Quicksort Is Optimal For Many Equal Keys

Sebastian Wild[*]

November 1, 2017

I prove that the average number of comparisons for median-of-k Quicksort (with fat-pivot a.k.a. three-way partitioning) is asymptotically only a constant α_k times worse than the lower bound for sorting random multisets of n elements with $\Omega(n^\varepsilon)$ duplicates of each value (for any $\varepsilon > 0$). The constant is $\alpha_k = \ln(2)/(H_{k+1} - H_{(k+1)/2})$, which converges to 1 as $k \to \infty$, so median-of-k Quicksort is asymptotically optimal for inputs with many duplicates. This partially resolves a conjecture by Sedgewick and Bentley (1999, 2002) and constitutes the first progress on the analysis of Quicksort with equal elements since Sedgewick's 1977 article.

1. Introduction

Sorting is one of the basic algorithmic tasks that is used as a fundamental stepping stone for a multitude of other, more complex challenges. Quicksort is the method of choice for sorting in practice. Any undergraduate student learns as a justification for its quality that the average number of comparisons is $\Theta(n \log n)$ for a random permutation of n distinct elements, putting it into the same complexity class as, e.g., Mergesort. By choosing random pivots we can make this average the *expected* behavior regardless of the order of the input. Moreover, the hidden constants in $\Theta(n \log n)$ are actually small: Quicksort needs $2\ln(2) n \operatorname{ld}(n) \pm O(n)$ comparisons in expectation, i.e., asymptotically only a factor $2\ln(2) \approx 1.39$ more than the information-theoretic lower bound for any comparison-based sorting method.[1] By choosing the pivot as median of k sample elements in each step, for k a fixed, odd integer, this constant can be reduced to $\alpha_k = \ln(2)/(H_{k+1} - H_{(k+1)/2})$ [37], where $H_k = \sum_{i=1}^{k} 1/i$. Note that α_k converges to 1 as $k \to \infty$, so Quicksort's expected behavior is asymptotically optimal on random permutations.

It is also well-known that Quicksort rarely deviates much from this expected behavior. Folklore calculations show that Quicksort needs, e.g., at most 7 times the expected number of comparisons with probability at least $1 - 1/n^2$. This result can be strengthened [32] to guarantee at most α times the expected costs for any $\alpha > 1$ with probability $1 - O(n^{-c})$ for all constants c (independent of α). Median-of-k Quicksort is hence optimal not only in expectation, but almost always!

These guarantees certainly justify the wide-spread use of Quicksort, but they only apply to inputs with n *distinct* elements. In many applications, duplicates (i.e., elements that are equal w.r.t. the order relation) appear naturally. The SQL clause `GROUP BY C`, for example, can be implemented by first sorting rows w.r.t. column `C` to speed up the computation of aggregating functions like `COUNT` or `SUM`. Many rows with equal `C`-entries are expected in such applications.

[*]David R. Cheriton School of Computer Science, University of Waterloo, Email: `wild@uwaterloo.ca`

[1]I write ld for $\log_2$ (*l*ogarithmus *d*ualis), and ln for $\log_e$ (*l*ogarithmus *n*aturalis).

At the *KnuthFest* celebrating Donald Knuth's 1000000_2th birthday, Robert Sedgewick gave a talk titled *"Quicksort is optimal"* [40],[2] presenting a result that is at least very nearly so: Quicksort with *fat-pivot* (a.k.a. *three-way*) partitioning uses $2 \ln 2 \approx 1.39$ times the number of comparisons needed by *any* comparison-based sorting method for *any* randomly ordered input, with or without equal elements. He closed with the conjecture that this factor can be made arbitrarily close to 1 by choosing the pivot as the median of k elements for sufficiently large k. This statement is referred to as the Sedgewick-Bentley conjecture.

The Sedgewick-Bentley conjecture is a natural generalization of the distinct-keys case (for which the statement is known to hold true), and sorting experts will not find the claim surprising, since it is the theoretical justification for a long-established best practice of general-purpose sorting: as observed many times in practice, Quicksort with the fat-pivot partitioning method by Bentley and McIlroy [4][3] and *ninther* (a.k.a. *pseudomedian of nine*) pivot sampling comes to within *ten percent* of the optimal expected comparison count for inputs with or without equal keys.

In this paper, I confirm the Sedgewick-Bentley conjecture for inputs with "many duplicates", i.e., where every key value occurs $\Omega(n^{\varepsilon})$ times for an arbitrary constant $\varepsilon > 0$. To do so I show a bound on the constant of proportionality, namely the same α_k mentioned above. (Sedgewick and Bentley did not include a guess about the constant or the speed of convergence in their conjecture). While Quicksort can certainly be outperformed for particular types of inputs, the combination of simple, efficient code and almost universal proven optimality is unsurpassed; it is good news that the latter also includes inputs with equal keys.

Confirming the Sedgewick-Bentley conjecture is not a surprising outcome, but it has been surprisingly resistant to all attempts to formally prove it: no progress had been made in the nearly two decades since it was posed (see Section 5 for some presumable reasons for that), so restricting the problem might be sensible. I would like to remark that my restriction of many duplicates is a technical requirement of the *analysis*, but we have no reason to believe that Quicksort performs much different when it is violated. More importantly, it is *not* the purpose of this paper to suggest sorting inputs with many duplicates as a natural problem per se, for which tailored methods shall be developed. (One is tempted to choose a hashing-based approach when we *know* that the number u of different values is small, but my results show that precisely for such instances, Quicksort will be competitive to any tailored algorithm!) It must be seen as an idiosyncrasy of the present analysis one might try to overcome in the future.

It is a strength of Quicksort is that it smoothly *adapts* to the actual amount of duplication without requiring explicit detection and special handling of that case. The concept is analogous (but orthogonal) to *adaptive* sorting methods [14] that similarly take advantage of existing (partial) *order* in the input. The purpose of this paper is thus to finally deliver a mathematical proof that median-of-k Quicksort is an optimal "entropy-adaptive" (a.k.a. *distribution sensitive* [43]) sorting method, at least for a large class of inputs.

Methods. For the analysis we will work in an alternative input model: instead of fixing the *exact* multiplicities of a multiset, we fix a *discrete probability distribution* over $[1..u]$ and draw n elements independently and identically distributed (i.i.d.) according to that distribution. We

[2]Sedgewick presented the conjecture in a less widely-known talk already in 1999 [39]. Since they never published their results in an article, I briefly reproduce their arguments in Section 4.

[3]The classic implementation by Bentley and McIlroy [4] from 1993 is used as default sorting methods in important programming libraries, e.g., the GNU C++ Standard Template Library (STL) and the Java runtime library (JRE). Since version 7, the JRE uses dual-pivot Quicksort instead, but if the two sampled pivots are found *equal*, it falls back to fat-pivot partitioning by Bentley and McIlroy.

will see that going from fixed to expected multiplicities only increases the average sorting costs, so the i.i.d. model provides an upper bound.

The analysis of Quicksort on discrete i.i.d. inputs proceeds in two steps: First, I use that costs of (median-of-k) Quicksort correspond to costs of searching in a (k-fringe-balanced) binary search tree (BST). A concentration argument shows that for inputs with many duplicates, the search tree typically reaches a stationary state after a short prefix of the input, and is thus independent of the (larger) rest of the input.

The second step is to determine the expected search costs in a k-fringe-balanced BST built by repeatedly inserting i.i.d. elements until all u values appear in the tree; (I call the tree *saturated* then). I show that the search costs are asymptotically α_k times the *entropy* of the universe distribution. To do so I derive lower and upper bounds from the recurrence of costs (with the *vector* of probability weights as argument) using the aggregation property of the entropy function. To the best of my knowledge, the analysis of search costs in fringe-balanced trees with equal keys is novel, as well.

Most used techniques are elementary, but keeping the error terms under control (so that $\Omega(n^\varepsilon)$ duplicates suffice for any $\varepsilon > 0$) made some refinements of classical methods necessary that might be of independent interest.

Outline. We start with some notation and a collection of known results (Section 2), and introduce the input models (Section 3). Section 4 is devoted to related work on sorting in the presence of equal elements. We continue in Section 5 with a formal statement of the main result and a briefly discussion why some previous attempts did not succeed. In Sections 6 – 9, we study the cost of fat-pivot Quicksort on discrete i.i.d. inputs as outlined above: Section 6 demonstrates that it suffices to study search costs in trees to analyze Quicksort, and Section 7 gives a stochastic description of these trees. In Section 8, we exploit the assumption of many equal keys to separate the influence of the input size from the influence of different universe distributions (the "first step" from above). In Section 9 we then derive an asymptotic approximation for the search costs in fringe-balanced trees (second step). We establish a lower bound for the i.i.d. model in Section 10. Section 11 proves the main result by combining all pieces. The results are summarized and put into context in Section 12.

Appendix A collects all notation used in this paper for reference. In Appendix B, we formally prove that fringe-balanced search trees have logarithmic height with high probability in the presence of equal keys.

2. Preliminaries

We start with some common notation and a precise description of fat-pivot Quicksort with median-of-k sampling, as well as the corresponding a search tree variant, the k-fringe-balanced trees. We collect further known results that will be used later onhere for the reader's convenience: some classical tail bounds (Section 2.5), a fact on the height concentration of randomly built search trees (Section 2.6), and a few properties of the entropy function used later on (Section 2.7).

2.1. Notation

This section contains the most important notation; a comprehensive list is given in Appendix A for reference.

I write vectors in bold font $\boldsymbol{x} = (x_1, \ldots, x_n)$ and always understand them as column vectors (even when written in a row in the text). By default, all operations on vectors are meant component-wise. By $\Sigma \boldsymbol{x}$, I denote $x_1 + \cdots + x_n$, the *total* of $\boldsymbol{x}$. The *profile* (or multiplicities vector) a multiset M over $[u] = \{1, \ldots, u\}$ is a vector $\boldsymbol{x} = (x_1, \ldots, x_u)$ where every values $v \in [u]$ occurs x_v times in M.

I use Landau-symbols (O, Ω etc.) in the formal sense of Flajolet and Sedgewick [15, Section A.2] and write $f(n) = g(n) \pm O(h(n))$ to mean $|f(n) - g(n)| = O(h(n))$. (I use $\pm$ to emphasize the fact that we specify $f(n)$ up to an additive error of $h(n)$ without restricting the sign of the error.)

For a random variable X, $\mathbb{E}[X]$ is its expectation and $\mathbb{P}[X = x]$ denotes the probability of the event $X = x$. For $\boldsymbol{q} \in [0,1]^u$ with $\Sigma \boldsymbol{q} = 1$, I write $U \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$ to say that U is a random variable with $\mathbb{P}[U = i] = q_i$ for $i \in [u]$; generally, $\overset{\mathcal{D}}{=}$ denotes equality in distribution. $\mathrm{Mult}(n, \boldsymbol{q})$ is a *multinomial distributed* variable with n trials from $\mathcal{D}(\boldsymbol{q})$, and $\mathrm{Beta}(\alpha, \beta)$ is a *beta distributed* variable with parameters $\alpha, \beta > 0$. By $\mathcal{H}$ or $\mathcal{H}_{\mathrm{ld}}$ I denote the binary *Shannon entropy* $\mathcal{H}(\boldsymbol{q}) = \sum_{i=1}^{u} q_i \operatorname{ld}(1/q_i)$; likewise $\mathcal{H}_{\ln}$ is the base e entropy.

I say an event $E = E(n)$ occurs *with high probability (w.h.p.)* if for *any* constant c holds $\mathbb{P}[E(n)] = 1 \pm O(n^{-c})$ as $n \to \infty$. Note that other sources use w.h.p. if any such $c \in \mathbb{N}$ exists, which is a much weaker requirement. Similarly for $X_1, X_2, \ldots$ a sequence of real random variables, I say "$X_n = O(g(n))$ w.h.p." if for every constant c there is a constant d so that $\mathbb{P}[|X_n| \le d|g(n)|] = 1 \pm O(n^{-c})$ as $n \to \infty$.

2.2. Reference Version of Fat-Pivot Quicksort

By "fat-pivot" I mean a partitioning method that splits the input into three parts: elements (strictly) smaller than the pivot, elements equal to the pivot, and elements (strictly) larger than the pivot. Instead of only one pivot element in binary partitioning, we now obtain a "fat" pivot segment that separates the left and right subproblems. This idea is more commonly known as *three-way partitioning,* but I prefer the vivid term fat pivot to make the distinction between fat-pivot partitioning and partitioning around *two pivots* clear. The latter has become fashionable again in recent years [23, 48, 47]. Since all duplicates of the pivot are removed before recursive calls, the number of partitioning rounds is bounded by the number of different values in the input.

To simplify the presentation, I assume in this work that partitioning *retains the relative order* of elements that go to the same segment. A corresponding reference implementation operating on linked lists is given in Algorithm 1. It uses the median of k sample elements as pivot, where the sample size $k = 2t + 1$, $t \in \mathbb{N}_0$, is a tuning parameter of the algorithm. We consider k a fixed constant in the analysis.

I always mean Algorithm 1 when speaking of Quicksort in this paper. Moreover, by its costs I always mean the number of ternary key comparisons, i.e., the number of calls to *cmp*. Apart from selecting the median, one partitioning step in Algorithm 1 uses exactly n ternary comparisons. Some of these will be redundant because they already happened while determining the median of the sample. We ignore this optimization here.

Algorithm 1 serves as precise specification for the analysis but it is not the method of choice in practice. However, our results apply to practical methods, as well (see the discussion in Section 6.1).

2.3. Quicksort and Search Trees

It has become folklore that the comparison costs of (classic) Quicksort are the same as the internal path length of a BST, which equals the cost to construct the tree by successive insertions. Hibbard [18] first described this fact in 1962, right after Hoare's publication of

Algorithm 1. Fat-pivot median-of-k Quicksort.

```
Quicksort_k(L)
 1  if (L.length) ≤ k − 1
 2      return Insertionsort(L)
 3  end if
 4  P := Median(L[1..k])
 5  L_1, L_2, L_3 := new List
 6  while ¬ L.empty
 7      U := L.removeFirstElement()
 8      case distinction on cmp(U, P)
 9          in case < do L_1.append(U)
10          in case = do L_2.append(U)
11          in case > do L_3.append(U)
12      end cases
13  end while
14  L_1 := Quicksort_k(L_1)
15  L_3 := Quicksort_k(L_3)
16  return L_1.append(L_2).append(L_3)
```

Quicksort itself [19, 20]. Formally we associate a *recursion tree* to each execution of Quicksort: Each partitioning step contributes a node labeled with the used pivot value. Its left and right children are the recursion trees of the left and right recursive calls, respectively. Figure 1 shows an example.

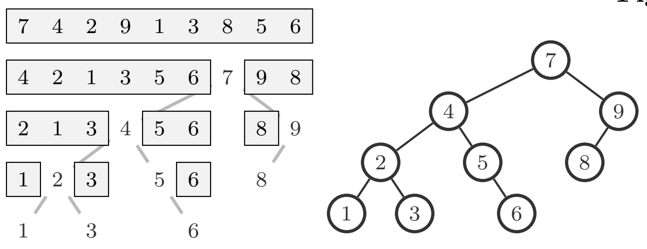

Figure 1: Execution trace of Quicksort without pivot sampling and its recursion tree. The recursion tree coincides with the BST obtained by successively inserting the original input.An animated version of this example (and its extensions to finge-balancing and equal keys) is available online: `youtu.be/yi6syj9nksk`.

Recursion trees obviously fulfill the search tree property; in fact, the recursion tree is *exactly* the binary search tree that results from successively inserting the input elements into an initially empty tree (in the order they appear in the original input) if Quicksort always uses the first element of the list as pivot and partitions so that the relative order of elements smaller resp. larger than the pivot is retained (as is done in Algorithm 1). Even the *same set* of comparisons is then used in both processes, albeit in a different order.

2.4. Fringe-Balanced Trees

The correspondence extends to median-of-k Quicksort with an appropriate *fringe-balancing rule* for BSTs. This is a little less widely known, but also well researched [11]. Upon constructing a k-fringe-balanced search tree, we *collect* up to $k-1$ elements in a leaf. (This corresponds to truncating the recursion in Quicksort for $n \le k-1$ and leave small subproblems for Insertionsort.) Once a leaf has collected $k = 2t+1$ elements, it is *split:* Simulating the pivot selection process in Quicksort, we find the median from the k elements in the leaf and use it as the label of a new inner node. Its children are two new leaves with the elements that did not become pivots: the t smaller elements go to the left, the t larger to the right. Because of the dynamic process of splitting leaves, the correspondence is best shown in motion; the supplementary animation (`youtu.be/yi6syj9nksk`) includes fringe-balanced trees.

More formally, a k-fringe-balanced tree, for $k = 2t + 1$ an odd integer, is a binary search tree whose leaves can store between 0 and $k - 1$ elements. An empty fringe-balanced tree is represented as *Leaf*(), a single empty leaf. The k-fringe-balanced tree $\mathcal{T}$ corresponding to a sequence of elements $x_1, \ldots, x_n$ is obtained by successively inserting the elements into an initially empty tree using Algorithm 2; more explicitly, with $T_0 = \mathit{Leaf}()$ and $T_i = \textsc{Insert}_k(\mathcal{T}_{i-1}, x_i)$ for $1 \le i \le n$, we have $\mathcal{T} = \mathcal{T}_n$.

Algorithm 2. Insert into k-fringe-balanced tree.

```
Insert_k($\mathcal{T}, x$)
if $\mathcal{T}$ is $Leaf(\boldsymbol{U})$
    Append $x$ to $\boldsymbol{U}$
    if $|\boldsymbol{U}| \le k-1$ then return $Leaf(\boldsymbol{U})$ end if
    // Else: Split the leaf
    $P :=$ Median($U_1, \ldots, U_k$)
    $C_1, C_2 :=$ new empty list
    for each $U$ in $\boldsymbol{U}$
        case distinction on $cmp(U, P)$
            in case $U < P$ do append $U$ to $C_1$
            in case $U > P$ do append $U$ to $C_2$
            // In case $U == P$, we drop $U$.
        end cases
    end for
    return $Inner\big(P, Leaf(C_1), Leaf(C_2)\big)$
else $\mathcal{T}$ is $Inner(P, \mathcal{T}_1, \mathcal{T}_2)$
    case distinction on $cmp(x, P)$
        in case $x == P$ do return $\mathcal{T}$    // tree unchanged
        in case $x < P$ do return $Inner\big(P, \textsc{Insert}_k(\mathcal{T}_1, x), \mathcal{T}_2\big)$
        in case $x > P$ do return $Inner\big(P, \mathcal{T}_1, \textsc{Insert}_k(\mathcal{T}_2, x)\big)$
    end cases
end if
```

Growing trees with Algorithm 2 enforces a certain balance upon the lowest subtrees, i.e., at the *fringe* of the tree, hence the name *fringe-balanced.* Searching an element in a fringe-balanced tree works as in an ordinary BST, except for the leaves, where we sequentially search through the buffer;Algorithm 3 shows pseudocode for completeness. (Note that elements collected in leaves are not sorted.)

Algorithm 3. Search in fringe-balanced trees.

```
Search($\mathcal{T}, x$)
if $\mathcal{T}$ is $Leaf(\boldsymbol{U})$
    return SequentialSearch($\boldsymbol{U}, x$)
else $\mathcal{T}$ is $Inner(P, \mathcal{T}_1, \mathcal{T}_2)$
    case distinction on $cmp(U, P)$
        in case $U == P$ do return "Found"
        in case $U < P$ do return Search($\mathcal{T}_1, x$)
        in case $U > P$ do return Search($\mathcal{T}_2, x$)
    end cases
end if
```

Duplicate Insertions. Since our analysis of Quicksort with equal keys builds on the precise performance of fringe-balanced trees, we put a particular emphasis is on the treatment of duplicate insertions in Algorithm 2. If a key x is already present during insertion and x appears as key of an *inner* node, the (repeated) insertion has no effect; but if x is found in a *leaf*, another copy of x is appended to the buffer of that leaf.

This unequal treatment might seem peculiar at first sight, but does exactly what we need: duplicates do play a role for selecting pivots—likely values contribute more duplicates to a random sample and are thus more likely to be selected as pivot—but once a pivot has been selected, all its copies are removed in this single partitioning step no matter how many there are.

2.5. Tail Bounds

For the reader's convenience, we collect a few basic tail bounds here: a classical and a less known Chernoff concentration bound, and a bound for the far end of the lower tail of the binomial distribution $\mathrm{Bin}(n, p)$.

Lemma 2.1 (Chernoff Bound): *Let $X \overset{\mathcal{D}}{=} \mathrm{Bin}(n, p)$ for $n \in \mathbb{N}$ and $p \in (0, 1)$ and let $\delta \geq 0$. Then*

$$\mathbb{P}\left[\left|\frac{X}{n} - p\right| \geq \delta\right] \quad\leq\quad 2\exp(-2\delta^2 n). \qquad \square$$

This bound appears, e.g., as Theorem 2.1 of McDiarmid [31].

The following lemma is a handy, but less well-known bound for the *multinomial* distribution that appears—indeed rather hidden—as Lemma 3 in a paper by Devroye [6] from 1983. (Its proof is also discussed on *math stack exchange:* `math.stackexchange.com/q/861058`.)

Lemma 2.2 (Chernoff Bound for Multinomial): *Let $\boldsymbol{X} \overset{\mathcal{D}}{=} \mathrm{Mult}(n, \boldsymbol{p})$ for $n \in \mathbb{N}$ and $\boldsymbol{p} \in (0, 1)^u$ with $\Sigma\boldsymbol{p} = 1$. Further, let $\delta \in (0, 1)$ with $\delta \geq \sqrt{20u/n}$ be given. Then*

$$\mathbb{P}\left[\sum_{i=1}^{u}\left|\frac{X_i}{n} - p_i\right| \;\geq\; \delta\right] \quad\leq\quad 3\exp(-\delta^2 n/25). \qquad \square$$

Finally, we also use the following elementary observation.

Lemma 2.3 (Far-End Left-Tail Bound): *Let $X^{(n)} \overset{\mathcal{D}}{=} \mathrm{Bin}(n, p^{(n)})$ be a sequence of random variables and $k \in \mathbb{N}$ a constant, where $p^{(n)}$ satisfies $p^{(n)} = \omega(\frac{\log n}{n})$ as $n \to \infty$ and is bounded away from 1, i.e., there is a constant $\varepsilon > 0$ so that $p^{(n)} \leq 1 - \varepsilon$ for all n. Then $\mathbb{P}[X^{(n)} \leq k] = o(n^{-c})$ as $n \to \infty$ for any constant c.*

Proof: For better readability, we drop the superscript from $p^{(n)}$ when n is clear from the context. Let c be an arbitrary constant.

$$\begin{aligned}
\mathbb{P}[X^{(n)} \leq k] \cdot n^c &= n^c \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} \\
&\leq n^c \sum_{i=0}^{k} \frac{1}{i!} \underbrace{\left(\frac{p}{1-p}\right)^i}_{\leq (\frac{1-\varepsilon}{\varepsilon})^i} n^i (1-p)^n \\
&\leq n^{c+k}(1-p)^n \cdot O(1) \\
&= \exp\Bigl(n\ln(1-p) + (c+k)\ln(n)\Bigr) \cdot O(1)
\end{aligned}$$

using $\ln(x) \le x-1$, this is

$$\begin{aligned} &\le \exp\Bigl(-np \pm O(\log n)\Bigr) \cdot O(1) \\ &\to 0 \end{aligned}$$

since $p = \omega(\frac{\log n}{n})$. This proves the claim. □

The requirement that $p^{(n)}$ is bounded away from 1 can be lifted, but it simplifies the proof. For the application in the present paper the version is sufficient as is.

2.6. Logarithmic Tree Height w.h.p.

It is a folklore theorem that randomly grown search trees have logarithmic height with high probability. For the present work, the following result is sufficient.

Proposition 2.4 (Probability of Height-Degeneracy): *For any fixed (odd) k holds: the probability that a k-fringe-balanced search tree built from n randomly ordered elements (with or without duplicates) has height $> 13\ln n$ is in $O(1/n^2)$ as $n \to \infty$.* □

We give a formal proof of Proposition 2.4 and a more general discussion in Appendix B.

2.7. Properties of the Entropy Function

This section collects a few useful properties of the entropy function. We start with some observations that follow from well-known results of real analysis. Proofs are given in my Ph.D. thesis [47].

Lemma 2.5 (Elementary Properties of the Entropy Function):
Let $\mathcal{H}_{\ln} : [0,1]^u \to \mathbb{R}_{\ge 0}$ with $\mathcal{H}_{\ln}(\boldsymbol{x}) = \sum_{i=1}^{u} x_i \ln(1/x_i)$ be the base e entropy function.

(a) $\mathcal{H}_{\ln}(\boldsymbol{x}) = \ln(2)\mathcal{H}_{\mathrm{ld}}(\boldsymbol{x})$.

(b) *For all $\boldsymbol{x} \in [0,1]^u$ with $\Sigma\boldsymbol{x} = 1$ we have that $0 \le \mathcal{H}_{\ln}(\boldsymbol{x}) \le \ln(u)$.*

(c) *$\mathcal{H}_{\ln}$ is Hölder-continuous in $[0,1]^u$ for any exponent $h \in (0,1)$, i.e., there is a constant $C = C_h$ such that $|f(\boldsymbol{y}) - f(\boldsymbol{x})| \le C_h u \cdot \|\boldsymbol{y}-\boldsymbol{x}\|_\infty^h$ for all $\boldsymbol{x},\boldsymbol{y} \in [0,1]^u$.*

A possible choice for C_h is given by

$$C_h \;=\; \biggl(\int_0^1 \bigl|\ln(t)+1\bigr|^{\frac{1}{1-h}}\biggr)^{1-h} \qquad (1)$$

For example, $h = 0.99$ yields $C_h \approx 37.61$. □

A beta distributed variable $\mathit{\Pi}$ can be seen as a *random* probability. We can then ask for the *expected* entropy of a Bernoulli trial with probability $\mathit{\Pi}$. (The special case of Lemma 2.6 when $t = 0$ appears Section 5.0 of Bayer [3] and as Exercise 6.2.2–37 of Knuth [27].)

Lemma 2.6 (Expected Entropy of Beta Variables): *Let $t \in \mathbb{N}_0$. For $\mathit{\Pi} \overset{\mathcal{D}}{=} \mathrm{Beta}(t+1, t+1)$ and $k = 2t+1$ we have*

$$\mathbb{E}[\mathcal{H}_{\ln}(\mathit{\Pi}, 1-\mathit{\Pi})] \;=\; H_{k+1} - H_{t+1}. \qquad (2)$$

Proof: We need the following integral, which is a special case of Equation (4.253-1), p. 540, of Gradshteyn and Ryzhik [16] with $r = 1$:

$$\int_0^1 z^{a-1}(1-z)^{b-1}\ln(z)\,dz \;=\; \mathrm{B}(a,b)\big(\psi(a)-\psi(a+b)\big), \qquad (a,b>0). \tag{3}$$

Here $\psi(z) = \frac{d}{dz}\ln(\mathit{\Gamma}(z))$ is the digamma function.

The proof is now simply by computing. By symmetry we have $\mathbb{E}[\mathcal{H}_{\ln}(\mathit{\Pi}, 1-\mathit{\Pi})] = -2\mathbb{E}[\mathit{\Pi}\ln(\mathit{\Pi})]$; using the above integral and the relation $\psi(n+1) = H_n - \gamma$ (Equation (5.4.14) of the DLMF [9]) we find that

$$\begin{aligned}
\mathbb{E}[\mathit{\Pi}\ln(\mathit{\Pi})] &= \int_0^1 x\ln(x)\frac{x^t(1-x)^t}{\mathrm{B}(t+1,t+1)}\,dx \\
&= \frac{\mathrm{B}(t+2,t+1)}{\mathrm{B}(t+1,t+1)}\int_0^1 \ln(x)\frac{x^{t+1}(1-x)^t}{\mathrm{B}(t+2,t+1)}\,dx \\
&\underset{(3)}{=} \frac{t+1}{k+1}\big(\psi(t+2)-\psi(k+2)\big) \\
&= \frac{t+1}{2t+2}\big(H_{t+1}-H_{k+1}\big) \\
&= \frac{1}{2}\big(H_{t+1}-H_{k+1}\big)\;.
\end{aligned}$$

Inserting yields the claim. □

Finally, using the Chernoff bound for the multinomial distribution (Lemma 2.2), we obtain the following concentration property of the entropy of a normalized multinomial variable.

Lemma 2.7 (Concentration of Entropy of Multinomials): *Let $u \in \mathbb{N}$ and $\boldsymbol{p} \in (0,1)^u$ with $\mathit{\Sigma}\boldsymbol{p} = 1$, and $\boldsymbol{X} \overset{\mathcal{D}}{=} \mathrm{Mult}(n,\boldsymbol{p})$. Then it holds that*

$$\mathbb{E}\left[\mathcal{H}_{\ln}\left(\frac{\boldsymbol{X}}{n}\right)\right] \;=\; \mathcal{H}_{\ln}(\boldsymbol{p}) \;\pm\; \rho, \tag{4}$$

where we have for any $\delta \in (0,1)$ with $\delta \ge \sqrt{20u/n}$, $h \in (0,1)$ and C_h as in Equation (1) that

$$\rho \;\le\; C_h\delta^h(1-3e^{-\delta^2 n/25}) \;+\; 3u\ln(u)e^{-\delta^2 n/25}.$$

If $u = O(n^\nu)$ as $n\to\infty$ for a constant $\nu\in[0,1)$, then Equation (4) holds with an error of $\rho = o(n^{-(1-\nu)/2+\varepsilon})$ for any fixed $\varepsilon > 0$.

Proof: We start with the multinomial Chernoff bound: for any $\delta \ge \sqrt{20u/n}$ we have that

$$\begin{aligned}
\mathbb{P}[\|\boldsymbol{X}-\boldsymbol{x}\|_\infty \;\ge\; \delta n] &\;\le\; \mathbb{P}\left[\sum_{i=1}^{u}\left|\frac{X_i}{n}-q_i\right| \;\ge\; \delta\right] \\
&\underset{\text{Lemma 2.2}}{\le} 3\exp(-\delta^2 n/25).
\end{aligned} \tag{5}$$

To use this in estimating $\mathbb{E}\big[|\mathcal{H}_{\ln}(\frac{\boldsymbol{X}}{n}) - \mathcal{H}_{\ln}(\boldsymbol{p})|\big]$, we divide the domain of $\frac{\boldsymbol{X}}{n}$ into the region of values with $\|\cdot\|_\infty$-distance at most δ from $\boldsymbol{p}$, and all others. By Lemma 2.5, $\mathcal{H}_{\ln}$ is Hölder-continuous for any exponent $h \in (0,1)$ with Hölder-constant C_h. Using this and the boundedness of $\mathcal{H}_{\ln}$ (Lemma 2.5–(b)) yields

$$\begin{aligned}\mathbb{E}\Bigl[\Bigl|\mathcal{H}_{\ln}\Bigl(\frac{\boldsymbol{X}}{n}\Bigr) - \mathcal{H}_{\ln}(\boldsymbol{p})\Bigr|\Bigr] &\underset{(5)}{\le} \sup_{\boldsymbol{\xi}:\|\boldsymbol{\xi}\|_\infty<\delta} \bigl|\mathcal{H}_{\ln}(\boldsymbol{p}+\boldsymbol{\xi}) - \mathcal{H}_{\ln}(\boldsymbol{p})\bigr| \cdot (1-3e^{-\delta^2 n/25}) \\ &\qquad + \sup_{\boldsymbol{x}} \bigl|\mathcal{H}_{\ln}(\boldsymbol{x}) - \mathcal{H}_{\ln}(\boldsymbol{p})\bigr| \cdot 3e^{-\delta^2 n/25} \\ &\underset{\text{Lemma 2.5}}{\le} C_h\delta^h \cdot (1-3e^{-\delta^2 n/25}) \;+\; 3\ln(u)e^{-\delta^2 n/25}.\end{aligned}$$

This proves the first part of the claim.

For the second part, we assume $u = O(n^\nu)$, thus $u \le dn^\nu$ for a constant d and large enough n. We obtain an asymptotically valid choice of δ when $\delta = \omega(n^{(\nu-1)/2})$; then for large enough n we will have $\delta > \sqrt{20}dn^{(\nu-1)/2} \ge \sqrt{20u/n}$.

Let now an $\varepsilon > 0$ be given and set $\tilde\varepsilon = \varepsilon + \nu/2$. We may further assume that $\tilde\varepsilon < \frac12$ since the claim is vacuous for larger $\tilde\varepsilon$. We choose a Hölder exponent $h \in (0,1)$ so that $h > \frac{1-2\tilde\varepsilon}{1-\nu}$ (this is possible since $\frac{1-2\tilde\varepsilon}{1-\nu} < 1$ for $\tilde\varepsilon > \nu/2$); this is equivalent to the relation

$$\frac{\nu-1}{2} \;<\; -\frac{\frac12 - \tilde\varepsilon}{h}.$$

We can thus pick c between these two values, e.g., $c = (\frac{\nu-1}{2} - \frac{1/2-\tilde\varepsilon}{h})/2$. Since $c > -(\nu-1)/2$, the choice $\delta = n^c$ guarantees $\delta \ge \sqrt{20u/n}$ for large enough n and we can apply Equation (4).

As we now show, these choices are sufficient to prove the claim $\rho = o(n^{-1/2+\tilde\varepsilon})$. To streamline the computations, we note that (by its definition) we can write h as

$$\begin{aligned} h &= \frac{1-2\tilde\varepsilon}{1-\nu} + \frac{4}{1-\nu}\lambda && \text{for some constant } \lambda > 0 \text{ and} \\ h &= \frac{1-2\tilde\varepsilon}{1-\nu-2\lambda''} = \frac{1-2\tilde\varepsilon}{1+\nu-2\lambda'} && \text{for constants } \lambda'' > 0 \text{ resp. } \lambda' > \nu,\end{aligned}$$

which implies $h\cdot c + (\frac12 - \tilde\varepsilon) = -\lambda < 0$ and $2c+1 = \lambda' > 0$. With these preparations we find (for n large enough to have $u \le dn^\nu$ and $\delta \ge \sqrt{20u/n}$) that

$$\begin{aligned}\rho\cdot n^{1/2-\tilde\varepsilon} &\le C_h\delta^h n^{1/2-\tilde\varepsilon}(1-3\exp(-\delta^2 n/25)) \;+\; 3n^{1/2-\tilde\varepsilon}\ln(u)\exp(-\delta^2 n/25) \\ &\le \underbrace{C_h n^{-\lambda}}_{\to 0}\cdot(1-\underbrace{3\exp(-n^{\lambda'}/25)}_{\to 0}) \;+\; \underbrace{3\nu\ln(dn)\exp\Bigl(-3n^{\lambda'} + (\tfrac12-\tilde\varepsilon)\ln(n)\Bigr)}_{\to 0} \\ &\to 0, \qquad (n\to\infty),\end{aligned}$$

which implies the claim. □

3. Input Models

This section formally defines the input models and some related notation.

3.1. Multiset Model

In the *random multiset permutation model* (a.k.a. *exact-profile model*), we have parameters $u \in \mathbb{N}$, the *universe size,* and $\boldsymbol{x} \in \mathbb{N}^u$, the fixed *profile.* An input under this model always has size $n = \Sigma\boldsymbol{x}$, and is given by a uniformly chosen random permutation of

$$\underbrace{1,\ldots,1}_{x_1 \text{ copies}},\, \underbrace{2,\ldots,2}_{x_2 \text{ copies}},\, \ldots,\, \underbrace{u,\ldots,u}_{x_u \text{ copies}},$$

i.e., the multiset with x_v copies of the number v for $v = 1, \ldots, u$. The random multiset permutation model is a natural generalization of the classical random permutation model (which considers permutations of an ordinary set). I write $C_{\boldsymbol{x}}^{(k)}$ for the (random) number of (ternary) comparisons used by Algorithm 1 to sort a random multiset permutation with profile $\boldsymbol{x}$; I will in the following use $C_{\boldsymbol{x}}$ (omitting k) for conciseness.

3.2. I.I.D. Model

In the *(discrete) i.i.d. model* (a.k.a. *probability model* [26] or *expected-profile model* [47]) with parameters $u \in \mathbb{N}$ and $\boldsymbol{q} \in (0,1)^u$ with $\Sigma\boldsymbol{q} = 1$, an input of size n consists of n i.i.d. (independent and identically distributed) random variables $U_1, \ldots, U_n$ with $U_i \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$ for $i = 1, \ldots, n$. The domain $[u]$ is called the *universe,* and $\boldsymbol{q}$ the (probability vector of the) *universe distribution.*

I denote by X_v, for $v \in [u]$, the number of elements U_i that have value v; the vector $\boldsymbol{X} = (X_1, \ldots, X_u)$ of all these *multiplicities* is called the *profile* of the input $\boldsymbol{U} = (U_1, \ldots, U_n)$. Clearly, $\boldsymbol{X}$ has a multinomial distribution, $\boldsymbol{X} \overset{\mathcal{D}}{=} \mathrm{Mult}(n, \boldsymbol{q})$, with mean $\mathbb{E}[\boldsymbol{X}] = n\boldsymbol{q}$.

The discrete i.i.d. model is a natural complement of the random-permutation model: we draw elements from a *discrete* distribution in the former whereas the latter is equivalent to drawing i.i.d. elements from any *continuous* distribution. By $C_{n,\boldsymbol{q}} = C_{n,\boldsymbol{q}}^{(k)}$, I denote the number of (ternary) comparisons used by Algorithm 1 to sort n i.i.d. $\mathcal{D}(\boldsymbol{q})$ elements (again usually omitting the dependence on k).

3.3. Relation of the Two Models

The two input models—random multiset permutations with profile $\boldsymbol{x}$ resp. $n = \Sigma\boldsymbol{x}$ i.i.d. elements with distribution $\boldsymbol{q} = \boldsymbol{x}/n$—are closely related. Both can be described using an urn with (initially) n balls, where for every $v \in [u]$ exactly x_v balls bear the label v. The multiset model corresponds to drawing n balls from this urn without replacement (ultimately emptying the urn completely), so that the profile of the input is always $\boldsymbol{x}$. The i.i.d. model corresponds to drawing n balls from the the urn *with* replacement; this leaves the urn unchanged and the n draws are mutually independent.

In our reference Quicksort (Algorithm 1), we use the first k elements, i.e., the first k balls drawn from the urn, to determine the pivot: it is the median of this sample. By that, we try to estimate the median of all n balls (initially) in the urn, so as to obtain a balanced split. In the multiset model, the k elements are drawn *without* replacement, whereas in the i.i.d. model, they are drawn *with* replacement from the urn. The algorithm is of course the same in both cases (and indeed chooses sample elements *without* replacement), but the mutual independence of elements in the i.i.d. model implies that the sampling process is (in this latter case) equivalent to sampling *with* replacement.

Now it is well-known that sampling without replacement yields a *better* (i.e., less variable) estimate of the true median than sampling with replacement, and thus leads to more favorable splits in Quicksort! The above reasoning applies for recursive calls, as well, so that on average the multiset model is no more costly for Quicksort than the i.i.d. model: $\mathbb{E}[C_{\boldsymbol{x}}] \le \mathbb{E}[C_{n,\boldsymbol{x}/n}]$. We will thus focus on analyzing the i.i.d. model.[4]

[4]The full version of this paper will discuss the multiset model and the above reasoning in more detail.

4. Previous Work

In the *multiset sorting problem*, we are given a permutation of a *fixed* multiset, where value $v \in [u]$ appears $x_v \in \mathbb{N}$ times, for a total of $n = \Sigma \boldsymbol{x}$ elements. Neither u nor $\boldsymbol{x}$ are known to the algorithm. We consider only comparison-based sorting; unless stated otherwise, all results concern the number of *ternary* comparisons, i.e., one comparison has as result $<$, $=$ or $>$.

4.1. Multiset Sorting

Multiset sorting attracted considerable attention in the literature. Munro and Raman [35] prove a lower bound of $n \operatorname{ld} n - \sum_{i=1}^{u} x_i \operatorname{ld} x_i - n \operatorname{ld} e \pm O(\log n)$ (ternary) comparisons, which can equivalently be written in terms of the entropy as $\mathcal{H}(\boldsymbol{x}/n)n - n \operatorname{ld} e \pm O(\log n)$. We reproduce their main argument in Section 10 and extend it to i.i.d. inputs.

The conceptually simplest algorithm coming close to this bound is to insert elements into a splay tree, collecting all duplicates in linear lists inside the nodes. By "static optimality" of splay trees (Theorem 2 of Sleator and Tarjan [45]), this needs $O(\mathcal{H}(\boldsymbol{x}/n)n)$ comparisons and so is optimal up to a constant factor. That factor is at least 2 (using *semi-splaying*), and we need linear extra space.

Already in 1976, Munro and Spira [34] described simple variants of Mergesort and Heapsort that *collapse* duplicate elements whenever discovered. They are optimal up to an $O(n)$ error term w.r.t. comparisons, but do not work in place. (Their Heapsort requires a non-standard extract-min variant that does not work in place.)

The first in-place method was the adapted Heapsort of Munro and Raman [35]; it does not use the element-collapsing technique, but rather removes all duplicates from the heap in one bulk extract-min operation. None of these methods made it into practical library implementations since they incur significant overhead w.r.t. existing sorting methods when there are not many equal keys in the input.

4.2. Quicksort on Multiset Permutations

Building on Burge's analysis of BSTs [5], Sedgewick analyzed several Quicksort variants on random permutations of multisets in his 1977 article [38]. For fat-pivot Quicksort without sampling, he found the exact average-case result (when every permutation of the multiset is equally likely): $2\mathcal{H}_Q(\boldsymbol{x}) + n - u$ ternary comparisons, where $\mathcal{H}_Q(\boldsymbol{x}) = \sum_{1 \le i < j \le u} x_i x_j / (x_i + \cdots + x_j)$ is the so-called "Quicksort entropy". Interestingly, Sedgewick found fat-pivot partitioning not advisable for practical use at that time; this only changed with the success of the implementation of Bentley and McIlroy [4].

Two decades later, Sedgewick and Bentley [39, 40] combined this exact, but somewhat unwieldy result with the bound $\mathcal{H}_Q(\boldsymbol{q}) \le \mathcal{H}(\boldsymbol{q}) \ln 2$ and concluded that with at most $(2\ln(2)\mathcal{H}(\boldsymbol{x}/n) + 1)n$ ternary comparisons on average, fat-pivot Quicksort is asymptotically optimal in the average case for sorting a random permutation of any fixed multiset[5]—up to the constant factor $2 \ln 2$. The bound $\mathcal{H}_Q(\boldsymbol{q}) \le \mathcal{H}(\boldsymbol{q}) \ln 2$ was noted in a seemingly unrelated context by Allen and Munro [1, Theorem 3.2] that appeared just one year after Sedgewick's Quicksort paper [38]. Allen and Munro studied the move-to-root heuristic for self-organizing BSTs, which they found to have the *same* search costs in the long run as a BST built by

[5] Sedgewick and Bentley [39, 40] compare this number against $\operatorname{ld}\bigl(\binom{n}{x_1,\ldots,x_u}\bigr) = \operatorname{ld}\bigl(n!/(x_1! \cdots x_u!)\bigr)$, i.e., the logarithm of the number of different input orderings (given by the multinomial coefficient). This information-theoretic argument lower bounds the number of needed yes/no questions (i.e., binary comparisons), but more elaboration is necessary for *ternary* comparisons. The lower bound of Munro and Raman [35] (cf. Section 10) uses a reduction to distinct elements and yields the desired bound for ternary comparisons.

inserting elements drawn i.i.d. according to the access distribution until saturation. We will consider this connection between Quicksort and search trees in detail in Section 6.

Katajainen and Pasanen considered Quicksort-based approaches for multiset sorting. They argued (indirectly) that a fat-pivot Quicksort uses on average $2n\ln(n) - \sum_{v=1}^{u} x_v \operatorname{ld}(x_v) \pm O(n)$ comparisons (their Theorem 3), since "Due to the three-way partitions, all redundant comparisons between a pivot and elements equal to the pivot are avoided" [24]. Note however that this only shows that we use at most $\mathcal{H}(\boldsymbol{x}/n)n + (\frac{2}{\operatorname{ld} e} - 1)n \ln n \pm O(n)$ comparisons, which is *not* entropy-optimal.

In a companion paper [25] they described a stable Quicksort version with exact median selection and showed that it needs $O(\mathcal{H}(\boldsymbol{x}/n)n)$ comparisons even in the worst case; however the constant of proportionality is one plus the constant for deterministic median selection, and thus at least 3 [10].

4.3. Fringe-Balanced Trees

The concept of fringe balancing (see Section 2.4) appears under a handful of other names in the literature: *locally balanced search trees* [46], *diminished trees* [17], and *iR / SR trees* [21, 22]. I use the term *fringe-balanced trees* since it is the most vivid term and since it is by now widely adopted in the analysis-of-algorithms community, see, e.g., the relatively recent monograph [11] by Drmota. The name "fringe balanced" itself has its origins in a technique called *fringe analysis*, which Poblete and Munro [36] applied to BSTs that use what they called a "fringe heuristic". The earliest occurrence of "fringe balanced" seems to be in the title of a paper by Devroye [8]; curiously enough, Devroye did not use this term in the main text of the paper.

Along with the different names come slight variations in the definitions; I remark that our definition (deliberatively) differs a bit in the base cases from usual definitions to precisely mimic Quicksort recursion trees.

Many parameters like path length, height and profiles of fringe-balanced trees have been studied when the trees are built from a random permutation of n distinct elements, see, e.g., Drmota [11]. The case of equal elements has not been considered except for the unbalanced case $k = 1$, i.e., ordinary BSTs; see Kemp [26], Archibald and Clément [2].

5. New Results

We now state the main claim of this paper. Note that the error terms of asymptotic approximations in the results mentioned above only involved n, so there was no need to specify an explicit relation between the profile of the multiset $\boldsymbol{x} = (x_1, \ldots, x_u)$, the universe size u, and the number of elements n; here we are not so fortunate. We hence include n as sub- or superscript whenever the dependence on the input size is important.

Theorem 5.1 (Main Result): *Let $(u_n)_{n\in\mathbb{N}}$ be a sequence of integers and $(\boldsymbol{q}^{(n)})_{n\in\mathbb{N}}$ be a sequence of vectors with $\boldsymbol{q}^{(n)} \in (0,1)^{u_n}$ and $\Sigma\boldsymbol{q}^{(n)} = 1$ for all $n \in \mathbb{N}$. Assume further that $(\boldsymbol{q}^{(n)})$ has "many duplicates", i.e., there is a constant $\varepsilon > 0$ so that $\min_{v\in[u_n]} q_v^{(n)} n = \Omega(n^\varepsilon)$ as $n \to \infty$. Abbreviate the corresponding (binary) entropy by $\mathcal{H}_n = \mathcal{H}(\boldsymbol{q}^{(n)})$.*

The number $C_{n,\boldsymbol{q}^{(n)}}$ of (ternary) comparisons used by median-of-k Quicksort with fat-pivot partitioning to sort n elements drawn i.i.d. according to $\mathcal{D}(\boldsymbol{q}^{(n)})$ fulfills

$$\mathbb{E}[C_{n,\boldsymbol{q}^{(n)}}] \;=\; \alpha_k \mathcal{H}_n n \;\pm\; O\Big((\mathcal{H}_n^{1-\delta} + 1)n\Big), \qquad (n \to \infty),$$

for any constant $\delta < \frac{2}{k+5}$. This number is asymptotically optimal up to the factor α_k.

Previous Approaches And Why They Fail for $k > 1$. Sedgewick's analysis [38] is based on explicitly solving the recurrence for the expected number of comparisons. Since it has a *vector,* namely the profile $\boldsymbol{x}$, as parameter, tricky differencing operations are required to obtain a telescoping recurrence. They rely on symmetries that are only present for the most basic version of Quicksort: it has now been 40 years since Sedgewick's article appeared, and not the slightest generalization of the analysis to, say, median-of-3 Quicksort, has been found.

Following our approach outlined in the introduction, we can alternatively compute the expected costs to search each element of the multiset in a BST built by inserting the same elements in random order. The random number of comparisons can thus be written as the scalar product $\boldsymbol{\Gamma}^T\boldsymbol{x}$, where $\boldsymbol{\Gamma}$ is the node-depth vector of the BST (cf. Section 6). For an ordinary BST, once an element is present in the tree, any further insertions of the *same* value are without effect; so we obtain the same tree no matter how many duplicates of this element later follow. This means that the resulting tree has *exactly the same shape* as when we insert elements drawn i.i.d. according to $\mathcal{D}(\boldsymbol{q})$ with $\boldsymbol{q} = \boldsymbol{x}/n$ until saturation. The expected search costs in the latter case are found to be precisely $2\mathcal{H}_Q(\boldsymbol{q}) + 1$ by a comparatively simple argument[6] [1]; multiplying by n gives the Quicksort costs.

For median-of-k Quicksort we obtain k-fringe-balanced trees, and now a certain number of duplicate insertions *do* affect the shape of the tree; after all, this is the way the balancing is achieved in first place (see Section 2.4). As the multiset model corresponds to drawing elements without replacement, the probabilities for the values *change* after each insertion. Analyzing the search cost then essentially reduces to solving the vector-recurrence for Quicksort with pivot sampling that has resisted all attempts for 40 years. One might conclude that the case $k = 1$ can be explicitly solved precisely because we were effectively working in the i.i.d. model instead of the multiset model.

My Assumption. The only hope I see to make progress for $k > 1$ is thus to cling to the i.i.d. model even though it is not equivalent to the multiset model anymore. We thereby retain independent insertions and the analysis of search costs in fringe-balanced trees becomes conceivable. However, we face a new problem: How often we search each value v is now a random variable X_v and the random number of comparisons is $\boldsymbol{\Gamma}^T\boldsymbol{X}$, the *product* of two random variables. Since $\boldsymbol{\Gamma}$ is the node-depth vector of a tree built by inserting a multiset with profile $\boldsymbol{X}$ (in random order), the two random variables are *not independent.*

My assumption—$\Omega(n^\varepsilon)$ expected occurrences of each value—is a simple sufficient condition to circumvent this complication (see Section 8). It is possible to slightly weaken it (at the price of a more clumsy criterion): We can tolerate values with minuscule multiplicities in $O(n^\delta)$ as long as $\delta < \varepsilon$ (so that we have a nonempty *separating range* (δ, ε)), and the total number of these rare values is $O(n^{1-\varepsilon})$.

6. Quicksort and Search Trees with Duplicates

The correspondence discussed in Sections 2.3/2.4 extends to inputs with equal keys if we consider a *weighted* path length in the tree, where the weight of each node is the multiplicity X_v of its key value v. This is best seen in a concrete example.

[6] Assume we search $v \in [u]$. We always have one final comparison with outcome $=$; the remaining comparisons are on the search path and compare v to some $j \neq v$. We compare v to j iff among the values between v and j, j was the *first* to be inserted into the tree, which happens with probability $q_j/(q_v + \cdots + q_j)$. Adding up the indicator variables for these events and multiplying by the probability q_v to search that value v, we obtain $1 + \sum_{v=1}^{u} q_v \sum_{j\neq v} \frac{q_j}{q_v+\cdots+q_j} = 1 + 2\sum_{1\le i<j\le u} \frac{q_i q_j}{q_i+\cdots+q_j} = 2\mathcal{H}_Q(\boldsymbol{x}/n) + 1$.

Example. Let the universe size be $u = 5$ with profile $\boldsymbol{X} = (X_1, \ldots, X_5)$. Assume that $\boldsymbol{X} \geq k$ so that each of the five values is used as a pivot in *exactly* one partitioning step, and the leaves in the final tree will be empty. Assume we obtain the recursion tree shown in Figure 2.

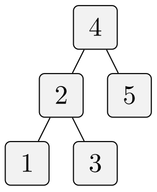

Figure 2: Exemplary recursion tree for $u = 5$. Each node represents a partitioning step, with the given pivot value. Child links correspond to child recursive calls. Empty leaves are not shown. The node-depths vector for this tree is $\boldsymbol{\Gamma} = (3, 2, 3, 1, 2)$.

The traditional recurrence for Quicksort sums up the costs of all partitioning steps. Each such uses one comparison per element, plus the comparisons for selecting the pivot, say, at most $c \cdot k$; ignoring the latter, we essentially sum up the subproblem sizes of all recursive calls. For our recursion tree this yields

$$\begin{cases} X_1 + X_2 + X_3 + X_4 + X_5 & \pm\ c \cdot k \\ X_1 + X_2 + X_3 & \pm\ c \cdot k \\ X_1 & \pm\ c \cdot k \\ \qquad\qquad X_3 & \pm\ c \cdot k \\ \qquad\qquad\qquad\qquad\quad X_5 & \pm\ c \cdot k \end{cases}$$

comparisons, (each row corresponding to one partitioning step). For example, the step with pivot 2 gets as input all elements smaller than 4, i.e., $X_1 + X_2 + X_3$ many, so the number of comparisons used in this step is $X_1 + X_2 + X_3 \pm c \cdot k$ (for some constant c depending on the median-selection algorithm).

The key observation is that we can also read the result *column-wise,* aligning the rows by key values: up to the given error terms, we find that *sorting costs are the cost of searching each input element in the (final) recursion tree!* For example, searching 3 in the tree from Figure 2, we first go left, then right and then find 3 as the pivot, so the costs are 3—which is precisely the coefficient of X_3 in the overall costs. In vector form, we can write the search costs as $\boldsymbol{\Gamma}^T \boldsymbol{X}$, where $\boldsymbol{\Gamma}$ is the *node-depths vector* of the recursion tree, i.e., the vector of depths of nodes sorted by their keys. In the example $\boldsymbol{\Gamma} = (3, 2, 3, 1, 2)$. This is nothing else than a weighted path length; (more precisely, a weighted internal path length where we include *both* endpoints to determine the length of a path).

6.1. Recursion Trees

We formalize the result of the above example in two lemmas. The first one formally captures the correspondence of Quicksort and search trees.

Lemma 6.1 (Recursion Trees): *For any input $\boldsymbol{U} = (U_1, \ldots, U_n)$ (with or without duplicates) the following processes execute the same set of (ternary) key comparisons and produce the same tree shape:*

(1) sorting $\boldsymbol{U}$ with Algorithm 1 and storing the recursion tree, ignoring any calls to Insertionsort,

(2) inserting $\boldsymbol{U}$ successively into an initially empty k-fringe-balanced tree using Algorithm 2.

Proof: We prove the equivalence by induction on n. If $n \leq k - 1$, Quicksort stops (it passes control directly to Insertionsort which we ignore), so the recursion tree consists of one leaf only.

Likewise in the search tree, all elements are gathered in the single leaf and no comparisons happen. So assume the claim holds for inputs with less than k elements.

If now $n \geq k$, Quicksort chooses a pivot P from the first k elements, compares all elements to P, and divides the input into segments $\boldsymbol{U}^{(1)}$ and $\boldsymbol{U}^{(2)}$ containing the elements strictly smaller resp. strictly larger than P. All duplicates of P are put in place and vanish from the recursion tree.

Now consider what happens upon inserting the kth element in the search tree. This is the first time a leaf is split and the key for the inner node (the root of the tree) is chosen from the first k inserted elements overall. We thus choose same value P that was chosen as pivot in Quicksort. The other elements from the leaf are compared to P and inserted into one of the two new leaves (unless they are duplicates of P). Any later insertions must start at the root, so each of these elements are also compared to P before the insertion continues in one of the subtrees, or stops if a duplicate of P is found.

So we execute the same set of comparisons in both processes at the root of the tree. Towards applying the inductive hypothesis for recursive calls resp. subtrees, we note that the relative order of elements is retained in both processes, so the elements inserted in the left/right child of the root are exactly $\boldsymbol{U}^{(1)}$ resp. $\boldsymbol{U}^{(2)}$ in both cases. The claim thus follows by induction. □

The proof relies on the fact that Algorithm 1 retains the relative order of elements, but practical non-stable, in-place implementations (e.g., those in [4, 41]) do not fulfill this requirement. However, a weaker version of Lemma 6.1 remains valid for any fat-pivot partitioning method: the two processes always have the same *distribution* of the number of comparisons and tree shapes over *randomly ordered* inputs. In this general case, sorting the input corresponds to inserting a (different, but uniquely determined) permutation of the input into a fringe-balanced tree; my Ph.D. thesis [47] gives some more details on that. Such a distributional version of Lemma 6.1 is sufficient for all results in this paper, so our results apply to practical implementations, as well.

6.2. Search Costs

The second lemma relates the costs to build a search tree to the cost of searching in the final tree.

Lemma 6.2 (Search Costs): *Let $\boldsymbol{U} = (U_1, \ldots, U_n)$ be an input (with or without duplicates), let $\mathcal{T}$ be built from $\boldsymbol{U}$ by successive insertions using Algorithm 2 and let $B_{\boldsymbol{U}}$ be the number of comparisons done in this process. Assume there are I inner nodes in $\mathcal{T}$. Searching each element of $\boldsymbol{U}$ in $\mathcal{T}$ using Algorithm 3 (ignoring the sequential search in leaves) uses $B_{\boldsymbol{U}} \pm c \cdot Ik$ comparisons.*

Proof: The elements in the leaves of $\mathcal{T}$ require the exact same comparisons for insert and search to find the path to their leaf. (We ignore the sequential search within the leaf). So consider the key $v \in [u]$ of an inner node. The first occurrences of v in $\boldsymbol{U}$ are simply added to the leaf that later becomes the inner node with label v. Each of these entails one comparison more when searching for it in $\mathcal{T}$ as we paid when inserting it (namely the last comparison with v that identifies them as equal). However, there are be at most k such elements before the leaf overflows, and for the remaining duplicate insertions of v, we *do* the last comparison (same as in the search). So searching pays up to $I \cdot k$ comparisons more. On the other hand, whenever a leaf overflows, we need a certain number of comparisons to select the median, say at most $c \cdot k$ for some constant c. So insertion pays up to $I \cdot ck$ comparisons more than insertion. □

In general, I can be as large as $n/2$, but it is certainly bounded by the number u of distinct keys in the input. Together this yields the formal generalization of our example.

Corollary 6.3 (Quicksort and Search Costs): *Let $\boldsymbol{U} = (U_1, \ldots, U_n)$ be an input over universe $[u]$. Let $\mathcal{T}$ be built from $\boldsymbol{U}$ by successive insertions using Algorithm 2 and denote by $\boldsymbol{\Gamma}$ its node-depth vector. The cost of Algorithm 1 on $\boldsymbol{U}$ (ignoring Insertionsort) is $\boldsymbol{\Gamma}^T \boldsymbol{X} \pm cuk$ for $\boldsymbol{X}$ the profile of $\boldsymbol{U}$.* $\square$

Up to an error term of $O(u)$, we can thus consider search costs in fringe-balanced trees instead of sorting costs in Quicksort. For a special class of such trees, we will be able to determine the search costs: the *saturated fringe-balanced trees.* They are the subject of the next section.

7. Saturated Fringe-Balanced Trees

Consider a k-fringe-balanced tree $\mathcal{T}$ built by successively inserting elements drawn i.i.d. $\mathcal{D}(\boldsymbol{q})$ (for a fixed universe distribution $\boldsymbol{q}$) into an initially empty tree. How does this tree evolve if we continue inserting indefinitely? Since the universe $[u]$ is finite and duplicate insertions do not alter $\mathcal{T}$, the process reaches a stationary state almost surely. We call the trees corresponding to such stationary states *saturated trees* (w.r.t. the given, fixed universe). The expected search costs in saturated trees will play a key role in the analysis of Quicksort.

We start by developing a stochastic description of the shape of a random saturated tree $\mathcal{T}$ and set up a recurrence equation for the expected search costs.

7.1. Stochastic Description

Let $\boldsymbol{q} \in (0,1)^u$ with $\Sigma\boldsymbol{q} = 1$ be given. The distribution function of the universe distribution $U \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$ is $F_U(v) = \mathbb{P}[U \le v] = \sum_{i=1}^{\lfloor v \rfloor} q_i$ for $v \in [0, u+1)$, and I denote its (generalized) inverse by $F_U^{-1} : (0,1) \to [1..u]$ with $F_U^{-1}(x) = \inf\{v \in [1..u] : F_U(v) \ge x\}$.

Let P be the label of the root of $\mathcal{T}$; the distribution of P is a key ingredient to (recursively) describe saturated trees. (P is also the pivot chosen in the first partitioning step of median-of-k Quicksort.) When the first leaf overflows, P is chosen as the median of the first $k = 2t+1$ inserted values, which are i.i.d. $\mathcal{D}(\boldsymbol{q})$ distributed, so it is given by

$$P \;\overset{\mathcal{D}}{=}\; f_U^{-1}(\mathit{\Pi}), \tag{6}$$

where $\mathit{\Pi}$ has a $\mathrm{Beta}(t+1, t+1)$ distribution, i.e., it has density $f_{\mathit{\Pi}}(z) = z^t(1-z)^t / \mathrm{B}(t+1, t+1)$. $\mathrm{B}(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ is the *beta function.* This is the generalized *inversion method of random sampling,* see Devroye [7, Sec. V.3.4], illustrated in our Figure 3, which is based on the fact that $\mathrm{Beta}(t+1, t+1)$ is the distribution of the median of $2t+1$ i.i.d. uniformly in $(0,1)$ distributed random variables (see, e.g., [7, Sec. I.4.3]). For convenient notation, we write $\boldsymbol{D} = (D_1, D_2) = (\mathit{\Pi}, 1-\mathit{\Pi})$ for the induced *spacings*; see Figure 4.

We further denote by V_1 and V_2 the probability that a random element $U \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$ belongs to the left resp. right subtree of the root, and by $H = \mathbb{P}[U = P]$ the probability to "hit" the root's value. These quantities are fully determined by P (see also Figure 4):

$$V_1 \;=\; q_1 + \cdots + q_{P-1}, \tag{7.1}$$

$$V_2 \;=\; q_{P+1} + \cdots + q_u, \tag{7.2}$$

$$H \;=\; q_P. \tag{7.3}$$

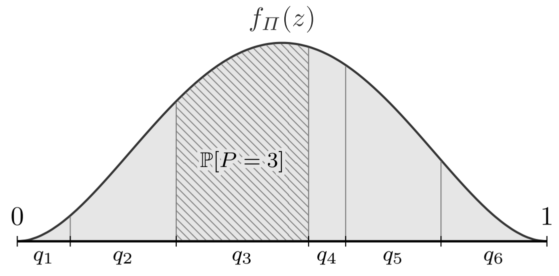

Figure 3: Illustration of pivot sampling in the i.i.d. modelwith $u=6$. Π is the x-coordinate of a point uniformly chosen in the gray area (the area under the curve), and P is the index of the interval this point lies in.

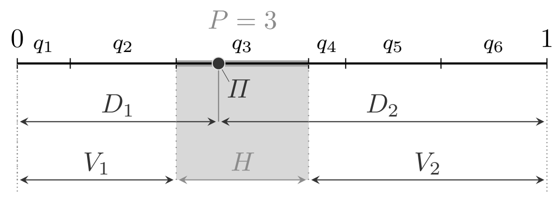

Figure 4: Relation of the different quantities in the stochastic description.

(In the boundary case $P=1$, we have $V_1=0$, and similarly $V_2=0$ for $P=u$.) Finally, we denote by $\boldsymbol{Z_1}$ and $\boldsymbol{Z_2}$ the "zoomed-in" universe distributions in the left resp. right subtree:

$$\boldsymbol{Z_1} = \left(\frac{q_1}{V_1},\ldots,\frac{q_{P-1}}{V_1}\right), \tag{8.1}$$

$$\boldsymbol{Z_2} = \left(\frac{q_{P+1}}{V_2},\ldots,\frac{q_u}{V_2}\right). \tag{8.2}$$

$\boldsymbol{Z_1}$ is not well-defined for $P=1$; we set it to the *empty* vector $\boldsymbol{Z_1}=()$ in this case. Similarly $\boldsymbol{Z_2}=()$ for $P=u$.

7.2. Search Costs

Let $\mathcal{T}$ be a random k-fringe-balanced tree resulting from inserting i.i.d. $\mathcal{D}(\boldsymbol{q})$ elements until saturation. Each value $v\in[u]$ then appears as the key of one inner node of $\mathcal{T}$; let $\varGamma_v$ denote its depth, i.e., the (random) number of nodes on the path (including endpoints) from the root to the node containing v in the (random) tree $\mathcal{T}$. The vector $\boldsymbol{\varGamma}=(\varGamma_1,\ldots,\varGamma_u)$ is called the (random) *node-depths vector* of $\mathcal{T}$ (see Figure 2 (page 15) for an example). Finally, we write $A_{\boldsymbol{q}}=\boldsymbol{\varGamma}^T\boldsymbol{q}$. This is the average depth of a node drawn according to $\mathcal{D}(\boldsymbol{q})$ in the (random) tree $\mathcal{T}$; note that we *average* the costs over the searched *key*, but consider the *tree* fixed; so $A_{\boldsymbol{q}}$ is a random variable since $\mathcal{T}$ remains random: the (weighted) average node depth in a random saturated k-fringe-balanced tree.

The expected node depth, or equivalently, the expected search cost in the tree, can be described recursively: The root contributes one comparison to any searched element, and with probability H the search stops there. Otherwise, the sought element is in the left or right subtree with probabilities V_1 resp. V_2, and the expected search costs in the subtrees are given recursively by $A_{\boldsymbol{Z_1}}$ and $A_{\boldsymbol{Z_2}}$. With the notation from above, this yields a *distributional recurrence* for $A_{\boldsymbol{q}}$:

$$A_{\boldsymbol{q}} \overset{\mathcal{D}}{=} 1 + V_1 A^{(1)}_{\boldsymbol{Z_1}} + V_2 A^{(2)}_{\boldsymbol{Z_2}}, \qquad (u\ge 1), \tag{9.1}$$

$$A_{()} = 0, \tag{9.2}$$

where $(A^{(1)}_{\boldsymbol{q}})$ and $(A^{(2)}_{\boldsymbol{q}})$ are independent copies of $(A_{\boldsymbol{q}})$, which are also independent of $(V_1,V_2,\boldsymbol{Z_1},\boldsymbol{Z_2})$.

8. Quicksort With Many Duplicates

As discussed in Section 5, the attempts to analyze median-of-k Quicksort in full generality have failed; I therefore confine myself to the following restricted i.i.d. model.

Definition 8.1 (Many Duplicates): *Let $(\boldsymbol{q}^{(n)})_{n\in\mathbb{N}}$ be a sequence of stochastic vectors, where $\boldsymbol{q}^{(n)}$ has u_n entries, i.e., $\boldsymbol{q}^{(n)} \in (0,1)^{u_n}$ and $\Sigma\boldsymbol{q}^{(n)} = 1$, for all $n \in \mathbb{N}$. An input of size $n \in \mathbb{N}$ under the i.i.d. model for $(\boldsymbol{q}^{(n)})_{n\in\mathbb{N}}$ consists of the n i.i.d. $\mathcal{D}(\boldsymbol{q}^{(n)})$ distributed random variables. $(\boldsymbol{q}^{(n)})_{n\in\mathbb{N}}$ is said to have many duplicates if there is a constant $\varepsilon > 0$ so that $\mu_n = \Omega(n^{-1+\varepsilon})$ as $n \to \infty$ where $\mu_n := \min_r q_r^{(n)}$ is the smallest probability.*

This condition ensures that every value occurs $\Omega(n^{\varepsilon})$ times in expectation. (It might hence be more appropriate to say many duplicates *of each kind,* but I refrain from doing so for conciseness.) With many duplicates, we expect few *degenerate* inputs in the following sense.

Definition 8.2 (Profile-Degenerate Inputs): *Let $\nu \in [0,1)$ and $k \in \mathbb{N}$. An input vector $\boldsymbol{U} = (U_1,\ldots,U_n) \in [u]^n$ of size n is called (ν,k)-profile-degenerate if not all u elements of the universe appear at least k times in the first $n_T = \lceil n^{\nu}\rceil$ elements $U_1,\ldots,U_{n_T}$ of $\boldsymbol{U}$. If the parameters are clear from the context or are not important, we call $\boldsymbol{U}$ simply profile-degenerate.*

For non-degenerate inputs, the recursion tree will depend only on the first n_T elements, and the profile of the remaining $n - n_T$ elements is *independent of this tree.* By choosing n_T large enough to have non-degenerate inputs w.h.p., but small enough to keep the contribution of the first n_T elements to the search costs negligible, we obtain the following theorem.

Theorem 8.3 (Separation Theorem): *Consider median-of-k Quicksort with fat-pivot partitioning under a discrete i.i.d. model with many duplicates. The expected number of comparisons fulfills*

$$\mathbb{E}[C_{n,\boldsymbol{q}^{(n)}}] \;=\; \mathbb{E}[A_{\boldsymbol{q}^{(n)}}]\cdot n \;\pm\; O(n^{1-\varepsilon}), \qquad (n\to\infty), \tag{10}$$

for a constant $\varepsilon > 0$; more precisely, we need $\varepsilon \in (0,\tilde{\varepsilon})$ when $\mu_n = \min_r q_r^{(n)} = \Omega(n^{-1+\tilde{\varepsilon}})$.

Recall that $\mathbb{E}[A_{\boldsymbol{q}^{(n)}}]$ is the expected search cost in a *saturated* k-fringe-balanced tree built from $\mathcal{D}(\boldsymbol{q}^{(n)})$. It depends only on the universe distribution $\boldsymbol{q}^{(n)}$ (and k), but *not* (directly) on n. We have therefore separated the influence of n and $\boldsymbol{q}$ (for inputs with many duplicates), and can investigate $\mathbb{E}[A_{\boldsymbol{q}}]$ in isolation in the next section. The remainder of this section is devoted to the proof of the separation theorem.

Proof of Theorem 8.3: By Corollary 6.3 we can study search costs in fringe-balanced trees; the main challenge is that this tree is built form the same input that is used for searching. In other words, $\boldsymbol{\Gamma}$ and $\boldsymbol{X}$ are not independent; for non-degenerate inputs, they are however almost so.

We start noting the following basic fact that we will use many times: In any discrete i.i.d. model, $u_n \le \frac{1}{\mu_n}$ for all n. In particular, an i.i.d. model with many duplicates has $u_n = O(n^{1-\varepsilon})$. This is easy to see: Since μ_n is the smallest entry of $\boldsymbol{q}^{(n)}$ we have $1 = \Sigma\boldsymbol{q}^{(n)} \ge u_n\mu_n$, so $u_n \le 1/\mu_n$. The second part follows directly from Definition 8.1.

Probability of degenerate profiles. We can bound the probability of degenerate inputs using Chernoff bounds. An elementary far-end lower-tail bound (Lemma 2.3) actually yields the following slightly stronger asymptotic result.

Lemma 8.4 (Non-Degenerate w.h.p.): *Assume an i.i.d. model with* $\mu_n = \min_v q_v^{(n)} = \Omega(n^{-\rho})$ *for* $\rho \in [0,1)$ *and let* $k \in \mathbb{N}$ *and* $\rho < \nu < 1$. *Then the probability of an input of size* n *to be* (ν, k)*-profile-degenerate is in* $o(n^{-c})$ *as* $n \to \infty$ *for any constant* c.

Proof: Let $\rho \in [0,1)$, k and $\nu \in (\rho, 1)$ be given. Set $\varepsilon = \nu - \rho > 0$ and denote by $\boldsymbol{Y} = \boldsymbol{Y}^{(n)}$ the profile of the first $n_T = \lceil n^\nu \rceil$ elements of the input. Clearly $\boldsymbol{Y}^{(n)} \overset{\mathcal{D}}{=} \mathrm{Mult}(n_T; \boldsymbol{q}^{(n)})$. Assume w.l.o.g. that the minimal probability is always $q_1^{(n)} = \mu_n$. A standard application of the *union bound* yields

$$\begin{aligned} \mathbb{P}[\neg \boldsymbol{Y}^{(n)} \geq k] &= \mathbb{P}\left[\bigvee_{v=1}^{u_n} Y_v^{(n)} < k\right] \\ &\leq \sum_{v=1}^{u_n} \mathbb{P}[Y_v^{(n)} < k] \\ &\leq u_n \cdot \mathbb{P}[Y_1^{(n)} < k]. \end{aligned} \tag{11}$$

Now $Y_1^{(n)} \overset{\mathcal{D}}{=} \mathrm{Bin}(n_T, \mu_n)$ with $\mu_n = \Omega(n^{-\rho}) = \Omega(n_T^{-\rho/\nu}) = \omega(\frac{\log n_T}{n_T})$, and we always have $\mu_n \leq \frac{1}{2} < 1$, so we can apply Lemma 2.3: for any given constant c, we have $\mathbb{P}[Y_1^{(n)} < k] = o(n_T^{-(c+1)/\nu})$. Since $u_n = o(\frac{n}{\log n}) = o(n)$ we find that

$$\begin{aligned} n^c \cdot \mathbb{P}[\neg \boldsymbol{Y}^{(n)} \geq k] &\underset{(11)}{\leq} n^c \, u_n \, \mathbb{P}[Y_1^{(n)} < k] \\ &= o(n^{c+1}) \cdot o\Big(n_T^{-\frac{c+1}{\nu}}\Big) \\ &= o(1), \end{aligned}$$

since $n_T \sim n^\nu$. So the input is (ν, k)-degenerate with high probability. $\square$

In Theorem 8.3, we assume an i.i.d. model with many duplicates, i.e., $\mu_n = \Omega(n^{-1+\tilde{\varepsilon}})$ with $\tilde{\varepsilon} \in (0,1]$, and an $\varepsilon \in (0, \tilde{\varepsilon})$ is given. We set

$$\nu \;:=\; \frac{(1-\tilde{\varepsilon}) + (1-\varepsilon)}{2} \;=\; 1 - \frac{\tilde{\varepsilon} + \varepsilon}{2} \;\in\; (1-\tilde{\varepsilon}, 1-\varepsilon).$$

Then, by Lemma 8.4, an input of size n is (ν, k)-degenerate with probability in $o(n^{-c})$ for all c. This also means that the overall cost contribution of degenerate inputs to expected costs is in $o(n^{-c})$ for all c, and hence covered by the error term in Equation (10), since costs for any input are at most quadratic in n.

We will thus, for the remainder of this proof, assume that the input is not (ν, k)-degenerate, i.e., each of the values of the universe appears at least k times among the first $n_T = \lceil n^\nu \rceil$ elements.

Independence of Profiles and Trees. We now turn to the distribution of the recursion trees. The shape of the recursion tree is determined by at most $u \cdot k$ elements: we have at most u partitioning rounds since each of the u elements of the universe becomes a pivot in at most one partitioning step, and each partitioning step inspects k elements for choosing its pivot.

Also, for each of the u values in the universe, at most the first k occurrences in the input, reading from left to right, can influence the tree: if a value $v \in [u]$ is already contained in an inner node, all further duplicates of v are ignored. Otherwise, all occurrences of v must appear in a single leaf, which can hold up to $k-1$ values, so there are never more than $k-1$ copies of

v in the tree. The leaf will overflow at the latest upon inserting the kth occurrence of v, and then a new internal node with pivot v is created.

In a non-degenerate input, the first k duplicates appear among the first n_T elements $U_1, \ldots, U_{n_T}$ of the input, so all pivots are chosen based on these elements only. Moreover, after these n_T insertions, all u values appear as labels of inner nodes. All leaves are empty then and remain so for good: the recursion tree has reached a saturated state.

We denote by $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(\boldsymbol{q}^{(n)})$ the node-depths vector for the final recursion tree and by $\tilde{\boldsymbol{X}}$ the profile of $U_{n_T+1}, \ldots, U_n$. Since they are derived from disjoint ranges of the i.i.d. input, $\boldsymbol{\Gamma}$ and $\tilde{\boldsymbol{X}}$ are stochastically independent.

Overall Result. We now have all ingredients to compute the overall costs of Quicksort. Recall that $u_n = O(n^{1-\tilde{\varepsilon}})$ and $n_T \sim n^{\nu}$ with $1 - \tilde{\varepsilon} < \nu < 1 - \varepsilon$. Since a recursion tree cannot have a path longer than u, we always have $\boldsymbol{\Gamma} \le cu$ for a fixed constant c that depends only on the cost measure, i.e., $\Gamma_v = O(n^{1-\tilde{\varepsilon}})$ for all $v \in [u]$. However, this estimate is very pessimistic; we know that randomly grown trees have logarithmic height w.h.p. (Section 2.6 resp. Appendix B), so $\Gamma_v = O(\log n)$ for all v with high probability. To be concrete, the height is $> 13 \ln n$ with probability in $O(n^{-2})$ by Proposition 2.4.

We therefore further split the set of non-profile-degenerate inputs into *"height-degenerate"* ones where the height of the resulting recursion tree is $> 13 \ln n$ and all other ones. This gives the following stochastic representation conditional on the input being not (ν, k)-profile-degenerate.

$$\begin{aligned}
C_{n,\boldsymbol{q}^{(n)}} &= \boldsymbol{\Gamma}^T \boldsymbol{X} \;\pm\; O(u) \\
&= \boldsymbol{\Gamma}^T \tilde{\boldsymbol{X}} \;\pm\; n_T \|\boldsymbol{\Gamma}\|_\infty \;\pm\; O(u) \\
&\overset{\mathcal{D}}{=} \boldsymbol{\Gamma}^T \hat{\boldsymbol{X}} \;\pm\; 2n_T \|\boldsymbol{\Gamma}\|_\infty \;\pm\; O(u) \\
&= \boldsymbol{\Gamma}^T \hat{\boldsymbol{X}} \;\pm\; O\Big(n_T \big(\mathbb{1}_{\{\text{not height-deg.}\}} \cdot \log n + \mathbb{1}_{\{\text{height-deg.}\}} \cdot u\big)\Big) \\
&= \boldsymbol{\Gamma}^T \hat{\boldsymbol{X}} \;\pm\; O\Big(n^{\nu} \big(\mathbb{1}_{\{\text{not height-deg.}\}} \cdot \log n + \mathbb{1}_{\{\text{height-deg.}\}} \cdot n^{1-\tilde{\varepsilon}}\big)\Big),
\end{aligned}$$

where $\hat{\boldsymbol{X}}$ is independent of $\boldsymbol{\Gamma}$ and $\hat{\boldsymbol{X}} \overset{\mathcal{D}}{=} \mathrm{Mult}(n, \boldsymbol{q}^{(n)})$. We thus find that

$$C_{n,\boldsymbol{q}^{(n)}} \overset{\mathcal{D}}{=} \boldsymbol{\Gamma}^T \hat{\boldsymbol{X}} \;\pm\; O(n^{1-\varepsilon}), \qquad \text{(input neither profile- nor height-degenerate).} \tag{12}$$

Taking expectations over all non-degenerate inputs in Equation (12), exploiting independence, and inserting $\mathbb{P}[\text{height-deg.}] = O(1/n^2)$ (Proposition 2.4) yields

$$\begin{aligned}
\mathbb{E}[C_{n,\boldsymbol{q}^{(n)}}] &= \mathbb{E}[\boldsymbol{\Gamma}^T \hat{\boldsymbol{X}}] \;\pm\; O(n^{1-\varepsilon}) \\
&= \mathbb{E}[\boldsymbol{\Gamma}]^T \cdot \mathbb{E}[\hat{\boldsymbol{X}}] \;\pm\; O(n^{1-\varepsilon}) \\
&= \underbrace{\big(\mathbb{E}[\boldsymbol{\Gamma}]^T \cdot \boldsymbol{q}^{(n)}\big)}_{\mathbb{E}[A_{\boldsymbol{q}^{(n)}}]} \cdot n \;\pm\; O(n^{1-\varepsilon}),
\end{aligned} \tag{13}$$

with $A_{\boldsymbol{q}^{(n)}}$ as given in Equation (9) on page 18. As argued above, the contribution of profile-degenerate inputs is in $o(n^{-c})$ for any c and thus covered by $O(n^{1-\varepsilon})$, so Equation (13) holds also for the unconditional expectation. This concludes the proof of Theorem 8.3. □

9. Expected Search Costs in Saturated Trees

The remaining step of the analysis consists in computing $\mathbb{E}[A_{\boldsymbol{q}}]$, the expected search costs in saturated k-fringe-balanced trees built from i.i.d. $\mathcal{D}(\boldsymbol{q})$ elements. Previous work only covers the unbalanced BST case ($k = 1$), where the result is known *exactly*: $\mathbb{E}[A_{\boldsymbol{q}}] = 2\mathcal{H}_Q(\boldsymbol{q}) + 1 \le 2\mathcal{H}_{\ln}(\boldsymbol{q}) + 1$ where $\mathcal{H}_Q(\boldsymbol{q}) = \sum_{1\le i<j\le u} q_i q_j/(q_i + \cdots + q_j)$ [1, Theorems 3.1 and 3.4].

Since the recurrence equation for the expected search costs (Equation (9) on page 18) seems hard to solve, we try to relate it to a quantity that we already know: the *entropy of the universe distribution.* The entropy function satisfies an *aggregation property:* intuitively speaking, we do not change the entropy of the final outcomes if we *delay* some decisions in a random experiment by first deciding among whole groups of outcomes and then continuing within the chosen group. Indeed, this property was Shannon's third fundamental requirement when he introduced his entropy function in 1948 [44, p. 393].

The aggregation property can nicely be formalized in terms of trees, see Lemma 6.2.2E of Knuth [27, p. 444], and we can use it in a search tree to express the entropy of node access probabilities as the entropy of the split corresponding to the root plus the entropies of its subtrees, weighted by their respective total probabilities. More specifically in our setting, we have $\boldsymbol{q} \in [0,1]^u$ with $\Sigma\boldsymbol{q} = 1$ and we condition on the value $P \in [u]$ in the root. Using the notation from Section 7.1 —the zoomed-in distributions of the subtrees $\boldsymbol{Z}$, the probability to go into these subtrees $\boldsymbol{V}$, and the probability to access the root H—we find that

$$\mathcal{H}(\boldsymbol{q}) \;=\; \mathcal{H}(V_1, H, V_2) \;+\; \sum_{r=1}^{2} V_r \mathcal{H}(\boldsymbol{Z_r})\,. \tag{14}$$

(By linearity, $\mathcal{H}$ can be w.r.t. any base; we will use it with $\mathcal{H}_{\ln}$ below.)

If we likewise condition on the root value P in the recursive description of the search costs, we find for the expected search costs that

$$\mathbb{E}[A_{\boldsymbol{q}}] \;=\; 1 \;+\; \sum_{r=1}^{2} V_r \mathbb{E}[A_{\boldsymbol{Z_r}}]\,. \tag{15}$$

$\mathcal{H}(\boldsymbol{q})$ and $\mathbb{E}[A_{\boldsymbol{q}}]$ fulfill the *same form* of recurrence, only with a different toll function: 1 instead of $\mathcal{H}(V_1, H, V_2)$! We thus try to relate these two toll functions by obtaining bounds on $\mathcal{H}(V_1, H, V_2)$, and then extend this relation inductively to $\mathcal{H}(\boldsymbol{q})$ and $\mathbb{E}[A_{\boldsymbol{q}}]$. The technical difficulties in doing so are that $\mathcal{H}(V_1, H, V_2)$ is very sensitive to $\boldsymbol{q}$, so we have to do a case distinction to obtain bounds on it. We hence cannot give matching upper and lower bounds for $\mathcal{H}(V_1, H, V_2)$, which necessitates the introduction of second order terms to account for the slack (cf. the constant d below). Separately deriving upper and lower bounds on $\mathbb{E}[A_{\boldsymbol{q}}]$ from that in terms of $\mathcal{H}(\boldsymbol{q})$, and making them match asymptotically in the leading term, we obtain the following result.

Theorem 9.1 (Expected Search Costs): *Let a sequence of universe distributions $(\boldsymbol{q}^{(n)})_{n\in\mathbb{N}}$ be given for which $\mathcal{H}_n := \mathcal{H}_{\mathrm{ld}}(\boldsymbol{q}^{(n)}) \to \infty$ as $n \to \infty$. The expected search costs of a saturated k-fringe-balanced tree (with $k = 2t + 1$) built from i.i.d. $\mathcal{D}(\boldsymbol{q}^{(n)})$ keys is given by*

$$\mathbb{E}[A_{\boldsymbol{q}^{(n)}}] \;=\; \alpha_k \mathcal{H}_{\mathrm{ld}}(\boldsymbol{q}^{(n)}) \;\pm\; O\Big({\mathcal{H}_n}^{\frac{t+2}{t+3}} \log(\mathcal{H}_n)\Big), \qquad (n \to \infty).$$

Proof: We start with the upper bound. We actually derive a whole class of upper bounds characterized by a parameter ε.

Lemma 9.2 (Upper Bound): *Let $\mathbb{E}[A_{\boldsymbol{q}}]$ satisfy Equation* (15), *and let $\varepsilon \in (0,1)$ be given. Define*

$$\begin{aligned} c \;=\; c_\varepsilon \;&=\; \frac{1}{\tilde{H} - 4\varepsilon\tilde{h}}\,, \\ d \;=\; d_\varepsilon \;&=\; \frac{(t+1)\,\mathrm{B}(t+1,t+1)}{\varepsilon^{t+2}(1-\varepsilon)^t}\,, \\ \textit{where}\;\; \tilde{H} \;&=\; H_{k+1} - H_{t+1} \\ \textit{and}\;\; \tilde{h} \;&=\; H_k - H_t\,. \end{aligned}$$

If $c \ge 0$, we have that $\mathbb{E}[A_{\boldsymbol{q}}] \le c \cdot \mathcal{H}_{\ln}(\boldsymbol{q}) + d$ for all stochastic vectors $\boldsymbol{q}$.

Proof: Let ε with $c = c_\varepsilon \ge 0$ be given. Note that $d = d_\varepsilon \ge 0$ holds for all ε. The proof is by induction on u, the size of the universe. If $u = 0$, i.e., $\boldsymbol{q} = ()$, we have $\mathbb{E}[A_{\boldsymbol{q}}] = 0$, see Equation (9.2). Since $d \ge 0$ and here $\mathcal{H}_{\ln}(\boldsymbol{q}) = 0$, the claim holds.

Now assume that $u \ge 1$ and the claim holds for all (strictly) smaller universe sizes. We start by taking expectations in Equation (9) and conditioning on the pivot value P:

$$\begin{aligned} \mathbb{E}[A_{\boldsymbol{q}}] \;&=\; 1 \,+\, \mathbb{E}_P\Biggl[\sum_{r=1}^{2} \mathbb{E}[V_r A_{\boldsymbol{Z_r}} \mid P]\Biggr] \\ &=\; 1 \,+\, \mathbb{E}_P\Biggl[\sum_{r=1}^{2} V_r\,\mathbb{E}[A_{\boldsymbol{Z_r}} \mid P]\Biggr] \end{aligned}$$

using the inductive hypothesis

$$\begin{aligned} &\le\; 1 \,+\, \mathbb{E}_P\Biggl[\sum_{r=1}^{2} V_r\,(c\mathcal{H}_{\ln}(\boldsymbol{Z_r}) + d)\Biggr] \\ &=\; 1 \,+\, c\cdot\mathbb{E}\Biggl[\sum_{r=1}^{2} V_r\,\mathcal{H}_{\ln}(\boldsymbol{Z_r})\Biggr] \,+\, d\cdot\mathbb{E}[\Sigma\boldsymbol{V}] \\ &\underset{(14)}{=}\; c\cdot\mathcal{H}_{\ln}(\boldsymbol{q}) \,+\, \underbrace{1 \,-\, c\cdot\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)] \,+\, d\cdot\mathbb{E}[\Sigma\boldsymbol{V}]}_{\varpi} \end{aligned} \tag{16}$$

It remains to show that $\varpi \le d$. We consider two cases depending on the maximal probability $\lambda = \max_{1\le v\le u} q_v$.

1. Case $\lambda < \varepsilon$:
   In this case, all individual probabilities are smaller than ε, so it is plausible that we can bound the expected entropy of partitioning $\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)]$ from below. The subuniverse probabilities V_r are quite close to the continuous spacings D_r: by definition (see also Figure 4 on page 18) we have in interval-arithmetic notation

$$V_r \;=\; D_r \,+\, (-2\varepsilon, 0), \qquad (r = 1,2). \tag{17}$$

   For the expected partitioning entropy, this means

$$\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)] \;\ge\; \sum_{r=1}^{2} \mathbb{E}[V_r \ln(1/V_r)]$$

using Equation (17) and $x\log(1/x) \ge (x-\varepsilon)\log(1/(x+\varepsilon'))$ for $\varepsilon,\varepsilon' \ge 0$ and $x \in [0,1]$, this is

$$
\begin{aligned}
&\ge \sum_{r=1}^{2} \mathbb{E}\bigl[(D_r - 2\varepsilon)\ln(1/D_r)\bigr] \\
&= \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{D})] \;+\; 2\varepsilon \sum_{r=1}^{2} \mathbb{E}[\ln(D_r)] \\
&\underset{\text{Lemma 2.6, (3)}}{=} \tilde{H} \;+\; 2\varepsilon \sum_{r=1}^{2} \bigl(\psi(t+1) - \psi(k+1)\bigr) \\
&= \tilde{H} \;-\; 4\varepsilon(H_k - H_t) \\
&= \tilde{H} \;-\; 4\varepsilon\tilde{h} \\
&= 1/c\,.
\end{aligned}
$$

Hence c satisfies $c \ge 1/\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)] \ge 0$, which implies

$$
\begin{aligned}
\varpi &\le 1 - \frac{1}{\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)]} \cdot \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)] \;+\; d \cdot \underbrace{\mathbb{E}[\Sigma\boldsymbol{V}]}_{\le 1} \\
&\le d.
\end{aligned}
$$

The inductive step is proven in this case.

2. Case $\lambda \ge \varepsilon$:
In the second case, there is a *likely* value $v \in [u]$ with $q_v \ge \varepsilon$. We will show a lower bound for having this value as label of the root.
We have $\Pi \overset{\mathcal{D}}{=} \mathrm{Beta}(t+1,t+1)$ and hence $f_\Pi(z) = \frac{z^t(1-z)^t}{\mathrm{B}(t+1,t+1)}$. Recall that $P = f_U^{-1}(\Pi)$ (Figure 3), so we can bound the probability to draw v from below by the smallest value of the integral over any ε-wide strip of the density:

$$
\begin{aligned}
\mathbb{P}[P=v] &\ge \min_{0\le\zeta\le1-\varepsilon} \int_{z=\zeta}^{\zeta+\varepsilon} f_{\Pi_r}(z)\,dz \\
&= \min_{0\le\zeta\le1-\varepsilon} \int_{z=\zeta}^{\zeta+\varepsilon} \frac{z^t(1-z)^t}{\mathrm{B}(t+1,t+1)}\,dz \\
&= \int_{z=0}^{\varepsilon} \frac{z^t(1-z)^t}{\mathrm{B}(t+1,t+1)}\,dz \\
&\ge \int_{0}^{\varepsilon} \frac{z^t(1-\varepsilon)^t}{\mathrm{B}(t+1,t+1)}\,dz \\
&= \frac{\varepsilon^{t+1}(1-\varepsilon)^t}{(t+1)\,\mathrm{B}(t+1,t+1)}\,.
\end{aligned}
\qquad (18)
$$

For the expected hitting probability, we thus have for any $\boldsymbol{q}$ with a $q_v \ge \varepsilon$ that

$$
\mathbb{E}[H] \;\ge\; q_v \cdot \mathbb{P}[P=v] \underset{(18)}{\ge} \frac{\varepsilon^{t+2}(1-\varepsilon)^t}{(t+1)\,\mathrm{B}(t+1,t+1)} \;=\; 1/d, \qquad (19)
$$

so $d \ge 1/\mathbb{E}[H]$. This implies

$$
\begin{aligned}
\varpi - d &\le 1 - c\cdot\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V},H)] + d\cdot\mathbb{E}[\Sigma\boldsymbol{V}] - d \\
&\le 1 - (1-\mathbb{E}[\Sigma\boldsymbol{V}]) \cdot \frac{1}{\mathbb{E}[H]} \\
&= 0.
\end{aligned}
$$

This concludes the inductive step for the second case.

The inductive step is thus done in both cases, and the claim holds for all stochastic vectors $\boldsymbol{q}$ by induction. $\square$

The lower bound on $\mathbb{E}[A_{\boldsymbol{q}}]$ uses basically the same techniques; only a few details differ.

Lemma 9.3 (Lower Bound): *Let $\mathbb{E}[A_{\boldsymbol{q}}]$ satisfy Equation (15), and let $\varepsilon \in (0, 1/e)$ be given. Define*

$$\begin{aligned} c \;=\; c_\varepsilon &\;=\; \frac{1}{\tilde{H} + 4\varepsilon + \varepsilon\ln(1/\varepsilon)}, \\ d \;=\; d_\varepsilon &\;=\; (c_\varepsilon \ln(3) \;-\; 1)\,\frac{(t+1)\,\mathrm{B}(t+1,t+1)}{\varepsilon^{t+2}(1-\varepsilon)^t}, \\ \text{where}\quad \tilde{H} &\;=\; H_{k+1} - H_{t+1}\,. \end{aligned}$$

If $d \ge 0$, we have that $\mathbb{E}[A_{\boldsymbol{q}}] \ge c \cdot \mathcal{H}_{\ln}(\boldsymbol{q}) - d$ all stochastic vectors $\boldsymbol{q}$.

Proof: Let $\varepsilon \in (0, 1/e)$ with $d = d_\varepsilon \ge 0$ be given; $c = c_\varepsilon \ge 0$ holds for any ε. The proof is similar to that of Lemma 9.2, so we emphasize the differences and skip identical parts. If $u = 0$, i.e., $\boldsymbol{q} = ()$, the claim holds since $d \ge 0$.

Now assume $u \ge 1$ and that the claim holds for all (strictly) smaller universe sizes. As for the upper bound, we find from the recurrence using the inductive hypothesis

$$\begin{aligned} \mathbb{E}[A_{\boldsymbol{q}}] &\;\ge\; 1 \;+\; \mathbb{E}_P\left[\sum_{r=1}^{2} V_r\,(c\mathcal{H}_{\ln}(\boldsymbol{Z_r}) - d)\right] \\ &\underset{(14)}{=}\; c\cdot\mathcal{H}_{\ln}(\boldsymbol{q}) \;+\; \underbrace{1 \;-\; c\cdot\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)] \;-\; d\cdot\mathbb{E}[\Sigma\boldsymbol{V}]}_{\varpi}, \end{aligned} \tag{20}$$

and it remains to show that $\varpi \ge -d$. We consider the same two cases for $\lambda = \max_{1\le v\le u} q_v$.

1. Case $\lambda < \varepsilon$:
   In this case, we bound the expected entropy of partitioning, $\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$, from *above.* Similar to the computation for the upper bound, we find

   $$\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)] \;=\; \sum_{r=1}^{2}\mathbb{E}[V_r\ln(1/V_r)] \;+\; \mathbb{E}[H\ln(1/H)]$$

   using Equation (17) and $(x-\varepsilon)\log(1/(x-\varepsilon)) \le x\ln(1/x) + \varepsilon'$ for $0 \le \varepsilon \le \varepsilon'$ and $x \in [0,1]$, and that $x\ln(1/x)$ is increasing for $x \in [0, 1/e]$, this is

   $$\begin{aligned} &\le\; \sum_{r=1}^{2}\mathbb{E}\big[D_r\ln(1/D_r) + 2\varepsilon\big] \;+\; \varepsilon\ln(1/\varepsilon) \\ &\underset{(2)}{=}\; \tilde{H} \;+\; 4\varepsilon \;+\; \varepsilon\ln(1/\varepsilon) \end{aligned}$$

   So c satisfies $c \le 1/\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$, which implies

   $$\begin{aligned} \varpi &\;\ge\; 1 \;-\; \frac{1}{\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]}\cdot\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)] \;-\; d\cdot\underbrace{\mathbb{E}[\Sigma\boldsymbol{V}]}_{\le 1} \\ &\;\ge\; -d. \end{aligned}$$

   The inductive step is proven in this case.

2. Case $\lambda \ge \varepsilon$:
   In the second case, there is a *likely* value $v \in [u]$ with $q_v \ge \varepsilon$. By the same arguments as in the proof of Lemma 9.2, we find

$$\mathbb{E}[H] \underset{(19)}{\ge} \frac{\varepsilon^{t+2}(1-\varepsilon)^t}{(t+1)\,\mathrm{B}(t+1,t+1)},$$

   so that $d \ge (c\ln(3)-1)/\mathbb{E}[H]$. This implies

$$\begin{aligned}\varpi + d &= 1 - c\cdot\mathbb{E}\big[\underbrace{\mathcal{H}_{\ln}(\boldsymbol{V},H)}_{\le \ln(3)}\big] + d(1-\mathbb{E}[\Sigma\boldsymbol{V}]) \\ &\ge 1 - c\cdot\ln(3) + \frac{c\ln(3)-1}{\mathbb{E}[H]}\cdot\mathbb{E}[H] \\ &= 0.\end{aligned}$$

   This concludes the inductive step also in the second case, so the claim holds for all stochastic vectors $\boldsymbol{q}$ by induction. □

We observe that c_ε converges to $1/\tilde{H}$ for both bounds as $\varepsilon \to 0$, so there is hope to show $\mathbb{E}[A_{\boldsymbol{q}}] \sim \mathcal{H}_{\ln}(\boldsymbol{q})/\tilde{H}$. We have to be precise about the limiting process and error terms, though.

Let $(\boldsymbol{q}^{(i)})_{i\in\mathbb{N}}$ be a sequence of universe distributions for which $\mathcal{H}_i := \mathcal{H}_{\ln}(\boldsymbol{q}^{(i)}) \to \infty$ as $i \to \infty$. Now, consider c_ε and d_ε from Lemma 9.2. As functions in ε, they satisfy for $\varepsilon \to 0$

$$c_\varepsilon = \frac{1}{\tilde{H}} \pm O(\varepsilon), \qquad d_\varepsilon = O(\varepsilon^{-t-2}) \tag{21}$$

Since the bounds above hold simultaneously for all feasible values of ε, we can let ε depend on $\mathcal{H}_i$. If we set

$$\varepsilon = \varepsilon_i = \mathcal{H}_i^{-\frac{1}{t+3}} \tag{22}$$

we have $\varepsilon_i \to 0$ as $i \to \infty$ and so $c_{\varepsilon_i} > 0$ for large enough i. Then we have by Lemma 9.2

$$\mathbb{E}[A_{\boldsymbol{q}^{(i)}}] \le c_{\varepsilon_i}\mathcal{H}_i + d_{\varepsilon_i} \underset{(21)}{=} \frac{\mathcal{H}_i}{\tilde{H}} \pm O\Big(\mathcal{H}_i^{\frac{t+2}{t+3}}\Big), \qquad (i \to \infty). \tag{23}$$

Now consider the lower bound. For c_ε and d_ε from Lemma 9.3 we similarly find as $\varepsilon \to 0$ that

$$c_\varepsilon = \frac{1}{\tilde{H}} \pm O(\varepsilon\log\varepsilon), \qquad d_\varepsilon = O(\varepsilon^{-t-2}). \tag{24}$$

With the same ε_i as above (Equation (22)) we have $d_{\varepsilon_i} \ge 0$ for large enough i, so by Lemma 9.3 it holds that

$$\mathbb{E}[A_{\boldsymbol{q}^{(i)}}] \ge \frac{\mathcal{H}_i}{\tilde{H}} \pm O\Big(\mathcal{H}_i^{\frac{t+2}{t+3}}\log\mathcal{H}_i\Big), \qquad (i \to \infty). \tag{25}$$

Together with Equation (23), this concludes the proof of Theorem 9.1. □

10. Lower Bound For I.I.D. Sorting

We follow the elegant argument of Munro and Raman [35] for multiset sorting to obtain a lower bound for the discrete i.i.d. model. By averaging over the profiles, we obtain essentially the result of their Theorem 4, but with a weaker error term.

Theorem 10.1 (Lower Bound): *Let $u = O(n^\nu)$ for a constant $\nu \in [0,1)$ and $\boldsymbol{q} \in (0,1)^u$ with $\Sigma\boldsymbol{q} = 1$. For any constant $\varepsilon > 0$, $\mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n - n/\ln(2) \pm o\big(n^{(1+\nu)/2+\varepsilon}\big)$ ternary comparisons are necessary in expectation as $n \to \infty$ to sort n i.i.d. $\mathcal{D}(\boldsymbol{q})$ elements by any comparison-based algorithm.*

We remark that one might expect to require at least $\mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n$ comparisons since this is the entropy of a vector of n i.i.d. $\mathcal{D}(\boldsymbol{q})$ elements; but such entropy arguments must be taken with care: for u much larger than n, $u \gg n$, we might have $\mathcal{H}(\boldsymbol{q}) \gg \mathrm{ld}\, n$; then $n\mathcal{H}(\boldsymbol{q}) \gg n \,\mathrm{ld}\, n$ is certainly *not* a lower bound for sorting. (Theorem 10.1 does not make a statement for such a case since the error bound dominates then.)

Proof of Theorem 10.1: Let $U_1, \ldots, U_n$ be n i.i.d. $\mathcal{D}(\boldsymbol{q})$ numbers and $V_1, \ldots, V_n$ be a random permutation of $[n]$. The n vectors (U_i, V_i) are then all distinct, and all $n!$ relative rankings w.r.t. lexicographic order are equally likely.

Assume we can sort $U_1, \ldots, U_n$ with $\mathbb{E}[C_{n,\boldsymbol{q}}]$ ternary comparisons on average. We use this method to partially sort the n vectors $(U_1, V_1), \ldots, (U_n, V_n)$ according to the first component only. We can then complete the sorting using Mergesort (separately) on each of the u classes of elements with same first component. (Any sorting method must already have determined the borders between these classes while sorting according to $U_1, \ldots, U_n$.) The total number of comparisons we use is then no more than

$$\mathbb{E}[C_{n,\boldsymbol{q}}] \;+\; \sum_{v=1}^{u} \mathbb{E}\big[X_v \,\mathrm{ld}(X_v)\big] \;=\; \mathbb{E}[C_{n,\boldsymbol{q}}] \;+\; n\mathbb{E}\underbrace{\Bigg[\sum_{v=1}^{u} \frac{X_v}{n} \,\mathrm{ld}\bigg(\frac{X_v}{n}\bigg)\Bigg]}_{=\mathcal{H}_{\mathrm{ld}}(\boldsymbol{X}/n)} \;+\; n\,\mathrm{ld}(n)$$

with ρ as in Lemma 2.7

$$\begin{aligned} &= \; n\,\mathrm{ld}(n) + \mathbb{E}[C_{n,\boldsymbol{q}}] + n\mathcal{H}_{\mathrm{ld}}(\boldsymbol{q}) \;\pm\; n\,\frac{\rho}{\ln(2)} \\ &\geq \; n\,\mathrm{ld}(n) - n/\ln(2) \;\pm\; O(\log n) \end{aligned}$$

since the latter is the well-known lower bound on the average number of (ternary or binary) comparisons for sorting a random permutation of n distinct elements, see, e.g., Equation 5.3.1–(37) of Knuth [27]. It follows that

$$\mathbb{E}[C_{n,\boldsymbol{q}}] \;\geq\; \mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n - \frac{n}{\ln(2)} \;\pm\; n\,\frac{\rho}{\ln(2)}.$$

For $u = O(n^\nu)$ with $\nu \in [0,1)$ we get asymptotically for any $\varepsilon > 0$

$$\mathbb{E}[C_{n,\boldsymbol{q}}] \;\geq\; \mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n - \frac{n}{\ln(2)} \;\pm\; o\big(n^{\frac{1+\nu}{2}+\varepsilon}\big). \qquad \square$$

11. Proof of Main Result

With all these preparations done, the proof of our main result reduces to properly combining the ingredients developed above.

Proof of Theorem 5.1: Let the sequence $(\boldsymbol{q}^{(n)})_{n\in\mathbb{N}}$ be given. By assumption the model has many duplicates, i.e., $\mu_n = \Omega(n^{-1+\varepsilon})$. We thus obtain from Theorem 8.3 that

$$\mathbb{E}[C_{n,\boldsymbol{q}^{(n)}}] \;=\; \mathbb{E}[A_{\boldsymbol{q}^{(n)}}] \cdot n \;\pm\; O(n^{1-\delta'}), \qquad (n \to \infty),$$

for any $\delta' \in (0, \varepsilon)$. If $\mathcal{H}_n = \mathcal{H}(\boldsymbol{q}^{(n)}) \to \infty$ as $n \to \infty$, we can continue with Theorem 9.1 right away. To cover the case that $(\mathcal{H}_n)_{n\in\mathbb{N}}$ contains an infinite subsequence that is bounded, we add an error bound of $O(n)$; this dominates $\mathbb{E}[A_{\boldsymbol{q}^{(n)}}] \cdot n$ (making the claim is essentially vacuous) for such inputs. So in any case we have that

$$\begin{aligned}\mathbb{E}[C_{n,\boldsymbol{q}^{(n)}}] &= \alpha_k \mathcal{H}_n \cdot n \;\pm\; O(\mathcal{H}_n^{1-\delta} n + n + n^{1-\delta'}) \\ &= \alpha_k \mathcal{H}_n \cdot n \;\pm\; O(\mathcal{H}_n^{1-\delta} n + n), \qquad (n \to \infty)\end{aligned}$$

for any $\delta \in (0, \frac{1}{t+3}) = (0, \frac{2}{k+5})$. The optimality claim follows directly by comparing with the lower bound in Theorem 10.1. $\square$

12. Conclusion

Computer scientists are so accustomed to the random-permutation model that the possibility of duplicate keys in sorting is easily overlooked. The following formulation (which is equivalent to random permutations) makes the presence of an underlying restriction more obvious: we sort n numbers drawn independently from the same *continuous* probability distribution. The assumption of i.i.d. samples is as natural as declaring all orderings equally likely, and certainly adequate when no specific knowledge is available; even more so for Quicksort where randomly shuffling the input prior to sorting is best practice. The use of a *continuous* distribution, though, is a restriction worth questioning; what happens if we use a discrete distribution instead?

The most striking difference certainly is that with a discrete distribution, we expect to see equal elements appear in the input. There are more differences, though. If the distribution is (absolutely) continuous (i.e., attains values in a real interval and has a continuous density), almost surely all elements are different and the ranks form a random permutation [29], no matter how the continuous distribution itself looks like. By contrast, different discrete universe distributions each yield a *different* input model; in particular the universe size u can have a great influence.

In this paper, I presented the first analysis of median-of-k Quicksort under the model of independently and identically distributed numbers from a *discrete* universe distribution. I showed that it uses asymptotically a factor $\alpha_k = \ln(2)/(H_{k+1} - H_{(k+1)/2})$ more comparisons than necessary in expectation. Analytical complications necessitated the restriction to the case where every element is expected to appear $\Omega(n^{\varepsilon})$ times for some $\varepsilon > 0$, but I conjecture that the result generalizes beyond the limitations of the current techniques (see the comments below).

The very same statement—asymptotically a factor α_k over the lower bound—holds in the i.i.d. model with a continuous distribution, so, apart from the above restriction, we can say that median-of-k Quicksort is asymptotically optimal to within the constant factor α_k for *any* randomly ordered input. It is reassuring (and somewhat remarkable given the big differences in the derivations that lead to it) that we obtain the same constant in both cases: the relative benefit of pivot sampling is independent of the presence or absence of equal keys. Table 1 shows the first values of α_k; most savings happen for small k.

k	α_k	saving over $k = 1$
1	1.38629	—
3	1.18825	14.3 %
5	1.12402	18.9 %
7	1.09239	21.2 %
9	1.07359	22.6 %
∞	1	27.9 %

Table 1: A few values of α_k and the relative improvement over the case without sampling.

Exploiting the correspondence of Quicksort and search trees, we can also express the result in terms of trees: The expected search costs in search trees built from *continuous* i.i.d. (or randomly permuted and *distinct*) data is of order $\log n$, whereas for *discrete* i.i.d. data (with many duplicates), it is of order $\mathcal{H}(\boldsymbol{q})$. Fringe-balancing allows to lower the constant of proportionality, α_k, and has the *same relative effect* in both cases.

A particularly simple example of distributions covered by our analysis is the uniform distribution, $\boldsymbol{q} = (\frac{1}{u}, \ldots, \frac{1}{u})$, where $u = u_n = O(n^{1-\varepsilon})$. This model coincides with the random u-ary files studied by Sedgewick [38] in his 1977 article. Since the entropy of the discrete uniform distribution is simply $\mathcal{H}(\boldsymbol{q}) = \mathrm{ld}(u)$, we obtain $\mathbb{E}[C_{n,\boldsymbol{q}}] \sim \alpha_k n \,\mathrm{ld}(u)$—the same form as for random permutation only with $\mathrm{ld}\, n$ replaced by $\mathrm{ld}\, u$. For the special case of the uniform distribution, we can also strengthen the error bound of Theorem 9.1 for $\mathbb{E}[A_{\boldsymbol{q}}]$ to $O(1)$ using different techniques; see my Ph.D. thesis [47, Sec. 8.7.7] for details.

I surveyed previous results on multiset sorting, which uses a different input model. We cannot directly relate the result, but we have seen (in Section 3.3) that the i.i.d. model yields at least an upper bound. (A more fine-grained discussion of the relation of the two models is deferred to a full version of this article.)

Since $H_{k+1} - H_{(k+1)/2} \sim \ln(k+1) - \ln((k+1)/2) = \ln(2)$ as $k \to \infty$, we have $\alpha_k \to 1$, so we have confirmed the conjecture of Sedgewick and Bentley for inputs with many duplicates: by increasing k, we obtain sorting methods that are arbitrarily close to the asymptotically optimal number of comparisons for randomly ordered inputs, and in particular for random permutations of a multiset.

Extensions and Future Directions. The methods used in this paper can readily be generalized to analyze skewed sampling, i.e., when we pick an order statistic other than the median of the sample. It is known that the number of comparisons is minimal for the median [30], so this might not be very interesting in its own right, but it can be used to analyze *ninther* sampling and similar schemes [47, Sec. 6.4].

I conjecture that $\alpha_k \mathcal{H}(\boldsymbol{x}/n) n$ is the correct leading term for any profile $\boldsymbol{x} \in \mathbb{N}^u$. If we have, however, a discrete i.i.d. model with $q_v \ll n^{-1}$ so that v is unlikely to be present in the input *at all*, a modified "entropy" must appear instead of $\mathcal{H}(\boldsymbol{q})$. (This case cannot occur in the multiset model since $x_v \geq 1$.) In the limit when all individual q_i are small, this modified entropy would have to equal $\mathrm{ld}(n)$. Identifying such a unified expression is an interesting challenge.

Another line of future research is to extend the present analysis to multiway Quicksort, e.g., the Yaroslavskiy-Bentley-Bloch dual-pivot Quicksort used in Java. The uniform case is known [47], but the techniques of the present paper do not carry over: the partitioning costs for such methods also depend on $\boldsymbol{q}$, which means that we do not get matching lower and upper bounds for $\mathbb{E}[A_{\boldsymbol{q}}]$ using the method of Section 9 any more.

Acknowledgments

I am very grateful for inspiring discussions with Conrado Martínez, Markus Nebel, Martin Aumüller and Martin Dietzfelbinger, which laid the ground for the present paper. Moreover I thank Sándor Fekete for many helpful comments that improved the presentation and framing of the results. Finally, I thank my anonymous referees for their scrutiny and thorough reviews; their comments helped to clarify the presentation significantly.

Appendix

A. Index of Notation

In this appendix, I collect the notations used in this work.

A.1. Generic Mathematical Notation

$\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$	natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, integers $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$, rational numbers $\mathbb{Q} = \{p/q : p \in \mathbb{Z} \wedge q \in \mathbb{N}\}$, real numbers $\mathbb{R}$, and complex numbers $\mathbb{C}$.
$\mathbb{R}_{>1}$, $\mathbb{N}_{\ge 3}$ etc.	restricted sets $X_{\mathrm{pred}} = \{x \in X : x \text{ fulfills pred}\}$.
$\ln(n)$, $\mathrm{ld}(n)$	natural and binary logarithm; $\ln(n) = \log_e(n)$, $\mathrm{ld}(n) = \log_2(n)$.
$\boldsymbol{x}$	to emphasize that $\boldsymbol{x}$ is a vector, it is written in **bold;** components of the vector are not written in bold: $\boldsymbol{x} = (x_1, \ldots, x_d)$; unless stated otherwise, all vectors are column vectors.
X	to emphasize that X is a random variable it is Capitalized.
$[a, b)$	real intervals, the end points with round parentheses are excluded, those with square brackets are included.
$[m..n]$, $[n]$	integer intervals, $[m..n] = \{m, m+1, \ldots, n\}$; $[n] = [1..n]$.
$\lVert\boldsymbol{x}\rVert_p$	p-norm; for $\boldsymbol{x} \in \mathbb{R}^d$ and $p \in \mathbb{R}_{\ge 1}$ we have $\lVert\boldsymbol{x}\rVert_p = \bigl(\sum_{r=1}^{d} \lvert x_r\rvert\bigr)^{1/p}$.
$\lVert\boldsymbol{x}\rVert_\infty$	∞-norm or maximum-norm; for $\boldsymbol{x} \in \mathbb{R}^d$ we have $\lVert\boldsymbol{x}\rVert_\infty = \max_{r=1,\ldots,d} \lvert x_r\rvert$.
$\boldsymbol{x} + 1$, $2^{\boldsymbol{x}}$, $f(\boldsymbol{x})$	element-wise application on vectors; $(x_1, \ldots, x_d) + 1 = (x_1 + 1, \ldots, x_d + 1)$ and $2^{\boldsymbol{x}} = (2^{x_1}, \ldots, 2^{x_d})$; for any function $f : \mathbb{C} \to \mathbb{C}$ write $f(\boldsymbol{x}) = (f(x_1), \ldots, f(x_d))$ etc.
$\Sigma\boldsymbol{x}$	"total" of a vector; for $\boldsymbol{x} = (x_1, \ldots, x_d)$, we have $\Sigma\boldsymbol{x} = \sum_{i=1}^{d} x_i$.
$\boldsymbol{x}^T$, $\boldsymbol{x}^T\boldsymbol{y}$	"transpose" of vector/matrix $\boldsymbol{x}$; for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, I write $\boldsymbol{x}^T\boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$.
H_n	nth harmonic number; $H_n = \sum_{i=1}^{n} 1/i$.
$O(f(n))$, $\pm O(f(n))$, Ω, Θ, $\sim$	asymptotic notation as defined, e.g., by Flajolet and Sedgewick [15, Section A.2]; $f = g \pm O(h)$ is equivalent to $\lvert f - g\rvert \in O(\lvert h\rvert)$.
$x \pm y$	x with absolute error $\lvert y\rvert$; formally the interval $x \pm y = [x - \lvert y\rvert, x + \lvert y\rvert]$; as with O-terms, I use "one-way equalities": $z = x \pm y$ instead of $z \in x \pm y$.
$\Gamma(z)$	the gamma function, $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$.
$\psi(z)$	the digamma function, $\psi(z) = \frac{d}{dz} \ln(\Gamma(z))$.

$\mathrm{B}(\alpha,\beta)$	beta function; $\mathrm{B}(\alpha,\beta)=\int_0^1 z^{\alpha-1}(1-z)^{\beta-1}\,dz=\Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$.
$I_{a,b}(\alpha,\beta)$	incomplete regularized beta function; $I_{a,b}(\alpha,\beta)=\int_a^b z^{\alpha-1}(1-z)^{\beta-1}\,dz\,\big/\,\mathrm{B}(\alpha,\beta)$.

A.2. Stochastics-related Notation

$\mathbb{P}[E]$, $\mathbb{P}[X=x]$	probability of an event E resp. probability for random variable X to attain value x.
$\mathbb{E}[X]$	expected value of X; I write $\mathbb{E}[X \mid Y]$ for the conditional expectation of X given Y, and $\mathbb{E}_X[f(X)]$ to emphasize that expectation is taken w.r.t. random variable X.
$X \overset{\mathcal{D}}{=} Y$	equality in distribution; X and Y have the same distribution.
$\mathbb{1}_E$, $\mathbb{1}_{\{X\le 5\}}$	indicator variable for event E, i.e., $\mathbb{1}_E$ is 1 if E occurs and 0 otherwise; $\{X \le 5\}$ denotes the event induced by the expression $X \le 5$.
$\mathrm{B}(p)$	Bernoulli distributed random variable; $p \in [0,1]$.
$\mathcal{D}(\boldsymbol{p})$	discrete random variable with weights $\boldsymbol{p}$; for $\boldsymbol{p} \in [0,1]^d$, for $I \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{p})$, we have $I \in [1..d]$ and $\mathbb{P}[I=i]=p_i$ for $i \in [d]$ and 0 otherwise.
$\mathrm{Beta}(\alpha,\beta)$	beta distributed random variable with shape parameters $\alpha \in \mathbb{R}_{>0}$ and $\beta \in \mathbb{R}_{>0}$.
$\mathrm{Bin}(n,p)$	binomial distributed random variable with $n \in \mathbb{N}_0$ trials and success probability $p \in [0,1]$; $X \overset{\mathcal{D}}{=} \mathrm{Bin}(n,p)$ is equivalent to $(X, n-X) \overset{\mathcal{D}}{=} \mathrm{Mult}(n; p, 1-p)$.
$\mathrm{Mult}(n,\boldsymbol{p})$	multinomially distributed random variable; $n \in \mathbb{N}_0$ and $\boldsymbol{p} \in [0,1]^d$ with $\Sigma\boldsymbol{p}=1$.
$\mathrm{HypG}(k,r,n)$	hypergeometrically distributed random variable; $n \in \mathbb{N}$, $k, r, \in \{1,\ldots,n\}$.
$\mathcal{H}(\boldsymbol{p})$, $\mathcal{H}_{\mathrm{ld}}(\boldsymbol{p})$, $\mathcal{H}_{\ln}(\boldsymbol{p})$	Shannon entropy of information theory; $\mathcal{H}_{\mathrm{ld}}(p_1,\ldots,p_d)=\sum_{r=1}^d p_r \,\mathrm{ld}(1/p_r)$; similarly $\mathcal{H}_{\ln}$ is the base-e entropy. $\mathcal{H}_{\ln}(p_1,\ldots,p_d)=\sum_{r=1}^d p_r \ln(1/p_r)$; I write $\mathcal{H}$ for $\mathcal{H}_{\mathrm{ld}}$.
stochastic vector	A vector $\boldsymbol{p}$ is called stochastic if $0 \le \boldsymbol{p} \le 1$ and $\Sigma\boldsymbol{p}=1$.
w.h.p. (event)	Let $E=E(n)$ be an event that depends on a parameter n. I say "E occurs w.h.p." if $\mathbb{P}[E(n)]=1\pm O(n^{-c})$ as $n\to\infty$ for *any* constant c.
w.h.p. (bound)	Let $X_1, X_2, \ldots$ be a sequence of real random variables. I say "$X_n=O(g(n))$ w.h.p." if for every constant c there is a constant d so that $\mathbb{P}[\lvert X_n\rvert \le d\lvert g(n)\rvert]=1\pm O(n^{-c})$ as $n\to\infty$.

A.3. Notation for the Algorithm

n	length of the input array, i.e., the input size.
u	universe size $u \in \mathbb{N}$.
$\boldsymbol{x}$	fixed profile in the multiset model; $\boldsymbol{x} \in \mathbb{N}^u$, $\Sigma\boldsymbol{x}=n$
$\boldsymbol{q}$	probability weights of the (discrete) i.i.d. model; $\boldsymbol{q} \in (0,1)^u$, $\Sigma\boldsymbol{q}=1$.
U_i	ith element of the input; in the i.i.d. model, $U_i \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$, for all i and $(U_1,\ldots,U_n)$ are (mutually) independent.
k, t	sample size $k=2t+1$ for $t \in \mathbb{N}_0$;
ternary comparison	operation to compare two elements, outcome is either $<$, $=$ or $>$.
$\boldsymbol{X}$	profile of the input; X_v is the number of occurrences of value v in $U_1,\ldots,U_n$; $\boldsymbol{X} \overset{\mathcal{D}}{=} \mathrm{Mult}(n,\boldsymbol{q})$.

A.4. Notation for the Analysis

$C_{\boldsymbol{x}}$	(random) number of (ternary) comparisons needed by Algorithm 1 to sort a random permutation of a multiset with profile $\boldsymbol{x}$; the dependence on k is left implicit.
$C_{n,\boldsymbol{q}}$	(random) number of (ternary) comparisons needed by Algorithm 1 to sort n i.i.d. $\mathcal{D}(\boldsymbol{q})$ numbers; the dependence on k is left implicit.
$\mathit{\Pi}$	continuous pivot value; $\mathit{\Pi} \overset{\mathcal{D}}{=} \mathrm{Beta}(t+1, t+1)$
$\boldsymbol{D} = (D_1, D_2)$	continuous spacings of the unit interval $(0,1)$ induced by $\mathit{\Pi}$, i.e., $\boldsymbol{D} = (\mathit{\Pi}, 1 - \mathit{\Pi})$.
P	(random) value of chosen pivot in the first partitioning step; $P = F_U^{-1}(\mathit{\Pi})$, for F_U the cumulative distribution function of $\mathcal{D}(\boldsymbol{q})$.
$\boldsymbol{V} = (V_1, V_2)$	non-pivot class probabilities, see Equation (7).
H	hitting probability, Equation (7).
$\boldsymbol{Z} = (\boldsymbol{Z_1}, \boldsymbol{Z_2})$	zoomed-in distributions; see Equation (8).
$\boldsymbol{\mathit{\Gamma}}$	$\boldsymbol{\mathit{\Gamma}} \in \mathbb{N}_0^u$; node-depths vector in search tree; $\mathit{\Gamma}_v$ is the random cost of searching value $v \in [u]$.
$A_{\boldsymbol{q}}$	random $\boldsymbol{q}$-weighted average node depth of a k-fringe-balanced tree built from inserting $\mathcal{D}(\boldsymbol{q})$ elements till saturation, $A_{\boldsymbol{q}} = \boldsymbol{\mathit{\Gamma}}^T \boldsymbol{q}$; see Equation (9).

B. Height of Recursion Trees

In this appendix, we recapitulate the well-known folklore result that Quicksort recursion trees—or equivalently binary search trees—have logarithmic height with high probability. We will show that the same is true for fringe-balanced trees (and thus median-of-k Quicksort) and inputs with equal keys; this is not surprising as the latter are more balanced than ordinary BSTs, but a formal proof is in order for two reasons: we (a) employ a stricter notion of "w.h.p." (cf. Section 2.1) than often used, and (b) there is a pitfall in the folklore proof using good-split-bad-split indicators that has not been rigorously addressed anywhere (to my knowledge).

We start with a result of Mahmoud [28] (page 101) for ordinary binary search trees that is a prototypical for the types of results we are after in this section. (We will confine ourselves to slightly less accurate statements, though.)

Lemma B.1 (BSTs Have Log-Height with High Probability):
For any $\varepsilon > 0$ it holds: The probability that a binary search tree built from a random permutation of n distinct elements has height $\ge (\alpha + \varepsilon) \ln n$ is at most $K c_{\alpha+\varepsilon} n^{-\eta_{\alpha+\varepsilon}} = O(n^{-\eta_{\alpha+\varepsilon}})$ where $K > 0$ is some constant (independent of ε), $c_x = 1/(\Gamma(x)(1 - 2/x))$, $\eta_x = -(x \ln(2/x) + x - 1)$ and $\alpha \approx 4.31107$ is the unique root of η_x for $x \in (4,5)$. □

By Lemma 6.1 this result translates immediately to the height of Quicksort recursion trees.

Mahmoud's derivation is based on extensive knowledge on random BSTs (also given in [28]), in particular he uses the exact expected number of leaves at any given level. Generalizing this for fringe-balanced trees with duplicates seems a daunting task.

Variants of Lemma B.1 are sometimes covered in (advanced) algorithms courses and textbooks, see, e.g., Exercise 4.20 of Mitzenmacher and Upfal [33] Section 2.4 of Dubhashi and Panconesi [12], and Section 4.7 of Erickson [13]. There, a more intuitive argument is given by bounding the probability of many "bad" splits at nodes using Chernoff bounds, however neither of these resources gives a detailed formal proof. The argument is appealingly simple, and fairly easy to extend, but it indeed requires some care to really guarantee the needed independence. We therefore give a detailed proof below and obtain the following result.

Lemma B.2 (Logarithmic Height of Recursion Trees With High Probability): *Let $\mathcal{T}$ be a k-fringe-balanced search tree built by inserting n i.i.d. $\mathcal{D}(\boldsymbol{q})$ numbers into an initially empty tree. Then $\mathcal{T}$ has height $O(\log n)$ w.h.p.; more precisely $\mathcal{T}$ has height $\ge c\ln n$ with probability $\le 2n^{\eta}$ for $n \ge n_0 = e^{30\,000}$ (i.e., with probability $O(n^{\eta})$), where*

$$\begin{aligned}
\eta &= 1 - 2c\delta^2\\
\delta &= p - \frac{1}{c}\cdot\left(\frac{1}{\ln(1/\alpha)} + 1\right)\\
p &= 0.99 - 2I_{\alpha-0.01,1}(t+1,t+1) \;=\; 0.99 - 2\,\frac{k!}{t!\,t!}\int_{\alpha-0.01}^{1} x^t(1-x)^t\,dx
\end{aligned}$$

where $\alpha \in (\frac{1}{2}, 1)$ is a parameter that can be chosen arbitrarily as long as $\delta > 0$. This result is independent of $\boldsymbol{q}$, and holds in particular when $\boldsymbol{q}$ depends on n.

k	c	η	k	c	η
1	12	−1.72	5	7	−0.70
1	13	−2.86	5	8	−2.24
1	20	−11.53	5	20	−22.45
3	9	−1.94	∞	4	−2.03
3	10	−3.37	∞	5	−4.01
3	20	−19.02	∞	20	−33.71

Table B.1: A few exemplary values for the constants in Lemma B.2.

The given value for n_0 is clearly ridiculous; the intention of Lemma B.2 is to give a rigorous proof of the asymptotic result. As we will see in the proof, one can trade larger c values for smaller n_0, and all constants could be improved by a stronger version of the Chernoff bound. It is not my intention to do so here.

We note that for Quicksort, an alternative route is to analyze a modified version of the algorithm that makes some technicalities vanish and performs no better than the original Quicksort; see, e.g., Seidel [42]. Moreover, much stronger concentration results are known for the overall number of *comparisons* in Quicksort, see McDiarmid and Hayward [32] or the streamlined description in Section 7.6 of Dubhashi and Panconesi [12]. There the exponent of the probability bound is arbitrarily large for one *fixed* bound $c\ln n$. It seems not possible to obtain such a stronger bound for the *height* of the tree, though.

Concentration results are typically framed in terms of randomized Quicksort. To emphasize the relation between Quicksort recursion trees and search trees, our reference Quicksort (Algorithm 1) is not randomized, but deterministically chooses the first elements for the sample. Causing an exceptionally high recursion tree can hence be attributed to a specific input in our scenario; the following definition expresses that idea.

Definition B.3 (Height-degenerate Inputs): *An input of length n is called h-height-degenerate (w.r.t. our reference fat-pivot median-of-k Quicksort) if the recursion tree of* $\textsc{Quicksort}_k$ *on this input has a height $> h\ln(n)$.*

From Lemma B.2 we immediately obtain the following fact.

Corollary B.4 (Probability of Height-Degeneracy): *For any k, the probability that an input of n elements is 13-height-degenerate is in $O(1/n^2)$ as $n \to \infty$.* □

We will in the following simply use "height-degenerate" to mean 13-height-degenerate.

* * *

Proof of Lemma B.2: Let $k = 2t + 1$ be given. We will follow the folklore proof that the height of randomly grown BSTs is typically logarithmic: we determine a constant probability $p > 0$ (independent of n) so that a single partitioning step yields a reasonably balanced split; since long paths in the recursion tree cannot contain more than a certain number of such balanced nodes, we can bound the probability of seeing such a long path in a recursion tree.

Outline. A few technicalities need to be addressed in the formal proof:

1. To tweak constants, we will introduce a parameter $\alpha \in (1/2, 1)$ and require subproblems at a node v to contain at most $\alpha n(v)$ elements, where $n(v)$ is the size of the sublist that v corresponds to.
2. The probability to lose a given fraction of elements depends on the subproblem size (a discretization effect) and in our case of inputs with duplicates also on $\boldsymbol{q}$. We can therefore only *lower bound* the probability for a balanced node by a constant p. To obtain a term for p that we can actually evaluate, we resort to asymptotic approximations, which are only a valid lower bound for sublist sizes larger than a threshold n_0.
3. Since the precise probability to lose a fraction of elements depends on the sublist size, also the events that a certain node be balanced are dependent.

The last point is a problem since the standard Chernoff bounds requires mutual independence; nevertheless this issue is not addressed in the sources cited above. Since the pivot choices themselves are done independently, we only need a little trick to obtain independent events: A node is only considered good when it is balanced and additionally a biased coin flip yields heads, where the bias is chosen so that the overall probability for a good node is the same for all nodes. This gives us a sequence of independent indicator variables to which we can apply our machinery as usual.

Balanced nodes. We use the following notation in this proof. v denotes a node of the recursion tree. By $n(v)$ we mean the sublist size at v, i.e., the $\underline{\text{n}}$umber of elements in the subproblem of the recursive call that v corresponds to. $d(v)$ denotes the $\underline{\text{d}}$epth of v, i.e., the number of nodes on the path from the root to v, including endpoints. Finally, if $n(v) \ge k$, we use $J_r(v)$, $r = 1, 2$, to the denote the size of the rth subproblem at v; this is the subproblem size of v's left resp. right child.

We are now in the position to formalize the notion of *balanced* nodes: Let $\alpha \in (1/2, 1)$ be a fixed number and $n_0 \ge k$ a constant (to be chosen later). We call an inner node *α-balanced* if $n(v) \le n_0$ or if $J_r(v)/n(v) \le \alpha$ for both $r = 1, 2$. ((α, n_0)-balanced would be more appropriate; the dependence on n_0 is understood implicitly). An α-balanced node hence has no subproblem with more than an α-fraction of the elements, or has a negligibly small sublist anyway.

The key idea is now that any path in a recursion tree for n elements can contain at most

$$\log_\alpha(n_0/n) \;=\; \log_{1/\alpha}(n/n_0) \;=\; \frac{1}{\ln(1/\alpha)}\ln(n/n_0) \;\le\; \frac{1}{\ln(1/\alpha)}\ln(n)$$

α-balanced nodes before reaching a node v with $n(v) \le n_0$ since considering only the size reduction at these α-balanced nodes already reduces the n initial elements to $\le \alpha^{\log_\alpha(n_0/n)} n = n_0$ elements. From there on, at most n_0 additional α-balanced nodes can follow since each reduces the subproblem size by at least one.

Balanced is not good enough. We are now formalizing the idea sketched above to obtain *independent* indicator variables. For a node with $n(v) \ge n_0$, we define $p_b(v) = \mathbb{P}[v \ \alpha\text{-balanced} \,|\, n(v)]$. Note that $p_b(v)$ only depends on $n(v)$ and k, but since the number of its possible subproblems sizes is finite, $p_b(v)$ will necessarily differ for different values of $n(v)$, even without pivot sampling ($k = 1$) and without our threshold ($n_0 = 0$).

However, we will show below that we can find choices for n_0 and α so that at least $p_b(v) \ge p$ for a given constant $p = p(\alpha)$ in all possible trees.

For such a p, we call a node v (α, p)-*good* if it is α-balanced *and additionally* $B(v) = 1$, where $B(v) \overset{\mathcal{D}}{=} \mathrm{B}(\frac{p}{p_b(v)})$ which is independent of all other random variables. The distribution of $B(v)$ is conditional on the given tree via $n(v)$, so one might imagine first drawing a random recursion tree, and then assigning its nodes labels good or bad.

Since every good node is also balanced, we cannot have more than $\frac{1}{\ln(1/\alpha)} \ln(n) + n_0$ good nodes on any path in a recursion tree for input size n.

Probability of long paths. We can now bound the probability of having a tree of height $\ge c \ln n$. First note that the overall number of nodes is bounded by n (at least one pivot is removed in each step), so the number of leaves in the recursion tree is trivially bounded by n. By the union bound, the probability that *any* of these leaves has depth $\ge h$ is at most n times the probability that one leaf has depth $\ge h$. Let hence v be one of the leaves in the recursion tree and let $v_1, \ldots, v_{d(v)} = v$ be the nodes on the path from the root v_1 down to v; recall that $d(v)$ is the depth of leaf v.

The sublist corresponding to v is the result of $d(v) - 1$ successive partitioning steps, each of which is either (α, p)-good or not. Let $G_1, \ldots, G_{d(v)-1}$ be the corresponding indicator random variables where G_i is 1 if and only if v_i is good. By construction, we have $\mathbb{P}[G_i = 1] = p_b(v) \cdot \frac{p}{p_b(v)} = p$ independently of the tree. We now extend $G_1, \ldots, G_{d(v)-1}$ to an infinite sequence of random variables by i.i.d. $\mathrm{B}(p)$ variables for all $i \ge d(v)$; the G_i then form an infinite sequence of i.i.d. random variables.

Now recall that the number of α-balanced nodes on any path is at most $n_0 + \frac{1}{\ln(1/\alpha)} \ln n$. For any $h \in \mathbb{N}$, we thus have

$$\begin{aligned} \mathbb{P}[d(v) \ge h] &\le \mathbb{P}\Big[G_1 + \cdots + G_h \le n_0 + \frac{1}{\ln(1/\alpha)} \ln n\Big] \\ &= \mathbb{P}\Big[X_h \le \tilde{\gamma} \ln n\Big] \\ &\le \mathbb{P}\Big[X_h \le \gamma \ln n\Big], \end{aligned}$$

where

$$\begin{aligned} X_h &\overset{\mathcal{D}}{=} \mathrm{Bin}(h, p), \\ \tilde{\gamma} &= \frac{1}{\ln(1/\alpha)} + \frac{n_0}{\ln n}, \\ \gamma &= \frac{1}{\ln(1/\alpha)} + 1 \ \ge \ \tilde{\gamma}, \qquad (n \ge e^{n_0}). \end{aligned}$$

For $n \ge e^{n_0}$ and h so that $\delta := p - \gamma \ln(n)/h > 0$ we then have

$$\begin{aligned}
\mathbb{P}[d(v) \ge h] &\le \mathbb{P}[X_h \le \gamma \ln n] \\
&= \mathbb{P}\biggl[p - \frac{X_h}{h} \ge p - \gamma \frac{\ln(n)}{h}\biggr]; \\
&\le \mathbb{P}\biggl[\biggl|\frac{X_h}{h} - p\biggr| \ge \delta\biggr] \\
&\underset{\text{Lemma 2.1}}{\le} 2\exp(-2\delta^2 h).
\end{aligned}$$

With $h = c\ln(n)$, we have $\delta = p - \frac{\gamma}{c}$, which is independent of n, and positive for any $c > \gamma/p$. With this bound we finally find

$$\begin{aligned}
\mathbb{P}[\mathcal{T} \text{ has height } \ge c\ln n] &\le n\,\mathbb{P}[d(v) \ge c\ln n] \\
&\le 2n\exp(-2c\delta^2 \ln n) \\
&= 2n^{1-2c\delta^2},
\end{aligned}$$

which implies the claim.

A lower bound for balanced nodes: how to choose the constants. It remains to show that we can actually find values n_0 and $p = p(\alpha)$ (at least for some choices of α) so that so that $p_b(v) \ge p$ in all nodes v in all possible trees $\mathcal{T}$.

To this end, we derive an *upper* bound for the probability $1 - p_b(v)$ that the root v of $\mathcal{T}$ is *not* (α, n_0)-balanced. For $n \le n_0$, we are done since $p_b(v) = 1$ and any $p > 0$ will do. So assume $n > n_0$. By the union bound we have

$$\mathbb{P}[v \text{ not } \alpha\text{-balanced}] \;=\; \mathbb{P}[\exists r : J_r^{(n)} \ge \alpha n] \;\le\; \sum_r \mathbb{P}[J_r^{(n)} \ge \alpha n], \qquad (26)$$

so it suffices to consider the subproblems $r = 1, 2$ in isolation.

Recall that after a pivot value P is chosen according to Equation (6) on page 17, the probabilities V_r for any other element to belong to the rth subproblem are fully determined. P in turn is fully determined by the choice of $\boldsymbol{D}$. Hence, conditionally on $\boldsymbol{D}$, we have $\mathrm{Bin}(n-k, V_r)$ elements that go to the rth subproblem, plus up to t from the sample. Moreover we always have $V_r \le D_r$ (cf. Figure 4). Conditional on $\boldsymbol{D}$, $J_r^{(n)}$ is hence smaller than $\tilde{J}_r \overset{\mathcal{D}}{=} \mathrm{Bin}(n, D_r) + t$ in stochastic order, i.e., for all j we have that $\mathbb{P}[J_r \ge j \mid \boldsymbol{D}] \le \mathbb{P}[\tilde{J}_r \ge j \mid \boldsymbol{D}]$. (By averaging over all choices for $\boldsymbol{D}$ the same relation holds also unconditionally.)

This is nothing but the precise formulation of the fact that (in stochastic order) subproblem sizes for inputs with duplicates are no larger than for random-permutation inputs, since we potentially exclude duplicates of pivots from recursive calls.

The good thing about D_r is that—unlike J_r—it does not depend on n: $D_r \overset{\mathcal{D}}{=} \mathrm{Beta}(t+1, t+1)$ in every node. Also—unlike V_r—it does not depend on $\boldsymbol{q}$. Since $\tilde{J}_r$ is concentrated around nD_r, J_r is likely to be $\ge \alpha n$ only for $D_r > \alpha$. Precisely for a constant $\delta > 0$ we have

$$\begin{aligned}
\mathbb{P}\Bigl[\tilde{J}_r \ge (D_r + \delta)n \Bigm| D_r\Bigr] &\le \mathbb{P}\biggl[\biggl|\frac{\tilde{J}_r}{n} - D_r\biggr| \ge \delta \biggm| D_r\biggr] \\
&\underset{\text{Lemma 2.1}}{\le} 2\exp(-2\delta^2 n) \\
&\le 2\exp(-2\tilde{\delta}^2 n) \qquad \text{for any } \tilde{\delta} < \delta.
\end{aligned}$$

Using this and separately considering D_r larger resp. smaller $\alpha - \delta$ yields

$$\begin{aligned}\mathbb{P}[J_r \geq \alpha n] &\leq \mathbb{P}[\tilde{J}_r \geq \alpha n] \\ &= \mathbb{E}_D\big[\mathbb{1}_{\{D_r<\alpha-\delta\}} \cdot \mathbb{P}[J_r \geq \alpha n \mid D_r = d, d < \alpha - \delta]\big] \\ &\qquad + \mathbb{E}_D\big[\mathbb{1}_{\{D_r>\alpha-\delta\}} \cdot \mathbb{P}[J_r \geq \alpha n \mid D_r = d, d > \alpha - \delta]\big] \\ &\leq \mathbb{P}[D_r < \alpha - \delta] \cdot 2\exp(-2\delta^2 n) \;+\; \mathbb{P}[D_r > \alpha - \delta] \cdot 1\end{aligned}$$

if we choose, say $\delta = 0.01$, we have $2\exp(-2\delta^2 n) \leq 0.005$ for $n \geq n_0 = 30\,000$

$$\leq \mathbb{P}[D_r > \alpha - 0.01] \;+\; 0.005 \qquad (n \geq n_0).$$

Plugging in above, we find that with $n_0 = 30\,000$, we can choose

$$p \;=\; 0.99 \;-\; 2 \cdot I_{\alpha-0.01,1}(t+1, t+1) \;\leq\; \mathbb{P}[v\ \alpha\text{-balanced}] \qquad (n \geq n_0). \tag{27}$$

Since $p = p(\alpha)$ is continuous and ≥ 0.97 for $\alpha = 1$ there is always a valid choice $\alpha < 1$ with $p > 0$. We are free to choose any such α; the resulting constant c for the achieved height bound then has to satisfy $c > \big(1 + \frac{1}{\ln(1/\alpha)}\big)/p$. It is not clear in general which choice yields the best bounds, so we keep it as a parameter in the analysis. $\square$

Two further remarks are in order about Lemma B.2.

- **Height-bound for any input.**
  The attentive reader might have noticed that we do not make use of the assumption that the input consists of $\mathcal{D}(\boldsymbol{q})$ elements. In fact, the above proof works for *any* randomly permuted input, since we actually compute the subproblem sizes in the most unfavorable case: when all elements are distinct.

  For randomized Quicksort, the random-order assumption is also vacuous; we thus have proved the more general statement that randomized Quicksort has $O(\log n)$ recursion depth w.h.p. for any input. In particular, this proves Proposition 2.4.

- **Height-bound in terms of $\boldsymbol{q}$.**
  For *saturated* trees, our bound on the height in terms of n is meaningless. By similar arguments as above we can show that the height is in $O(\log(1/\mu))$ with high probability as $1/\mu \to \infty$, where μ is the smallest probability q_v: Intuitively, after $c\ln(1/\mu)$ balanced subdivisions of the unit interval, we are left with segments of size less than μ, so after so many partitioning rounds, we have reduced the subuniverse sizes to 1. The subproblems are then solved in one further partitioning step.

  This bound is intuitively more appealing, but for our use case in the proof of the separation theorem (Theorem 8.3), we are dealing with non-saturated trees and the $\log n$ bound turns out to be more convenient. (There we only require that $1/\mu$ does not grow *too fast* with n, but we do not have any guarantee that it grows at all. The height-bound $c\log(1/\mu)$ only holds with high probability as $1/\mu$ goes to infinity; we would then need a case distinction on the growth rate of $1/\mu$ …)

References

[1] B. Allen and I. Munro. Self-organizing binary search trees. *Journal of the ACM*, 25(4):526–535, October 1978. doi: 10.1145/322092.322094.

[2] M. Archibald and J. Clément. Average depth in a binary search tree with repeated keys. In *Colloquium on Mathematics and Computer Science*, volume 0, pages 309–320, 2006.

[3] P. J. Bayer. *Improved Bounds on the Cost of Optimal and Balanced Binary Search Trees.* Master's Thesis, Massachusetts Institute of Technology, 1975.

[4] J. L. Bentley and M. D. McIlroy. Engineering a sort function. *Software: Practice and Experience*, 23(11):1249–1265, 1993.

[5] W. H. Burge. An analysis of binary search trees formed from sequences of nondistinct keys. *Journal of the ACM*, 23(3):451–454, July 1976. doi: 10.1145/321958.321965.

[6] L. Devroye. The equivalence of weak, strong and complete convergence in l_1 for kernel density estimates. *The Annals of Statistics*, 11(3):896–904, September 1983. doi: 10.1214/aos/1176346255.

[7] L. Devroye. *Non-Uniform Random Variate Generation.* Springer New York, 1986. (available on author's website `http://luc.devroye.org/rnbookindex.html`).

[8] L. Devroye. On the expected height of fringe-balanced trees. *Acta Informatica*, 30(5):459–466, May 1993. doi: 10.1007/BF01210596.

[9] DLMF. NIST Digital Library of Mathematical Functions. Release 1.0.10; Release date 2015-08-07. URL `http://dlmf.nist.gov`.

[10] D. Dor and U. Zwick. Median selection requires $(2+\varepsilon)n$ comparisons. *SIAM Journal on Discrete Mathematics*, 14(3):312–325, January 2001. doi: 10.1137/S0895480199353895.

[11] M. Drmota. *Random Trees.* Springer, 2009. ISBN 978-3-211-75355-2.

[12] D. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms.* Cambridge University Press, 2009.

[13] J. Erickson. Tail inequalities. `https://courses.engr.illinois.edu/cs473/sp2017/notes/04-chernoff.pdf`, 2017.

[14] V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys*, 24(4):441–476, 1992.

[15] P. Flajolet and R. Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009. ISBN 978-0-52-189806-5. (available on author's website: `http://algo.inria.fr/flajolet/Publications/book.pdf`).

[16] I. Gradshteyn and I. Ryzhik. *Table of Integrals, Series, and Products.* Academic Press, 7th edition, 2007. ISBN 978-0-12-373637-6.

[17] D. H. Greene. *Labelled formal languages and their uses.* Ph.D. thesis, Stanford University, January 1983.

[18] T. N. Hibbard. Some combinatorial properties of certain trees with applications to searching and sorting. *Journal of the ACM*, 9(1):13–28, January 1962. doi: 10.1145/321105.321108.

[19] C. A. R. Hoare. Algorithm 64: Quicksort. *Communications of the ACM*, 4(7):321, July 1961.

[20] C. A. R. Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, January 1962.

[21] S.-H. S. Huang and C. K. Wong. Binary search trees with limited rotation. *BIT*, (4):436–455, December 1983. doi: 10.1007/BF01933619.

[22] S.-H. S. Huang and C. K. Wong. Average number of rotations and access cost in iR-trees. *BIT*, 24 (3):387–390, September 1984. doi: 10.1007/BF02136039.

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

[24] J. Katajainen and T. Pasanen. Stable minimum space partitioning in linear time. *BIT*, 32(4): 580–585, December 1992. ISSN 0006-3835. doi: 10.1007/BF01994842.

[25] J. Katajainen and T. Pasanen. Sorting multisets stably in minimum space. *Acta Informatica*, 31 (4):301–313, April 1994. doi: 10.1007/BF01178508.

[26] R. Kemp. Binary search trees constructed from nondistinct keys with/without specified probabilities. *Theoretical Computer Science*, 156(1-2):39–70, March 1996. doi: 10.1016/0304-3975(95)00302-9.

[27] D. E. Knuth. *The Art Of Computer Programming: Searching and Sorting.* Addison Wesley, 2nd edition, 1998. ISBN 978-0-20-189685-5.

[28] H. M. Mahmoud. *Evolution of Random Search Trees.* Wiley, 1992. ISBN 0-471-53228-2.

[29] H. M. Mahmoud. *Sorting: A distribution theory.* John Wiley & Sons, 2000. ISBN 1-118-03288-8.

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

[31] C. J. H. McDiarmid. Concentration. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–248. Springer, Berlin, 1998.

[32] C. J. H. McDiarmid and R. B. Hayward. Large deviations for Quicksort. *Journal of Algorithms*, 21 (3):476–507, November 1996. doi: 10.1006/jagm.1996.0055.

[33] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005. ISBN 0-521-83540-2.

[34] I. Munro and P. M. Spira. Sorting and searching in multisets. *SIAM Journal on Computing*, 5(1): 1–8, March 1976. doi: 10.1137/0205001.

[35] J. I. Munro and V. Raman. Sorting multisets and vectors in-place. In *Workshop on Algorithms and Data Structures (WADS)*, volume 519 of *LNCS*, pages 473–480, Berlin/Heidelberg, 1991. Springer. doi: 10.1007/BFb0028285.

[36] P. V. Poblete and J. I. Munro. The analysis of a fringe heuristic for binary search trees. *Journal of Algorithms*, 6(3):336–350, September 1985. doi: 10.1016/0196-6774(85)90003-3.

[37] R. Sedgewick. The analysis of Quicksort programs. *Acta Informatica*, 7(4):327–355, 1977.

[38] R. Sedgewick. Quicksort with equal keys. *SIAM Journal on Computing*, 6(2):240–267, 1977.

[39] R. Sedgewick and J. Bentley. New research on theory and practice of sorting and searching (talk slides), 1999. URL `http://www.cs.princeton.edu/~rs/talks/Montreal.pdf`.

[40] R. Sedgewick and J. Bentley. Quicksort is optimal (talk slides), 2002. URL `http://www.cs.princeton.edu/~rs/talks/QuicksortIsOptimal.pdf`.

[41] R. Sedgewick and K. Wayne. *Algorithms.* Addison-Wesley, 4th edition, 2011. ISBN 978-0-32-157351-3.

[42] R. Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, pages 37–67. Springer, 1993. doi: 10.1007/978-3-642-58043-7_3.

[43] S. Sen and N. Gupta. Distribution-sensitive algorithms. *Nordic Journal of Computing*, 6(2):194, 1999.

[44] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3): 379–423, July 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[45] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3): 652–686, July 1985. doi: 10.1145/3828.3835.

[46] A. Walker and D. Wood. Locally balanced binary trees. *The Computer Journal*, 19(4):322–325, April 1976. doi: 10.1093/comjnl/19.4.322.

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

[48] S. Wild and M. E. Nebel. Average case analysis of Java 7's dual pivot Quicksort. In L. Epstein and P. Ferragina, editors, *European Symposium on Algorithms (ESA)*, volume 7501 of *LNCS*, pages 825–836. Springer, 2012. URL `http://arxiv.org/abs/1310.7409`.