alpha} \frac{f(x)}{(1-x)^3}\, dx \;+\; \frac{\alpha^3}{3} \int_{x=\alpha}^{1} \frac{f(x)}{x^3}\, dx. \end{aligned}$$

Inserting yields the integral equation for $\boldsymbol{t} = (0,0,0)$,

(7)

$$\begin{aligned} f(\alpha) \;=\; a_{\mathcal{F}}(\alpha) + \\ 2\Bigg(&\alpha^2 \int_{\alpha}^{1} \frac{f(x)}{x^3}\, dx \;-\; \alpha^3 \int_{\alpha}^{1} \frac{f(x)}{x^4}\, dx \\ &+ (1-\alpha)^2 \int_0^{\alpha} \frac{f(x)\, dx}{(1-x)^3} - (1-\alpha)^3 \int_0^{\alpha} \frac{f(x)\, dx}{(1-x)^4} \\ &+ \frac{(1-\alpha)^3}{3} \int_0^{\alpha} \frac{f(x)}{(1-x)^3}\, dx \;+\; \frac{\alpha^3}{3} \int_{\alpha}^{1} \frac{f(x)}{x^3}\, dx \Bigg), \end{aligned}$$

and taking derivatives four times yields

$$\frac{d^4 f}{d\alpha^4} \;=\; \frac{d^4 a_{\mathcal{F}}}{d\alpha^4} \;+\; 2 \cdot \frac{1-3\alpha(1-\alpha)}{\alpha^2(1-\alpha)^2} \cdot \frac{d^2 f}{d\alpha^2}. \tag{8}$$

For comparisons $a_{\mathcal{C}}(\alpha) = 19/12$ and for scanned elements $a_{\mathcal{SE}}(\alpha) = 4/3$. More generally, if $a_{\mathcal{F}}(\alpha) = a$ for some

constant a we can solve (8) to get

$$\begin{aligned} f(\alpha) \;=\; & C_1 + C_2\cdot\alpha \\ & + C_3\cdot(1-(1-\alpha)\ln(1-\alpha)-\alpha\ln(\alpha)) \\ & + C_4\Big(\tfrac{3}{10}\alpha^5 - \tfrac{3}{4}\alpha^4 + \tfrac{1}{6}\alpha^3 + \tfrac{1}{2}\alpha^2 \\ & \qquad - (1-\alpha)\ln(1-\alpha) + 1 - \alpha\Big) \end{aligned}$$

for some constants C_i, $i = 1,\dots,4$, to be determined. If $a_{\mathcal{F}}(\alpha)$ is symmetric, that is, $a_{\mathcal{F}}(\alpha) = a_{\mathcal{F}}(1-\alpha)$ for any $\alpha \in [0,1]$ then $f(\alpha)$ is also symmetric, and this entails $C_2 = C_4 = 0$. Therefore $f(\alpha) = C_1 + C_3\cdot(1+h(\alpha))$ where $h(\alpha) = -(1-\alpha)\ln(1-\alpha) - \alpha\ln(\alpha)$. We have $a_{\mathcal{F}}(\alpha) = a$ for some constant a, and from § 4, we then know $\overline{f} = 2a$. Moreover, we can also determine $\mathbb{E}[F_{n,1}]$ with the DMT (see also [39]) and find $f(0) = 3a_{\mathcal{F}}(0)/2$ (this equality holds in terms of right limits when $\alpha \to 0^+$). These two equations determine C_1 and C_3 and we obtain

$$f^{[\mathrm{YQS}]}(\alpha) \;=\; a^{[\mathrm{YQS}]}\left(\frac{3}{2} + h(\alpha)\right). \tag{9}$$

Recall that for standard quickselect $f^{[\mathrm{CQS}]}(\alpha) = a^{[\mathrm{CQS}]}(2 + 2h(x))$ (§ 1.1). We stress here that the values for a are different for CQS and YQS.

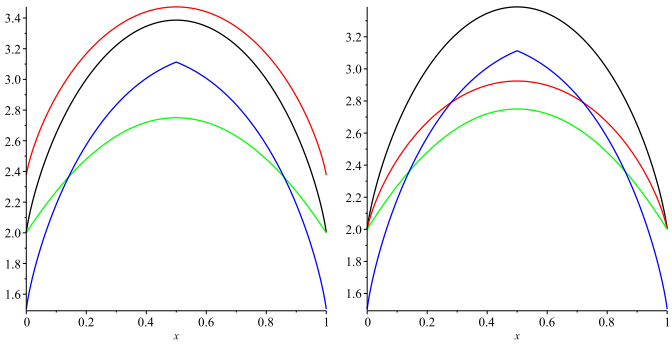

Figure 2: Key comparions, $c(\alpha)$, (**left**) and scanned elements, $se(\alpha)$, (**right**) for standard Quickselect (black), YBB-select (red), median-of-three Quickselect (green) and proportion-from-2 (blue).

Discussion. Classic Quickselect (CQS) uses fewer comparisons than YBB-Select (YQS) not only in the grand average, but for *any* fixed relative rank, (see Fig. 2 left). Similarly for write accesses, which we omit due to space constraints. For scanned elements, however, YQS scans *less* elements on average than CQS for *any* relative rank α, (Fig. 2 right)! The difference $se^{[\mathrm{CQS}]}(\alpha) - se^{[\mathrm{YQS}]}(\alpha) = \frac{2}{3}h(\alpha)$ is positive for all $\alpha \in (0,1)$ and reaches a maximum of about 13.6% more at $\alpha = 1/2$ (approx. 3.386 vs. 2.924). As we know from § 4 (Tab. 1), $\overline{se}^{[\mathrm{CQS}]} = 3$ and $\overline{se}^{[\mathrm{YQS}]} = 2.\overline{6}$, i.e., on average for random ranks, YQS scans 11.1% less elements than CQS.

Fig. 2 shows two further Quickselect variants. Median-of-three Quickselect (M3) beats YQS on all ranks, but the comparison is not quite fair because of the larger sample. *Proportion-from-2* (PROP2), however, uses the same sample size as YQS: It selects the smaller resp. larger of two sampled elements depending on whether $\alpha \le 1/2$ or $\alpha > 1/2$ holds.[5] This adaptive variant was considered in [27]; it is optimal w.r.t. comparisons for sample size 2 and beats YQS w.r.t. scanned elements for extremal α (roughly when $\alpha \le 0.281$ or $\alpha \ge 0.719$) and grand average ($\overline{se}^{[\mathrm{PROP2}]} \approx 2.598$). The dual-pivot equivalent of proportion-from-2 that we study next will improve on this significantly.

7 Sesquickselect

Like PROP2, Sesquickselect (SQS) uses two sample elements. If $\alpha < \nu$ for a parameter $\nu \in [0, \frac{1}{2}]$, we use the smaller element in the sample to partition the array, if $\alpha > 1-\nu$ use the larger element, and if $\alpha \in [\nu, 1-\nu]$ use *both* elements as pivots in YBB partitioning. We now analyze Sesquickselect on linear ranks following the same steps as in § 6.

Provided $f(\alpha) = \lim_{n\to\infty, m/n\to\alpha} \mathbb{E}[F_{n,m}]/n$ exists (recall § 5), it will be a piecewise-defined function: $f(\alpha) = f_1(\alpha)$ for $\alpha \in I_1 := [0,\nu)$, $f(\alpha) = f_2(\alpha)$ for $\alpha \in I_2 := [\nu, 1-\nu]$, and $f(\alpha) = f_3(\alpha)$ for $\alpha \in I_3 := (1-\nu, 1]$. Moreover, since $a_{\mathcal{F}}(\alpha)$ is symmetric (i.e., $a_{\mathcal{F}}(\alpha) = a_{\mathcal{F}}(1-\alpha)$) so is $f(\alpha)$, which implies $f_3(\alpha) = f_1(1-\alpha)$ and $f_2(\alpha) = f_2(1-\alpha)$. Only two "pieces", say f_1 and f_2, thus have to be determined. For $f_2(\alpha)$, we find that it satisfies the very same differential equation, (8), as $f^{[\mathrm{YQS}]}$. This is not surprising; f_2 is the YQS branch of SQS and the differential equation only uses local properties of f (as pointed out in § 5). And inside I_2, $f = f_2$.

Likewise, $f_1(\alpha)$ satisfies the same differential equation as the function f_1 in PROP2 (see [27]):

$$\frac{d^4 f_1}{d\alpha^4} \;=\; \frac{d^4 a_{\mathcal{F}}}{d\alpha^4} \;+\; \frac{2}{\alpha^2}\cdot\frac{d^2 f_1}{d\alpha^2} \;+\; \frac{2}{(1-\alpha)}\cdot\frac{d^3 f_1}{d\alpha^3}.$$

The derivatives of $a_{\mathcal{F}}(\alpha)$ vanish (inside any I_v), and we can reduce the order of both differential equations using $\phi_1 = f_1''$ and $\phi_2 = f_2''$. This yields

$$\begin{aligned} f_1(x) \;=\; & C_1\Big(\tfrac{1}{6}x^3 + \tfrac{1}{2}x^2 - x - (1-x)\ln(1-x)\Big) \\ & + C_2 h(x) \;+\; C_3 x \;+\; C_6, \end{aligned}$$

[5] The idea generalizes to proportion-from-k (PROPk) for any sample size k, where further cutoffs are introduced. In *proportion-from-3*, for example, if $\alpha < \nu$ for a parameter $\nu \in [0, \frac{1}{2}]$, the smallest element of three elements is used as the pivot; if $\alpha > 1-\nu$, the largest element is used, and the median of the sample is used whenever $\alpha \in [\nu, 1-\nu]$.

		Comparisons		Scanned elements	
α	PROP2	YQS	ν^*-SQS	YQS	ν^*-SQS
0	1.5	2.375	1.5	2	1.5
1/2	3.113^+	3.472^+	3.252^+	2.924^+	2.843^+
avg	2.598^+	$3.1\overline{6}$	2.733^+	$2.\overline{6}$	2.500^+

Table 2: Some special values of $c(\alpha)$ ($\overline{c}$) and $se(\alpha)$ ($\overline{se}$) for several variants: YQS ($\nu = 0$), PROP2 ($\nu = 1/2$) and ν^*-SQS. (Recall that $c^{[\text{PROP2}]} = se^{[\text{PROP2}]}$.)

$$f_2(\alpha) \;=\; C_4 \;+\; C_5 h(x),$$

for constants $C_1, \ldots, C_6$ that depend on the cost measure and threshold ν. We will write $f(\alpha) = f_\nu(\alpha)$ resp. ν-SQS to stress this latter dependence. The symmetry of f_2 was already taken into account. Since $f_1(0) = \frac{3}{2} a_{\mathcal{F}}(0)$ we can eliminate $C_6 = \frac{3}{2} a_{\mathcal{F}}(0)$. To determine the remaining constants, we insert the general expression for f_1 and f_2 into the integral equation and equate. The process is laborious, but doable with computer algebra; we report explicit expressions for $C_1, \ldots, C_5$ for comparisons and scanned elements in the extended online version.

Discussion. We will focus on scanned elements. To understand how ν-SQS behaves for different ν, we consider $g_1(\nu) = \lim_{\alpha \to \nu^-} f_{1,\nu}(\alpha)$ and $g_2(\nu) = \lim_{\alpha \to \nu^+} f_{2,\nu}(\alpha)$, the values of the two branches of f_ν at ν. We have $g_1(0) = \lim_{\nu \to 0^+} g_1(\nu) = 1.\overline{6}$ and $g_2(0) = \lim_{\nu \to 0^+} g_2(\nu) = 2$, but $g_1(\frac{1}{2}) \approx 3.11$ and $g_2(\frac{1}{2}) \approx 2.91$ Since g_1 and g_2 are continuous and strictly increasing for $\nu \in (0, \frac{1}{2})$, they cross at a unique point $\nu = \nu^* \in (0, \frac{1}{2})$. This point is indeed the right choice:

Theorem 7.1 (Optimal Sesquickselect): *There exists an optimal value of $\nu^* \approx 0.265\,717$ such that $se_{1,\nu^*}(\nu^*) = se_{2,\nu^*}(\nu^*)$. ν^*-SQS scans fewer elements than other ν-SQS, i.e., $se_{\nu^*}(\alpha) \le se_\nu(\alpha)$, for all $\nu \in [0, \frac{1}{2}]$ and all $\alpha \in [0, 1]$; in particular, $se_{\nu^*}(\alpha) \le se^{[\text{YQS}]}(\alpha)$ and $se_{\nu^*}(\alpha) \le se^{[\text{PROP2}]}(\alpha)$ for any $\alpha \in [0, 1]$.*

The proof is similar to [27, Thm 5.1], we give the details in the extended online version.

Since ν^* is optimum across all relative ranks, ν^* minimizes $se_\nu(1/2)$ and $\overline{se}_\nu$: we have $se_{\nu^*}(1/2) \approx 2.843$ and $\overline{se}_{\nu^*} \approx 2.5004$, (see also Tab. 2).

7.1 Sesquickselect with larger samples.

The idea of Sesquickselect naturally extends to more than two sample elements: SQSk adaptively chooses one or two elements as pivot(s) from a sample of k. (SQS is simply SQS2 in this notation). For larger k, there are many options to do this and guidance is needed to select good variants. With pivot sampling, scanned-element costs for

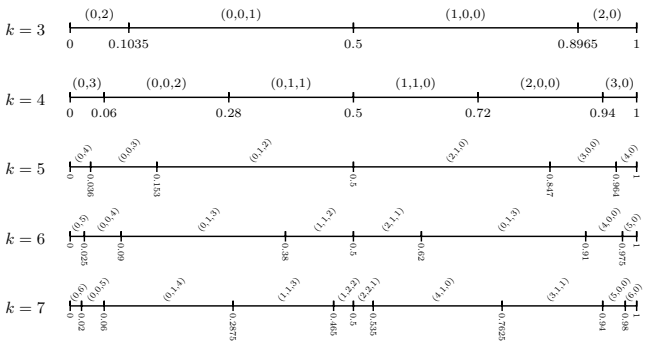

Figure 3: Our conjectured (approx.) optimal SQSk variants for small k. If α falls in the interval (delimited by the values given below the line), the vector above the line is used for $\boldsymbol{t}$. When $\boldsymbol{t}$ has two entries, classic partitioning is used; where three entries are given, we use YBB-partitioning for $\alpha \le \frac{1}{2}$ and BBY-partitioning for $\alpha > \frac{1}{2}$.

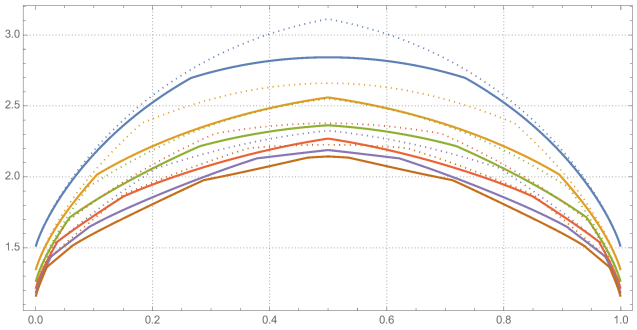

Figure 4: $se(\alpha)$ for SQSk (thick) and (optimally biased) PROPk (dotted) for $k = 2$ (blue), $k = 3$ (yellow), $k = 4$ (green), $k = 5$ (red), $k = 6$ (purple), and $k = 7$ (brown).

YBB partitioning are $a_{\mathcal{SE}} = 1+(t_1+1)/(k+1)$; when $t_3 < t_1$, we can improve this to $1+(t_3+1)/(k+1)$ using "BBY partitioning", a symmetric variant of YBB partitioning. We hence assume here that $a_{\mathcal{SE}} = 1 + \frac{\min\{t_1,t_3\}+1}{k+1}$.

We could give a complete analysis for SQS2, but for larger k the higher-order differential equations seem to withstand analytic solutions. We can, however, numerically solve the integral equation (6) to get insight into which adaptive variants are promising algorithms. The code is available online: `https://github.com/sebawild/quickselect-integral-equation`. Although numeric convergence was very good in all our explorations, we do not prove the validity of the numeric procedures. The smoothness requirement for Thm. 5.1 seemed likewise to be fulfilled in all cases, but it remains a working hypothesis for this section.

We conjecture that the variants given in Fig. 3 are the (approximately) optimal choices for the given sample size; they have been found by extensive albeit non-exhaustive

search. Fig. 4 compares the scanned-elements cost for Sesquickselect and biased proportion-from-k for small k.

Discussion. We observe that for $k \leq 7$ we are still far away from the optimal leading term of $1+\min\{\alpha, 1-\alpha\}$. For example, $\overline{se}^{[\mathrm{SQS7}]} \approx 1.841$, almost 50% more than the optimal 1.25. This is quite different in sorting, where median-of-7 Quicksort is less than 10% above optimum in the leading term.

A possible explanation is that the variance of the pivot ranks is too big. Consider, e.g., $k = 7$ and $\alpha \in [0.465, 0.5]$. The probability to recurse on the middle segment for, e.g., $\boldsymbol{t} = (2, 0, 3)$ is only roughly 1%. We must therefore use rather balanced sampling vectors (here $\boldsymbol{t} = (1, 2, 2)$) and thus lose the ability to reduce the problem size to much less than $\frac{1}{3}n$ in one step.

8 Conclusion

Despite the asymptotic optimality of the Floyd-Rivest algorithm, practical implementations use Quickselect variants with a fixed-size sample. Since they hence look very similar to sorting methods based on Quicksort, it is tempting to copy optimizations that fair well in sorting blindly to the selection routines. However, our results show that the similarities are misleading. While multiway partitioning is vital in Quicksort for saving memory transfers – a cost measure of increasing relevance – more than two pivots are *not* helpful in Quickselect.

Moreover, Quickselect offers a large potential for optimization that has no counterpart in sorting whatsoever: adapting the strategy to the (relative) sought rank $\alpha = \frac{m}{n}$. The biased proportion-from-k variants of single-pivot Quickselect proposed in [27] minimize the number of *comparisons*; in terms of scanned elements, however, Sesquickselect – a novel combination of single-pivot and dual-pivot Quickselect – outperforms proportion-from-k significantly.

In the limit for large sample sizes $k \to \infty$, Sesquickselect converges to the Floyd-Rivest algorithm. This limit is "degenerate" in that we always choose *two* pivots (Sesquickselect only uses a single pivot for extreme α), and that the middle segment has size $o(n)$ (in expectation). In that case, also the savings of dual-pivot partitioning over its simulation by two binary partitioning rounds are negligible.

For practical sample sizes, one cannot rely on this connection to design a good selection method, though – unlike for single-pivot variants, mimicking Floyd-Rivest too closely can result in performance much worse than non-adaptive Quickselect. We need analyses that explicitly take the effect of fixed-size samples into account; such are initiated in this article.

8.1 Future Work.

We had to leave many interesting questions about Sesquickselect open; some are not even known for single-pivot Quickselect.

- How fast do the costs converge to the optimum as k grows? Only an upper bound for median-of-k seems known [13, Thm. 4].
- In Fig. 3, the number of intervals seems to grow linearly with k; can we avoid the use of many different versions in adaptive methods while still achieving (close to) optimal costs?
- Do the theoretical improvements translate to faster running time? Preliminary explorations were promising although the relative improvements are small.
- What is the order of the second term / the speed of convergence in the asymptotic expansion of the costs for fixed quantiles?
- How does adaptive sampling affect the variance, higher moments or full distribution of costs? Some results for PROPk are shown in [21].
- Does adaptive sampling also improve the number of symbol comparisons?

References

[1] Andrei Alexandrescu. Fast deterministic selection. In Costas S. Iliopoulos, Solon P. Pissis, Simon J. Puglisi, and Rajeev Raman, editors, *International Symposium on Experimental Algorithms (SEA 2017)*, volume 75 of *LIPIcs*, pages 24:1–24:19, 2017. doi:10.4230/LIPIcs.SEA.2017.24.

[2] D.H. Anderson and R. Brown. Combinatorial aspects of C.A.R. Hoare's FIND algorithm. *Australasian Journal of Combinatorics*, 5:109–119, 1992.

[3] Martin Aumüller and Martin Dietzfelbinger. Optimal partitioning for dual-pivot Quicksort. *ACM Transactions on Algorithms*, 12(2):18:1–18:36, 2015. doi:10.1145/2743020.

[4] Martin Aumüller, Martin Dietzfelbinger, and Pascal Klaue. How good is multi-pivot quicksort? *ACM Transactions on Algorithms*, 13(1):8:1–8:47, 2016. doi:10.1145/2963102.

[5] Jon Bentley. Programming pearls: how to sort. *Communications of the ACM*, 27(4):287–291, 1984.

[6] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973. doi:10.1016/S0022-0000(73)80033-9.

[7] Julien Clément, James Allen Fill, Thu Hien Nguyen Thi, and Brigitte Vallée. Towards a realistic analysis of the QuickSelect algorithm. *Theory of Computing Systems*, 58(4):528–578, 2015. doi:10.1007/s00224-015-9633-5.

[8] Walter Cunto and J. Ian Munro. Average case selection. *Journal of the ACM*, 36(2):270–279, 1989. doi:10.1145/62044.62047.

[9] Luc Devroye. Exponential bounds for the running time of a selection algorithm. *Journal of Computer and System Sciences*, 29(1):1–7, 1984. doi:10.1016/0022-0000(84)90009-6.

[10] Robert W. Floyd and Ronald L. Rivest. Expected time bounds for selection. *Communications of the ACM*, 18(3):165–172, 1975. doi:10.1145/360680.360691.

[11] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation For Computer Science*. Addison-Wesley, 1994.

[12] Rudolf Grübel. Hoare's selection algorithm: A Markov chain approach. *Journal of Applied Probability*, 35(01):36–45, 1998. doi:10.1239/jap/1032192549.

[13] Rudolf Grübel. On the median-of-k version of Hoare's selection algorithm. *RAIRO – Theoretical Informatics and Applications*, 33(2):177–192, 1999. doi:10.1051/ita:1999112.

[14] Rudolf Grübel and Uwe Rösler. Asymptotic distribution theory for Hoare's selection algorithm. *Advances in Applied Probability*, 28(01):252–269, 1996. URL: https://doi.org/10.2307/1427920, doi:10.2307/1427920.

[15] C. A. R. Hoare. Algorithm 65: Find. *Communications of the ACM*, 4(7):321–322, 1961.

[16] Hsien-Kuei Hwang and Tsung-Hsi Tsai. Quickselect and the Dickman function. *Combinatorics, Probability and Computing*, 11(04), 2002. doi:10.1017/s0963548302005138.

[17] Java Core Library Development Mailing List. Replacement of quicksort in java.util.arrays with new dual-pivot quicksort, 2009. URL: https://www.mail-archive.com/core-libs-dev@openjdk.java.net/msg02608.html.

[18] P. Kirschenhofer, H. Prodinger, and C. Martínez. Analysis of Hoare's FIND algorithm with median-of-three partition. *Random Structures and Algorithms*, 10(1-2):143–156, 1997. doi:10.1002/(sici)1098-2418(199701/03)10:1/2<143::aid-rsa7>3.0.co;2-v.

[19] Peter Kirschenhofer and Helmut Prodinger. Comparisons in Hoare's Find algorithm. *Combinatorics, Probability and Computing*, 7(01):111–120, 1998.

[20] Krzysztof C. Kiwiel. On Floyd and Rivest's SELECT algorithm. *Theoretical Computer Science*, 347(1):214–238, 2005. doi:10.1016/j.tcs.2005.06.032.

[21] Diether Knof and Uwe Roesler. The analysis of Find and versions of it. *Discrete Mathematics & Theoretical Computer Science*, 14, 2012. URL: https://dmtcs.episciences.org/581.

[22] Donald E. Knuth. Mathematical analysis of algorithms. In *IFIP Congress (1)*, pages 19–27, 1971.

[23] Donald E. Knuth. *Selected Papers on Analysis of Algorithms*, volume 102 of *CSLI Lecture Notes*. Center for the Study of Language and Information Publications, 2000.

[24] Daniel Krenn. An extended note on the comparison-optimal dual-pivot quickselect. In *2017 Proceedings of the Fourteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974775.11.

[25] Shrinu Kushagra, Alejandro López-Ortiz, Aurick Qiao, and J. Ian Munro. Multi-pivot Quicksort: Theory and experiments. In *Meeting on Algorithm Engineering and Experiments (ALENEX)*, pages 47–60. SIAM, 2014. doi:10.1137/1.9781611973198.6.

[26] Hosam M. Mahmoud, Reza Modarres, and Robert T. Smythe. Analysis of quickselect : an algorithm for order statistics. *RAIRO – Theoretical Informatics and Applications*, 29(4):255–276, 1995.

[27] Conrado Martínez, Daniel Panario, and Alfredo Viola. Adaptive sampling strategies for quickselect. *ACM Transactions on Algorithms*, 6(3):1–45, 2010. doi:10.1145/1798596.1798606.

[28] Conrado Martínez and Salvador Roura. Optimal sampling strategies in Quicksort and Quickselect. *SIAM Journal on Computing*, 31(3):683–705, 2001. doi:10.1137/S0097539700382108.

[29] David R. Musser. Introspective Sorting and Selection Algorithms. *Software: Practice and Experience*, 27(8):983–993, 1997.

[30] Markus E. Nebel, Sebastian Wild, and Conrado Martínez. Analysis of pivot sampling in dual-pivot Quicksort. *Algorithmica*, 75(4):632–683, 2016. doi:10.1007/s00453-015-0041-7.

[31] Volkert Paulsen. The moments of FIND. *Journal of Applied Probability*, 34(04):1079–1082, 1997. doi:10.2307/3215021.

[32] Salvador Roura. Improved master theorems for divide-and-conquer recurrences. *Journal of the ACM*, 48(2):170–205, 2001.

[33] Robert Sedgewick. Implementing Quicksort programs. *Communications of the ACM*, 21(10):847–857, 1978.

[34] Houshang H. Sohrab. *Basic Real Analysis*. Springer Birkhäuser, 2nd edition, 2014.

[35] Henning Sulzbach, Ralph Neininger, and Michael Drmota. A Gaussian limit process for optimal FIND algorithms. *Electronic Journal of Probability*, 19(0), 2014. doi:10.1214/ejp.v19-2933.

[36] John D. Valois. Introspective sorting and selection revisited. *Software: Practice and Experience*, 30(6):617–638, 2000. doi:10.1002/(sici)1097-024x(200005)30:6<617::aid-spe311>3.0.co;2-a.

[37] Sebastian Wild. *Dual-Pivot Quicksort and Beyond: Analysis of Multiway Partitioning and Its Practical Potential*. Doktorarbeit (Ph. D. thesis), Technische Universität Kaiserslautern, 2016. ISBN 978-3-00-054669-3. URL: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:hbz:386-kluedo-44682.

[38] Sebastian Wild. Dual-pivot and beyond: The potential of multiway partitioning in quicksort. *it – Information Technology*, 60(3):173–177, 2018. doi:10.1515/itit-2018-0012.

[39] Sebastian Wild, Markus E. Nebel, and Hosam Mahmoud. Analysis of Quickselect under Yaroslavskiy's dual-pivoting algorithm. *Algorithmica*, 74(1):485–506, 2016. doi:10.1007/s00453-014-9953-x.

[40] Sebastian Wild, Markus E. Nebel, and Ralph Neininger. Average case and distributional analysis of dual pivot Quicksort. *ACM Transactions on Algorithms*, 11(3):22:1–22:42, 2015. doi:10.1145/2629340.

[41] Vladimir Yaroslavskiy. Dual-Pivot Quicksort. 2009. URL: http://iaroslavski.narod.ru/quicksort/DualPivotQuicksort.pdf.