J. W. Goethe-Universität Frankfurt am Main
Fachbereich 12
Institut für Mathematik

Probabilistic Analysis of Dual-Pivot Quicksort "Count"

Masterarbeit

von Jasmin Straub

Matrikelnummer: 4954615

E-Mail: jstraub@math.uni-frankfurt.de

Betreuer:	Prof. Dr. Ralph Neininger
Zweitgutachter:	Prof. Dr. Götz Kersting
eingereicht am:	26. Juli 2017

Erklärung zur Masterarbeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig abgefasst und keine anderen Hilfsmittel als die angegebenen benutzt habe.

Ich erkläre ferner, dass diejenigen Stellen der Arbeit, die anderen Werken wörtlich oder dem Sinne nach entnommen sind, in jedem einzelnen Fall unter Angabe der Quellen kenntlich gemacht sind.

(Ort, Datum)	(Unterschrift)

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Sorting is present in our everyday lives and at least since the development of computers, it has become an important field of research. Thus, over the last decades, lots of different sorting algorithms have been developed. One of the most efficient and most widely used algorithms is Quicksort. Published by Tony Hoare in 1961, Quicksort is a divide-and-conquer algorithm which puts an input sequence[*] in increasing order and works as follows: First, we select an arbitrary element p of the list—the so-called pivot. The main idea of Quicksort is to partition the remaining elements into two groups: the elements smaller than p and the elements larger than p. After this partitioning step, the pivot p is at its final position and the partitioning strategy is then recursively repeated on both parts. A possible implementation of classic Quicksort can be found in the appendix (Algorithm 2). If we consider the input sequence to be a random permutation of its elements, classic Quicksort needs on average $2n\log(n)+\mathrm{O}(n)$ comparisons and $\frac{1}{3}n\log(n)+\mathrm{O}(n)$ swaps, see e.g. Sedgewick [19, p. 334].

In order to make the sorting procedure more efficient, numerous variants of classic Quicksort have been developed over the last decades. One modification is to use $k \geq 2$ pivots instead of one single pivot and to partition the remaining elements into $k+1$ groups. The case $k = 2$ is called dual-pivot Quicksort and uses two pivots p and q (where we assume p to be the smaller pivot). The remaining $n-2$ elements are then partitioned into three parts: the elements smaller than p (*small* elements), the elements between p and q (*medium* elements) and the elements larger than q (*large* elements).

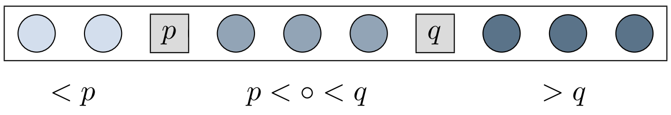


FIGURE 1: Example of the dual-pivot Quicksort procedure after the first partitioning step.

In contrast to standard Quicksort, where each of the $n-1$ remaining elements has to be compared once to the pivot, the number of comparisons for dual-pivot Quicksort depends on the concrete partitioning strategy. In his Ph.D. thesis [20], Robert Sedgewick suggested a dual-pivot Quicksort variant which uses on average $\frac{32}{15}n\log(n)+\mathrm{O}(n)$ comparisons and $0.8n\log(n)+\mathrm{O}(n)$ swaps and therefore is inferior to standard Quicksort. Also other studies (e.g. Hennequin [9]) suggested that the multi-pivot approach would not lead to significant improvements in comparison to classic Quicksort. For this reason, multi-pivot Quicksort did not receive much attention in the following years.

However, in 2009, Vladimir Yaroslavskiy proposed a dual-pivot Quicksort variant which he had developed together with Jon Bentley and Joshua Bloch. In the following, we will denote

[*] For our analysis, we assume the input to consist of n distinct elements. Nevertheless, Quicksort also works for non-distinct data.

this version by Yaroslavskiy-Bentley-Bloch (YBB) Quicksort. As running time experiments had suggested a very good performance of YBB Quicksort, it has been used as standard sorting method for Oracle's Java 7 runtime library since then. In 2012, Wild and Nebel [24] proved that a slightly modified version of YBB Quicksort (Algorithm 3 in this work) uses on average $1.9n\log(n) + \mathrm{O}(n)$ comparisons and $0.6n\log(n) + \mathrm{O}(n)$ swaps. Moreover, Wild, Nebel and Neininger [25] analyzed the number of Java Bytecode instructions and found out that YBB Quicksort executes about 20 % more Java Bytecodes than classic Quicksort. Thus, they conjectured that the superiority of YBB Quicksort in practice is not caused by a smaller number of instructions but by "*advanced features of modern processors*" ([25], p. 25).

In 2014, Kushagra et al. [12] suggested an explanation for the superiority of YBB Quicksort: They conjectured that the running time improvements of YBB Quicksort are due to its better cache behavior. More precisely, they considered another cost measure, the number of "cache misses", and came to the conclusion that YBB Quicksort needs significantly less cache misses than classic Quicksort. Furthermore, they suggested and analyzed a three-pivot Quicksort variant with promising results in theory and in practice. However, the number of cache misses is machine-dependent and not convenient to analyze. For this reason, Nebel, Wild and Martínez [15] introduced a different but closely related cost measure, the number of *scanned elements*, which is defined as *the total distance covered by all scanning indices.* They show that YBB Quicksort needs much less element scans than classic Quicksort ($1.6n\log(n)+\mathrm{O}(n)$ vs $2n\log(n) + \mathrm{O}(n)$ on average) which is, together with the results mentioned before, a convincing explanation for the superior performance of YBB Quicksort in running time studies.

In their article [1], Martin Aumüller and Martin Dietzfelbinger studied and analyzed several Quicksort versions. Among other strategies, they suggest a partitioning method "Count" (Strategy $\mathcal{C}$ in [1, Section 6]) with the aim of minimizing the expected number of key comparisons among all dual-pivot Quicksort variants. They show that this minimum is $1.8n\log(n) + \mathrm{O}(n)$ and that the algorithm "Count" (Algorithm 7 of [1]) achieves this lower bound. In their experimental study, YBB Quicksort and the three-pivot Quicksort algorithm of Kushagra et al. [12] belong to the fastest algorithms, whereas classic Quicksort and the algorithm "Count" are noticeably slower. To be more precise, Classic Quicksort is about 8 % and "Count" Quicksort about 15 % slower than YBB Quicksort (see [1, Section 8.2]). They try to explain the observed differences by looking at the average instruction count and the cache behavior of the algorithms and come to the conclusion that "*neither standard measures like the average comparison count or the average swap count, nor empirical measures like the average instruction count or the cache behavior predict running time correctly when considered in isolation*" (Aumüller and Dietzfelbinger [1, p. 26]).

In this work, we will analyze the dual-pivot Quicksort version "Count" which was suggested by Aumüller and Dietzfelbinger [1, Section 6 and Algorithm 7]. Aumüller et al. [2] have already derived an exact expression for the average number of key comparisons and proved that the strategy “Count” minimizes the expected number of comparisons among all algorithmic dual-pivot strategies. The aim of this master’s thesis is to identify the asymptotic distributions of both the number of comparisons and the number of swaps when using the dual-pivot strategy “Count”. To be more precise, we will derive exact and asymptotic expressions for the average number of swaps in the dual-pivot Quicksort algorithm “Count”. Moreover, we will show that both the number of key comparisons and the number of swaps converge—suitably normalized—to a limit law. The limiting distributions can be identified implicitly by stochastic fixed point equations from which we can compute asymptotic variances. In particular, we will see that for large n, both the number of key comparisons and the number of swaps are concentrated around their mean.

This work is organized as follows: After introducing some notation and the input model in Section 2, Section 3 describes the dual-pivot Quicksort algorithm “Count”. In Section 4, we give a short overview on the main results which are proven in Sections 5–9. Finally, Section 10 compares the obtained results to other Quicksort variants and presents the results of simulation studies.

2 Preliminaries

2.1 Some Notation

◇ As is common, we write

$$\mathbb{1}_{\{\text{expression}\}} = \begin{cases} 1, & \text{if expression is true,} \\ 0, & \text{if expression is false.} \end{cases}$$

◇ For a random variable X, we denote by $\mathcal{L}(X)$, $\mathbb{E}[X]$ and $\operatorname{Var}(X)$ its distribution, expectation and variance, respectively. Furthermore, we write $X \overset{d}{=} Y$ if the random variables X and Y have the same distribution. The covariance between two random variables X and Y is denoted by $\operatorname{Cov}(X, Y)$ and the correlation coefficient by $\operatorname{Corr}(X, Y)$.

◇ We say that a random variable X is L_p-integrable ($1 \le p < \infty$) if and only if $\mathbb{E}[|X|^p]$ is finite. In this case, let $\|X\|_p := \mathbb{E}[|X|^p]^{1/p}$ denote the L_p-norm for X. A sequence $(X_n)_{n\ge1}$ of random variables is said to converge in L_p to a random variable X (denoted by $X_n \xrightarrow{L_p} X$) if and only if $\lim\limits_{n\to\infty} \|X_n - X\|_p = 0$.

◇ We define the *Wasserstein* metric ℓ_2 on the space of probability measures on $\mathbb{R}^d$ ($d \ge 1$) with existing second moments by

$$\ell_2(\mu, \nu) := \inf \{\|X - Y\|_2 : \mathcal{L}(X) = \mu, \mathcal{L}(Y) = \nu\}.$$

The space of the centered probability distributions on $\mathbb{R}^d$ with existing second moments is a complete metric space with regard to the ℓ_2-metric. Furthermore, a sequence

$(X_n)_{n\geq 0}$ converges in ℓ_2 to X if and only if $(X_n)_{n\geq 0}$ converges in distribution and with second moments to X (see Bickel and Freedman [4]).

- ◇ The multinomial distribution with n trials and (non-negative) success probabilities $p_1, \dots, p_k$ (with $p_1 + \dots + p_k = 1$) is denoted by $\mathrm{M}(n; p_1, \dots, p_k)$.
- ◇ We use the common asymptotic notation, i.e. for functions $f, g : \mathbb{N} \to \mathbb{R}$ with $g(n) \neq 0$ for n large enough, we write
  - ▷ $f(n) = \mathrm{O}(g(n))$ if there exist positive numbers C and n_0 such that $|f(n)| \leq C|g(n)|$ for all $n \geq n_0$,
  - ▷ $f(n) = o(g(n))$ if $\frac{f(n)}{g(n)} \to 0$ as $n \to \infty$,
  - ▷ $f(n) \sim g(n)$ if $\frac{f(n)}{g(n)} \to 1$ as $n \to \infty$.
- ◇ We always denote by "log" the natural logaritm to base e and set

$$\mathcal{H}_n = \sum_{k=1}^{n} \frac{1}{k}, \qquad \mathcal{H}_n^{\mathrm{alt}} = \sum_{k=1}^{n} \frac{(-1)^k}{k} \qquad \text{and} \qquad \mathcal{H}_n^{\mathrm{odd}} = \sum_{k=1}^{n} \frac{\mathbb{1}_{\{k \text{ odd}\}}}{k}.$$

- ◇ The following asymptotic expansion is well-known (see e.g. Section 9 of Graham, Knuth and Patashnik [7]):

$$\mathcal{H}_n = \log(n) + \gamma + \mathrm{O}\left(\frac{1}{n}\right),$$

  where $\gamma = 0.5772156649\ldots$ is the Euler–Mascheroni constant. Furthermore, we have (see Lemma 8.1 of Aumüller et al. [2]):

$$\mathcal{H}_n^{\mathrm{alt}} = -\log(2) + \mathrm{O}\left(\frac{1}{n}\right).$$

2.2 Input Model

When analyzing different cost measures of the dual-pivot Quicksort algorithm "Count", we will assume that the input sequence $A = (A[1], \dots, A[n])$ is a random permutation of $\{1, \dots, n\}$ (meaning that each of the $n!$ permutations of $\{1, \dots, n\}$ has equal probability $1/n!$ to become the input). In this setting, we can choose the two outer elements as pivots, i.e. $p := \min\{A[1], A[n]\}$ and $q := \max\{A[1], A[n]\}$.

For the analysis of the limiting distribution of the normalized costs, it will be more convenient to assume that the input consists of a sequence $U_1, \dots, U_n$ of n independent and uniformly on the unit interval $[0, 1]$ distributed random variables. In this case, the input elements are pairwise different almost surely and their ranks form a random permutation of $\{1, \dots, n\}$ (see e.g. Mahmoud [14, Section 1.10]).

3 The Dual-Pivot Quicksort Algorithm “Count”

3.1 Description of the Partitioning Strategy “Count”

As already mentioned in the introduction, the number of comparisons depends on the concrete partitioning strategy. This is different from classic Quicksort, where the number of key comparisons during the first partitioning step is always $n-1$ (since each of the $n-1$ non-pivot elements has to be compared once to the pivot). When using two pivots instead of one single pivot, some of the $n-2$ remaining elements have to be compared to both pivots. More precisely, the following elements have to be compared to both p and q:

- ◇ each medium element,
- ◇ each small element which is compared to the larger pivot q first,
- ◇ each large element which is compared to the smaller pivot p first.

Therefore, in order to minimize the number of key comparisons, small elements should be compared to p first and large elements should be compared to q first. Of course, we do not know the type of an element before having it classified. Hence, we need a strategy to determine whether the next element should be compared to the smaller pivot p or to the larger pivot q first. A simple strategy would be: *“Always compare to the smaller pivot p first”.* However, this strategy does not take into account that we could include the types of the previously classified elements in our decision. For example, if all of the previously classified elements were large, this increases the probability that the next element is also large. We will now present the strategy “Count” which is described in Section 6 of Aumüller and Dietzfelbinger [1]. The basic idea of this strategy is that the next element is compared to the larger pivot q first if and only if we have seen more large than small elements so far. We set $s_0 = l_0 = 0$ and denote by s_i (l_i, respectively) the number of small (large, respectively) elements among the first i classified elements ($i \geq 1$).

> **Strategy “Count”:** *If $s_{i-1} \geq l_{i-1}$, compare the i-th element to the smaller pivot p first. Otherwise, compare to the larger pivot q first.*

It can be shown (see Theorem 15.2 of Aumüller et al. [2]) that this strategy minimizes the expected number of key comparisons among all algorithmic dual-pivot partitioning strategies.

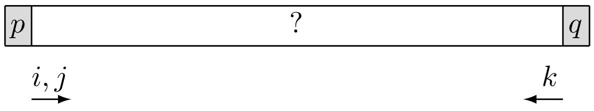
3.2 The Dual-Pivot Quicksort Algorithm “Count”

The following algorithm (Algorithm 1) is—slightly modified—taken from Aumüller et al. [2, Algorithm 1] and gives a possible pseudocode for the strategy “Count”. The only difference, marked in blue, is that Algorithm 1 of Aumüller et al. [2] additionally ensures that $p = A[left]$ and $q = A[right]$ by swapping $A[left]$ and $A[right]$ if $A[right] < A[left]$.

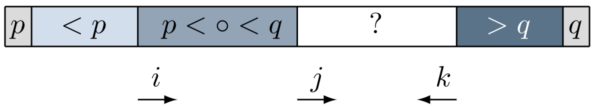
Algorithm 1 Dual-Pivot Quicksort Algorithm “Count”

```
1:  procedure COUNT($A$, $left$, $right$)
2:      if $right \le left$ then
3:          return
4:      if $A[right] < A[left]$ then
5:          $p \leftarrow A[right]$; $q \leftarrow A[left]$    // in [2]: swap $A[left]$ and $A[right]$
6:      else
7:          $p \leftarrow A[left]$; $q \leftarrow A[right]$    // in [2]: $p \leftarrow A[left]$; $q \leftarrow A[right]$ instead of lines 6 and 7
8:      $i \leftarrow left + 1$; $k \leftarrow right - 1$; $j \leftarrow i$
9:      $d \leftarrow 0$
10:     while $j \le k$ do
11:         if $d \ge 0$ then
12:             if $A[j] < p$ then
13:                 swap $A[i]$ and $A[j]$
14:                 $i \leftarrow i + 1$; $j \leftarrow j + 1$; $d \leftarrow d + 1$
15:             else
16:                 if $A[j] < q$ then
17:                     $j \leftarrow j + 1$
18:                 else
19:                     swap $A[j]$ and $A[k]$
20:                     $k \leftarrow k - 1$; $d \leftarrow d - 1$
21:         else
22:             if $A[k] > q$ then
23:                 $k \leftarrow k - 1$; $d \leftarrow d - 1$
24:             else
25:                 if $A[k] < p$ then
26:                     // Perform a cyclic rotation to the left, i.e.,
27:                     // tmp $\leftarrow A[k]$; $A[k] \leftarrow A[j]$; $A[j] \leftarrow A[i]$; $A[i] \leftarrow$ tmp
28:                     $rotate3(A[k], A[j], A[i])$
29:                     $i \leftarrow i + 1$; $d \leftarrow d + 1$
30:                 else
31:                     swap $A[j]$ and $A[k]$
32:                 $j \leftarrow j + 1$
33:     $A[left] \leftarrow A[i-1]$ and $A[i-1] \leftarrow p$    // in [2]: swap $A[left]$ and $A[i-1]$
34:     $A[right] \leftarrow A[k+1]$ and $A[k+1] \leftarrow q$    // in [2]: swap $A[right]$ and $A[k+1]$
35:     COUNT($A$, $left$, $i - 2$)
36:     COUNT($A$, $i$, $k$)
37:     COUNT($A$, $k + 2$, $right$)
```

The above algorithm uses three pointers i, j and k, whereby i and j start at the left end and k starts at the right end. Thus, we have the following initial state (if we assume that $A[left] < A[right]$, i.e. $p = A[left]$ and $q = A[right]$):



The idea is to move the pointers so that $i-1$ marks the position of the rightmost small element, $k+1$ marks the position of the leftmost large element and that the elements $A[j], \ldots, A[k]$ are not classified yet. Hence, the array always has the following form:



Recall that our partitioning strategy “Count” is: *“Compare the next element to q first if and only if we have seen more large than small elements so far”*. Considering that, we use a variable d storing the difference of the number of small and large elements. When classifying the next element, there are two possibilities:

(1) We have classified as least as many small as large elements so far (i.e. $d \geq 0$). In this case, the next element to be classified is $A[j]$. According to our strategy “Count”, we first compare $A[j]$ to the smaller pivot p and then, if necessary, additionally to q.

- ▷ If $A[j]$ is small (i.e. $A[j] < p$), it has to be swapped with $A[i]$.
- ▷ If $A[j]$ is medium (i.e. $A[j] > p$ and $A[j] < q$), no swap is needed.
- ▷ If $A[j]$ is large (i.e. if $A[j] > p$ and $A[j] > q$), we need to swap $A[j]$ and $A[k]$.

(2) We have classified more large than small elements so far (i.e. $d < 0$). In this case, the next element to be classified is $A[k]$. According to our strategy “Count”, we first compare $A[k]$ to the larger pivot q and then, if necessary, to p.

- ▷ If $A[k]$ is large (i.e. $A[k] > q$), no swap is needed.
- ▷ If $A[k]$ is medium (i.e. $A[k] < q$ and $A[k] > p$), it has to be swapped with $A[j]$.
- ▷ If $A[k]$ is small (i.e. if $A[k] < q$ and $A[k] < p$), we perform a cyclic rotation which has the same effect as swapping $A[k]$ and $A[j]$ and then swapping $A[j]$ and $A[i]$.

In each of these cases, the pointers have to be moved as described in the algorithm. As soon as $j > k$, all elements have been classified. We then swap the pivots to their final positions and sort the three sub-arrays recursively.

4 Summary of the Results

In this section, we give a short overview on the main results of this master's thesis.

In Section 5, we will consider a Pólya urn with initially three balls of different types (one small, one medium and one large ball). In each step, we draw a ball uniformly at random from the urn and return it to the urn together with an additional ball of the drawn type. We will derive some elementary properties of this modified Pólya urn model (Section 5.1) and discuss its relation to the dual-pivot Quicksort strategy "Count" (Section 5.2). The main result of Section 5.1 (Theorem 5.2) is the following:

Conditioned on the event "more large than small balls during the first i draws", the probability of drawing another large ball in the next step is exactly $\frac{1}{2}$ for all $i \geq 1$.

Applied to the dual-pivot Quicksort strategy "Count", this implies that once an element is compared to the larger pivot q first, there is a 50 percent chance that this element is indeed large (see Section 5.2).

Section 6 deals with the contraction method and establishes a distributional recurrence for the costs when sorting a random permutation with the dual-pivot Quicksort algorithm "Count". We then define the normalized costs by subtracting the expected values and dividing by n. In the following section (Section 7), we use the contraction method in order to identify the limiting distribution of the normalized number of key comparisons in the dual-pivot Quicksort algorithm "Count". More precisely, Theorem 7.3 and Corollary 7.4 state the following result:

The normalized number $\mathcal{C}_n^* = \frac{\mathcal{C}_n - \mathbb{E}[\mathcal{C}_n]}{n}$ of key comparisons in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of $\{1, \ldots, n\}$ converges in distribution and with second moments to a random variable $\mathcal{C}^*$ whose distribution $\mathcal{L}(\mathcal{C}^*)$ is implicitly characterized by a stochastic fixed point equation. Moreover, as $n \to \infty$, we have

$$\operatorname{Var}(\mathcal{C}_n) \sim \sigma_C^2 n^2,$$

where $\sigma_C^2 = \frac{1609}{300} - \frac{27}{50}\pi^2 + \frac{3}{10}\log(2) = 0.241691110\ldots$.

The results concerning the Pólya urn model of Section 5 are used in Section 8 to show that the average number $\mathbb{E}[\mathcal{S}_n]$ of swaps in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of $\{1, \ldots, n\}$ is

$$\mathbb{E}[\mathcal{S}_n] = \frac{3}{4} n \log(n) + An + \frac{3}{4}\log(n) + \mathrm{O}(1),$$

where $A = -\frac{4}{5} + \frac{3}{4}\gamma - \frac{1}{20}\log(2) = -0.40174561\ldots$ and $n \to \infty$. In fact, we even obtain an *exact* result for the average number of swaps (see Theorem 8.1). We then use the contraction method again in order to obtain a limiting distribution of the normalized number of swaps from which we can compute asymptotic variances (see Theorem 8.3 and Corollary 8.4):

The normalized number $\mathcal{S}_n^* = \frac{\mathcal{S}_n - \mathbb{E}[\mathcal{S}_n]}{n}$ *of swaps in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of* $\{1, \ldots, n\}$ *converges in distribution and with second moments to a random variable* $\mathcal{S}^*$ *whose distribution* $\mathcal{L}(\mathcal{S}^*)$ *is implicitly characterized by a stochastic fixed point equation. Moreover, as* $n \to \infty$*, we have*

$$\mathrm{Var}(\mathcal{S}_n) \sim \sigma_S^2 \, n^2,$$

where $\sigma_S^2 = \frac{47}{48} - \frac{3}{32}\pi^2 + \frac{3}{32}\log(2) = 0.118873802\ldots$*.*

Subsequently, in Section 9, we analyze the correlation between the number of key comparisons and the number of swaps by using the bivariate contraction method and obtain the following result (see Corollary 9.2):

The covariance between the number $\mathcal{C}_n$ *of key comparisons and the number* $\mathcal{S}_n$ *of swaps in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of* $\{1, \ldots, n\}$ *is*

$$\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n) \sim \sigma_{C,S} \, n^2,$$

where $\sigma_{C,S} = \frac{43}{20} - \frac{9}{40}\pi^2 + \frac{7}{40}\log(2) = 0.0506397663\ldots$ *and* $n \to \infty$*. Moreover, we obtain that the correlation between* $\mathcal{C}_n$ *and* $\mathcal{S}_n$ *is, as* $n \to \infty$*,*

$$\mathrm{Corr}(\mathcal{C}_n, \mathcal{S}_n) = \frac{\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n)}{\sqrt{\mathrm{Var}(\mathcal{C}_n)}\sqrt{\mathrm{Var}(\mathcal{S}_n)}} \sim 0.298755\ldots.$$

Finally, we will compare our results to classic Quicksort and YBB Quicksort and we will perform a simulation study in "R" in order to experimentally validate our results.

5 A Related Pólya Urn Model

In this section, we will establish a connection between dual-pivot Quicksort and a (modified) Pólya urn model. Thus, we can obtain results concerning dual-pivot Quicksort by analyzing the Pólya urn model.

5.1 Description of the Model and First Results

Imagine the following (modified) Pólya urn model. At the beginning, the urn contains three balls: one small, one medium and one large ball. We now draw a ball randomly from the urn, return it to the urn and additionally add a ball of the drawn type to the urn. We then repeat this selection procedure. For each $i \geq 0$, let S_i, M_i and L_i denote the numbers of small, medium and large balls added to the urn during the first i draws (i.e. $S_0 = M_0 = L_0 = 0$ and $S_i + M_i + L_i = i$ for any $i \geq 0$).

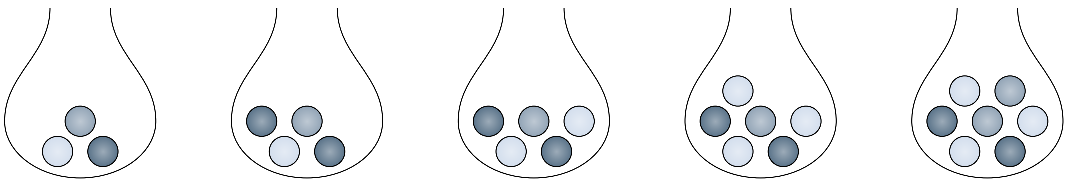


FIGURE 2: A possible evolution of the Pólya urn during the first four steps. The corresponding drawn types in this example are *large* (●), *small* (●), *small* (●) and *medium* (●), which means that we have $(S_4, M_4, L_4) = (2, 1, 1)$.

By a short combinatorial argument, the distribution of (S_i, M_i, L_i) is easily determined:

LEMMA 5.1. *For any i, s, m, $\ell \in \mathbb{N}_0$ with $s + m + \ell = i$, we have*

$$\mathbb{P}(S_i = s, M_i = m, L_i = \ell) = \frac{2}{(i+1)(i+2)},$$

i.e. (S_i, M_i, L_i) is uniformly distributed on the set $\{(s, m, \ell) \in \mathbb{N}_0^3 \mid s + m + \ell = i\}$.

Proof. There are $\binom{i}{s,m,\ell} := \frac{i!}{s!\,m!\,\ell!}$ ways of arranging s small, m medium and ℓ large elements. Any fixed order occurs with probability

$$\frac{1 \cdot \ldots \cdot s \cdot 1 \cdot \ldots \cdot m \cdot 1 \cdot \ldots \cdot \ell}{3 \cdot 4 \cdot \ldots \cdot (i+2)} = 2\,\frac{s!\,m!\,\ell!}{(i+2)!}.$$

Hence, we conclude

$$\mathbb{P}(S_i = s, M_i = m, L_i = \ell) = \binom{i}{s, m, \ell} \cdot 2\,\frac{s!\,m!\,\ell!}{(i+2)!} = \frac{2}{(i+1)(i+2)}.$$ □

The main result of this section is the following: Conditioned on the event *"more large than small balls after i draws"*, the probability that the next ball is large is exactly $\frac{1}{2}$.

THEOREM 5.2. *For any $i \geq 1$,*

$$\mathbb{P}(L_{i+1} = L_i + 1 \mid L_i > S_i) = \frac{1}{2}.$$

The proof of Theorem 5.2 is prepared by the following two lemmas. We first compute the probability that the urn contains more large than small balls after $i \geq 1$ draws. Clearly, this probability should be near $\frac{1}{2}$ if the number i of draws is large.

LEMMA 5.3. *For any $i \geq 1$,*

$$\mathbb{P}(L_i > S_i) = \begin{cases} \frac{i}{2(i+1)}, & \textit{if } i \textit{ is even,} \\ \frac{i+1}{2(i+2)}, & \textit{if } i \textit{ is odd.} \end{cases}$$

Proof. First, by Lemma 5.1 and the law of total probability, we have

$$\begin{aligned} \mathbb{P}(L_i = S_i) &= \sum_{m=0}^{i} \mathbb{P}(L_i = S_i, M_i = m) \\ &= \sum_{m=0}^{i} \mathbb{P}\Big(M_i = m, S_i = L_i = \frac{i-m}{2}\Big) \\ &= \sum_{m=0}^{i} \mathbb{1}_{\{i-m \text{ even}\}} \frac{2}{(i+1)(i+2)} \\ &= \begin{cases} \frac{i+2}{2} \cdot \frac{2}{(i+1)(i+2)} = \frac{1}{i+1}, & \text{if } i \text{ is even,} \\ \frac{i+1}{2} \cdot \frac{2}{(i+1)(i+2)} = \frac{1}{i+2}, & \text{if } i \text{ is odd.} \end{cases} \end{aligned}$$

Due to symmetry, we obtain

$$\mathbb{P}(L_i > S_i) = \frac{1}{2}\,\mathbb{P}(L_i \neq S_i) = \begin{cases} \frac{i}{2(i+1)}, & \text{if } i \text{ is even,} \\ \frac{i+1}{2(i+2)}, & \text{if } i \text{ is odd,} \end{cases}$$

which finishes the proof. □

The following lemma computes the probability of the event *"we have more large than small balls after i draws and the next ball is large"*.

LEMMA 5.4. *For any $i \geq 1$,*

$$\mathbb{P}(L_{i+1} = L_i + 1, L_i > S_i) = \begin{cases} \frac{i}{4(i+1)}, & \text{if } i \text{ is even,} \\ \frac{i+1}{4(i+2)}, & \text{if } i \text{ is odd.} \end{cases}$$

Proof. Since $\{L_i > S_i\} = \bigcup_{\ell=1}^{i} \bigcup_{\substack{s=0 \\ s \leq \ell-1}}^{i-\ell} \{S_i = s, M_i = i-s-\ell, L_i = \ell\}$, we obtain from the law of total probability

$$\mathbb{P}(L_{i+1} = L_i + 1, L_i > S_i) = \sum_{\ell=1}^{i} \sum_{\substack{s=0 \\ s \leq \ell-1}}^{i-\ell} \mathbb{P}(L_{i+1} = L_i + 1, S_i = s, M_i = i-s-\ell, L_i = \ell)$$

$$= \sum_{\ell=1}^{i} \sum_{\substack{s=0 \\ s \leq \ell-1}}^{i-\ell} \mathbb{P}(S_i = s, M_i = i-s-\ell, L_i = \ell)\ \mathbb{P}(L_{i+1} = L_i + 1 | S_i = s, M_i = i-s-\ell, L_i = \ell)$$

$$= \sum_{\ell=1}^{i} \sum_{\substack{s=0 \\ s \leq \ell-1}}^{i-\ell} \frac{2}{(i+1)(i+2)} \cdot \frac{\ell+1}{i+3}$$

$$= \frac{2}{(i+1)(i+2)(i+3)} \left(\sum_{\ell=1}^{\lfloor i/2 \rfloor} \ell(\ell+1) + \sum_{\ell=\lfloor i/2 \rfloor+1}^{i} (i-\ell+1)(\ell+1) \right)$$

$$= \begin{cases} \frac{i}{4(i+1)}, & \text{if } i \text{ is even,} \\ \frac{i+1}{4(i+2)}, & \text{if } i \text{ is odd.} \end{cases}$$

The last step follows from the well-known identities $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ and $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$ for $n \geq 1$. $\square$

Proof (of Theorem 5.2). The desired equality now follows directly by combining Lemma 5.3 and Lemma 5.4. $\square$

Hence, we have shown that conditioned on the event *"more large than small balls after i draws"*, the probability that the next ball is large is exactly $\frac{1}{2}$. In addition to this, we will also need the probability of the event *"we have more large than small balls after i draws and the next ball is small"*.

LEMMA 5.5. *For any $i \geq 1$,*

$$\mathbb{P}(L_i > S_i, S_{i+1} = S_i + 1) = \begin{cases} \frac{i(i+4)}{12(i+1)(i+3)}, & \text{if } i \text{ is even,} \\ \frac{1}{12}, & \text{if } i \text{ is odd.} \end{cases}$$

Proof. Just as in the proof of Lemma 5.4, we obtain from the law of total probability

$$\begin{aligned}
\mathbb{P}(L_i > S_i, S_{i+1} = S_i + 1) &= \sum_{\ell=1}^{i} \sum_{\substack{s=0\\ s\le \ell-1}}^{i-\ell} \mathbb{P}(S_i = s, M_i = i-s-\ell, L_i = \ell, S_{i+1} = S_i + 1)\\
&= \sum_{\ell=1}^{i} \sum_{\substack{s=0\\ s\le \ell-1}}^{i-\ell} \mathbb{P}(S_i = s, M_i = i-s-\ell, L_i = \ell)\, \mathbb{P}(S_{i+1} = S_i + 1 | S_i = s, M_i = i-s-\ell, L_i = \ell)\\
&= \sum_{\ell=1}^{i} \sum_{\substack{s=0\\ s\le \ell-1}}^{i-\ell} \frac{2}{(i+1)(i+2)} \cdot \frac{s+1}{i+3}\\
&= \frac{2}{(i+1)(i+2)(i+3)} \left(\sum_{\ell=1}^{\lfloor i/2 \rfloor} \frac{\ell(\ell+1)}{2} + \sum_{\ell=\lfloor i/2\rfloor+1}^{i} \frac{(i-\ell+1)(i-\ell+2)}{2} \right)\\
&= \begin{cases} \frac{i(i+4)}{12(i+1)(i+3)}, & \text{if } i \text{ is even,}\\ \frac{1}{12}, & \text{if } i \text{ is odd.} \end{cases}
\end{aligned}$$

□

5.2 Relation between the Pólya Urn and Dual-Pivot Quicksort

For a fixed $n \ge 2$, let $V'_n = \big((S'_i, M'_i, L'_i)\big)_{0\le i\le n-2}$ denote the stochastic process describing the evolution of the numbers of small, medium and large elements when partitioning a random permutation of $\{1,\dots,n\}$ with the strategy “Count”. To be more precise, we denote by S'_i, M'_i and L'_i the numbers of small, medium and large elements after having classified i elements with the strategy “Count” ($0 \le i \le n-2$). Then, the process V'_n has the same distribution as the stochastic process $V_n = \big((S_i, M_i, L_i)\big)_{0\le i\le n-2}$ describing the evolution of the numbers of types in the previously introduced Pólya urn model with $n-2$ draws. The idea is that in both models, the probability of a concrete path $v_n = \big((s_i, m_i, \ell_i)\big)_{0\le i\le n-2}$ depends only on the final values s_{n-2}, m_{n-2} and ℓ_{n-2}.

LEMMA 5.6. *For $v_n = \big((s_i, m_i, \ell_i)\big)_{0\le i\le n-2} \in (\mathbb{N}_0^3)^{n-1}$, we have*

$$\mathbb{P}(V'_n = v_n) = \mathbb{P}(V_n = v_n).$$

Proof. If $v_n = \big((s_i, m_i, \ell_i)\big)_{0\le i\le n-2}$ is a possible realization of V'_n (meaning that the probability $\mathbb{P}(V'_n = v_n)$ is positive), we have

$$\begin{aligned}
\mathbb{P}(V'_n = v_n) = \mathbb{P}&\big((S'_{n-2}, M'_{n-2}, L'_{n-2}) = (s_{n-2}, m_{n-2}, \ell_{n-2})\big)\\
&\cdot \mathbb{P}\big(V'_n = v_n \,|\, (S'_{n-2}, M'_{n-2}, L'_{n-2}) = (s_{n-2}, m_{n-2}, \ell_{n-2})\big).
\end{aligned}$$

As $(S'_{n-2}, M'_{n-2}, L'_{n-2}) = (s_{n-2}, m_{n-2}, \ell_{n-2})$ is equivalent to *“the pivots are* $s_{n-2}+1$ *and*

$s_{n-2} + m_{n-2} + 2$", the first factor equals $\frac{1}{\binom{n}{2}}$. Moreover, once we condition on the final values $S'_{n-2} = s_{n-2}$, $M'_{n-2} = m_{n-2}$ and $L'_{n-2} = \ell_{n-2}$, every of the possible $\binom{n}{s_{n-2},\, m_{n-2},\, \ell_{n-2}}$ paths is equally likely. Hence, we get

$$\mathbb{P}(V'_n = v_n) = \frac{1}{\binom{n}{2}} \cdot \frac{1}{\binom{n-2}{s_{n-2},\, m_{n-2},\, \ell_{n-2}}} = 2\, \frac{s_{n-2}!\, m_{n-2}!\, \ell_{n-2}!}{n!}.$$

On the other hand, the proof of Lemma 5.1 shows that

$$\mathbb{P}(V_n = v_n) = 2\, \frac{s_{n-2}!\, m_{n-2}!\, \ell_{n-2}!}{n!}$$

as well. Thus, both processes have the same distribution. □

We note that the proof of the preceding lemma uses arguments similar to those in the proof of Lemma 3.1 and Lemma 11.2 in Aumüller et al. [2]. Also Section 2.4.7 of Wild's Ph.D. thesis [23] characterizes the relation between dual-pivot Quicksort and the Pólya urn model. In the remainder of this section, we will apply the results from Section 5.1 to the dual-pivot Quicksort strategy "Count".

For $n \geq 2$, let S_n^+, M_n^+ and L_n^+ denote the numbers of small, medium and large elements compared to q first when partitioning an input sequence of length n with the strategy "Count". Recall that we compare an element to the larger pivot q first if and only if we have classified more large than small elements so far. The following graphic (Figure 3) illustrates the definition of S_n^+.

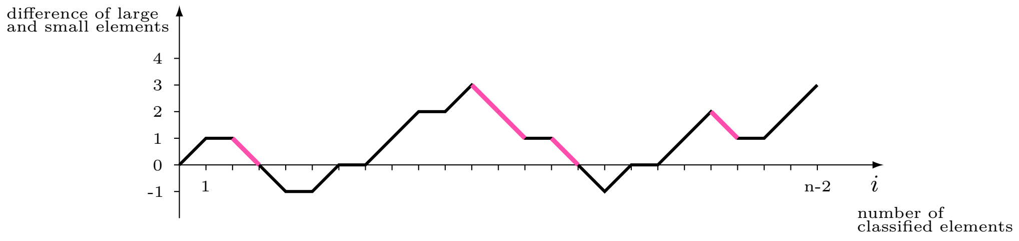

difference of large and small elements
4
3
2
1
0
-1
1
n-2
i
number of classified elements

FIGURE 3: Path describing the sequence of classified types: Small elements correspond to down-steps, medium elements to horizontal steps and large elements to up-steps. The next element is compared to q first if and only if the path is above the horizontal axis. Thus, S_n^+ counts every down-step above the horizontal axis (colored in pink).

The following lemma shows that the expected value of $S_n^+ + M_n^+$ equals the expected value of L_n^+. This means that on average, we have just as many large elements which are compared to q first as small or medium elements which are compared to q first.

LEMMA 5.7. *For any $n \geq 2$, we have*

$$\mathbb{E}[S_n^+ + M_n^+ - L_n^+] = 0.$$

Proof. The relation of Lemma 5.6 between dual-pivot Quicksort and the Pólya urn implies

$$S_n^+ \overset{d}{=} \sum_{i=1}^{n-3} \mathbb{1}_{\{L_i > S_i,\ S_{i+1} = S_i + 1\}},$$

$$M_n^+ \overset{d}{=} \sum_{i=1}^{n-3} \mathbb{1}_{\{L_i > S_i,\ M_{i+1} = M_i + 1\}},$$

$$L_n^+ \overset{d}{=} \sum_{i=1}^{n-3} \mathbb{1}_{\{L_i > S_i,\ L_{i+1} = L_i + 1\}}.$$

Thus, using Theorem 5.2, we obtain

$$\begin{aligned}
&\mathbb{E}[S_n^+ + M_n^+ - L_n^+] \\
&= \sum_{i=1}^{n-3} \mathbb{P}(L_i > S_i,\ S_{i+1} = S_i + 1) + \mathbb{P}(L_i > S_i,\ M_{i+1} = M_i + 1) - \mathbb{P}(L_i > S_i,\ L_{i+1} = L_i + 1) \\
&= \sum_{i=1}^{n-3} \mathbb{P}(L_i > S_i)\Big(\mathbb{P}(L_{i+1} = L_i \,|\, L_i > S_i) - \mathbb{P}(L_{i+1} = L_i + 1 \,|\, L_i > S_i)\Big) \\
&= 0.
\end{aligned}$$

□

LEMMA 5.8. *For any $n \geq 2$, we have*

$$\mathbb{E}[S_n^+] = \frac{1}{12}n - \frac{7}{24} + \frac{1}{8(n - \mathbb{1}_{\{n\ even\}})}.$$

Proof. Using Lemma 5.5, we obtain

$$\begin{aligned}
\mathbb{E}[S_n^+] &= \mathbb{E}\left[\sum_{i=1}^{n-3} \mathbb{1}_{\{L_i > S_i,\ S_{i+1} = S_i + 1\}}\right] \\
&= \sum_{i=1}^{n-3} \mathbb{P}(L_i > S_i,\ S_{i+1} = S_i + 1) \\
&= \sum_{i=1}^{n-3} \left(\mathbb{1}_{\{i \text{ even}\}} \frac{i(i+4)}{12(i+1)(i+3)} + \mathbb{1}_{\{i \text{ odd}\}} \frac{1}{12}\right) \\
&= \frac{1}{12}n - \frac{7}{24} + \frac{1}{8(n - \mathbb{1}_{\{n \text{ even}\}})}.
\end{aligned}$$

Note that the last step follows easily by induction on n. □

6 The Contraction Method

Introduced in 1991 by Uwe Rösler [17], the contraction method is a useful tool when analyzing random recursive structures and algorithms. As the application of the contraction method to dual-pivot Quicksort has already been worked out in Section 4 of Wild, Nebel and Neininger [25], most results of the remainder of this section can be found there.

6.1 A General Convergence Theorem

This section presents a convergence theorem which we will frequently use in the following sections. The results can be found in Rösler [18] for the univariate case and in Neininger [16] for the multivariate case.

Let X be a centered, square-integrable random variable in $\mathbb{R}^d$ $(d \geq 1)$. We assume that there exist $K \geq 1$, square-integrable random $d \times d$ matrices $A_1, \dots, A_K$ and a centered, square-integrable d-dimensional random vector b such that

$$X \stackrel{d}{=} \sum_{r=1}^{K} A_r X^{(r)} + b, \tag{1}$$

where $X^{(1)}, \dots, X^{(K)}$ and $(A_1, \dots, A_K, b)$ are independent and $X^{(r)}$ is distributed as X for all $r \in \{1, \dots, K\}$. We denote by $\|B\|_{\text{op}} := \sup_{\|x\|=1} \|Bx\|$ the operator norm of a matrix B and by B^t the transposed matrix. Then, Lemma 3.1 in Neininger [16] implies the following result.

THEOREM 6.1. *If* $\sum_{r=1}^{K} \mathbb{E}[\|A_r^t A_r\|_{op}] < 1$*, then there is a unique solution* $\mathcal{L}(X)$ *to* (1) *among all centered, square-integrable distributions.*

The idea is to see the stochastic fixed point equation (1) as a map within the space of probability distributions on $\mathbb{R}^d$. Restricted to the space of all centered, square-integrable distributions, this map is a contraction with respect to the ℓ_2-metric. A proof of this assertion can be found in Neininger [16, Proof of Lemma 3.1]. As the space of the centered probability measures on $\mathbb{R}^d$ with existing second moments is a complete metric space with regard to the ℓ_2-metric, the Banach fixed point theorem implies the existence of a unique fixed point.

Now, let $(X_n)_{n\geq 0}$ be a sequence of random variables in $\mathbb{R}$ or $\mathbb{R}^2$ satisfying the distributional recursion

$$X_n \stackrel{d}{=} \sum_{r=1}^{K} A_r^{(n)} X_{I_r^{(n)}}^{(r)} + b^{(n)}, \quad n \geq n_0, \tag{2}$$

for some $n_0 \geq 1$ and $K \geq 1$. We assume that $I^{(n)} = (I_1^{(n)}, \dots, I_K^{(n)})$ is a vector of random integers in $\{0, \dots, n\}$. In the univariate case, $A_r^{(n)}$ and $b^{(n)}$ are real-valued random variables and in the bivariate case, $A_r^{(n)}$ is a 2×2 matrix for all $r \in \{1, \dots, K\}$ and $b^{(n)}$ is a random vector in $\mathbb{R}^2$.

Moreover, we assume that the following conditions are satisfied:

- ◇ $(X_j^{(1)})_{0\leq j\leq n}$, …, $(X_j^{(K)})_{0\leq j\leq n}$ and $(A_1^{(n)},\dots,A_K^{(n)},b^{(n)},I^{(n)})$ are independent for all $n\geq n_0$.
- ◇ $X_j^{(r)}$ has the same distribution as X_j for all $r\in\{1,\dots,K\}$ and $j\geq 0$.
- ◇ X_n, $A_r^{(n)}$ and $b^{(n)}$ are square-integrable for all n and $r\in\{1,\dots,K\}$.
- ◇ For all $n\geq 0$, X_n and $b^{(n)}$ are centered, i.e. $\mathbb{E}[X_n]=\mathbb{E}[b^{(n)}]=0$.

Under these assumptions, we have the following general convergence theorem, see Theorem 4.1 of Neininger [16]:

THEOREM 6.2. *Let $(X_n)_{n\geq 0}$ satisfy the distributional recursion* (2)*. In addition to the preceding assumptions, we assume the following conditions:*

(A1) $(A_1^{(n)},\dots,A_K^{(n)},b^{(n)})\xrightarrow{\ell_2}(A_1,\dots,A_K,b)$ *as* $n\to\infty$.

(A2) $\sum_{r=1}^{K}\mathbb{E}[\|A_r^tA_r\|_{\text{op}}]<1$.

(A3) $\mathbb{E}\Big[\mathbb{1}_{\{I_r^{(n)}\leq\ell\}\cup\{I_r^{(n)}=n\}}\left\|(A_r^{(n)})^tA_r^{(n)}\right\|_{\text{op}}\Big]\to 0$ *as* $n\to\infty$*, for* $\ell\geq 0$ *and* $r\in\{1,\dots,K\}$.

Then, the sequence $(X_n)_{n\geq 0}$ converges in ℓ_2 to some random variable X, where $\mathcal{L}(X)$ is the (among all centered, square-integrable distributions) unique solution of (1)*. In other words, the sequence $(X_n)_{n\geq 0}$ converges in distribution and with second (mixed) moments to X.*

6.2 Preliminary Considerations

In the following sections, we will assume the input to be a sequence $U_1,\dots,U_n$ of n independent and uniformly on $[0,1]$ distributed random variables. This is in accordance with our input model as the ranks of $U_1,\dots,U_n$ form a random permutation of $\{1,\dots,n\}$. Note that the order given in the input sequence is not necessarily the order in which Algorithm 1 processes the elements. However, we do not change the distribution of the costs if we assume U_1 and U_2 to be the outermost elements and if we assume $U_3, U_4,\dots,U_n$ to be the sequence of the remaining elements in the order in which they are read by Algorithm 1. As in the previous sections, the two outermost elements U_1 and U_2 are chosen as pivots, i.e. $P=\min\{U_1,U_2\}$ and $Q=\max\{U_1,U_2\}$. We now denote by $D=(D_1,D_2,D_3)$ the "spacings" between the pivots on the unit intervall. More precisely, we set (see Figure 4)

$$D=(D_1,D_2,D_3)=(P,Q-P,1-Q),$$

which means that D contains the probabilities for an element to be small, medium or large, respectively. Obviously, we have $D_1+D_2+D_3=1$, so that D_3 is determined by D_1 and D_2.

It can be proven (see e.g. David and Nagaraja [5], p. 133f) that (D_1, D_2) has the following density:

$$f_{(D_1,D_2)}(x,y) = \begin{cases} 2, & \text{for } x, y \geq 0 \text{ and } x + y \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

For a measurable function $g : [0,1]^3 \to \mathbb{R}$ with $\mathbb{E}[|g(D_1, D_2, D_3)|] < \infty$, we therefore have

$$\mathbb{E}[g(D_1, D_2, D_3)] = 2 \int_0^1 \int_0^{1-x} g(x, y, 1-x-y) \, \mathrm{d}y \, \mathrm{d}x. \tag{3}$$

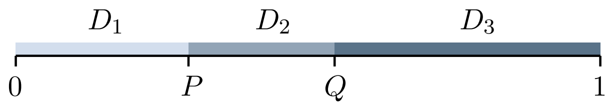


FIGURE 4: The Definition of $D = (D_1, D_2, D_3)$

In what follows, we denote by $I^{(n)} = (I_1^{(n)}, I_2^{(n)}, I_3^{(n)})$ the numbers of small, medium and large elements of the input sequence. It is easy to determine the marginal distribution of $I_1^{(n)}$, $I_2^{(n)}$ and $I_3^{(n)}$.

LEMMA 6.3. *The sizes $I_1^{(n)}$, $I_2^{(n)}$ and $I_3^{(n)}$ of the three subproblems are identically distributed with*

$$\mathbb{P}(I_r^{(n)} = k) = \frac{n-k-1}{\binom{n}{2}}$$

for $r \in \{1,2,3\}$ and $k \in \{0, \dots, n-2\}$. Furthermore, we have

$$\mathbb{E}[I_r^{(n)}] = \frac{n-2}{3}$$

for all $r \in \{1,2,3\}$.

Proof. We just consider the case $r = 1$ as the cases $r = 2$ and $r = 3$ follow similarly. The number $I_1^{(n)}$ of small elements is determined by the rank of the smaller pivot element P. More precisely, $\left\{I_1^{(n)} = k\right\} = \{\mathrm{rank}(P) = k+1\} = \{\mathrm{rank}(P) = k+1, \mathrm{rank}(Q) \in \{k+2, \dots, n\}\}$ for $k \in \{0, \dots, n-2\}$. Hence, there are $\binom{n}{2}$ possibilities to choose the two pivots and $n-k-1$ possibilities to choose the pivots such that we have exactly k small elements. The rest follows directly by using the fact that $I_1^{(n)} + I_2^{(n)} + I_3^{(n)} = n-2$. $\square$

As already mentioned, given $D = (d_1, d_2, d_3)$, the probability that an element is small, medium or large is d_1, d_2 or d_3, respectively. Since the random variables $U_1, \dots, U_n$ are independent, we get: Given $D = (d_1, d_2, d_3)$, the vector $I^{(n)}$ is multinomially $M(n-2; d_1, d_2, d_3)$ distributed. We write

$$\mathcal{L}(I^{(n)}) = M(n-2; D_1, D_2, D_3).$$

Since we will use the contraction method in the following sections, we already state two convergence lemmas.

LEMMA 6.4. *For any $r \in \{1,2,3\}$, we have*

$$\frac{I_r^{(n)}}{n} \to D_r$$

as $n \to \infty$, both almost surely and in L_2.

Proof. Recall that given $D = (d_1, d_2, d_3)$, $I^{(n)}$ is multinomially $M(n-2; d_1, d_2, d_3)$ distributed. Therefore, conditioned on $D = (d_1, d_2, d_3)$, $\frac{I_r^{(n)}}{n}$ converges to d_r almost surely by the strong law of large numbers ($r \in \{1,2,3\}$). It follows for $r \in \{1,2,3\}$:

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{I_r^{(n)}}{n} = D_r\right) = \mathbb{E}\Big[\mathbb{1}_{\left\{\lim\limits_{n\to\infty} \frac{I_r^{(n)}}{n} = D_r\right\}}\Big] = \mathbb{E}\Big[\mathbb{E}\Big[\mathbb{1}_{\left\{\lim\limits_{n\to\infty} \frac{I_r^{(n)}}{n} = D_r\right\}}\Big| D_r\Big]\Big] = 1.$$

From the dominated convergence theorem, we also get L_2-convergence (note that both $\frac{I_r^{(n)}}{n}$ and D_r are in $[0,1]$):

$$\lim_{n\to\infty} \mathbb{E}\left[\Big(\frac{I_r^{(n)}}{n} - D_r\Big)^2\right] = \mathbb{E}\left[\lim_{n\to\infty}\Big(\frac{I_r^{(n)}}{n} - D_r\Big)^2\right] = 0. \qquad \square$$

LEMMA 6.5. *For any $r \in \{1,2,3\}$, we have*

$$\frac{I_r^{(n)}}{n} \log\left(\frac{I_r^{(n)}}{n}\right) \to D_r \log(D_r)$$

as $n \to \infty$, both almost surely and in L_2.

Proof. The almost sure convergence directly follows from Lemma 6.4 and the continuity of the function $x \mapsto x\log(x)$ on $[0,1]$ (with the convention $0\log(0) := 0$). The L_2-convergence follows as in the proof of the previous lemma from the dominated convergence theorem (note that the function $x \mapsto x\log(x)$ is bounded on [0,1]). $\square$

6.3 The Dual-Pivot Quicksort Recurrence

In what follows, we will analyze different cost measures of the dual-pivot Quicksort algorithm "Count". Let C_n denote the costs when sorting a random permutation of $\{1, \ldots, n\}$ with the dual-pivot Quicksort algorithm "Count" and let T_n denote the costs during the first partitioning procedure. We have

$$C_n = T_n + \text{ costs for the three subproblems.}$$

Let $I^{(n)} = (I_1^{(n)}, I_2^{(n)}, I_3^{(n)})$ denote, as previously, the sizes of the three subproblems. Hennequin [8] showed the following implication:

> *If the input is a random permutation of $\{1, \dots, n\}$ and if every key comparison involves a pivot element of the current partitioning step, then the subproblems are also random permutations of their elements.*

This implies that the ranks of the elements of each subproblem again form a random permutation. Moreover, conditional on their sizes, the three subproblems are independent of each other. Thus, for $n \geq 2$, we obtain the following distributional recursion:

$$C_n \overset{d}{=} \sum_{r=1}^{3} C^{(r)}_{I_r^{(n)}} + T_n, \tag{4}$$

where $(C_j^{(1)})_{0\leq j\leq n}$, $(C_j^{(2)})_{0\leq j\leq n}$, $(C_j^{(3)})_{0\leq j\leq n}$ and $(T_n, I^{(n)})$ are independent and $C_j^{(r)}$ is distributed as C_j for $r \in \{1, 2, 3\}$ and $j \geq 0$.

6.3.1 The Recurrence for the Expected Costs

In particular, for $n \geq 2$, we get the following recursive form for the expected costs:

$$\mathbb{E}[C_n] = \sum_{r=1}^{3} \mathbb{E}[C^{(r)}_{I_r^{(n)}}] + \mathbb{E}[T_n] = 3\ \mathbb{E}[C_{I_1^{(n)}}] + \mathbb{E}[T_n].$$

Conditioning on the size of the first subproblem and using Lemma 6.3 yields

$$\mathbb{E}[C_{I_1^{(n)}}] = \mathbb{E}[\mathbb{E}[C_{I_1^{(n)}} | I_1^{(n)}]] = \sum_{k=0}^{n-2} \mathbb{P}(I_1^{(n)} = k)\, \mathbb{E}[C_k] = \sum_{k=0}^{n-2} \frac{n-k-1}{\binom{n}{2}}\, \mathbb{E}[C_k].$$

Hence, the expected costs satisfy $\mathbb{E}[C_0] = \mathbb{E}[C_1] = 0$ and the following recurrence for $n \geq 2$:

$$\mathbb{E}[C_n] = \frac{6}{n(n-1)} \sum_{k=0}^{n-2} (n-k-1)\, \mathbb{E}[C_k] + \mathbb{E}[T_n].$$

This recurrence has already been derived and analyzed before (see e.g. Section 4.2 of Wild's master's thesis [22]). In Section 4.2.1 of [22], Wild presents an elementary derivation of the closed form of $\mathbb{E}[C_n]$ and obtains the following result for $n \geq 4$:

$$\begin{aligned}\mathbb{E}[C_n] =& \frac{1}{\binom{n}{4}} \sum_{i=5}^{n} \binom{i}{4} \sum_{j=3}^{i-2} \left(\mathbb{E}[T_{j+2}] - \frac{2j}{j+2}\mathbb{E}[T_{j+1}] + \frac{j(j-1)}{(j+2)(j+1)}\mathbb{E}[T_j] \right) \\ &+ \frac{n+1}{5} \left(\mathbb{E}[T_4] + \frac{1}{2}\mathbb{E}[T_2] \right).\end{aligned}$$

6.3.2 The Recurrence for the Normalized Costs

We now define the normalized costs as $C_0^* = 0$ and

$$C_n^* = \frac{C_n - \mathbb{E}[C_n]}{n} \qquad \text{for } n \geq 1.$$

Thus, $(C_n^*)_{n\geq 0}$ is a sequence of centered, square-integrable random variables. Using (4) leads to the following distributional recurrence for C_n^* $(n \geq 2)$:

$$C_n^* \stackrel{d}{=} \sum_{r=1}^{3} A_r^{(n)} C^{*(r)}_{I_r^{(n)}} + b^{(n)},$$

where $(C_j^{*(1)})_{0\leq j\leq n}$, $(C_j^{*(2)})_{0\leq j\leq n}$, $(C_j^{*(3)})_{0\leq j\leq n}$ and $(b^{(n)}, I^{(n)})$ are independent and $C_j^{*(r)}$ is distributed as C_j^* for $r \in \{1,2,3\}$ and $j \geq 0$,

$$A_r^{(n)} = \frac{I_r^{(n)}}{n} \quad \text{and} \quad b^{(n)} = \frac{1}{n}\left(T_n - \mathbb{E}[C_n] + \sum_{r=1}^{3} \mathbb{E}[C_{I_r^{(n)}} | I_r^{(n)}]\right).$$

In order to use the contraction method (Theorem 6.2) later, it will be necessary to determine the limit of $b^{(n)}$. Therefore, we will need the following result.

LEMMA 6.6. *If there exist constants A and B such that $\mathbb{E}[C_n] = An\log(n) + Bn + o(n)$ as $n \to \infty$, then*

$$\frac{1}{n}\left(-\mathbb{E}[C_n] + \sum_{r=1}^{3} \mathbb{E}[C_{I_r^{(n)}} | I_r^{(n)}]\right) \xrightarrow{L_2} A\sum_{r=1}^{3} D_r \log(D_r).$$

Proof. Using that $I_r^{(n)} \in \{0,\dots,n-2\}$ and the fact that $\sum_{r=1}^{3} I_r^{(n)} = n-2$, we obtain

$$\begin{aligned}
&\frac{1}{n}\left(-\mathbb{E}[C_n] + \sum_{r=1}^{3} \mathbb{E}[C_{I_r^{(n)}} | I_r^{(n)}]\right)\\
&= \frac{1}{n}\left(-An\log(n) - Bn - o(n) + \sum_{r=1}^{3}\left(AI_r^{(n)}\log(I_r^{(n)}) + BI_r^{(n)} + o(I_r^{(n)})\right)\right)\\
&= -A\log(n) + A\sum_{r=1}^{3}\frac{I_r^{(n)}}{n}\log(I_r^{(n)}) + o(1)\\
&= -A\log(n) + A\sum_{r=1}^{3}\frac{I_r^{(n)}}{n}\log\left(\frac{I_r^{(n)}}{n}\right) + A\sum_{r=1}^{3}\frac{I_r^{(n)}}{n}\log(n) + o(1)\\
&= A\sum_{r=1}^{3}\frac{I_r^{(n)}}{n}\log\left(\frac{I_r^{(n)}}{n}\right) + o(1).
\end{aligned}$$

The assertion now follows from Lemma 6.5. □

7 Analysis of the Number of Key Comparisons

We will first analyze the distribution of the number of key comparisons. For this, we denote by $\mathcal{C}_n$ the number of key comparisons of the dual-pivot Quicksort algorithm “Count” when sorting a random permutation of $\{1,\dots,n\}$ and by $T_C(n)$ the number of key comparisons during the first partitioning step.

7.1 The Average Number of Key Comparisons

Theorem 12.1 of Aumüller et al. [2] contains an exact result for the expected value of $\mathcal{C}_n$ for $n \geq 4$:

$$\begin{aligned}\mathbb{E}[\mathcal{C}_n] =& \frac{9}{5}n\mathcal{H}_n - \frac{1}{5}n\mathcal{H}_n^{\mathrm{alt}} - \frac{89}{25}n + \frac{67}{40}\mathcal{H}_n - \frac{3}{40}\mathcal{H}_n^{\mathrm{alt}} - \frac{83}{800} + \frac{(-1)^n}{10}\\ &- \frac{\mathbb{1}_{\{n\ \mathrm{even}\}}}{320}\left(\frac{1}{n-3} + \frac{3}{n-1}\right) + \frac{\mathbb{1}_{\{n\ \mathrm{odd}\}}}{320}\left(\frac{3}{n-2} + \frac{1}{n}\right).\end{aligned}$$

Plugging in the asymptotic expansions of $\mathcal{H}_n$ and $\mathcal{H}_n^{\mathrm{alt}}$, we obtain the following result.

THEOREM 7.1. *The average number of key comparisons in the dual-pivot Quicksort algorithm “Count” when sorting a random permutation of* $\{1,\dots,n\}$ *is*

$$\mathbb{E}[\mathcal{C}_n] = \frac{9}{5}n\log(n) + An + \frac{67}{40}\log(n) + \mathrm{O}(1),$$

where $A = \frac{9}{5}\gamma + \frac{1}{5}\log(2) - \frac{89}{25} = -2.38238236\dots$ *and* $n \to \infty$.

7.2 Distributional Analysis of the Number of Key Comparisons

As discussed in Section 6.3, the number $\mathcal{C}_n$ of key comparisons satisfies the following distributional recurrence for $n \geq 2$:

$$\mathcal{C}_n \overset{d}{=} \sum_{r=1}^{3} \mathcal{C}^{(r)}_{I_r^{(n)}} + T_C(n),$$

where $(\mathcal{C}_j^{(1)})_{0\leq j\leq n}$, $(\mathcal{C}_j^{(2)})_{0\leq j\leq n}$, $(\mathcal{C}_j^{(3)})_{0\leq j\leq n}$ and $(T_C(n), I^{(n)})$ are independent and $\mathcal{C}_j^{(r)}$ is distributed as $\mathcal{C}_j$ for $r \in \{1,2,3\}$ and $j \geq 0$. We now set $\mathcal{C}_0^* = 0$ and

$$\mathcal{C}_n^* = \frac{\mathcal{C}_n - \mathbb{E}[\mathcal{C}_n]}{n} \qquad \text{for } n \geq 1.$$

Just as in Section 6.3.2, we obtain the following distributional recurrence for the normalized number $\mathcal{C}_n^*$ of key comparisons ($n \geq 2$):

$$\mathcal{C}_n^* \overset{d}{=} \sum_{r=1}^{3} A_r^{(n)} \mathcal{C}^{*(r)}_{I_r^{(n)}} + b_C^{(n)},$$

where $(\mathcal{C}_j^{*(1)})_{0\le j\le n}$, $(\mathcal{C}_j^{*(2)})_{0\le j\le n}$, $(\mathcal{C}_j^{*(3)})_{0\le j\le n}$ and $(b_C^{(n)}, I^{(n)})$ are independent and $\mathcal{C}_j^{*(r)}$ is distributed as $\mathcal{C}_j^*$ for $r \in \{1,2,3\}$ and $j \ge 0$,

$$A_r^{(n)} = \frac{I_r^{(n)}}{n} \quad \text{and} \quad b_C^{(n)} = \frac{1}{n}\left(T_C(n) - \mathbb{E}[\mathcal{C}_n] + \sum_{r=1}^{3} \mathbb{E}[\mathcal{C}_{I_r^{(n)}} | I_r^{(n)}]\right).$$

In order to apply Theorem 6.2, we have to check the following three conditions:

(A1) $(A_1^{(n)}, A_2^{(n)}, A_3^{(n)}, b_C^{(n)}) \xrightarrow{\ell_2} (A_1, A_2, A_3, b_C)$ as $n \to \infty$.

(A2) $\sum\limits_{r=1}^{3} \mathbb{E}[A_r^2] < 1$.

(A3) $\mathbb{E}\left[\mathbb{1}_{\{I_r^{(n)} \le \ell\} \cup \{I_r^{(n)} = n\}} (A_r^{(n)})^2\right] \to 0$ for $\ell \ge 0$, $r \in \{1,2,3\}$ and $n \to \infty$.

(A1): As we have already shown in Lemma 6.4, $A_r^{(n)} = \frac{I_r^{(n)}}{n}$ converges in L_2 to $A_r := D_r$ for $r \in \{1,2,3\}$ and as $n \to \infty$. Furthermore, according to Lemma 6.6 and Theorem 7.1, as $n \to \infty$, we have

$$\frac{1}{n}\left(-\mathbb{E}[\mathcal{C}_n] + \sum_{r=1}^{3} \mathbb{E}[\mathcal{C}_{I_r^{(n)}} | I_r^{(n)}]\right) \xrightarrow{L_2} \frac{9}{5} \sum_{r=1}^{3} D_r \log(D_r).$$

It therefore remains to show the convergence of $\frac{1}{n} T_C(n)$. Recall that S_n^+, M_n^+ and L_n^+ denote the numbers of small, medium and large elements which are compared to the larger pivot q first. The number $T_C(n)$ of key comparisons during the first partitioning phase includes

- ◇ 1 comparison for the pivots U_1 and U_2,
- ◇ $2\,I_2^{(n)}$ comparisons for the medium elements (each medium element has to be compared to both pivots),
- ◇ $I_1^{(n)} + S_n^+$ comparisons for the small elements (each of the S_n^+ small elements which are compared to q first have to be compared to both pivots),
- ◇ $2\,I_3^{(n)} - L_n^+$ comparisons for the large elements (each of the $I_3^{(n)} - L_n^+$ large elements which are compared to p first have to be compared to both pivots).

Altogether, using $I_1^{(n)} + I_2^{(n)} + I_3^{(n)} = n - 2$, we obtain

$$T_C(n) = 1 + I_1^{(n)} + 2\,I_2^{(n)} + 2\,I_3^{(n)} + S_n^+ - L_n^+ = n - 1 + I_2^{(n)} + I_3^{(n)} + S_n^+ - L_n^+$$

and

$$\frac{T_C(n)}{n} = \frac{n-1}{n} + \frac{I_2^{(n)}}{n} + \frac{I_3^{(n)}}{n} + \frac{S_n^+}{n} - \frac{L_n^+}{n}.$$

From Lemma 6.4, we get the L_2-convergence $\frac{I_r^{(n)}}{n} \to D_r$ as $n \to \infty$. The convergence of the two remaining terms $\frac{S_n^+}{n}$ and $\frac{L_n^+}{n}$ is examined in the following lemma.

LEMMA 7.2. *As $n \to \infty$, we have*

$$\frac{S_n^+}{n} \xrightarrow{L_2} \mathbb{1}_{\{D_3>D_1\}} D_1, \quad \frac{M_n^+}{n} \xrightarrow{L_2} \mathbb{1}_{\{D_3>D_1\}} D_2 \quad \textit{and} \quad \frac{L_n^+}{n} \xrightarrow{L_2} \mathbb{1}_{\{D_3>D_1\}} D_3.$$

Proof. We define the stochastic process $W = (W_i)_{i\geq 0}$ by $W_0 := 0$ and

$$W_i = \sum_{j=1}^{i} \left(\mathbb{1}_{\{\text{the } j\text{-th classified element is large}\}} - \mathbb{1}_{\{\text{the } j\text{-th classified element is small}\}}\right)$$

for $i \geq 1$. In other words, W_i holds the difference of the number of large and small elements after having classified i elements. Given $D = (d_1, d_2, d_3)$, W is a random walk on $\mathbb{Z}$ with the following transition probabilities for $i \geq 0$, $x, y \in \mathbb{Z}$:

$$\mathbb{P}(W_{i+1} = y \,|\, W_i = x, D = (d_1, d_2, d_3)) = \begin{cases} d_1, & \text{if } y = x-1, \\ d_2, & \text{if } y = x, \\ d_3, & \text{if } y = x+1, \\ 0, & \text{otherwise.} \end{cases}$$

The following graphic (Figure 5) shows a possible realization of the first steps of W.

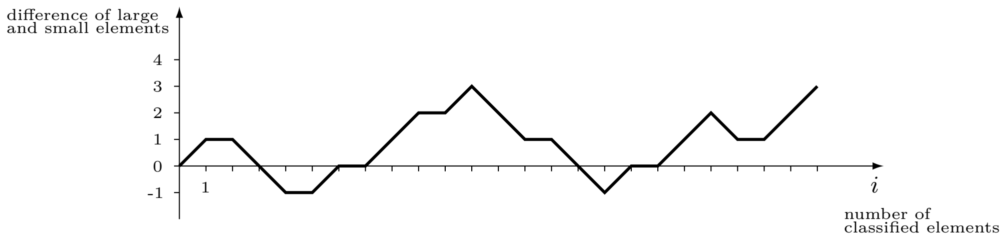


FIGURE 5: A possible evolution of W. Given $D = (d_1, d_2, d_3)$, W goes one step down with probability d_1, stays at its current state with probability d_2 and goes one step up with probability d_3.

We now distinguish between two cases:

(1) If $d_1 < d_3$, the random walk W has a positive drift. From the strong law of large numbers, we obtain that in this case, W drifts to $+\infty$ almost surely. We conclude that on $\{D_3 > D_1\}$, there exists almost surely some random $n_0 \in \mathbb{N}$ such that $W_i > 0$ for all $i \geq n_0$ (which means that from index n_0 on, we always compare to q first). Therefore, the following estimates hold on $\{D_3 > D_1\}$:

$$\frac{I_1^{(n)} - n_0}{n} \leq \frac{S_n^+}{n} \leq \frac{I_1^{(n)}}{n}.$$

Since both the term on the left-hand side and the term on the right-hand side converge to D_1 almost surely, we get that $\frac{S_n^+}{n}$ converges to D_1 almost surely on $\{D_3 > D_1\}$.

(2) If $d_1 > d_3$, the random walk W has a negative drift and, by the strong law of large numbers, drifts to $-\infty$ almost surely. Hence, on $\{D_1 > D_3\}$, there exists almost surely some random $n_0 \in \mathbb{N}$ such that $W_i \leq 0$ for all $i \geq n_0$. As a consequence, the following estimates hold on $\{D_1 > D_3\}$:

$$0 \leq \frac{S_n^+}{n} \leq \frac{n_0}{n}.$$

Hence, we obtain that $\frac{S_n^+}{n}$ converges to 0 almost surely on $\{D_3 > D_1\}$.

Finally, we conclude that

$$\frac{S_n^+}{n} = \mathbb{1}_{\{D_3>D_1\}} \frac{S_n^+}{n} + \mathbb{1}_{\{D_3<D_1\}} \frac{S_n^+}{n} + \mathbb{1}_{\{D_3=D_1\}} \frac{S_n^+}{n} \to \mathbb{1}_{\{D_3>D_1\}} D_1$$

almost surely as $n \to \infty$. The L_2-convergence again follows from the dominated convergence theorem. With the same arguments, we also obtain

$$\frac{M_n^+}{n} \xrightarrow{L_2} \mathbb{1}_{\{D_3>D_1\}} D_2 \quad \text{and} \quad \frac{L_n^+}{n} \xrightarrow{L_2} \mathbb{1}_{\{D_3>D_1\}} D_3. \qquad \square$$

It follows as $n \to \infty$

$$\frac{T_C(n)}{n} \xrightarrow{L_2} 1 + D_2 + D_3 + \mathbb{1}_{\{D_3>D_1\}} D_1 - \mathbb{1}_{\{D_3>D_1\}} D_3 = 1 + D_2 + \min\{D_1, D_3\}.$$

Remark: Intuitively, the limit is not very surprising. Every element needs to be compared at least once, each medium element has to be compared twice and the term $\min\{D_1, D_3\}$ contains the additional comparisons for the small and large elements which are compared to the "wrong" pivot first: If $D_3 > D_1$, almost all elements are compared to q first and we have additional comparisons for almost all small elements. If $D_1 > D_3$, on the contrary, almost all elements are compared to p first and we have additional comparisons for almost all large elements.

Altogether, we obtain as $n \to \infty$

$$(A_1^{(n)}, A_2^{(n)}, A_3^{(n)}, b_C^{(n)}) \xrightarrow{L_2} \Big(D_1, D_2, D_3, \frac{9}{5}\sum_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\}\Big).$$

(A2): The "spacings" D_1, D_2 and D_3 are identically distributed with distribution function $\mathbb{P}(D_1 \leq x) = \mathbb{P}(\min\{U_1, U_2\} \leq x) = 1 - (1-x)^2 = 2x - x^2$ for $x \in [0,1]$. Thus, D_1 has the density function $2(1-x)\mathbb{1}_{\{x\in[0,1]\}}$. We therefore have

$$\sum_{r=1}^{3} \mathbb{E}[A_r^2] = 3\ \mathbb{E}[D_1^2] = 3\int_0^1 2(1-x)x^2 \mathrm{d}x = \frac{1}{2} < 1.$$

(A3): For any $\ell \geq 0$ and $r \in \{1,2,3\}$, we have $(A_r^{(n)})^2 \leq 1$ and therefore, as $n \to \infty$,

$$\mathbb{E}\Big[\mathbb{1}_{\{I_r^{(n)}\leq \ell\}\cup\{I_r^{(n)}=n\}} (A_r^{(n)})^2\Big] \leq \mathbb{P}(I_r^{(n)} \leq \ell) \to 0.$$

We can therefore apply Theorem 6.2 and obtain the following result.

THEOREM 7.3. *The normalized number $\mathcal{C}_n^* = \frac{\mathcal{C}_n - \mathbb{E}[\mathcal{C}_n]}{n}$ of key comparisons when sorting a random permutation of $\{1,\dots,n\}$ with the dual-pivot Quicksort algorithm "Count" converges in distribution and with second moments to a random variable $\mathcal{C}^*$ whose distribution $\mathcal{L}(\mathcal{C}^*)$ is the (among all centered, square-integrable distributions) unique solution of the following equation:*

$$\mathcal{C}^* \stackrel{d}{=} \sum_{r=1}^{3} D_r \mathcal{C}^{*(r)} + \frac{9}{5}\sum_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\},$$

where $\mathcal{C}^{*(1)}$, $\mathcal{C}^{*(2)}$, $\mathcal{C}^{*(3)}$ and (D_1, D_2, D_3) are independent and $\mathcal{C}^{*(r)}$ is distributed as $\mathcal{C}^*$ for $r \in \{1,2,3\}$.

Considering that the convergence holds as well for the second moments of $\mathcal{C}_n^*$, we can asymptotically compute the variance $\mathrm{Var}(\mathcal{C}_n)$ of the number of key comparisons.

COROLLARY 7.4. *As $n \to \infty$, we have*

$$\mathrm{Var}(\mathcal{C}_n) \sim \sigma_C^2\, n^2,$$

where $\sigma_C^2 = \frac{1609}{300} - \frac{27}{50}\pi^2 + \frac{3}{10}\log(2) = 0.241691110\dots$.

Proof. By definition of $\mathcal{C}_n^*$, we have

$$\frac{1}{n^2}\,\mathrm{Var}(\mathcal{C}_n) = \frac{1}{n^2}\,\mathrm{Var}(n\mathcal{C}_n^* + \mathbb{E}[\mathcal{C}_n]) = \mathrm{Var}(\mathcal{C}_n^*) = \mathbb{E}\left[(\mathcal{C}_n^*)^2\right],$$

which converges to $\mathbb{E}[(\mathcal{C}^*)^2]$ as $n \to \infty$ by Theorem 7.3. In order to compute $\mathbb{E}[(\mathcal{C}^*)^2]$, we set $b_C := \frac{9}{5}\sum_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\}$ and obtain

$$\begin{aligned}
\mathbb{E}[(\mathcal{C}^*)^2] &= \mathbb{E}\left[\left(\sum_{r=1}^{3} D_r \mathcal{C}^{*(r)} + b_C\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{r=1}^{3} D_r \mathcal{C}^{*(r)}\right)^2\right] + 2\,\mathbb{E}\left[b_C \sum_{r=1}^{3} D_r \mathcal{C}^{*(r)}\right] + \mathbb{E}[b_C^2] \\
&= \mathbb{E}\left[\sum_{r=1}^{3} D_r^2 (\mathcal{C}^{*(r)})^2\right] + \mathbb{E}\left[\sum_{r\neq s} D_r \mathcal{C}^{*(r)} D_s \mathcal{C}^{*(s)}\right] + 2\sum_{r=1}^{3}\mathbb{E}[D_r b_C]\mathbb{E}[\mathcal{C}^{*(r)}] + \mathbb{E}[b_C^2] \\
&= \sum_{r=1}^{3}\mathbb{E}[D_r^2]\mathbb{E}[(\mathcal{C}^{*(r)})^2] + \mathbb{E}[b_C^2] \\
&= \frac{1}{2}\,\mathbb{E}[(\mathcal{C}^*)^2] + \mathbb{E}[b_C^2].
\end{aligned}$$

Solving for $\mathbb{E}[(\mathcal{C}^*)^2]$ and using (3) and a computer algebra system, we find

$$\begin{aligned}\mathbb{E}[(\mathcal{C}^*)^2] = 2\ \mathbb{E}[b_C^2] &= 2\ \mathbb{E}\left[\left(\frac{9}{5}\sum_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\}\right)^2\right]\\ &= \frac{1609}{300} - \frac{27}{50}\pi^2 + \frac{3}{10}\log(2) = 0.241691110\ldots,\end{aligned}$$

which provides the above result. $\square$

7.3 The Existence of a Smooth Density

In this section, we will show that the limit $\mathcal{C}^*$ of the normalized number $\mathcal{C}_n^*$ of key comparisons has a bounded and infinitely differentiable density function. To this end, we use a general theorem of Leckey [13] which is based on techniques of Fill and Janson [6]. Note that the limit $\mathcal{C}^*$ satisfies the following distributional equation:

$$\mathcal{C}^* \overset{d}{=} \sum_{r=1}^{\infty} A_r \mathcal{C}^{*(r)} + b_C,$$

where $\mathcal{C}^{*(1)}$, $\mathcal{C}^{*(2)}$, … and $(A_1, A_2, \ldots, b_C)$ are independent, $\mathcal{C}^{*(r)}$ is distributed as $\mathcal{C}^*$ for $r \geq 1$, $A_r = D_r$ for $r \in \{1,2,3\}$, $A_r = 0$ for $r \geq 4$ and $b_C = \frac{9}{5}\sum_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\}$. Following Definition 4.1 of [13] with $m = 1$, we denote by $\alpha^{\max}$ and $\alpha^{\sec}$ the two largest elements in $(A_r)_{r\geq 1}$ and say that conditions (B1)–(B5) hold if for all $r \geq 1$:

(B1) $\mathbb{P}(\alpha^{\max} \geq a) = 1$ for some constant $a > 0$,

(B2) $\mathbb{P}(\alpha^{\sec} \leq x) \leq \lambda x^{\nu}$ for some $\lambda, \nu > 0$ and all $x > 0$,

(B3) $\mathbb{P}(A_r \leq 1) = 1$,

(B4) $\mathbb{P}(\mathcal{C}^* = c) < 1$ for all $c \in \mathbb{R}$,

(B5) $\mathbb{P}\left(\sum\limits_{r=1}^{\infty} \mathbb{1}_{\{A_r \in (0,1)\}} \geq 1\right) > 0.$

Theorem 4.2 of Leckey [13] with $m = 1$ then implies:

THEOREM 7.5. *If conditions* (B1)–(B5) *hold, then $\mathcal{C}^*$ admits a bounded density function $f \in \mathcal{C}^\infty(\mathbb{R})$.*

As can easily be seen, the above conditions are satisfied:

(B1) We set, for example, $a = \frac{1}{3}$. As $D_1 + D_2 + D_3 = 1$, we obtain that

$$\mathbb{P}(\alpha^{\max} \geq a) = \mathbb{P}(\max\{D_1, D_2, D_3\} \geq \frac{1}{3}) = 1.$$

(B2) We set, for example, $\lambda = 12$ and $\nu = 2$. Since $D_1 + D_2 + D_3 = 1$, we also have $\min\{D_1, D_2, D_3\} + \alpha^{\text{sec}} + \max\{D_1, D_2, D_3\} = 1$ and obtain for $x \in (0, \frac{1}{2})$

$$\begin{aligned}\mathbb{P}(\alpha^{\text{sec}} \le x) &\le \mathbb{P}(\max\{D_1, D_2, D_3\} \ge 1 - 2x) \\ &= \mathbb{P}(\{D_1 \ge 1 - 2x\} \cup \{D_2 \ge 1 - 2x\} \cup \{D_3 \ge 1 - 2x\}) \\ &\le 3\ \mathbb{P}(D_1 \ge 1 - 2x) \\ &= 3\ \mathbb{P}(\min\{U_1, U_2\} \ge 1 - 2x) = 3(2x)^2 = 12x^2 = \lambda x^\nu.\end{aligned}$$

For $x \ge \frac{1}{2}$, the inequality holds trivially.

(B3) Obviously, we have $A_r \le 1$ for all $r \ge 1$.

(B4) For all $c \in \mathbb{R}$, we have $\mathbb{P}(\mathcal{C}^* = c) < 1$ since the variance $\operatorname{Var}(\mathcal{C}^*)$ is positive.

(B5) follows directly from the fact that D_1, D_2 and D_3 are in $(0, 1)$ almost surely.

Thus, Theorem 7.5 implies the following result.

THEOREM 7.6. *The limit $\mathcal{C}^*$ of the normalized number $\mathcal{C}_n^* = \frac{\mathcal{C}_n - \mathbb{E}[\mathcal{C}_n]}{n}$ of key comparisons in the dual-pivot Quicksort algorithm "Count" admits a bounded density function $f_C \in \mathcal{C}^\infty(\mathbb{R})$.*

8 Analysis of the Number of Swaps

We now come to the analysis of the number of swaps. Usually, the term *swap* describes the exchange of the values of two variables. Recall that in addition to the *swap*-operations (see lines 13, 19, 31, 33 and 34 in Algorithm 1), we also use *rotate3*-operations in order to move elements around (see line 28 in Algorithm 1).

```
1: procedure SWAP($x$, $y$)
2:     tmp ← $x$
3:     $x$ ← $y$
4:     $y$ ← tmp
```

```
1: procedure ROTATE3($x$, $y$, $z$)
2:     tmp ← $x$
3:     $x$ ← $y$
4:     $y$ ← $z$
5:     $z$ ← tmp
```

This raises the question of how we should count the number of *rotate3*-operations in terms of the unit swaps. As Wild suggested in his Ph.D. thesis [23, Section 3.2.2], we could consider a different unit of cost—the number of *write accesses* to the array—instead of counting the number of swaps. One swap exactly needs two write accesses whereas a *rotate3*-operation needs three write accesses to the array. In the following, we will therefore count a *rotate3*-operation as 1.5 swaps.

Remark: Instead of using the *rotate3*-operation in line 28 of Algorithm 1, we could also use two swap-operations. However, this would increase the number of write accesses to the array since two swap-operations need exactly four write accesses.

For any $n \geq 0$, let $T_S(n)$ denote the number of swaps during the first partitioning step and let $\mathcal{S}_n$ denote the total number of swaps in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of $\{1, \ldots, n\}$.

8.1 The Average Number of Swaps

For $n \geq 2$, the number $T_S(n)$ of swaps during the first partitioning step includes

- ◇ $I_1^{(n)} + \frac{1}{2} S_n^+$ swaps for the small elements (since there is one swap for each small element compared to p first and a *rotate3*-operation, i.e. 1.5 swaps, for each small element compared to q first),
- ◇ $I_3^{(n)} - L_n^+$ swaps for the large elements compared to p first,
- ◇ M_n^+ swaps for the medium elements compared to q first and
- ◇ two swaps at the end in order to bring the pivots to their final positions (both line 33 and line 34 of Algorithm 1 need two write accesses to the array).

Adding this up, we obtain

$$T_S(n) = 2 + I_1^{(n)} + I_3^{(n)} + \frac{1}{2} S_n^+ + M_n^+ - L_n^+. \tag{5}$$

It therefore follows from Lemma 5.7, Lemma 5.8 and Lemma 6.3 that, for $n \geq 2$, the expected number of swaps during the first partitioning step is

$$\begin{aligned} \mathbb{E}[T_S(n)] &= 2 + \mathbb{E}[I_1^{(n)}] + \mathbb{E}[I_3^{(n)}] + \mathbb{E}[S_n^+ + M_n^+ - L_n^+] - \frac{1}{2}\mathbb{E}[S_n^+] \\ &= 2 + \frac{n-2}{3} + \frac{n-2}{3} + 0 - \frac{1}{2}\left(\frac{1}{12}n - \frac{7}{24} + \frac{1}{8(n - \mathbb{1}_{\{n \text{ even}\}})}\right) \\ &= \frac{5}{8}n + \frac{13}{16} - \frac{1}{16(n - \mathbb{1}_{\{n \text{ even}\}})}. \end{aligned} \tag{6}$$

We can now derive an exact expression for the expected number of swaps in the Quicksort algorithm "Count":

THEOREM 8.1. *For $n \geq 4$, the average number of swaps in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of $\{1, \ldots, n\}$ is*

$$\begin{aligned} \mathbb{E}[\mathcal{S}_n] =& \frac{3}{4}n\mathcal{H}_n + \frac{1}{20}n\mathcal{H}_n^{alt} - \frac{4}{5}n + \frac{3}{4}\mathcal{H}_n + \frac{1}{20}\mathcal{H}_n^{alt} - \frac{23}{160} - \frac{1}{40}(-1)^n \\ & - \frac{\mathbb{1}_{\{neven\}}}{320}\left(\frac{3}{n-1} + \frac{1}{n-3}\right) + \frac{\mathbb{1}_{\{nodd\}}}{320}\left(\frac{1}{n} + \frac{3}{n-2}\right). \end{aligned}$$

Proof. Recall that the expected number $\mathbb{E}[\mathcal{S}_n]$ of swaps for the sorting procedure and the expected number $\mathbb{E}[T_S(n)]$ of swaps during the first partitioning step are linked by the following equality for $n \geq 2$:

$$\mathbb{E}[\mathcal{S}_n] = \frac{6}{n(n-1)} \sum_{k=0}^{n-2} (n-k-1)\,\mathbb{E}[\mathcal{S}_k] + \mathbb{E}[T_S(n)].$$

Thus, we have to solve the above recurrence with $\mathbb{E}[S_0] = \mathbb{E}[S_1] = 0$. In his master's thesis [22, Section 4.2.1], Wild presents an elementary derivation of the closed form of $\mathbb{E}[\mathcal{S}_n]$ and obtains the following result for $n \geq 4$:

$$\begin{aligned}\mathbb{E}[\mathcal{S}_n] =& \frac{1}{\binom{n}{4}} \sum_{i=5}^{n} \binom{i}{4} \sum_{j=3}^{i-2} \left(\mathbb{E}[T_S(j+2)] - \frac{2j}{j+2}\mathbb{E}[T_S(j+1)] + \frac{j(j-1)}{(j+2)(j+1)}\mathbb{E}[T_S(j)]\right) \\ &+ \frac{n+1}{5}\left(\mathbb{E}[T_S(4)] + \frac{1}{2}\mathbb{E}[T_S(2)]\right). \end{aligned} \tag{7}$$

Note that $\mathbb{E}[\mathcal{S}_n]$ is linear in $\mathbb{E}[T_S(n)]$, such that we may compute the contributions for each summand of $\mathbb{E}[T_S(n)]$ separately. Using the representation (6), we have that $\mathbb{E}[T_S(n)]$ has the form $an + b - \frac{1}{16}\frac{1}{n-\mathbb{1}_{\{n \text{ even}\}}}$ (with $a = \frac{5}{8}$ and $b = \frac{13}{16}$). Due to linearity, we have that $\mathbb{E}[\mathcal{S}_n] = \mathbb{E}[\mathcal{S}_n^{(1)}] - \frac{1}{16}\mathbb{E}[\mathcal{S}_n^{(2)}]$, where $\mathbb{E}[\mathcal{S}_n^{(1)}]$ and $\mathbb{E}[\mathcal{S}_n^{(2)}]$ denote the contributions of the terms $an + b$ and $\frac{1}{n-\mathbb{1}_{\{n \text{ even}\}}}$ to $\mathbb{E}[\mathcal{S}_n]$, respectively. The case that the expected partitioning costs have the form $an+b$ has already been analyzed in Section 4.2.1.1 of Wild [22] and, for $n \geq 4$, we obtain

$$\mathbb{E}[\mathcal{S}_n^{(1)}] = \frac{3}{4}n\mathcal{H}_n - \frac{33}{40}n + \frac{3}{4}\mathcal{H}_n - \frac{27}{160}. \tag{8}$$

In order to compute the contribution $\mathbb{E}[\mathcal{S}_n^{(2)}]$ of the non-linear part, we set $\mu_n := \frac{1}{n-\mathbb{1}_{\{n \text{ even}\}}}$ for $n \geq 2$. Considering (7), we have to compute the double sum

$$\frac{1}{\binom{n}{4}} \sum_{i=5}^{n} \binom{i}{4} \sum_{j=3}^{i-2} \left(\mu_{j+2} - \frac{2j}{j+2}\mu_{j+1} + \frac{j(j-1)}{(j+2)(j+1)}\mu_j\right).$$

Plugging in the values of μ_j, μ_{j+1} and μ_{j+2} yields for $i \geq 5$

$$\begin{aligned}\sum_{j=3}^{i-2} \left(\mu_{j+2} - \frac{2j}{j+2}\mu_{j+1} + \frac{j(j-1)}{(j+2)(j+1)}\mu_j\right) &= \sum_{j=3}^{i-2} \frac{(-1)^j}{\binom{j+2}{2}} \\ &= -4\mathcal{H}_i^{\text{alt}} + \frac{2(-1)^i}{i} - \frac{17}{6}, \end{aligned}$$

where the last equality follows easily by induction on i. Furthermore, we get for $n \geq 4$ (e.g.

by induction on n) that

$$\frac{1}{\binom{n}{4}}\sum_{i=5}^{n}\binom{i}{4}\left(-4\mathcal{H}_i^{\mathrm{alt}}+\frac{2(-1)^i}{i}-\frac{17}{6}\right)=-\frac{17}{30}n-\frac{4}{5}n\mathcal{H}_n^{\mathrm{alt}}-\frac{4}{5}\mathcal{H}_n^{\mathrm{alt}}-\frac{17}{30}+\frac{2}{5}(-1)^n$$
$$+\frac{\mathbb{1}_{\{n \mathrm{even}\}}}{20}\left(\frac{3}{n-1}+\frac{1}{n-3}\right)-\frac{\mathbb{1}_{\{n \mathrm{odd}\}}}{20}\left(\frac{1}{n}+\frac{3}{n-2}\right).$$

Using (7), we obtain

$$\begin{aligned}\mathbb{E}[\mathcal{S}_n^{(2)}]&=\frac{1}{\binom{n}{4}}\sum_{i=5}^{n}\binom{i}{4}\sum_{j=3}^{i-2}\left(\mu_{j+2}-\frac{2j}{j+2}\mu_{j+1}+\frac{j(j-1)}{(j+2)(j+1)}\mu_j\right)+\frac{n+1}{5}\left(\mu_4+\frac{1}{2}\mu_2\right)\\&=-\frac{2}{5}n-\frac{4}{5}n\mathcal{H}_n^{\mathrm{alt}}-\frac{2}{5}-\frac{4}{5}\mathcal{H}_n^{\mathrm{alt}}+\frac{2}{5}(-1)^n\\&\quad+\frac{\mathbb{1}_{\{n \mathrm{even}\}}}{20}\left(\frac{3}{n-1}+\frac{1}{n-3}\right)-\frac{\mathbb{1}_{\{n \mathrm{odd}\}}}{20}\left(\frac{1}{n}+\frac{3}{n-2}\right).\end{aligned}\tag{9}$$

Considering that $\mathbb{E}[\mathcal{S}_n]=\mathbb{E}[\mathcal{S}_n^{(1)}]-\frac{1}{16}\mathbb{E}[\mathcal{S}_n^{(2)}]$, the claim follows from (8) and (9). ☐

An alternative proof of Theorem 8.1 uses generating functions and can be found in the appendix. From the preceding theorem, we can now derive an asymptotic expansion of the expected number of swaps.

COROLLARY 8.2. *The average number of swaps in the dual-pivot Quicksort algorithm "Count" when sorting a random permutation of* $\{1,\dots,n\}$ *is*

$$\mathbb{E}[\mathcal{S}_n]=\frac{3}{4}n\log(n)+An+\frac{3}{4}\log(n)+\mathrm{O}(1),$$

where $A=-\frac{4}{5}+\frac{3}{4}\gamma-\frac{1}{20}\log(2)=-0.40174561\dots$ *and* $n\to\infty$.

Proof. The result follows directly from Theorem 8.1 by plugging in the following asymptotic expansions (see Lemma 8.1 of Aumüller et. al [2] and Section 9 of Graham, Knuth and Patashnik [7]):

$$\mathcal{H}_n^{\mathrm{alt}}=-\log(2)+\mathrm{O}\left(\frac{1}{n}\right),$$
$$\mathcal{H}_n=\log(n)+\gamma+\mathrm{O}\left(\frac{1}{n}\right).$$ ☐

8.2 Distributional Analysis of the Number of Swaps

As already shown, the total number $\mathcal{S}_n$ of swaps satisfies the following distributional recursion for $n\geq 2$:

$$\mathcal{S}_n\overset{d}{=}\sum_{r=1}^{3}\mathcal{S}^{(r)}_{I_r^{(n)}}+T_S(n),$$

where $(\mathcal{S}_j^{(1)})_{0\le j\le n}$, $(\mathcal{S}_j^{(2)})_{0\le j\le n}$, $(\mathcal{S}_j^{(3)})_{0\le j\le n}$ and $(T_S(n), I^{(n)})$ are independent and $\mathcal{S}_j^{(r)}$ is distributed as $\mathcal{S}_j$ for $r\in\{1,2,3\}$ and $j\ge 0$. We now define the normalized number of swaps by $\mathcal{S}_0^*=0$ and

$$\mathcal{S}_n^* = \frac{\mathcal{S}_n - \mathbb{E}[\mathcal{S}_n]}{n} \qquad \text{for } n\ge 1.$$

Then, $(\mathcal{S}_n^*)_{n\ge 0}$ is a sequence of centered, square-integrable random variables with

$$\mathcal{S}_n^* \overset{d}{=} \sum_{r=1}^{3} A_r^{(n)} \mathcal{S}^{*(r)}_{I_r^{(n)}} + b_S^{(n)}, \quad n\ge 2,$$

where $(\mathcal{S}_j^{*(1)})_{0\le j\le n}$, $(\mathcal{S}_j^{*(2)})_{0\le j\le n}$, $(\mathcal{S}_j^{*(3)})_{0\le j\le n}$ and $(b_S^{(n)}, I^{(n)})$ are independent, $\mathcal{S}_j^{*(r)}$ is distributed as $\mathcal{S}_j^*$ for $r\in\{1,2,3\}$, $j\ge 0$ and

$$A_r^{(n)} = \frac{I_r^{(n)}}{n} \quad\text{and}\quad b_S^{(n)} = \frac{1}{n}\left(T_S(n) - \mathbb{E}[\mathcal{S}_n] + \sum_{r=1}^{3}\mathbb{E}[\mathcal{S}_{I_r^{(n)}} | I_r^{(n)}]\right).$$

Just as in the previous section, we check the following three conditions in order to apply Theorem 6.2:

(A1) $(A_1^{(n)}, A_2^{(n)}, A_3^{(n)}, b_S^{(n)}) \xrightarrow{\ell_2} (A_1, A_2, A_3, b_S)$ as $n\to\infty$.

(A2) $\sum\limits_{r=1}^{3} \mathbb{E}[A_r^2] < 1$.

(A3) $\mathbb{E}\Big[\mathbb{1}_{\{I_r^{(n)}\le \ell\}\cup\{I_r^{(n)}=n\}} (A_r^{(n)})^2\Big] \to 0$ for $\ell\ge 0$, $r\in\{1,2,3\}$ and $n\to\infty$.

(A1): We already know (see Lemma 6.4) that $A_r^{(n)} = \frac{I_r^{(n)}}{n}$ converges to $A_r := D_r$ in L_2 for $r\in\{1,2,3\}$ and as $n\to\infty$. Furthermore, according to Lemma 6.6 and Corollary 8.2, as $n\to\infty$, we have

$$\frac{1}{n}\left(-\mathbb{E}[\mathcal{S}_n] + \sum_{r=1}^{3}\mathbb{E}[\mathcal{S}_{I_r^{(n)}} | I_r^{(n)}]\right) \xrightarrow{L_2} \frac{3}{4}\sum_{r=1}^{3} D_r \log(D_r).$$

It therefore remains to show the convergence of $\frac{1}{n}T_S(n)$. Using representation (5), we have

$$\frac{T_S(n)}{n} = \frac{2}{n} + \frac{I_1^{(n)}}{n} + \frac{I_3^{(n)}}{n} + \frac{S_n^+}{2n} + \frac{M_n^+}{n} - \frac{L_n^+}{n}.$$

Lemma 6.4 and Lemma 7.2 now imply as $n\to\infty$:

$$\begin{aligned}\frac{T_S(n)}{n} &\xrightarrow{L_2} D_1 + D_3 + \mathbb{1}_{\{D_3>D_1\}}\Big(\frac{1}{2}D_1 + D_2 - D_3\Big)\\ &= \mathbb{1}_{\{D_3>D_1\}}\Big(\frac{3}{2}D_1 + D_2\Big) + \mathbb{1}_{\{D_3\le D_1\}}(D_1 + D_3).\end{aligned}$$

Remark: Once again, the limit is not very surprising: If $D_3 > D_1$, almost all elements are compared to q first and thus, we need one swap for almost every medium element and a *rotate3*-operation (1.5 swaps) for almost every small element. If $D_3 < D_1$, on the contrary, almost all elements are compared to p first and we have one swap for almost every small or large element.

The conditions **(A2)** and **(A3)** follow just as in the previous section. Theorem 6.2 then implies the following result.

THEOREM 8.3. *The normalized number $\mathcal{S}_n^* = \frac{\mathcal{S}_n - \mathbb{E}[\mathcal{S}_n]}{n}$ of swaps when sorting a random permutation of $\{1, \dots, n\}$ with the dual-pivot Quicksort algorithm "Count" converges in distribution and with second moments to a random variable $\mathcal{S}^*$ whose distribution $\mathcal{L}(\mathcal{S}^*)$ is the (among all centered, square-integrable distributions) unique solution of the following equation:*

$$\mathcal{S}^* \overset{d}{=} \sum_{r=1}^{3} D_r \mathcal{S}^{*(r)} + \frac{3}{4}\sum_{r=1}^{3} D_r \log(D_r) + D_1 + D_3 + \mathbb{1}_{\{D_3 > D_1\}}\Big(\frac{1}{2}D_1 + D_2 - D_3\Big),$$

where $\mathcal{S}^{*(1)}$, $\mathcal{S}^{*(2)}$, $\mathcal{S}^{*(3)}$ and (D_1, D_2, D_3) are independent and $\mathcal{S}^{*(r)}$ is distributed as $\mathcal{S}^*$ for $r \in \{1, 2, 3\}$.

Just as in the previous section, we can asymptotically compute the variance of the number $\mathcal{S}_n$ of swaps.

COROLLARY 8.4. *As $n \to \infty$, we have*

$$\operatorname{Var}(\mathcal{S}_n) \sim \sigma_S^2 \, n^2,$$

where $\sigma_S^2 = \frac{47}{48} - \frac{3}{32}\pi^2 + \frac{3}{32}\log(2) = 0.118873802\ldots$.

Proof. With $b_S := 0.75 \sum_{r=1}^{3} D_r \log(D_r) + D_1 + D_3 + \mathbb{1}_{\{D_3 > D_1\}}(\frac{1}{2}D_1 + D_2 - D_3)$, we have

$$\begin{aligned}\mathbb{E}[(\mathcal{S}^*)^2] = 2\ \mathbb{E}[b_S^2] &= 2\ \mathbb{E}\left[\left(\frac{3}{4}\sum_{r=1}^{3} D_r \log(D_r) + D_1 + D_3 + \mathbb{1}_{\{D_3 > D_1\}}\Big(\frac{1}{2}D_1 + D_2 - D_3\Big)\right)^2\right] \\ &= \frac{47}{48} - \frac{3}{32}\pi^2 + \frac{3}{32}\log(2) = 0.118873802\ldots.\end{aligned}$$

In combination with Theorem 8.3, the claim follows. □

Just as in Section 7.3, we obtain the existence of a smooth density function of the limit $\mathcal{S}^*$.

THEOREM 8.5. *The limit $\mathcal{S}^*$ of the normalized number $\mathcal{S}_n^* = \frac{\mathcal{S}_n - \mathbb{E}[\mathcal{S}_n]}{n}$ of swaps in the dual-pivot Quicksort algorithm "Count" admits a bounded density function $f_S \in \mathcal{C}^\infty(\mathbb{R})$.*

Proof. The existence of a smooth density follows directly from Theorem 7.5 as conditions (B1)–(B5) are satisfied (see Section 7.3). □

9 The Correlation between Key Comparisons and Swaps

In order to analyze the correlation between the number of key comparisons and the number of swaps, we will use the multivariate version of Theorem 6.2. We set

$$X_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{and} \quad X_n = \begin{pmatrix} \mathcal{C}_n^* \\ \mathcal{S}_n^* \end{pmatrix} \quad \text{for } n \geq 1,$$

where $\mathcal{C}_n^* = \frac{\mathcal{C}_n - \mathbb{E}[\mathcal{C}_n]}{n}$ is the normalized number of key comparisons and $\mathcal{S}_n^* = \frac{\mathcal{S}_n - \mathbb{E}[\mathcal{S}_n]}{n}$ the normalized number of swaps. Thus, $(X_n)_{n\geq 0}$ is a sequence of centered, square-integrable random variables in $\mathbb{R}^2$ satisfying the distributional recursion

$$X_n \stackrel{d}{=} \sum_{r=1}^{3} A_r^{(n)} X^{(r)}_{I_r^{(n)}} + b_X^{(n)}$$

for $n \geq 2$, where $(X_j^{(1)})_{0\leq j\leq n}$, $(X_j^{(2)})_{0\leq j\leq n}$, $(X_j^{(3)})_{0\leq j\leq n}$ and $(b_X^{(n)}, I^{(n)})$ are independent, $X_j^{(r)}$ is distributed as X_j for $r \in \{1,2,3\}$, $j \geq 0$ and

$$A_r^{(n)} = \frac{1}{n}\begin{pmatrix} I_r^{(n)} & 0 \\ 0 & I_r^{(n)} \end{pmatrix} \quad \text{and} \quad b_X^{(n)} = \frac{1}{n}\begin{pmatrix} T_C(n) - \mathbb{E}[\mathcal{C}_n] + \sum_{r=1}^{3} \mathbb{E}[\mathcal{C}_{I_r^{(n)}} | I_r^{(n)}] \\ T_S(n) - \mathbb{E}[\mathcal{S}_n] + \sum_{r=1}^{3} \mathbb{E}[\mathcal{S}_{I_r^{(n)}} | I_r^{(n)}] \end{pmatrix}.$$

We again check the three conditions (A1)–(A3) of Theorem 6.2.

(A1): The results of the previous two sections directly imply the convergence

$$(A_1^{(n)}, A_2^{(n)}, A_3^{(n)}, b_X^{(n)}) \xrightarrow{L_2} (A_1, A_2, A_3, b_X)$$

as $n \to \infty$, with $A_r = \begin{pmatrix} D_r & 0 \\ 0 & D_r \end{pmatrix}$ for $r \in \{1,2,3\}$ and

$$b_X = \begin{pmatrix} b_C \\ b_S \end{pmatrix} = \begin{pmatrix} \frac{9}{5}\sum_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\} \\ \frac{3}{4}\sum_{r=1}^{3} D_r \log(D_r) + D_1 + D_3 + \mathbb{1}_{\{D_3 > D_1\}}(\frac{1}{2}D_1 + D_2 - D_3) \end{pmatrix}.$$

(A2): We have $\left\|A_r^t A_r\right\|_{\text{op}} = D_r^2$ for $r \in \{1,2,3\}$ and therefore

$$\sum_{r=1}^{3} \mathbb{E}\left[\left\|A_r^t A_r\right\|_{\text{op}}\right] = \sum_{r=1}^{3} \mathbb{E}[D_r^2] = \frac{1}{2} < 1.$$

Condition **(A3)** follows just as before. Using Theorem 6.2, we get the following result.

THEOREM 9.1. *The normalized bivariate random vector X_n converges in distribution and with second mixed moments to a random variable X whose distribution $\mathcal{L}(X)$ is the (among all centered, square-integrable distributions) unique solution of the following equation:*

$$X \stackrel{d}{=} \sum_{r=1}^{3} \begin{pmatrix} D_r & 0 \\ 0 & D_r \end{pmatrix} X^{(r)} + \begin{pmatrix} \frac{9}{5} \sum\limits_{r=1}^{3} D_r \log(D_r) + 1 + D_2 + \min\{D_1, D_3\} \\ \frac{3}{4} \sum\limits_{r=1}^{3} D_r \log(D_r) + D_1 + D_3 + \mathbb{1}_{\{D_3 > D_1\}}(\frac{1}{2}D_1 + D_2 - D_3) \end{pmatrix},$$

where $X^{(1)}$, $X^{(2)}$, $X^{(3)}$ and (D_1, D_2, D_3) are independent and $X^{(r)}$ is distributed as X for $r \in \{1, 2, 3\}$.

We can now analyze the asymptotic covariance $\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n)$ between the number of key comparisons and swaps.

COROLLARY 9.2. *As $n \to \infty$, we have*

$$\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n) \sim \sigma_{C,S}\, n^2,$$

where $\sigma_{C,S} = \frac{43}{20} - \frac{9}{40}\pi^2 + \frac{7}{40}\log(2) = 0.0506397663\ldots$. Moreover, concerning the correlation coefficient between $\mathcal{C}_n$ and $\mathcal{S}_n$, as $n \to \infty$, we obtain

$$\mathrm{Corr}(\mathcal{C}_n, \mathcal{S}_n) = \frac{\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n)}{\sqrt{\mathrm{Var}(\mathcal{C}_n)}\sqrt{\mathrm{Var}(\mathcal{S}_n)}} \sim 0.298755\ldots.$$

Proof. By Theorem 9.1, we have as $n \to \infty$

$$\begin{aligned} \frac{1}{n^2}\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n) &= \frac{1}{n^2}\mathrm{Cov}(n\mathcal{C}_n^* + \mathbb{E}[\mathcal{C}_n], n\mathcal{S}_n^* + \mathbb{E}[\mathcal{S}_n]) = \mathrm{Cov}(\mathcal{C}_n^*, \mathcal{S}_n^*) \\ &= \mathbb{E}[\mathcal{C}_n^* \mathcal{S}_n^*] \to \mathbb{E}[X_1 X_2], \end{aligned}$$

where X_1 and X_2 denote the first and second component of the limit X from Theorem 9.1. From the preceding theorem, using that $X^{(1)}$, $X^{(2)}$, $X^{(3)}$ and (D_1, D_2, D_3) are independent and that $X^{(1)}$, $X^{(2)}$ and $X^{(3)}$ are centered, we get

$$\begin{aligned} \mathbb{E}[X_1 X_2] &= \mathbb{E}\left[\left(\sum_{r=1}^{3} D_r X_1^{(r)} + b_C\right)\left(\sum_{r=1}^{3} D_r X_2^{(r)} + b_S\right)\right] \\ &= \mathbb{E}\left[\sum_{r=1}^{3} D_r^2 X_1^{(r)} X_2^{(r)}\right] + \mathbb{E}[b_C\, b_S] \\ &= \frac{1}{2}\,\mathbb{E}[X_1 X_2] + \mathbb{E}[b_C\, b_S]. \end{aligned}$$

Solving for $\mathbb{E}[X_1X_2]$ and using (3) and a computer algebra system, we obtain

$$\mathbb{E}[X_1X_2] = 2\ \mathbb{E}\left[b_C\, b_S\right] = \frac{43}{20} - \frac{9}{40}\pi^2 + \frac{7}{40}\log(2) = 0.0506397663\ldots.$$

The rest follows by Corollary 7.4 and Corollary 8.4. □

Remark: The computed asymptotic variances and correlations hold as well for the version of "Count" Quicksort given in Aumüller et al. [2]. However, if we had replaced the *rotate3*-operation in line 28 of Algorithm 1 by two swap-operations, we would have obtained slightly different results. To be more precise, we would have had

$$T_S(n) = 2 + I_1^{(n)} + I_3^{(n)} + S_n^+ + M_n^+ - L_n^+.$$

Thus, due to Lemma 5.7, the expected number of swaps during the first partitioning step would have been

$$\mathbb{E}[T_S(n)] = 2 + \frac{n-2}{3} + \frac{n-2}{3} + 0 = \frac{2}{3}n + \frac{2}{3}.$$

Using Wild's results [22, Section 4.2.1.1], we would have obtained that the average number of swaps in the (modified) Quicksort algorithm "Count" when sorting an input sequence of length n is

$$\mathbb{E}[\mathcal{S}_n] = 0.8n\log(n) - An + 0.8\log(n) + \mathcal{O}(1),$$

where $A = -\frac{24}{25} + \frac{4}{5}\gamma = -0.498227\ldots$ and $n \to \infty$. Furthermore, we would have had

$$\frac{T_S(n)}{n} \xrightarrow{L_2} D_1 + \min\{D_1, D_3\} + \mathbb{1}_{\{D_3 > D_1\}} D_2$$

as $n \to \infty$ and we would have had the following asymptotic results: $\mathrm{Var}(\mathcal{S}_n) \sim 0.11966\ldots n^2$, $\mathrm{Cov}(\mathcal{C}_n, \mathcal{S}_n) \sim 0.044810\ldots n^2$ and $\mathrm{Corr}(\mathcal{C}_n, \mathcal{S}_n) \sim 0.26349\ldots$.

10 Conclusion

The main results of this work are summarized in Table 1. For the sake of comparison, Table 1 also contains the corresponding results about classic Quicksort and YBB Quicksort.

cost measure	error	**Classic Quicksort**	**YBB Quicksort**	**“Count” Quicksort**
Comparisons				
expectation	$\mathcal{O}(\log(n))$	$2n\log(n) - 1.51223n^{*}$	$1.9n\log(n) - 2.45829n^{\S}$	$1.8n\log(n) - 2.38238n^{\|}$
std. dev.	$o(n)$	$0.648277n^{\dagger}$	$0.508930n^{\P}$	$0.491620n$
Swaps				
expectation	$\mathcal{O}(\log(n))$	$\frac{1}{3}n\log(n) + 0.08129n^{*}$	$0.6n\log(n) - 0.12367n^{\S}$	$0.75n\log(n) - 0.40174n$
std. dev.	$o(n)$	$0.023725n^{\dagger}$	$0.328365n^{\P}$	$0.344780n$
Correlation				
Coefficient	$o(1)$	$-0.86404^{\ddagger}$	$-0.05121^{\P}$	0.29875

Table 1: Summary of the results: asymptotic means, standard deviations and correlation coefficients for the number of key comparisons and the number of swaps.

As already analyzed by Wild and Nebel [24] and Aumüller and Dietzfelbinger [1], respectively, YBB Quicksort needs 5 % and “Count” Quicksort 10 % less key comparisons than classic Quicksort in the asymptotic average. The asymptotic standard deviation of the number of key comparisons when using YBB Quicksort or “Count” Quicksort is more than 20 % smaller than the corresponding asymptotic standard deviation in classic Quicksort.

However, when considering the number of swaps, it is just the other way around: YBB Quicksort needs 80 % more swaps and “Count” Quicksort even 125 % more swaps than classic Quicksort in the asymptotic average. While the number of swaps in classic Quicksort is highly concentrated around its mean, the asymptotic standard deviation when using YBB Quicksort or “Count” Quicksort is much higher.

Regarding the asymptotic correlation coefficient between key comparisons and swaps, it is striking that the number of comparisons and the number of swaps are positively correlated in “Count” Quicksort, whereas they are almost perfectly negatively correlated in classic Quicksort. Wild, Nebel and Neininger [25] give the following explanation for the negative correlation in classic Quicksort: “*A “good” run w. r. t. comparisons needs balanced partitioning, but the more balanced partitioning becomes, the higher is the potential for misplaced elements that need to be moved.*” (Wild, Nebel and Neininger [25], p. 24).

* see e.g. Sedgewick [19, p. 334] or Table 1 of Wild [22].
† see e.g. Hennequin [8, p. 330] or Section 3.3.3 of Wild [22].
‡ see Table 1 of Neininger [16].
§ see Theorems 3.8 and 3.9 of Wild, Nebel and Neininger [25] with $M = 1$.
¶ see Theorems 4.3, 4.4 and 4.6 of Wild, Nebel and Neininger [25].
$^{\|}$ see Corollary 12.2 of Aumüller et al. [2].

In order to empirically validate the obtained results, we performed some simulation studies and compared the asymptotic results with sample means, sample standard deviations and sample correlation coefficients. We use the “R” implementation given in Appendix A.3 and 500 random permutations of $\{1, \ldots, n\}$ for each $n \in \{0.5 \cdot 10^4, 10^4, 1.5 \cdot 10^4, 2 \cdot 10^4, 2.5 \cdot 10^4, 3 \cdot 10^4\}$. Figure 6 shows that the sample means are very close to the asymptotic expressions, whereas the sample standard deviations and correlations in Figure 7 are more irregular. Nevertheless, the depicted results match our computed values from Table 1.

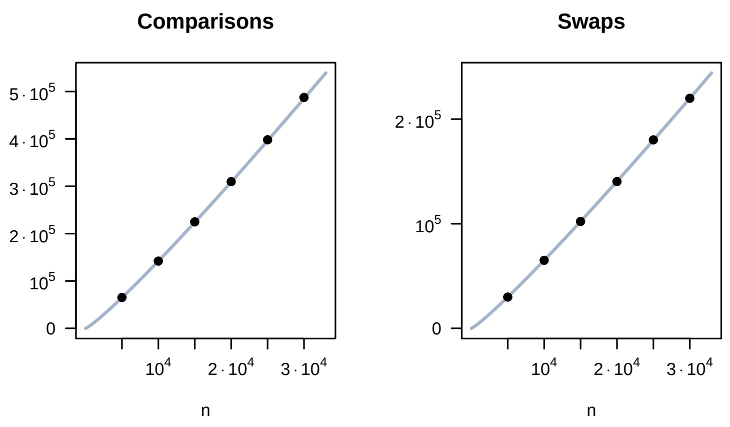


FIGURE 6: Comparison of sample means of $\mathcal{C}_n$ and $\mathcal{S}_n$ with the asymptotic results of Table 1 (the asymptotic results are shown in blue). For each n, the input sample consists of 500 random permutations.

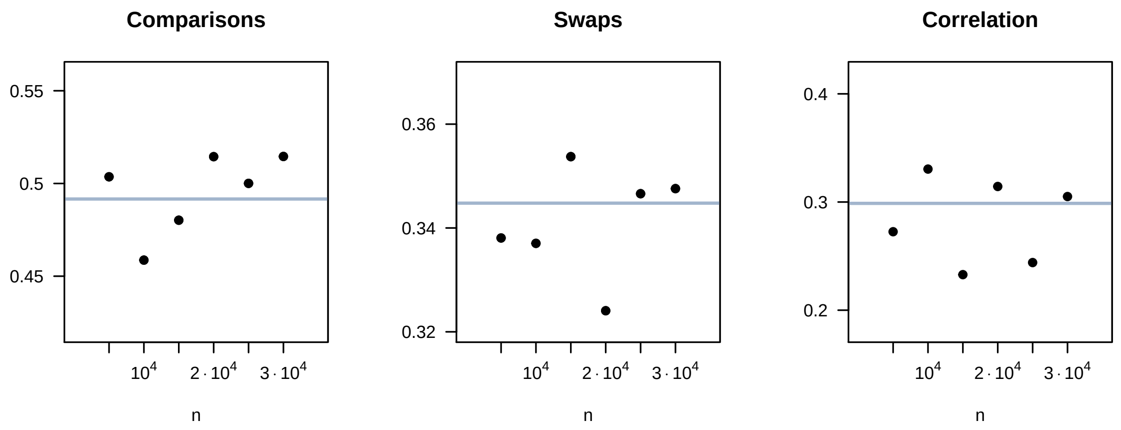


FIGURE 7: The two images on the left compare sample standard deviations of $\mathcal{C}_n^*$ and $\mathcal{S}_n^*$ with the asymptotic results of Table 1. The rightmost image compares sample correlations and the asymptotic correlation coefficient from Table 1. For each n, the input sample consists of 500 random permutations. The asymptotic results are shown in blue.

In the previous sections, we have seen that both the normalized number $\mathcal{C}_n^*$ of key comparisons and the normalized number $\mathcal{S}_n^*$ of swaps converge in distribution to random variables with bounded and smooth density functions. To get an idea of how these density functions may look like, Figure 8 shows histograms representing the relative frequencies of the values of $\mathcal{C}_n^*$ and $\mathcal{S}_n^*$ for a sample of 1000 random permutations of size $n = 10000$. In order to examine the relation between the number of comparisons and the number of swaps, Figure 8 additionally shows a scatter plot: Each of the 1000 datapoints is displayed as a dot, where the normalized number of key comparisons determines the position on the horizontal axis and the normalized number of swaps determines the position on the vertical axis. In accordance with our results of Section 9, the scatter plot suggests a positive correlation between the number of key comparisons and the number of swaps. For comparison, Figure 9 shows the corresponding results when sorting with classic Quicksort (Algorithm 2).

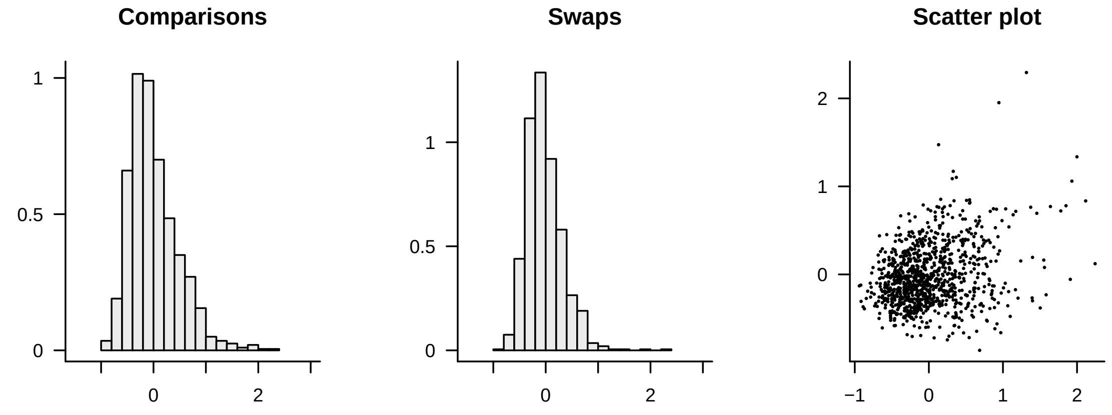


FIGURE 8: Histograms for the distributions of $\mathcal{C}_n^*$ and $\mathcal{S}_n^*$ and a scatter plot (plotting $\mathcal{S}_n^*$ against $\mathcal{C}_n^*$) when sorting with the dual-pivot Quicksort algorithm "Count". The input sample consists of 1000 random permutations of size $n = 10000$.

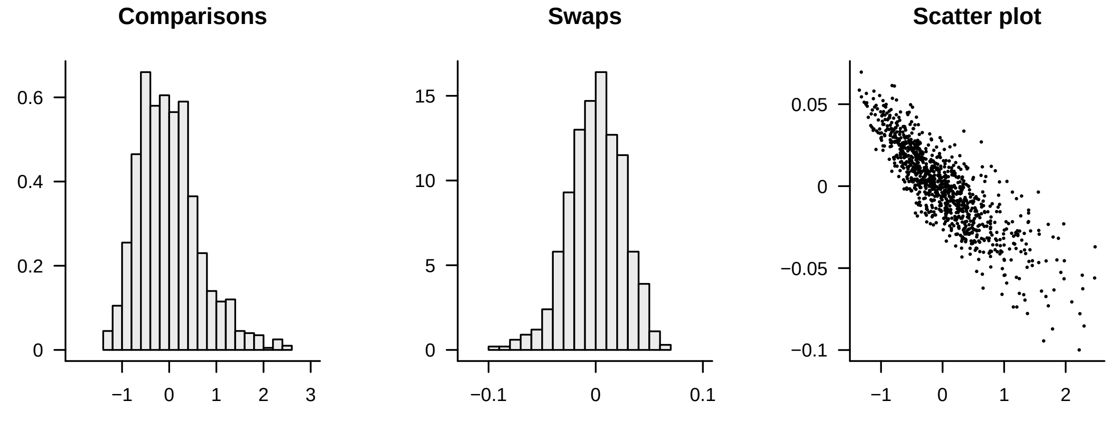


FIGURE 9: Histograms for the distributions of $\mathcal{C}_n^*$ and $\mathcal{S}_n^*$ and a scatter plot (plotting $\mathcal{S}_n^*$ against $\mathcal{C}_n^*$) when sorting with the classic Quicksort algorithm. The input sample consists of 1000 random permutations of size $n = 10000$.

A Appendix

A.1 An Alternative Proof of Theorem 8.1

Recall that the average number $\mathbb{E}[\mathcal{S}_n]$ of swaps and the average number $\mathbb{E}[T_S(n)]$ of swaps during the first partitioning step are linked by the recurrence

$$\mathbb{E}[\mathcal{S}_n] = \frac{6}{n(n-1)} \sum_{k=0}^{n-2} (n-k-1)\,\mathbb{E}[\mathcal{S}_k] + \mathbb{E}[T_S(n)], \quad (n \geq 2)$$

with $\mathbb{E}[\mathcal{S}_0] = \mathbb{E}[\mathcal{S}_1] = 0$. Following Hennequin [9] and Wild [22], we want to solve this recurrence by using generating functions. We set

$$S(z) := \sum_{n\geq 0} \mathbb{E}[\mathcal{S}_n]\, z^n \quad \text{and} \quad T(z) := \sum_{n\geq 0} \mathbb{E}[T_S(n)]\, z^n$$

and use the following lemma (which is Lemma 10.2 in Aumüller et al. [2]).

LEMMA A.1. *With the notation as above, we have*

$$S(z) = (1-z)^3 \int_0^z (1-y)^{-6} \int_0^y (1-x)^3\, T''(x)\ \mathrm{d}x\ \mathrm{d}y.$$

In Section 8.1, we have shown that, for $n \geq 2$, the average number of swaps during the first partitioning step is

$$\mathbb{E}[T_S(n)] = \frac{5}{8}n + \frac{13}{16} - \frac{1}{16(n - \mathbb{1}_{\{n \text{ even}\}})},$$

i.e. $\mathbb{E}[T_S(n)]$ has the form $an + b - \frac{1}{16}\frac{1}{n-\mathbb{1}_{\{n \text{ even}\}}}$ (with $a = \frac{5}{8}$ and $b = \frac{13}{16}$). As $\mathbb{E}[\mathcal{S}_n]$ is linear in $\mathbb{E}[T_S(n)]$, we can split $\mathbb{E}[T_S(n)]$ and compute the contributions of the linear term $an + b$ and the non-linear term separately. Due to linearity, we have $\mathbb{E}[\mathcal{S}_n] = \mathbb{E}[\mathcal{S}_n^{(1)}] - \frac{1}{16}\mathbb{E}[\mathcal{S}_n^{(2)}]$, where $\mathbb{E}[\mathcal{S}_n^{(1)}]$ and $\mathbb{E}[\mathcal{S}_n^{(2)}]$ denote the contributions of the terms $an + b$ and $\frac{1}{n-\mathbb{1}_{\{n \text{ even}\}}}$ to $\mathbb{E}[\mathcal{S}_n]$, respectively. With the result for linear partitioning costs in Section 4.2.1.1 of Wild [22], we obtain for $n \geq 4$:

$$\mathbb{E}[\mathcal{S}_n^{(1)}] = \frac{3}{4}n\mathcal{H}_n - \frac{33}{40}n + \frac{3}{4}\mathcal{H}_n - \frac{27}{160}. \tag{10}$$

In order to compute the contribution $\mathbb{E}[\mathcal{S}_n^{(2)}]$ of the non-linear part, we set

$$S_2(z) := \sum_{n\geq 2} \mathbb{E}[\mathcal{S}_n^{(2)}]\, z^n \quad \text{and} \quad T_2(z) := \sum_{n\geq 2} \frac{1}{n - \mathbb{1}_{\{n \text{ even}\}}}\, z^n.$$

Using the expansion $\sum\limits_{n>k} \frac{\mathbb{1}_{\{n-k \text{ odd}\}}}{n-k} z^n = z^k \operatorname{artanh}(z)$ which is valid for all $k \in \mathbb{Z}$ (where

$\operatorname{artanh}(z) := \frac{1}{2}\log(\frac{1+z}{1-z})$, see Aumüller et. al [2, p. 26]) for $k = 0$, we obtain

$$\begin{aligned} T_2(z) &= \sum_{n\geq 2}\left(\frac{\mathbb{1}_{\{n \text{ even}\}}}{n-1} + \frac{\mathbb{1}_{\{n \text{ odd}\}}}{n}\right) z^n = \sum_{n\geq 1}\frac{\mathbb{1}_{\{n \text{ odd}\}}}{n} z^{n+1} + \sum_{n\geq 2}\frac{\mathbb{1}_{\{n \text{ odd}\}}}{n} z^n \\ &= z\operatorname{artanh}(z) + \operatorname{artanh}(z) - z. \end{aligned}$$

Differentiating twice yields

$$T_2'(z) = \operatorname{artanh}(z) + \frac{z}{1-z^2} + \frac{1}{1-z^2} - 1,$$

$$T_2''(z) = 2\ \frac{z+1}{(1-z^2)^2}.$$

From Lemma A.1, we obtain

$$\begin{aligned} S_2(z) &= (1-z)^3 \int_0^z (1-y)^{-6} \int_0^y (1-x)^3\ T_2''(x)\ \mathrm{d}x\ \mathrm{d}y \\ &= (1-z)^3 \int_0^z (1-y)^{-6} \int_0^y (1-x)^3\, 2\ \frac{x+1}{(1-x^2)^2}\ \mathrm{d}x\ \mathrm{d}y \\ &= (1-z)^3 \int_0^z (1-y)^{-6} \int_0^y 2\ \frac{1-x}{1+x}\ \mathrm{d}x\ \mathrm{d}y \\ &= (1-z)^3 \int_0^z (1-y)^{-6}(4\log(y+1) - 2y)\ \mathrm{d}y \\ &= -\frac{2z}{5(1-z)^2} - \frac{1}{20}(1-z)^3\operatorname{artanh}(z) + \frac{4\log(1+z)}{5(1-z)^2} - \frac{1}{6}z^3 + \frac{9}{20}z^2 - \frac{7}{20}z. \end{aligned}$$

Thus, we have found a closed form of the generating function S_2 of $\mathbb{E}[\mathcal{S}_n^{(2)}]$ from which we can now derive the exact value of $\mathbb{E}[\mathcal{S}_n^{(2)}]$ by taking coefficients. To this end, we use several identities concerning generating functions. It is well-known that

$$\frac{z}{(1-z)^2} = \sum_{n\geq 0} nz^n,$$

Furthermore, since we have $\sum_{n>k} \frac{\mathbb{1}_{\{n-k \text{ odd}\}}}{n-k} z^n = z^k \operatorname{artanh}(z)$ for all $k \in \mathbb{Z}$ (see Aumüller et al. [2, p. 26]), we obtain

$$(1-z)^3\operatorname{artanh}(z) = \sum_{n\geq 1}\frac{\mathbb{1}_{\{n \text{ odd}\}}}{n} z^n - 3\sum_{n\geq 2}\frac{\mathbb{1}_{\{n \text{ even}\}}}{n-1} z^n + 3\sum_{n\geq 3}\frac{\mathbb{1}_{\{n \text{ odd}\}}}{n-2} z^n - \sum_{n\geq 4}\frac{\mathbb{1}_{\{n \text{ even}\}}}{n-3} z^n.$$

Moreover, we have (see Aumüller et al. [2], p. 27)

$$\frac{\log(1+z)}{(1-z)^2} = -\sum_{n\geq 1} n\mathcal{H}_n^{\text{alt}} z^n - \sum_{n\geq 1}\mathcal{H}_n^{\text{alt}} z^n + \frac{1}{2}\sum_{n\geq 0}(-1)^n z^n - \frac{1}{2}\sum_{n\geq 0} z^n.$$

Altogether, we obtain for $n \geq 4$:

$$\begin{aligned}\mathbb{E}[\mathcal{S}_n^{(2)}] = &-\frac{2}{5}n - \frac{4}{5}n\mathcal{H}_n^{\text{alt}} - \frac{2}{5} - \frac{4}{5}\mathcal{H}_n^{\text{alt}} + \frac{2}{5}(-1)^n \\ &- \frac{1}{20}\left[\mathbb{1}_{\{n \text{ odd}\}}\left(\frac{1}{n} + \frac{3}{n-2}\right) - \mathbb{1}_{\{n \text{ even}\}}\left(\frac{3}{n-1} + \frac{1}{n-3}\right)\right]. \end{aligned} \tag{11}$$

We can now compute the exact average number of swaps in the dual-pivot Quicksort algorithm "Count" for $n \geq 4$ by combining (10) and (11):

$$\begin{aligned}\mathbb{E}[\mathcal{S}_n] &= \mathbb{E}[\mathcal{S}_n^{(1)}] - \frac{1}{16}\mathbb{E}[\mathcal{S}_n^{(2)}] \\ &= \frac{3}{4}n\mathcal{H}_n + \frac{1}{20}n\mathcal{H}_n^{\text{alt}} - \frac{4}{5}n + \frac{3}{4}\mathcal{H}_n + \frac{1}{20}\mathcal{H}_n^{\text{alt}} - \frac{23}{160} - \frac{1}{40}(-1)^n \\ &\quad - \frac{\mathbb{1}_{\{n\text{even}\}}}{320}\left(\frac{3}{n-1} + \frac{1}{n-3}\right) + \frac{\mathbb{1}_{\{n\text{odd}\}}}{320}\left(\frac{1}{n} + \frac{3}{n-2}\right).\end{aligned}$$

A.2 Algorithms

Algorithm 2 Classic Quicksort with Sedgewick-Hoare Partitioning

```
1: procedure CLASSIC($A$, $left$, $right$)
2:     // We assume a sentinel value $A[left - 1] = -\infty$
3:     if $right - left \geq 1$ then
4:         $p \leftarrow A[right]$
5:         $i \leftarrow left - 1$; $j \leftarrow right$
6:         do
7:             do $i \leftarrow i + 1$ while $A[i] < p$
8:             do $j \leftarrow j - 1$ while $A[j] > p$
9:             if $j > i$ then swap $A[i]$ and $A[j]$
10:        while $j > i$
11:        swap $A[i]$ and $A[right]$
12:        CLASSIC($A$, $left$, $i - 1$)
13:        CLASSIC($A$, $i + 1$, $right$)
```

Algorithm 2 presents a pseudocode implementation for classic Quicksort by Sedgewick using Hoare's crossing-pointer technique (see Algorithm 1 in Wild's master's thesis [22]). It chooses the rightmost element as pivot (as in Program 1.2 of Sedgewick and Flajolet [21]) and uses two pointers i and j which scan the input from left and right, stop whenever they find a misplaced element and swap the misplaced elements. The following algorithm (Algorithm 3) shows a version of YBB Quicksort as it is presented in Algorithm 1 of Wild, Nebel and Neininger [25] with $M = 1$ (i.e. without using Insertionsort for short subproblems).

Algorithm 3 YBB Dual-Pivot Quicksort Algorithm by Yaroslavskiy, Bentley and Bloch

```
1:  procedure YBB($A$, $left$, $right$)
2:      if $right \leq left$ then
3:          return
4:      if $A[right] < A[left]$ then
5:          $p \leftarrow A[right]$; $q \leftarrow A[left]$
6:      else
7:          $p \leftarrow A[left]$; $q \leftarrow A[right]$
8:      $i \leftarrow left + 1$; $k \leftarrow right - 1$; $j \leftarrow i$
9:      while $j \leq k$ do
10:         if $A[j] < p$ then
11:             swap $A[i]$ and $A[j]$
12:             $i \leftarrow i + 1$
13:         else
14:             if $A[j] \geq q$ then
15:                 while $A[k] > q$ and $j < k$ do
16:                     $k \leftarrow k - 1$
17:                 if $A[k] \geq p$ then
18:                     swap $A[j]$ and $A[k]$
19:                 else
20:                     swap $A[j]$ and $A[k]$; swap $A[i]$ and $A[j]$; $i \leftarrow i + 1$
21:                 $k \leftarrow k - 1$
22:         $j \leftarrow j + 1$
23:     $A[left] \leftarrow A[i-1]$ and $A[i-1] \leftarrow p$
24:     $A[right] \leftarrow A[k+1]$ and $A[k+1] \leftarrow q$
25:     YBB($A$, $left$, $i-2$)
26:     YBB($A$, $i$, $k$)
27:     YBB($A$, $k+2$, $right$)
```

A.3 Implementation in R

```r

## the following implementations of dual-pivot Quicksort "Count" and classic Quicksort
## additionally count the number of key comparisons and the number of swaps

###########################################################################################
########## ALGORITHM "COUNT" (as it is described in Algorithm 1) ##########################
###########################################################################################

### Partitioning function "Count" ###
PartitionCount <- function(left, right){
  comparisons <<- comparisons + 1
  if (A[left] > A[right]){
    q <- A[left]; p <- A[right]}
  else{
    p <- A[left]; q <- A[right]}
  i <- left + 1; k <- right - 1; j <- i
  d <- 0     # holds the difference between small and large elements
  while (j <= k){
    if (d >= 0){
      comparisons <<- comparisons + 1
      if (A[j] < p){
        tmp <- A[i]; A[i] <<- A[j]; A[j] <<- tmp     # swap A[i] and A[j]
        swaps <<- swaps + 1
        i <- i+1; j <- j+1; d <- d+1}
      else{
        comparisons <<- comparisons + 1
        if (A[j] < q){j <- j+1}
        else {
          tmp <- A[j]; A[j] <<- A[k]; A[k] <<- tmp     # swap A[j] and A[k]
          swaps <<- swaps + 1
          k <- k-1; d <- d-1}}}
    else{
      comparisons <<- comparisons + 1
      while (A[k] > q){
        k <- k-1; d <- d-1; comparisons <<- comparisons + 1}
      if (j <= k){
        comparisons <<- comparisons + 1
        if (A[k] < p){
          tmp <- A[k]; A[k] <<- A[j]; A[j] <<- A[i]; A[i] <<- tmp     # rotate3(A[k],A[j],A[i])
          swaps <<- swaps + 1.5     # a rotate3-operation is counted as 1.5 swaps
          i <- i+1; d <- d+1}
        else{
          tmp <- A[j]; A[j] <<- A[k]; A[k] <<- tmp     # swap A[j] and A[k]
          swaps <<- swaps + 1}
        j <- j+1}}}
  A[left] <<- A[i-1]; A[i-1] <<- p     # swap p to its final position
  A[right] <<- A[k+1]; A[k+1] <<- q     # swap q to its final position
  swaps <<- swaps+2
  posp <- i-1; posq <- k+1     # posp and posq are the final positions of the pivots
  return(c(posp, posq))
}


### Dual-Pivot Quicksort Algorithm "Count" ###
DualPivotCount <- function(left, right){
  if (right-left >= 1){
    partition <- PartitionCount(left, right)
    DualPivotCount(left, partition[1]-1)
    DualPivotCount(partition[1]+1, partition[2]-1)
    DualPivotCount(partition[2]+1, right)}
}


```

```r

# for permutations
library("gtools", lib.loc="~/R/win-library/3.3")


##### Verification of E[T_S(n)] for small n (n <= 10) #####
max_n <- 10     # change the maximal value for n here

## our computed values for E[T_S(n)] are
exact_mean_swaps <- rep(0,max_n)
for (n in 2:max_n){
  if (n %% 2 == 0){exact_mean_swaps[n] <- 5/8 * n + 13/16 - 1/(16*(n-1))}
  else{exact_mean_swaps[n] <- 5/8 * n + 13/16 - 1/(16*n)}
}

## when using the partitioning strategy "Count", we obtain
mean_swaps <- rep(0,max_n)
for (n in 2:max_n){
  m <- factorial(n)     # number of permutations of {1,...,n}
  perm <- permutations(n,n)     # set of all permutations of {1,...,n}
  comp <- rep(0,m); swap <- rep(0,m)
  for (i in 1:m){
    A <- perm[i,]; comparisons <- 0; swaps <- 0
    PartitionCount(1, n)
    comp[i] <- comparisons; swap[i] <- swaps
  }
  mean_swaps[n] <- mean(swap)
}

print(exact_mean_swaps); print(mean_swaps)     # we obtain the same results
# [1] 0.000000 2.000000 2.666667 3.291667 3.925000 4.550000 5.178571 5.803571 6.430556 7.055556


##### Verification of E[S_n] for small n (n <= 9) #####
max_n <- 9     # change the maximal value for n here

## our computed values for E[S_n] are
exact_mean_swaps <- c(0,2,8/3,rep(0,max_n-3))
harm <- cumsum(1/(1:10))
harm_alt <- cumsum(1/(1:10)*c(-1,1))
for (n in 4:max_n){
  if (n %% 2 == 0){
    exact_mean_swaps[n] <- 3/4*n*harm[n] + 1/20*n*harm_alt[n] - 4/5*n + 3/4*harm[n] +
      1/20*harm_alt[n] - 23/160 - 1/40 - 1/320*(3/(n-1)+1/(n-3))}
  else{
    exact_mean_swaps[n] <- 3/4*n*harm[n] + 1/20*n*harm_alt[n] - 4/5*n + 3/4*harm[n] +
      1/20*harm_alt[n] - 23/160 + 1/40 + 1/320*(3/(n-2)+1/n)}
}

## when using the dual-pivot Quicksort algorithm "Count", we obtain
mean_swaps <- rep(0,max_n)
for (n in 1:max_n){
  m <- factorial(n)     # number of permutations of {1,..,n}
  perm <- permutations(n,n)     # set of all permutations of {1,...,n}
  comp <- rep(0,m); swap <- rep(0,m)
  for (i in 1:m){
    A <- perm[i,]; comparisons <- 0; swaps <- 0
    DualPivotCount(1, n)
    comp[i] <- comparisons; swap[i] <- swaps
  }
  mean_swaps[n] <- mean(swap)
}

print(exact_mean_swaps); print(mean_swaps)     ## we obtain the same results
# [1]  0.000000  2.000000  2.666667  4.291667  5.925000  7.675000  9.536905 11.489286 13.527381
```

```

########## Simulation studies ##############################################################

##### Empirical validation of the computed asymptotic results #####
# change the sample sizes and the input lengths here
values_for_n <- c(0.5*10^4,10^4,1.5*10^4,2*10^4,2.5*10^4,3*10^4)     # vector of input sizes
m <- 500     # number of generated permutations

N <- length(values_for_n)
mean_comp <- rep(0,N); mean_swap <- rep(0,N)
std_comp <- rep(0,N); std_swap <- rep(0,N); corr <- rep(0,N)
for (j in 1:N){
  n <- values_for_n[j]
  swap <- rep(0,m); comp <- rep(0,m)
  for (i in 1:m){
    A <- sample(1:n, n, replace = FALSE); comparisons <- 0; swaps <- 0
    DualPivotCount(1,n)
    comp[i] <- comparisons; swap[i] <- swaps
  }
  mean_comp[j] <- mean(comp); mean_swap[j] <- mean(swap)
  std_comp[j] <- sd(comp)/n; std_swap[j] <- sd(swap)/n; corr[j] <- cor(comp, swap)
}

# We obtained the following results (see Figure 6 and Figure 7)
# mean_comp:  [1]  65104.12 141981.03 224763.23 309740.30 398046.39 487445.48
# mean_swap:  [1]  29884.37  65011.14 102149.82 140316.41 180168.38 219945.66
# std_comp:   [1] 0.5036020 0.4586641 0.4801899 0.5144635 0.5000543 0.5145687
# std_swap:   [1] 0.3380717 0.3370387 0.3537372 0.3240614 0.3466012 0.3475795
# corr:       [1] 0.2725368 0.3304496 0.2328610 0.3143647 0.2440217 0.3050550


##### Histograms and Scatter Plot #####

n <- 10^4     # change the input length here
m <- 10^3     # change the number of permutations here

swap <- rep(0,m); comp <- rep(0,m)
for (i in 1:m){
  A <- sample(1:n, n, replace = FALSE); comparisons <- 0; swaps <- 0
  DualPivotCount(1,n)
  comp[i] <- comparisons; swap[i] <- swaps
}

## if n is even, the average number of comparisons and swaps is
exact_mean_comp <- 1.8*n*sum(1/(1:n)) - 0.2*n*sum(1/(1:n)*c(-1,1)) - 89/25*n + 67/40*sum(1/(1:n)) -
  3/40*sum(1/(1:n)*c(-1,1)) - 83/800 + 0.1 - 1/320*(1/(n-3)+3/(n-1))
exact_mean_swaps <- 0.75*n*sum(1/(1:n)) + 0.05*n*sum(1/(1:n)*c(-1,1)) - 0.8*n + 0.75*sum(1/(1:n)) +
  0.05*sum(1/(1:n)*c(-1,1)) - 23/160 - 1/40 - 1/320*(1/(n-3)+3/(n-1))

## create the images
par(mfrow=c(1,3))

hist((comp-exact_mean_comp)/n, breaks=15, freq=FALSE, col="grey92", main="Comparisons",
     xlab="", ylab="")
hist((swap-exact_mean_swaps)/n, breaks=15, freq=FALSE, col="grey92", main="Swaps",
     xlab="", ylab="")
plot((comp-exact_mean_comp)/n,(swap-exact_mean_swaps)/n, pch=19, cex=0.2, xlab="", ylab="",
     frame.plot = FALSE, main="Scatter plot")
```

```

###########################################################################################
########## CLASSIC QUICKSORT ALGORITHM (as it is described in Algorithm 2) #################
###########################################################################################

ClassicQuicksort <- function(left,right){
  # we assume that there is a sentinel value A[left-1] = -infinity
  if (right-left >= 1){
    p <- A[right]     # choose rightmost element as pivot
    i <- left-1; j <- right
    repeat{
      repeat{i <- i+1; comparisons <<- comparisons+1; if(A[i] >= p){break}}
      repeat{j <- j-1; comparisons <<- comparisons+1; if(A[j] <= p){break}}
      if (j > i){
        tmp <- A[i]; A[i] <<- A[j]; A[j] <<- tmp     # swap A[i] and A[j]
        swaps <<- swaps +1}
      if(j <= i){break}
    }
    tmp <- A[i]; A[i] <<- A[right]; A[right] <<- tmp     # swap p to its final position
    swaps <<- swaps +1
    ClassicQuicksort(left, i-1)
    ClassicQuicksort(i+1, right)}
}

## Note that we have to insert a sentinel value A[0] which is smaller than all elements in A,
## i.e. in order to sort a permutation A of {1,...,n}, we procede as follows:
## A <- c(0, A); ClassicQuicksort(2, n+1); A <- A[-1]



##### Verification of the average number of comparisons and swaps (n <= 9) #####
max_n <- 9     # change the maximal value for n here

## the exact expected number of comparisons and swaps when sorting a
## permutation of {1,...,n} is
exact_mean_comp <- c(0,3,6,rep(0,max_n-3))
exact_mean_swaps <- c(0,1,11/6,rep(0,max_n-3))
harm <- cumsum(1/(1:10))
harm_alt <- cumsum(1/(1:10)*c(-1,1))
for (n in 4:max_n){
  exact_mean_comp[n] <- 2 * (n+1) * (sum(1/(1:(n+1))) - 4/3)
  exact_mean_swaps[n] <- 1/3 * (n+1) * (sum(1/(1:(n+1))) - 1/3) - 1/2
}

## when using ClassicQuicksort, we obtain
mean_comp <- rep(0,max_n); mean_swaps <- rep(0,max_n)
for (n in 1:max_n){
  m <- factorial(n)     # number of permutations of {1,..,n}
  perm <- permutations(n,n)     # set of all permutations of {1,...,n}
  comp <- rep(0,m); swap <- rep(0,m)
  for (i in 1:m){
    A <- c(0, perm[i,]); comparisons <- 0; swaps <- 0
    ClassicQuicksort(2, n+1)
    comp[i] <- comparisons; swap[i] <- swaps
  }
  mean_comp[n] <- mean(comp); mean_swaps[n] <- mean(swap)
}

print(exact_mean_comp); print(mean_comp)
# [1]  0.00000  3.00000  6.00000  9.50000 13.40000 17.63333 22.15238 26.92143 31.91270
print(exact_mean_swaps); print(mean_swaps)
# [1] 0.000000 1.000000 1.833333 2.750000 3.733333 4.772222 5.858730 6.986905 8.152116
```

References

[1] Martin Aumüller and Martin Dietzfelbinger (2015) Optimal Partitioning for Dual-Pivot Quicksort, *ACM Trans. Algorithms* **12**, Art. 18, 36 pp.

[2] Martin Aumüller, Martin Dietzfelbinger, Clemens Heuberger, Daniel Krenn and Helmut Prodinger (2016+) *Dual-Pivot Quicksort: Optimality, Analysis and Zeros of Associated Lattice Paths.* Preprint. URL `http://arxiv.org/abs/1611.00258`.

[3] Martin Aumüller, Martin Dietzfelbinger and Pascal Klaue (2016) How Good Is Multi-Pivot Quicksort? *ACM Trans. Algorithms* **13**, Art. 8, 47 pp.

[4] Peter J. Bickel and David A. Freedman (1981) Some Asymptotic Theory for the Bootstrap. *The Annals of Statistics* **9**, 1196–1217.

[5] Herbert A. David and Haikady N. Nagaraja (2003) *Order Statistics.* Wiley-Interscience, third edition. ISBN 0-471-38926-9.

[6] James A. Fill and Svante Janson (2000) Smoothness and Decay Properties of the Limiting Quicksort Density Function. *Mathematics and Computer Science (Versailles, 2000)*, 53–64. Trends Math., Birkhäuser, Basel.

[7] Ronald L. Graham, Donald E. Knuth and Oren Patashnik (1994) *Concrete Mathematics. A Foundation for Computer Science.* second edition, Addison-Wesley.

[8] Pascal Hennequin (1989) Combinatorial Analysis of Quicksort Algorithm. *Informatique théorique et applications* **23**, 317–333.

[9] Pascal Hennequin (1991) *Analyse en moyenne d'algorithmes : tri rapide et arbres de recherche.* Ph.D. Thesis, Ecole Polytechnique, Palaiseau.

[10] C. A. R. Hoare (1962) Quicksort. *The Computer Journal* **5**, 10–15.

[11] Donald E. Knuth (1998) *The Art Of Computer Programming: Searching and Sorting.* Addison-Wesley, second edition.

[12] Shrinu Kushagra, Alejandro López-Ortiz, Aurick Qiao, and J. Ian Munro (2014) Multi-Pivot Quicksort: Theory and Experiments. In: McGeoch, C.C., Meyer, U. (ed.) SIAM Meeting on Algorithm Engineering and Experiments (ALENEX), pp. 47–60.

[13] Kevin Leckey (2016+) On Densities for Solutions to Stochastic Fixed Point Equations. Preprint. URL `http://arxiv.org/abs/1604.05787.`

[14] Hosam M. Mahmoud (2000) *Sorting: A Distribution Theory.* John Wiley & Sons.

[15] Markus E. Nebel, Sebastian Wild and Conrado Martínez (2016) Analysis of Pivot Sampling in Dual-Pivot Quicksort: A Holistic Analysis of Yaroslavskiy's Partitioning Scheme. *Algorithmica* **75**, 632–683.

[16] Ralph Neininger (2001) On a Multivariate Contraction Method for Random Recursive Structures with Applications to Quicksort. *Random Structures Algorithms* **19**, 498–524.

[17] Uwe Rösler (1991) A Limit Theorem for "Quicksort". *Informatique théorique et Applications/Theoretical Informatics and Applications* **25**, 85–100.

[18] Uwe Rösler (2001) On the Analysis of Stochastic Divide and Conquer Algorithms. *Algorithmica* **29**, 238–261.

[19] Robert Sedgewick (1977) The Analysis of Quicksort Programs. *Acta Informatica* **7**, 327–355.

[20] Robert Sedgewick (1980) *Quicksort.* Reprint of the author's Ph.D. Thesis.

[21] Robert Sedgewick and Philippe Flajolet (1996) *An Introduction to the Analysis of Algorithms.* Addison-Wesley- Longman.

[22] Sebastian Wild (2012) *Java 7's Dual Pivot Quicksort.* Master's Thesis. University of Kaiserslautern. URL `https://kluedo.ub.uni-kl.de/frontdoor/index/index/docId/3463`

[23] Sebastian Wild (2016) *Dual-Pivot Quicksort and Beyond: Analysis of Multiway Partitioning and Its Practical Potential.* Ph.D. Thesis. University of Kaiserslautern. URL `https://kluedo.ub.uni-kl.de/frontdoor/index/index/docId/4468`

[24] Sebastian Wild and Markus E. Nebel (2012) Average Case Analysis of Java 7's Dual Pivot Quicksort. In Leah Epstein and Paolo Ferragina, editors, *European Symposium on Algorithms 2012*, volume 7501 of LNCS, pages 825–836. Springer.

[25] Sebastian Wild, Markus E. Nebel and Ralph Neininger (2015) Average Case and Distributional Analysis of Dual-Pivot Quicksort, *ACM Trans. Algorithms* **11**, Art. 22, 42 pp.