Sorting Discrete i.i.d. Inputs: Quicksort Is Optimal

Sebastian Wild*

August 18, 2016

Abstract

We prove the Sedgewick-Bentley conjecture on median-of-k Quicksort on equal keys: The average number of comparisons for Quicksort with fat-pivot (a.k.a. three-way) partitioning is asymptotically only a constant times worse than the information-theoretic lower bound for sorting n i.i.d. elements, and that constant converges to 1 as $k \to \infty$. Hence, Quicksort with pivot sampling is an optimal distribution-sensitive algorithm for the i.i.d. sorting problem.

1 Introduction

Computer scientists are accustomed to the random permutation model for sorting: the input consists of numbers $1, \ldots, n$, where any ordering is equally likely. Despite its wide-spread use, I suggest an alternative formulation here that is more convenient for the present paper, and to me at least as natural:

The I.I.D. Sorting Problem. *Sort n (real) numbers $U_1, \ldots, U_n$ using pairwise comparisons*[1], *where $U_1, \ldots, U_n$ are independent and identically distributed (i.i.d.) according to a given probability distribution $\mathcal{U}$ (the universe distribution).*

When elements come from the same source, and no specific knowledge is available, i.i.d. samples are a natural assumption and in accordance with the *principle of maximum entropy.* If $\mathcal{U}$ is (absolutely) continuous, i.e., attains values in a real interval and has a continuous density f_U, it is known that a.s. all elements are different and the ranks form a random permutation of $1, \ldots, n$. We obtain the random permutation model again for *any continuous* $\mathcal{U}$ [23].

Research on sorting has focused on the random permutation model, which hides this vital restriction to continuous distributions. However, duplicates in the input appear regularly in many applications, for example during aggregation tasks in databases: The SQL clause `GROUP BY col` is often implemented by sorting rows by column `col`, and many rows with equal `col`-entries are to be expected in such applications. In this work, we are concerned with such inputs:

The Discrete I.I.D. Sorting Problem. *Sort n numbers $U_1, \ldots, U_n$ using pairwise comparisons, where $U_1, \ldots, U_n$ are drawn i.i.d. from a discrete probability distribution $\mathcal{U}$ with $\mathbb{P}[U_1 = i] = q_i$ for $i \in [1..u]$.*

Unlike in the continuous case, different discrete universe distributions yield *different* models here; in particular the universe size u has great influence. The interesting range is $1 \ll u \ll n$; for large u we approach the continuous case. In applications u can be very small, but is typically not known up front, so our methods should adapt automatically; in the words of Sen and Gupta [34] we seek *distribution-sensitive* algorithms.

Contributions. We study the performance of fat-pivot (a.k.a. three-way) Quicksort under the discrete i.i.d. model. The classic implementation of this algorithm by Bentley and McIlroy [4] from 1993 is still used as default sorting methods in important programming libraries, e.g., the GNU C++ STL and the Java runtime library[2]. We prove that this method as is comes to within *ten percent* of the optimal average comparison count for i.i.d. inputs, both continuous and discrete (under mild assumptions).

More generally, we confirm the following conjecture of Sedgewick and Bentley [31, 32], which had been open for almost two decades: *Fat-pivot Quicksort with median-of-k sampling is asymptotically optimal up to a constant factor (w.r.t. the expected comparison count), and this constant factor converges to 1 as $k \to \infty$.*

*Computer Science Department, University of Kaiserslautern, Email: `wild@cs.uni-kl.de`

[1]We consider *ternary* comparisons, i.e., with result $<$, $=$ or $>$.

[2]Since version 7, the JRE uses dual-pivot Quicksort instead, but for inputs with many equal keys, still the fat-pivot partitioning by Bentley and McIlroy takes over, namely whenever the two sampled pivots are equal.

Even though not explicitly mentioned, the analysis of median-of-k Quicksort is open since Sedgewick's seminal 1977 paper [30]; given the extensive body of knowledge on Quicksort with pivot sampling in the distinct keys / i.i.d. continuous case, it is surprising that no mathematical analysis with equal keys was available. This paper seems to be the first step towards closing this gap.

Outline. I summarize related work in the remainder of this section, and then briefly discuss fat-pivot Quicksort in Section 2. Section 3 introduces some preliminaries; a list of all used notation also appears in Appendix A. In Section 4, I give a lower bound for the discrete i.i.d. sorting problem. Sections 5–8 study the cost of fat-pivot Quicksort on discrete i.i.d. inputs. The result is summarized in Section 9.

1.1 Previous Work. The continuous i.i.d. model is implicitly used in analyses, but I could not find any mention of the discrete i.i.d. sorting problem per se. The related problem of multiset sorting attracted considerable attention though.

Multiset Sorting. Here we sort a random permutation of a *fixed* multiset, where value $i \in [1..u]$ appears $x_i \in \mathbb{N}$ times and we write $n = x_1 + \cdots + x_u$.

Munro and Raman [26] 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, and several methods are known to get close to this bound. Denoting by $\mathcal{H}$ the binary Shannon entropy, this bound can be written as $\mathcal{H}(\boldsymbol{x}/n)n - n \operatorname{ld} e \pm O(\log n)$.

The arguably simplest method is to insert elements into a splay tree, collecting all equal elements in a linear list per node. By static optimality (Theorem 2 of Sleator and Tarjan [35]), 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 *semisplaying*), and we need linear extra space.

Already in 1976, Munro and Spira [25] 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.)

The first in-place method was the adapted Heapsort of Munro and Raman [26]; it does not use the element-collapsing technique, but rather removes all duplicates from the heap in one bulk extract-min operation.

Quicksort on Multiset Permutations. Building on Burge's analysis of binary search trees (BSTs) [5], Sedgewick analyzed several Quicksort variants on random permutations of multisets in a 1977 article [30]. For fat-pivot Quicksort without sampling, he found the exact result: $2\mathcal{H}_Q(\boldsymbol{x}) + n - u$ ternary comparisons on average, where $\mathcal{H}_Q(\boldsymbol{x}) = \sum_{1 \le i < j \le u} \frac{x_i x_j}{x_i + \cdots + x_j}$. 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, in two talks in 1999 and 2002, Sedgewick and Bentley [31, 32] combined the exact result with the bound $\mathcal{H}_Q(\boldsymbol{q}) \le \mathcal{H}(\boldsymbol{q}) \ln 2$—which is given by Allen and Munro in a paper [1, Theorem 3.2] that appeared just one year after Sedgewick's—and concluded that with at most $(2\mathcal{H}(\boldsymbol{x}/n) + 1)n$ comparisons on average, fat-pivot Quicksort is asymptotically optimal up to a constant factor for sorting a random permutation of any fixed multiset. It is also in these talks that Sedgewick and Bentley conjectured that this constant $2 \ln 2$ can be reduced by pivot sampling and converges to 1 when the sample size approaches infinity.

The paper [1] that provided the missing link between $\mathcal{H}_Q$ and $\mathcal{H}$ does not mention Quicksort at all; 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 term as a BST built by inserting elements drawn i.i.d. according to the access distribution. We will consider this connection between Quicksort and search trees in detail in Section 5. Parameters of BSTs under the discrete i.i.d. model were studied in more detail later [21, 2].

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" [19]. 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 they described a stable Quicksort version with exact median selection and show that it needs $O(\mathcal{H}(\boldsymbol{x}/n)n)$ comparisons even in the worst case [20]; however the constant of proportionality is one plus the constant for deterministic median selection, and thus at least 3 [8].

2 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 get 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, also to make the distinction from dual-pivot partitioning clear.

Since all duplicates of the pivot are removed before recursive calls, the number of partitioning rounds is obviously bounded by u, the number of different values in the input.

To simplify presentation, I assume in this work that partitioning retains the relative order of elements that go to the same segment; a corresponding reference Quicksort appears in Appendix B. I always refer to this Quicksort variant in this paper. As most implementations in practice do, it uses as pivots the median of k sample elements; the sample size $k = 2t + 1$, $t \in \mathbb{N}_0$, is a tuning parameter of the algorithm that we consider a fixed constant in the analysis. Whenever I speak of costs, I mean the number of ternary key comparisons.

I would like to remark that all results derived in this paper carry over to other fat-pivot partitioning implementations that do not necessarily retain the relative order, as long as they use $n \pm O(n^{1-\varepsilon})$ comparisons; my Ph.D. thesis [37] gives details on that.

3 Preliminaries

This section lists important notations; a comprehensive list is given in Appendix A. We write vectors in bold font $\boldsymbol{x} = (x_1, \ldots, x_n)$ and always understand them as column vectors (even when written "in line"). By default, all operations on vectors are meant component-wise. By $\Sigma\boldsymbol{x}$, we denote $x_1 + \cdots + x_n$, the total of $\boldsymbol{x}$. $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the nth harmonic numbers.

We use Landau-symbols (O-classes etc.) in the sense of Flajolet and Sedgewick [10, Section A.2]. For a random variable X, $\mathbb{E}[X]$ is its expectation and $\mathbb{P}[X = x]$ denotes the probability for the event $X = x$. For $\boldsymbol{q} \in [0,1]^u$ with $\Sigma\boldsymbol{q} = 1$, we write $U \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$ to denote that $\mathbb{P}[U = i] = q_i$ for $i \in [u]$. By $\mathcal{H}$ or $\mathcal{H}_{\mathrm{ld}}$ we 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.

3.1 Discrete I.I.D. Model.

In the *discrete i.i.d. model* (a.k.a. *probability model* or *expected-profile model*) with parameters $u \in \mathbb{N}$ and $\boldsymbol{q} \in (0,1)^u$ with $\Sigma\boldsymbol{q} = 1$, a random input of size n consists of n i.i.d. 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*.

We denote by X_v, for $v \in [1..u]$, the number of elements U_j 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, $\boldsymbol{X} \overset{\mathcal{D}}{=} \mathrm{Mult}(n, \boldsymbol{q})$ and $\mathbb{E}[\boldsymbol{X}] = n\boldsymbol{q}$. This explains the name of the model: We do not fix the profile, but the expectation of the profile.

The distribution function of the universe distribution is $F_U(v) = \mathbb{P}[U \le v] = \sum_{i=1}^{\lfloor v \rfloor} q_i$ for $v \in [0, u+1)$, and we denote its (generalized) inverse by $F_U^{-1} : (0,1) \to [1..u]$ with $F_U^{-1}(x) = \inf\{v \in [1..u] : F_U(v) \ge x\}$. If we abbreviate by $c_j := \sum_{i=1}^{j} q_i$ for $j = 0, \ldots, u$ the cumulative sums of $\boldsymbol{q}$, we have

$$(1) \qquad F_U(v) \;=\; c_{\lfloor v \rfloor}, \text{for } 0 \le v < u + 1,$$

$$F_U^{-1}(x) \;=\; v \quad\iff\quad x \;\in\; (c_{v-1}, c_v],$$

$$(2) \qquad \text{for } 0 < x < 1,\ v \in [1..u].$$

The discrete i.i.d. model is a natural complement of the random-permutation model: we draw elements i.i.d. from a *discrete* distribution in the former instead of a *continuous* distribution in the latter. This adds the feature of equal elements, but not much more; in particular conditional on a profile, any relative ordering of elements is equally likely to appear.

Exact-Profile Model. In the exact-profile model (a.k.a. *multiset 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 formed as 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 number v for $v = 1,\ldots,u$.

4 Lower Bound For I.I.D. Sorting

The (binary) entropy of n i.i.d. $\mathcal{D}(\boldsymbol{q})$ distributed elements is $n\mathcal{H}(\boldsymbol{q})$, so we expect at least so many *binary* comparisons to sort such an input.[3] For a formal proof, and a lower bound w.r.t. *ternary* comparisons, we follow the elegant argument of Munro and Raman [26] for multiset sorting. By averaging over the profiles, we obtain essentially the same result as in their Theorem 4, except with a weaker error term.

Theorem 4.1. (Lower Bound) *Let $u = O(n^\nu)$ for $\nu \ge 0$. For any $\varepsilon > 0$, on average $\mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n - n/\ln(2) \pm o(n^{1/2+\nu+\varepsilon})$ ternary comparisons are necessary to sort n i.i.d. $\mathcal{D}(\boldsymbol{q})$ distributed elements as $n \to \infty$.*

A detailed proof appears in Appendix E.

The restriction that u may not grow too fast w.r.t. n is required to ensure tight concentration, and is a bit of a nuisance; a more general lower bound would be very welcome. For the present paper, however, Theorem 4.1 is sufficient: for u not too large and $\mathcal{H}(\boldsymbol{q})$ not too small, we asymptotically need $n\mathcal{H}(\boldsymbol{q})$ comparisons to sort.

[3]Such entropy arguments must be taken with care, though: if $u \gg n$ we might have $\mathcal{H}(\boldsymbol{q}) \gg \mathrm{ld}\, n$, and $n\mathcal{H}(\boldsymbol{q})$ is certainly *not* a lower bound for sorting any more.

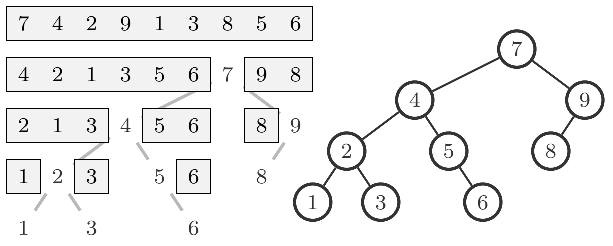

Figure 1: Execution trace of a simple Quicksort variant with $k = 1$ (no sampling) and its corresponding recursion tree. Here the first element is selected as pivot and the relative order of elements in subproblems is retained from the original input. The recursion tree coincides with the BST that results from successively inserting the original input.

5 Quicksort and Search Trees

It has become folklore that the comparison costs of Quicksort are the same as the internal path length of a BST if we have distinct elements and do not use pivot sampling; Hibbard [14] first described this fact in 1962, right after Hoare's publication of Quicksort itself [15, 16]. The correspondence extends to Quicksort with median-of-k with an appropriate fringe-balancing rule for BSTs. This is a little less widely known, but also well researched [9].

The correspondence extends naturally to inputs with equal keys, if consider a *weighted* path length, where the weight of each node corresponds to the multiplicity of that value in the input. This simple fact seems not to have found use in the literature up to now. In this section, we make the correspondence between Quicksort and search trees a bit more precise and specific, before we use it to analyze Quicksort with equal keys.

5.1 Recursion Trees.

To each execution of Quicksort we can associate a *recursion tree,* which turns out to be a binary search tree: Each partitioning step contributes an inner node, labeled with the used pivot value, and whose left and right children are the recursion trees of the left and right recursive calls, see Figure 1. Base cases, when $n \le k$ contribute a leaf node, labeled by the list of elements (in input order).

The recursion tree is exactly the binary search tree that results from successively inserting the elements in the order they appear in the original input, if we always take the first element of the list as pivot and partition so that the relative order of elements smaller resp. larger than the pivot is unaltered. In fact, the *same* set of comparisons is used in both processes, albeit in a different order.

5.2 Fringe-Balanced Search Trees. This correspondence extends to median-of-k Quicksort with *fringe-balanced* BSTs. They simulate the pivot-sampling process of Quicksort: upon constructing a search tree, we *collect* several new elements in a leaf. Once a leaf has collected $k = 2t + 1$ elements, it is *split:* From the sample of elements now available, we select the median P; it becomes the label of a new inner node, and stays so for good. Two new child leaves hold the elements that did not become pivots: the t smaller elements go to the left, the t larger to the right. Appendix C gives the full insertion procedure and some further remarks to clarify this definition.

5.3 Equal Keys. As for Quicksort, search trees are typically studied under the random-permutation model, i.e., assuming trees are built by inserting n *distinct* keys in random order. In this paper, we consider discrete i.i.d. inputs, so we have to specify how duplicates are dealt with. When we insert an element x already inserted before, what happens depends on where the element is: If x is a key in one of the inner nodes, the new insertion is without any effect; if x appears in a leaf, however, the new duplicate is appended to that leaf.

This different treatment might seem peculiar at first sight, but it is exactly the right way for our purposes: duplicates do play a role for selecting pivots—elements of the universe with higher probability contribute more duplicates in a random sample and are thus more likely selected as pivot—but once pivots have been selected, all duplicates are put in place in this one partitioning step, no matter how many of them we have.

5.4 Equivalence. A simple inductive argument yields the following correspondence.

FACT 5.1. (RECURSION TREES)
Consider median-of-k Quicksort with fat-pivot partitioning as described in Appendix B. For any input $\boldsymbol{U} = (U_1, \ldots, U_n)$ holds: Sorting $\boldsymbol{U}$ with Quicksort and inserting $\boldsymbol{U}$ successively into an initially empty k-fringe-balanced tree executes the same set of (ternary) key comparisons. □

The assumption that partitioning maintains the relative order among segments is not fulfilled for usual in-place partitioning methods, but with i.i.d. inputs, be it discrete or continuous, Fact 5.1 still holds *in distribution.* For our usage later, it is indeed sufficient that the resulting shapes of the trees have the same distribution.

6 Saturated Fringe-Balanced Trees

Consider a k-fringe-balanced tree $\mathcal{T}$ built from successively inserting elements drawn i.i.d. $\mathcal{D}(\boldsymbol{q})$ into an initially empty tree. Instead of fixing the number of insertions up front, in this section, we consider the process of indefinitely inserting i.i.d. elements. While it may happen that not all values in the universe appear in a given input, it becomes less and less likely not to have seen each possible outcomes if we keep drawing elements. Since the universe size is finite and insertions of values already present in an inner node are without effect, this means that $\mathcal{T}$ reaches one of finitely many stationary states almost surely. We call such trees *saturated* (w.r.t. the given, fixed universe).

6.1 Stochastic Model. 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

$$P \overset{\mathcal{D}}{=} f_U^{-1}(\Pi), \tag{3}$$

where Π has a Beta$(t + 1, t + 1)$ distribution, i.e., $f_\Pi(z) = z^t(1 - z)^t/\mathrm{B}(t + 1, t + 1)$ for $\mathrm{B}(a, b) = \Gamma(a)\Gamma(b)/\Gamma(a + b)$ the beta function. This is the generalized *inversion method of random sampling,* see Devroye [6, Sec. V.3.4], illustrated in our Figure 2. Recall that Beta$(t{+}1, t{+}1)$ is the distribution of the median of $2t{+}1$ i.i.d. $\mathcal{U}(0, 1)$ distributed random variables. For convenient notation, we write

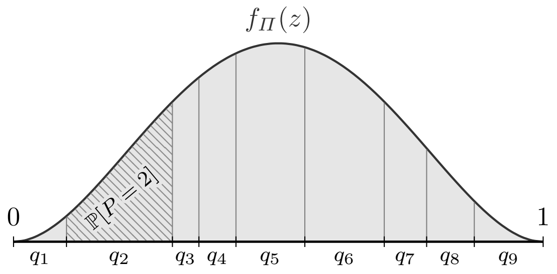

Figure 2: Illustration of pivot sampling under a discrete i.i.d. model with $u = 9$. Π is the x-coordinate of point uniformly chosen in the area under the curve, and P is the index of the interval this point lies in.

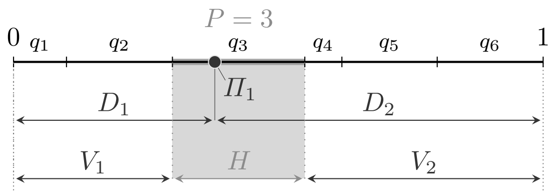

Figure 3: Relation of the quantities in the stochastic model.

$\boldsymbol{D} = (D_1, D_2) = (\Pi, 1-\Pi)$ for the induced spacings.

We further denote by V_r for $r = 1, 2$ the probability that a random element $U \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$ belongs to the rth child 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 3):

$$V_1 = q_1 + \cdots + q_{P-1}, \tag{4.1}$$

$$V_2 = q_{P+1} + \cdots + q_u, \tag{4.2}$$

$$H = q_P. \tag{4.3}$$

(In the boundary case $P = 1$, we have $V_1 = 0$, and similarly $V_2 = 0$ for $P = u$.) Finally, we denote by Z_r the "zoomed-in" universe distributions of the rth child subtree:

$$Z_1 = \left(\frac{q_1}{V_1}, \ldots, \frac{q_{P-1}}{V_1}\right), \tag{5.1}$$

$$Z_2 = \left(\frac{q_{P+1}}{V_2}, \ldots, \frac{q_u}{V_2}\right). \tag{5.2}$$

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

6.2 Weighted Path Length. Let $\mathcal{T}$ be a random k-fringe-balanced tree resulting from inserting i.i.d. $\mathcal{D}(\boldsymbol{q})$ until saturation. Each of values $v \in [u]$ appears as pivot in an inner node of $\mathcal{T}$; let Γ_v denote its depth, i.e., the number of nodes on the path from the root to that node, including endpoints. The vector $\boldsymbol{\Gamma} = (\Gamma_v)_{v\in[u]}$ is called the *node-depths vector* of $\mathcal{T}$. Finally, we write $A = A_{\boldsymbol{q}} = \boldsymbol{\Gamma}^T \boldsymbol{q}$ for the expected search cost in $\mathcal{T}$ for an element also drawn according to $\mathcal{D}(\boldsymbol{q})$. (Note that $A_{\boldsymbol{q}}$ is a random variable since $\mathcal{T}$ is random; we average over the searched key, but not over the tree it is searched in.)

The expected search costs can be described recursively: The root contributes one comparison to any search element. With probability V_r, $r = 1, 2$, the sought element is in the rth child subtree, whose expected search costs are given recursively by A_{Z_r}. With the notation from above, this yields a *distributional recurrence* for $A_{\boldsymbol{q}}$:

$$A_{\boldsymbol{q}} \overset{\mathcal{D}}{=} 1 + \sum_{r=1}^{2} V_r A^{(r)}_{Z_r}, \tag{6.1}$$

$$A_{()} = 0, \tag{6.2}$$

where $(A^{(r)}_{\boldsymbol{q}})$ are independent copies of $(A_{\boldsymbol{q}})$, which are also independent of $(\boldsymbol{V}, \boldsymbol{Z})$.

7 Quicksort With Many Duplicates

Let us denote by $C_{n,\boldsymbol{q}}$ the cost of Quicksort in the discrete i.i.d. model, i.e., the number of comparisons used by our reference Quicksort to sort n i.i.d. $\mathcal{D}(\boldsymbol{q})$ numbers using median-of-k sampling. The dependence on k will be left implicit to keep notation readable.

In this section, we derive an asymptotic approximation of $\mathbb{E}[C_{n,\boldsymbol{q}}]$. The key idea is to separate the influence of n from that of $\boldsymbol{q}$: we can approximate the costs of Quicksort by the costs of *searching* n random elements from the universe distribution in a random search tree, whose shape is independent of the searched elements. We first demonstrate this idea on an example, before we introduce two notions needed to formally state our separation theorem (Theorem 7.1).

7.1 Quicksort Costs Value-Wise. Consider an exemplary execution with universe size $u = 5$.

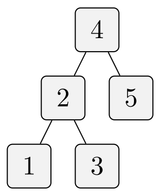

Figure 4: Exemplary recursion tree for $u = 5$. Each node represents a partitioning step, with the given pivot value. Child links correspond to child recursive calls. Empty leaves are not shown.

Assume k is small, say, $k = 3$, and n is large. Then it is very likely that each of the five values occurs at least k times. Let us assume this for the rest of the example.

Each of the five values is then used as a pivot in *exactly* one partitioning step; all duplicates of this element will be removed in this step, and only there. Leaves will thus always be empty, i.e., Insertionsort is always called for empty subranges. One possible outcome for the recursion tree is shown in Figure 4.

The cost of a single partitioning step is one comparison per element, plus the comparisons for selecting the pivot. For example, the partitioning step corresponding to pivot 2 gets as input all elements smaller than 4, i.e., $X_1 + X_2 + X_3$ many. The number of comparisons used in this step is hence $X_1 + X_2 + X_3 \pm c \cdot k$ for some global constant c depending on the median-selection algorithm.

The (random) total costs for sorting, conditional on the recursion tree from Figure 4, are then

(7)
$$C_n = \begin{cases} X_1 + X_2 + X_3 + X_4 + X_5 & \pm\ c \cdot k \\ X_1 + X_2 + X_3 & \pm\ c \cdot k \\ X_1 & \pm\ c \cdot k \\ \qquad\qquad X_3 & \pm\ c \cdot k \\ \qquad\qquad\qquad\qquad X_5 & \pm\ c \cdot k \end{cases}$$

Each line corresponds to one partitioning step and each column corresponds to one value of the universe. Reading the sum column-wise, we find that, up to the given error terms, *sorting costs are the cost of searching each input element in the (final) recursion tree:* For example, searching 3 in the tree from Figure 4, we first go left, then right and then find 2 as the pivot, so the costs are 3. And this is 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 in the above example $\boldsymbol{\Gamma} = (3, 2, 3, 1, 2)$. In general, $\boldsymbol{\Gamma}$ is exactly the node-depths vector of the recursion tree.

This "transposition" of cost contributions allows to exploit the independence in the input.

7.2 Many Duplicates & Degeneracy. The arguments used in the example above extend to the general case, but two issues need to be addressed:

1. The recursion tree was fixed in the example, but it is really a random object; in particular, the recursion tree and the profile of the input are *not independent.*

2. We have to bound the probability of degenerate cases, where not all elements occur at least k times.

To deal with these, we impose a (mild) restriction on how $\boldsymbol{q}$ (and thus u) may evolve with growing n.

Definition 7.1. (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 $(\boldsymbol{q}^{(n)})$ consists of the n i.i.d. $\mathcal{D}(\boldsymbol{q}^{(n)})$ distributed random variables $\boldsymbol{U} = (U_1, \ldots, U_n)$.

The i.i.d. model is said to have many duplicates if for $\mu_n := \min_r q_r^{(n)}$ holds $\mu_n = \Omega(n^{-1/3+\varepsilon})$ with a constant $\varepsilon > 0$.

This condition ensures that no value is extremely unlikely; it might hence be more appropriate to call it *many duplicates of each kind,* but we refrain from doing so for conciseness. With many duplicates, we expect few *degenerate* inputs in the sense of the following definition.

Definition 7.2. (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)-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 not important, we call $\boldsymbol{U}$ simply 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. Choosing n_T large enough so that the first k occurrences of all values are among the first n_T elements with high probability, and at the same time small enough to not make a large error in ignoring these elements' cost for the search costs, we obtain the following theorem; a detailed proof is given in Appendix F.

Theorem 7.1. (Separation Theorem) *Consider median-of-k Quicksort with fat-pivot partitioning under a discrete i.i.d. model with many duplicates. Then holds*

$$\mathbb{E}[C_{n,\boldsymbol{q}^{(n)}}] \;=\; \mathbb{E}[A_{\boldsymbol{q}^{(n)}}] \cdot n \;\pm\; O(n^{1-\varepsilon}) \tag{8}$$

for a constant $\varepsilon > 0$ as $n \to \infty$.

We can in fact be more precise: The error bound in Equation (8) holds for all $\varepsilon \in (0, 1-3\rho)$ when $\mu_n = \min_r q_r^{(n)} = \Omega(n^{-\rho})$.

Recall that $\mathbb{E}[A_{\boldsymbol{q}^{(n)}}]$ is the expected node depth in a saturated k-fringe-balanced tree built from $\mathcal{D}(\boldsymbol{q}^{(n)})$; it is given recursively by Equation (6). Note that $\mathbb{E}[A_{\boldsymbol{q}^{(n)}}]$ depends only on the universe distribution $\boldsymbol{q} = \boldsymbol{q}^{(n)}$ (and k), but not on n itself. We have therefore separated the influence of n and $\boldsymbol{q}$, and can investigate $\mathbb{E}[A_{\boldsymbol{q}}]$ in isolation.

8 The Expected Node Depth

In this section we are concerned with $\mathbb{E}[A_{\boldsymbol{q}}]$, the expected node depth of saturated k-fringe-balanced trees. As before we set $k = 2t+1$.

As mentioned above, previous results only cover the 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 $\mathcal{H}_Q(\boldsymbol{q}) = \sum_{1\le i<j\le u} \frac{q_i q_j}{q_i+\cdots+q_j}$ is the "Quicksort entropy" [1, Theorems 3.1 and 3.4]. For general fringe-balanced trees, no closed form seems in reach, but we can give an asymptotic approximation for large-entropy inputs.

Theorem 8.1. (Expected Node Depth) *Assume a sequence of universe distributions $(\boldsymbol{q}^{(i)})_{i\in\mathbb{N}}$ for which $\mathcal{H}_i := \mathcal{H}_{\ln}(\boldsymbol{q}^{(i)}) \to \infty$ as $i \to \infty$. The expected node depth of a saturated k-fringe-balanced tree built from i.i.d. $\boldsymbol{q}^{(i)}$ keys is for $i \to \infty$ given by*

$$\mathbb{E}[A_{\boldsymbol{q}^{(i)}}] \;=\; \frac{\mathcal{H}_{\ln}(\boldsymbol{q}^{(i)})}{H_{k+1} - H_{t+1}} \;\pm\; O\Big(\mathcal{H}_i^{\frac{t+2}{t+3}} \log(\mathcal{H}_i)\Big).$$

Table 1: A few values of α_k and the relative improvement over the no-sampling case.

t	α_{2t+1}	saving over $t=0$
0	1.38629	—
1	1.18825	14.3 %
2	1.12402	18.9 %
3	1.09239	21.2 %
4	1.07359	22.6 %
∞	1	27.9 %

We prove Theorem 8.1 in Appendix G; it relies on asymptotically matching upper and lower bounds: An inductive argument using a decomposition property of the entropy (Lemma E of Knuth [22, page 444]) shows that $\mathbb{E}[A_{\boldsymbol{q}}]$ is bounded from above and below by functions of the form $c \cdot \mathcal{H}_{\ln}(\boldsymbol{q}) + d$ for constants c and d. By introducing a parameter in the analysis, we can show the bounds for a whole family of pairs (c,d) in one shot, and we find that in a limiting sense, we can get the same c in the lower and upper bound.

9 Conclusion

Combining Theorems 7.1 and 8.1 we find that median-of-k Quicksort with fat-pivot partitioning uses on average and asymptotically

$$\mathbb{E}[C_{n,\boldsymbol{q}}] \;\sim\; \alpha_k \cdot \mathcal{H}(\boldsymbol{q})\, n \tag{9.1}$$

$$\text{with } \alpha_k \;=\; \frac{\ln 2}{H_{k+1} - H_{t+1}} \tag{9.2}$$

(ternary) comparisons to sort n i.i.d. numbers with discrete universe distribution $\boldsymbol{q}$, where $n \to \infty$, $\boldsymbol{q} = \boldsymbol{q}^{(n)}$ fulfills $\mathcal{H}(\boldsymbol{q}^{(n)}) \to \infty$ and has many duplicates, i.e., on expectation, each value of the universe occurs at least $\Omega(n^{2/3+\varepsilon})$ times in the input.

Since $H_{k+1} - H_{t+1} \to \ln(2t+2) - \ln(t+1) = \ln(2)$, this proves the conjecture of Sedgewick and Bentley in the limit for the given large-entropy inputs with many duplicates: $\alpha_k \to 1$ for $k \to \infty$. Note that α_k is the *same* constant that occurs for costs to sort random permutations; there holds $\mathbb{E}[C_n] \sim \alpha_k n \operatorname{ld} n$ [29].

Table 1 shows the first values of α_k; as for random permutations, most savings happen for the first few values of k. It is reassuring that we obtained the same constants for discrete and continuous i.i.d. models.

Technicalities aside, we can interpret our result as follows: The average node depth in search trees on continuous i.i.d. data is of order $\log n$, whereas for discrete i.i.d. data (with many duplicates), it is of order $\mathcal{H}(\boldsymbol{q})$. The fringe-balancing heuristic changes only the constant of proportionality, α_k, and is the same in both cases. Quicksort costs (on non-degenerate inputs) correspond to n random searches in such a tree, so we have to multiply by n. This point of view is a bit simplistic, but as we have seen in this paper, it gives the correct leading term.

It is good news that such a simple algorithm as fat-pivot Quicksort, which is already in wide-spread practical use, is actually very close to optimal for sorting i.i.d. inputs from many distributions, both continuous and discrete.

9.1 Extensions and Future Directions. The methods used in this paper can readily be generalized to analyze skewed sampling, i.e., when we pick another order statistic than the median of the sample. It is known that the number of comparisons is minimal when we pick the median, so this might not be very interesting in its own right, but it can be used to analyze *ninther* (a.k.a. pseudomedian-of-nine) sampling and similar schemes [37].

The restrictions on $\boldsymbol{q}^{(n)}$ are severe, even though they still allow for many sensible distributions. A simple example is $\mathcal{D}(\boldsymbol{q}) \overset{\mathcal{D}}{=} \mathcal{U}[1..u_n]$, the discrete uniform distribution, e.g., for $u_n = \lceil n^{0.3} \rceil$, then $\mathbb{E}[C_{n,\boldsymbol{q}}] \sim 0.3\alpha_k n \operatorname{ld} n$.[4]

I conjecture that $\alpha_k \, \mathcal{H}(\boldsymbol{q}) \, n$ is the correct leading term for a much larger range of parameters; but when a $q_i \ll n^{-1}$ so that i is unlikely to appear in the input *at all*, a modified entropy will have to be used. In the limit when all individual q_i are small, this modified entropy would have to equal $\operatorname{ld}(n)$. Proving such a unified term is an open problem.

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 technical challenge is that for such methods, the partitioning costs also depend on $\boldsymbol{q}$, and we do not get matching lower and upper bounds using the method of Appendix G.

[4]For this special case $\boldsymbol{q} = (\frac{1}{u}, \ldots, \frac{1}{u})$ better error bounds can be shown for $\mathbb{E}[A_{\boldsymbol{q}}]$ using Roura's continuous master theorem [28], see my Ph.D. thesis [37] for details.

Acknowledgments

The author is very grateful for several inspiring discussions with Conrado Martínez, Markus Nebel, Martin Aumüller and Martin Dietzfelbinger, which laid the ground for the present paper.

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. *Non-Uniform Random Variate Generation*. Springer New York, 1986.

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

[8] 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.

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

[10] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. ISBN 978-0-52-189806-5. URL `http://algo.inria.fr/flajolet/Publications/book.pdf`.

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

[12] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation For Computer Science*. Addison-Wesley, 1994. ISBN 978-0-20-155802-9.

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

[14] 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.

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

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

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

[18] 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.

[19] 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.

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

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] S. Roura. Improved Master Theorems for Divide-and-Conquer Recurrences. *Journal of the ACM*, 48 (2):170–205, 2001.

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

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

[31] 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`.

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

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

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

[35] 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.

[36] 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.

[37] S. Wild. *Dual-Pivot Quicksort and Beyond: Analysis of Multiway Partitioning and Its Practical Potential*. Doktorarbeit (Ph.D. thesis), Technische Universität Kaiserslautern, 2016.

Appendix

A Index of Notation

In this appendix, I collect the notations used in this work. Some might be seen as standard, including a few more is preferred to potential misunderstandings caused by omissions.

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 $\mathbb{C}$.

$\mathbb{R}_{>1}$, $\mathbb{N}_{\geq 3}$ etc.
: restricted sets $X_{\text{pred}} = \{x \in X : x \text{ fulfills pred}\}$.

$0.\overline{3}$ . . . . . . repeating decimal; $0.\overline{3} = 0.333\ldots = \frac{1}{3}$; numerals under the line form the repeated part of the decimal number.

$\ln(n)$, $\operatorname{ld}(n)$
: natural and binary logarithm; $\ln(n) = \log_e(n)$, $\operatorname{ld}(n) = \log_2(n)$.

$\boldsymbol{x}$ . . . . to emphasize that $\boldsymbol{x}$ is a vector, it is written in **bold;**
components of the vector are not written in bold: $\boldsymbol{x} = (x_1, \ldots, x_d)$;
we write $y \in \boldsymbol{x}$ also for vectors to mean $y \in \{x_1, \ldots, x_d\}$;
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, $[m..n] = \{m, m+1, \ldots, n\}$; $[n] = [1..n]$.

[stmt], $[x = y]$
: Iverson bracket, [stmt] $= 1$ if stmt is true, [stmt] $= 0$ otherwise.

$\boldsymbol{x} + 1$, $2^{\boldsymbol{x}}$, $f(\boldsymbol{x})$
: 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 $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, we 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 [10, Section A.2].

$x \pm y$ . . . . x with absolute error $|y|$; formally the interval $x \pm y = [x - |y|, x + |y|]$; as with O-terms, we use one-way equalities $z = x \pm y$ instead of $z \in x \pm y$.

Δ_d . . . . . . $(d-1)$-dimensional standard open simplex, $\Delta_d := \{\boldsymbol{x} \in \mathbb{R}^{d-1} : \boldsymbol{x} > 0 \wedge \Sigma\boldsymbol{x} < 1\} \subseteq (0, 1)^{d-1}$

$\Gamma(z)$ . . . . 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; defined in $\mathrm{B}(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha + \beta)$

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; we 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.

$\mathcal{U}(a, b)$ . . uniformly in $(a, b) \subset \mathbb{R}$ distributed random variable.

$\mathcal{U}[a..b]$ . . . discrete uniformly in $[a..b] \subset \mathbb{Z}$ distributed random variable.

$\mathcal{D}(\boldsymbol{p})$ . . . . 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.

$\operatorname{Beta}(\alpha, \beta)$
: Beta distributed random variable with shape parameters $\alpha \in \mathbb{R}_{>0}$ and $\beta \in \mathbb{R}_{>0}$.

$\operatorname{Bin}(n, p)$. binomial distributed random variable with $n \in \mathbb{N}_0$ trials and success probability $p \in [0, 1]$;
$X \overset{\mathcal{D}}{=} \operatorname{Bin}(n, p)$ is equivalent to $(X, n - X) \overset{\mathcal{D}}{=} \operatorname{Mult}(n; p, 1 - p)$.

Mult$(n, \boldsymbol{p})$
: multinomially distributed random variable; $n \in \mathbb{N}_0$ and $\boldsymbol{p} \in [0,1]^d$ with $\Sigma\boldsymbol{p} = 1$.

$\mathcal{H}(\boldsymbol{p})$, $\mathcal{H}_{\mathrm{ld}}(\boldsymbol{p})$, $\mathcal{H}_{\ln}(\boldsymbol{p})$
: Shannon entropy of information theory; $\mathcal{H}_{\mathrm{ld}}(p_1, \ldots, p_d) = \sum_{r=1}^{d} p_r \operatorname{ld}(1/p_r)$; similarly $\mathcal{H}_{\ln}$ is the base-e entropy. $\mathcal{H}_{\ln}(p_1, \ldots, p_d) = \sum_{r=1}^{d} p_r \ln(1/p_r)$; we write $\mathcal{H}$ for $\mathcal{H}_{\mathrm{ld}}$.

A.3 Notation for the Algorithm.

n .... length of the input array, i.e., the input size.

u .... universe size $u \in \mathbb{N}$.

$\boldsymbol{q}$ .... 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; $U_i \overset{\mathcal{D}}{=} \mathcal{D}(\boldsymbol{q})$, 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.

$C_{n,\boldsymbol{q}}$ .... (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.

$c_0, \ldots, c_u$ cumulative sums of $\boldsymbol{q}$; $c_j := \sum_{i=1}^{j} q_i$ for $j = 0, \ldots, u$.

Π ...... continuous pivot value; $\Pi \overset{\mathcal{D}}{=} \mathrm{Beta}(t+1, t+1)$

$\boldsymbol{D} = (D_1, D_2)$
: continuous spacings of the unit interval $(0,1)$ induced by Π i.e., $\boldsymbol{D} = (\Pi, 1 - \Pi)$.

P .... (random) value of chosen pivot in the first partitioning step; $P = F_U^{-1}(\Pi)$, for F_U the cumulative distribution function of $\mathcal{D}(\boldsymbol{q})$.

$\boldsymbol{V}$ ...... non-pivot class probabilities, see Equation (4).

$\boldsymbol{Z}$ .... zoomed-in distributions; see Equation (5).

H ...... hitting probability, Equation (4).

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.removeFirst(1)
 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)
```

$\boldsymbol{\Gamma}$ .... $\boldsymbol{\Gamma} \in \mathbb{N}_0^u$; node-depths vector in search tree; Γ_v is the random cost of searching value $v \in [u]$.

$A_{\boldsymbol{q}}$ ...... random internal path length of recursion tree, $A_{\boldsymbol{q}} = \boldsymbol{\Gamma}^T \boldsymbol{q}$; see Equation (6).

B Reference Quicksort

Algorithm 1 is our reference fat-pivot Quicksort implementation with median-of-k sampling. It is formulated on linked lists to allow a simple stable order-preserving partitioning method, which simplifies the description of the analysis a bit. Our average-case results apply equally well to more efficient array-based implementations as the one given by Bentley and McIlroy [4] or the simpler variant in Sedgewick and Wayne [33].

Apart from selecting the median, one partitioning step in Algorithm 1 uses exactly n ternary comparisons (*cmp* calls).

As we always use the first k elements as sample elements, and pass the t small sampled-out elements first to L_1 resp. the t large sampled-out elements to L_3, we exactly mimic the behavior of a k-fringe-balanced tree (see below); Fact 5.1 is obviously fulfilled.

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 $\boldsymbol{U} \setminus P$ // all except pivot
        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 $Inner\bigl(P, Leaf(C_1), Leaf(C_2)\bigr)$
else $\mathcal{T}$ is $Inner(P, \mathcal{T}_1, \mathcal{T}_2)$
    case distinction on $cmp(U, P)$
        in case $U == P$ do
            return "Already present."
        in case $U < P$ do
            return $Inner\bigl(P, \text{Insert}_k(\mathcal{T}_1, x), \mathcal{T}_2\bigr)$
        in case $U > P$ do
            return $Inner\bigl(P, \mathcal{T}_1, \text{Insert}_k(\mathcal{T}_2, x)\bigr)$
    end cases
end if
```

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
```

C Fringe-Balanced Trees

In this appendix, we give some details on fringe-balanced trees. Unfortunately, fringe-balanced trees appear under a handful of different names in the literature: they have also been called *locally balanced search trees* [36], *fringe-balanced trees* [27], *diminished trees* [13], and *iR / SR trees* [17, 18]. We use *fringe-balanced trees*, since it is the most vivid term and has been widely adopted at least in the analysis-of-algorithms community.

Along with the different names come slight variations in the definitions; we remark that our definition is again slightly different in the base cases than usual definitions to better match Quicksort recursion trees. To avoid ambiguities we describe our class of trees in some detail here.

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 $T_0 = Leaf()$ and $T_i = \text{Insert}_k(\mathcal{T}_{i-1}, x_i)$ for $1 \le i \le n$, we have $\mathcal{T} = \mathcal{T}_n$.

Growing trees with Algorithm 2 enforces certain shapes 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.)

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's monograph [9]. The case of equal elements has not been considered except for the trivial case $k = 1$, i.e., ordinary BSTs; see Kemp [21], Archibald and Clément [2].

D Properties of Entropy

The following properties of the entropy function follow from elementary facts of real analysis. (Detailed proofs are given in my Ph.D. thesis [37].)

Lemma D.1. *Let $\Delta_u \subset [0,1]^u$ be the set of all stochastic vectors with u components. 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. It holds:*

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

(b) $\max_{\boldsymbol{x}\in\Delta_u} \mathcal{H}_{\ln}(\boldsymbol{x}) = \ln(u)$.

(c) *$\mathcal{H}_{\ln}$ is Hölder-continuous in all of $[0,1]^u$ 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

$$C_h \;=\; \left(\int_0^1 |\ln(t)+1|^{\frac{1}{1-h}}\right)^{1-h}$$

For example, $h = 0.99$ yields $C_h < 38$.

□

D.1 Concentration. For the reader's convenience, we restate the classical Chernoff bound here. It appears as Theorem 2.1 of McDiarmid [24].

Lemma D.2. (Chernoff Bound)
Let $X \overset{\mathcal{D}}{=} \mathrm{Bin}(n,p)$ $(n \in \mathbb{N},\, p \in (0,1))$ and $\delta \ge 0$. Then

$$\text{(D.1)}\qquad \mathbb{P}\left[\left|\frac{X}{n} - p\right| \ge \delta\right] \;\le\; 2\exp(-2\delta^2 n).$$

□

Using the Chernoff bound for the binomial distribution, we obtain the following concentration property of the entropy of a normalized multinomial variable.

Lemma D.3. *Let $u \in \mathbb{N}$ and $\boldsymbol{p} \in (0,1)^u$ with $\Sigma\boldsymbol{p} = 1$, and $\boldsymbol{X} \overset{\mathcal{D}}{=} \mathrm{Mult}(n, \boldsymbol{p})$. Then holds*

$$\text{(D.2)}\qquad \mathbb{E}\left[\mathcal{H}_{\ln}\left(\frac{\boldsymbol{X}}{n}\right)\right] \;=\; \mathcal{H}_{\ln}(\boldsymbol{p}) \;\pm\; \rho,$$

where we have for any $\delta \ge 0$, $h \in (0,1)$ and C_h as in Lemma D.1 that

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

If $u = O(n^\nu)$ as $n \to \infty$ for a constant $\nu \in [0, \frac{1}{2})$, we have $\rho = o(n^{-1/2+\varepsilon})$ for any fixed $\varepsilon > \nu$.

Proof. By the *union bound* we have

(D.4)
$$\mathbb{P}\left[\left\|\frac{\boldsymbol{X}}{n} - \boldsymbol{p}\right\|_\infty \ge \delta\right] \;=\; \mathbb{P}\left[\bigvee_{i=1}^{u}\left|\frac{X_i}{n} - p_i\right| \ge \delta\right]$$

$$\text{(D.5)}\qquad \le\; \sum_{i=1}^{u}\mathbb{P}\left[\left|\frac{X_i}{n} - p_i\right| \ge \delta\right]$$

$$\text{(D.6)}\qquad \underset{\text{Lemma D.2}}{\le}\; 2u\exp(-2\delta^2 n).$$

To use this in bounding $\mathbb{E}[\mathcal{H}_{\ln}(\frac{\boldsymbol{X}}{n}) - \mathcal{H}_{\ln}(\boldsymbol{p})]$, we divide the domain $[0,1]^u$ 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 D.1, $\mathcal{H}_{\ln}$ is Hölder-continuous for any exponent $h \in (0,1)$ with Hölder-constant C_h. Using this, the boundedness of $\mathcal{H}_{\ln}$ and Equation (D.6), we find

(D.7)
$$\mathbb{E}\left[\mathcal{H}_{\ln}\left(\frac{\boldsymbol{X}}{n}\right) - \mathcal{H}_{\ln}(\boldsymbol{p})\right]$$
$$\le \sup_{\boldsymbol{\xi}:\|\boldsymbol{\xi}\|_\infty<\delta}\left|\mathcal{H}_{\ln}(\boldsymbol{p}+\boldsymbol{\xi}) - \mathcal{H}_{\ln}(\boldsymbol{p})\right| \cdot (1 - 2ue^{-2\delta^2 n})$$

(D.8)
$$+ \underbrace{\max_{\boldsymbol{x}\in[0,1]^u}\mathcal{H}_{\ln}(\boldsymbol{x})}_{\ln(u)} \cdot 2ue^{-2\delta^2 n}$$

(D.9)
$$\le\; C_h\delta^h u\cdot(1 - 2e^{-2\delta^2 n}) \;+\; 2u\ln(u)e^{-2\delta^2 n}.$$

This proves the first part of the claim.

For the second part, we choose a Hölder exponent $h \in (0,1)$ so that $\varepsilon > \frac{1-h}{2} + \nu$ holds, i.e., we can write $h = 1 - 2(\varepsilon - \nu) + 4\lambda$ for a constant $\lambda > 0$, and $h = (1 - 2(\varepsilon-\nu))/(1-2\lambda')$ for another constant $\lambda' > 0$. We may further assume $\varepsilon < \frac{1}{2}$; for larger values the claim is vacuous. Finally, we have $u \le dn^\nu$ for a constant d and large enough n. We then choose $\delta = n^c$ with $c = (-\frac{1}{2} - \frac{1/2-(\varepsilon-\nu)}{h})/2 = -\frac{1}{4} - \frac{1-2(\varepsilon-\nu)}{4h}$. For large n we thus have

(D.10)
$$\rho \cdot n^{1/2-\varepsilon}$$
$$\le\; C_h\delta^h n^{1/2-\varepsilon}u(1 - 2\exp(-2\delta^2 n))$$

(D.11)
$$+\; 2n^{1/2-\varepsilon}u\ln(u)\exp(-2\delta^2 n)$$

$$\leq \underbrace{C_h d n^{-\lambda}}_{\to 0} \cdot (1 - \underbrace{2\exp(-2n^{\lambda'})}_{\to 0})$$

(D.12)

$$+ \underbrace{2d\nu \ln(n) \exp\Bigl(-2n^{\lambda'} + (\tfrac{1}{2} - \varepsilon + \nu)\ln(n)\Bigr)}_{\to 0}$$

(D.13)

$$\to \;\; 0$$

for $n \to \infty$, which implies the claim. ☐

D.2 Expected Entropy of Beta Variables.

$\boldsymbol{D} = (\mathit{\Pi}, 1-\mathit{\Pi})$ can be seen as a random binary probability distribution, so it makes sense to ask for the expected entropy of this distribution.

Lemma D.4. (Expected Entropy) *For $\mathit{\Pi} \overset{\mathcal{D}}{=} \mathrm{Beta}(t+1, t+1)$, we have*

$$\text{(D.14)} \qquad \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{D})] \;=\; H_{k+1} - H_{t+1}$$

Proof. We need the following integral, which a special case of Equation (4.253-1), p. 540, of Gradshteyn and Ryzhik [11] with $r = 1$:

$$\text{(D.15)} \quad \int_0^1 z^{a-1}(1-z)^{b-1}\ln(z)\,dz$$
$$= \;\; \mathrm{B}(a,b)\bigl(\psi(a) - \psi(a+b)\bigr), \quad a, b > 0.$$

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 $\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{D})] = -2\mathbb{E}[\mathit{\Pi}\ln(\mathit{\Pi})]$; using the above integral and the relation $\psi(n+1) = H_n - \gamma$ (Equation (5.4.14) of the DLMF [7]) we find

(D.16)

$$\mathbb{E}[\mathit{\Pi}\ln(\mathit{\Pi})]$$

(D.17)

$$= \int_0^1 x\ln(x)\frac{x^t(1-x)^t}{\mathrm{B}(t+1,t+1)}\,dx$$

(D.18)

$$= \frac{\mathrm{B}(t+2,t+1)}{\mathrm{B}(t+1,t+1)}\int_0^1 \ln(x)\frac{x^{t+1}(1-x)^t}{\mathrm{B}(t+2,t+1)}\,dx$$

(D.19)

$$\underset{(D.15)}{=} \frac{t+1}{k+1}\bigl(\psi(t+2) - \psi(k+2)\bigr)$$

(D.20)

$$= \frac{t+1}{2t+2}\bigl(H_{t+1} - H_{k+1}\bigr)$$

(D.21)

$$= \frac{1}{2}\bigl(H_{t+1} - H_{k+1}\bigr)\,.$$

Inserting yields the claim. ☐

The special case $t = 0$ of Equation (D.14) appears Section 5.0 of Bayer [3]; his result appears also as Exercise 6.2.2–37 of Knuth [22].

E Proof of the Lower Bound

Proof of Theorem 4.1. 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

(E.22)

$$\mathbb{E}[C_{n,\boldsymbol{q}}] \;+\; \sum_{v=1}^{u}\mathbb{E}[X_v \,\mathrm{ld}(X_v)]$$

(E.23)

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

with ρ as in Lemma D.3

(E.24)

$$= \;\; n\,\mathrm{ld}(n) + \mathbb{E}[C_{n,\boldsymbol{q}}] + n\sum_{v=1}^{u} q_v\,\mathrm{ld}(q_v) \;\pm\; n\,\frac{\rho}{\ln(2)}$$

(E.25)

$$\geq \;\; n\,\mathrm{ld}(n) - n/\ln(2) \pm O(\log n)$$

since the latter is the lower bound on the average number of comparisons, see, e.g., Equation 5.3.1–(37) of Knuth [22]. It follows that

$$\text{(E.26)}\quad \mathbb{E}[C_{n,\boldsymbol{q}}] \;\geq\; \mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n - \frac{n}{\ln(2)} \;\pm\; n\,\frac{\rho}{\ln(2)}.$$

For $u = O(n^{\nu})$ with $\nu \in [0, \frac{1}{2})$ we get asymptotically for any $\varepsilon > 0$

(E.27)
$$\mathbb{E}[C_{n,\boldsymbol{q}}] \;\geq\; \mathcal{H}_{\mathrm{ld}}(\boldsymbol{q})n - \frac{n}{\ln(2)} \;\pm\; o(n^{\frac{1}{2}+\nu+\varepsilon}).$$

□

F Proof of the Separation Theorem

Proof of Theorem 7.1. The proof will basically have to show that degenerate inputs are so unlikely that we can ignore them, and use this to separate the distribution of recursion trees and search costs. We then address the costs of sampling pivots, and put the results together.

F.1 Probability of Degenerate Profiles. We start noting the following basic fact.

Fact F.1. *In any discrete i.i.d. model holds $u_n \leq \frac{1}{\mu_n}$ for all n. In particular, an i.i.d. model with many duplicates has $u_n = O(n^{1/3-\varepsilon})$.*

Proof. Since μ_n is the smallest entry of $\boldsymbol{q}^{(n)}$ we have $1 = \Sigma\boldsymbol{q}^{(n)} \geq u_n\mu_n$, so $u_n \leq 1/\mu_n$. The second part follows directly from the definition. □

Bounding the probability of degenerate profiles is essentially a standard application of the Chernoff bound (our Lemma D.2). To obtain a meaningful statement, we give an asymptotic result.

Lemma F.1. (Non-Degenerate W. H. P.) *Assume a discrete i.i.d. model with $\mu_n = \min_r q_r^{(u_n)} = \Omega(n^{-\rho})$ for $\rho \in [0, \frac{1}{2})$ and let $k \in \mathbb{N}$ and $\nu \in (2\rho, 1)$. Then holds: the probability of an input of size n to be (ν, k)-degenerate is in $o(n^{-c})$ for any constant c.*

Proof. Let $\rho \in [0, \frac{1}{2})$, k and $\nu \in (2\rho, 1)$ by given. Set $\varepsilon = \nu - 2\rho > 0$ 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* and the *Chernoff bound* yields

(F.28)
$$\mathbb{P}[\neg\boldsymbol{Y}^{(n)} \geq k] \;=\; \mathbb{P}\Bigg[\bigvee_{r=1}^{u_n} Y_r^{(n)} < k\Bigg]$$

$$\text{(F.29)}\qquad \leq\; \sum_{r=1}^{u_n}\mathbb{P}[Y_r^{(n)} < k]$$

$$\text{(F.30)}\qquad \leq\; u_n \cdot \mathbb{P}[Y_1^{(n)} < k]\;;$$

with $\delta = \mu_n - \frac{k}{n_T}$, this is

$$\text{(F.31)}\qquad =\; u_n \cdot \mathbb{P}\Bigg[\frac{X_1}{n_T} < \mu_n - \delta\Bigg]$$

$$\text{(F.32)}\qquad \underset{\text{Lemma D.2}}{\leq}\; 2u_n\exp\Bigg(-2\Big(\mu_n - \frac{k}{n_T}\Big)^2 n_T\Bigg)\,.$$

By assumption $\mu_n = \Omega(n^{-\rho})$, so for some constants n_μ and $c_\mu > 0$ we have $\mu_n \geq c_\mu n^{-\rho}$ for $n \geq n_\mu$, and by definition is $n_T = \lceil n^{\nu}\rceil = n^{2\rho+\varepsilon} \pm O(1)$. Inserting these, we find

(F.33)
$$\Big(\mu_n - \frac{k}{n_T}\Big)^2 n_T$$

(F.34)
$$\geq\; \Big(c_\mu n^{-\rho}(1 \;\pm\; O(n^{-\rho-\varepsilon}))\Big)^2 n^{\nu}(1 \pm O(n^{-\nu}))$$

(F.35)
$$=\; c_\mu^2 n^{\varepsilon}(1 \;\pm\; O(n^{-\rho-\varepsilon})).$$

By Fact F.1, also $u_n = O(n^{\rho})$, so $\ln(u_n) = O(\log n)$, and we obtain for any $c \geq 0$ that

(F.36)
$$n^c \cdot \mathbb{P}[\neg\boldsymbol{Y}^{(n)} \geq k]$$

(F.37)
$$\underset{\text{(F.32)}}{\leq} 2\exp\Bigg(c\ln(n) + \ln(u_n) \;-\; 2\Big(\mu_n - \frac{k}{n_T}\Big)^2 n_T\Bigg)$$

(F.38)
$$\underset{\text{(F.35)}}{=} 2\exp\Big(-2c_\mu^2 n^{\varepsilon}(1 \;\pm\; o(1))\Big)$$

(F.39)
$$\to\; 0, \qquad (n \to \infty)\,.$$

This concludes the proof of the lemma. □

In Theorem 7.1, we assume a discrete i.i.d. model with many duplicates, i.e., $\frac{1}{\mu_n} = O(n^\rho)$ with $\rho \in [0, \frac{1}{3})$. For given $\varepsilon \in (0, 1-3\rho)$, we set

$$\nu \;:=\; 1 - \varepsilon - \rho \;\in\; (2\rho, 1-\rho). \tag{F.40}$$

Then, by Lemma F.1, an input 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 (8), since costs for any input are at most quadratic.

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.

F.2 Independence of Recursion Trees. We now turn to the distribution of the recursion trees. By Fact 5.1 (page 5), recursion trees for median-of-k Quicksort have the same distribution as naturally grown k-fringe-balanced 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 of these chooses pivots after inspecting k elements.

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. At the latest upon inserting the kth occurrence of v will the leaf overflow and a new internal node containing a pivot v is created.

In our non-degenerate input, the first k duplicates appear among the n_T first 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 elements appear for sure in inner nodes, so all leaves are empty by that time, and remain so for good: the recursion tree has reached a stationary state.

As before, we denote by $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(\boldsymbol{q})$ the node-depths vector for the final recursion tree, and further, 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.

F.3 Contribution of Pivot Selection. With the search costs $\boldsymbol{\Gamma}^T \boldsymbol{X}$, we get the actual costs up to an error of $\pm ck$ per partitioning step, for some constant c depending on the median-selection algorithm. As argued above, we have at most u partitioning steps, so overall cost are $C_{n,\boldsymbol{q}} = \boldsymbol{\Gamma}^T \boldsymbol{X} \pm O(k \cdot u)$.

F.4 Result. We now have all ingredients to compute the overall costs of Quicksort. Recall that $u_n = O(n^\rho)$ and $n_T \sim n^\nu$ with $\nu < 1 - \rho$. Since $\boldsymbol{\Gamma} \le u$, we have for non-degenerate inputs the following stochastic representation:

$$\begin{aligned}
C_{n,\boldsymbol{q}} &= \boldsymbol{\Gamma}^T \boldsymbol{X} \;\pm\; O(k \cdot u) && \text{(F.41)}\\
&= \boldsymbol{\Gamma}^T \tilde{\boldsymbol{X}} \;\pm\; O(u \cdot n_T) && \text{(F.42)}\\
&= \boldsymbol{\Gamma}^T \tilde{\boldsymbol{X}} \;\pm\; O(n^{1-\varepsilon}). && \text{(F.43)}
\end{aligned}$$

Taking expectations over all non-degenerate inputs and exploiting independence yields

$$\begin{aligned}
&\text{(F.44)}\\
\mathbb{E}[C_{n,\boldsymbol{q}}] &= \mathbb{E}[\boldsymbol{\Gamma}^T \tilde{\boldsymbol{X}}] \;\pm\; O(n^{1-\varepsilon}) \\
&= \mathbb{E}[\boldsymbol{\Gamma}]^T \cdot \mathbb{E}[\tilde{\boldsymbol{X}}] \;\pm\; O(n^{1-\varepsilon}) && \text{(F.45)}\\
&= \underbrace{\left(\mathbb{E}[\boldsymbol{\Gamma}]^T \cdot \boldsymbol{q}^{(n)}\right)}_{\mathbb{E}[A_{\boldsymbol{q}}]} \cdot n \;\pm\; O(n^{1-\varepsilon}), && \text{(F.46)}
\end{aligned}$$

with $A_{\boldsymbol{q}}$ as given in Section 6.2. As argued above, the contribution of degenerate inputs is in $o(n^{-c})$ for any c and thus covered by $O(n^{1-\varepsilon})$, so Equation (F.46) holds also for the unconditional expectation. This concludes the proof of Theorem 7.1. □

G Proof of Expected Node Depth

Proof of Theorem 8.1. We first show upper and lower bounds, and afterwards combine them to obtain the claimed asymptotic expansion.

Before we start let us state for reference the following aggregation property for the entropy; It follows directly form Lemma E of Knuth [22, page 444]: Let $\boldsymbol{q} \in [0,1]^u$ with $\Sigma\boldsymbol{q} = 1$ and consider a fixed pivot value $P \in [u]$. Recall $\boldsymbol{Z}$, $\boldsymbol{V}$, H from Section 6.1. Then holds

$$\text{(G.47)} \quad \mathcal{H}(\boldsymbol{q}) \;=\; \mathcal{H}(V_1, H, V_2) + \sum_{r=1}^{2} V_r \mathcal{H}(Z_r) \,.$$

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

G.1 Upper Bound. We now show a class of upper bounds characterized by parameter ε.

Proposition G.1. (Upper Bound) *Let $A_{\boldsymbol{q}}$ satisfy Equation (6) on page 6, and let $\varepsilon \in (0,1)$ be given. Define*

$$\text{(G.48)} \qquad c \;=\; c_\varepsilon \;=\; \frac{1}{\tilde{H} - 4\varepsilon\tilde{h}}\,,$$

$$\text{(G.49)} \qquad d \;=\; d_\varepsilon \;=\; \frac{(t+1)\,\mathrm{B}(t+1,t+1)}{\varepsilon^{t+2}(1-\varepsilon)^t}\,,$$

$$\text{(G.50)} \qquad \textit{where} \;\; \tilde{H} \;=\; H_{k+1} - H_{t+1}$$

$$\text{(G.51)} \qquad \textit{and} \;\; \tilde{h} \;=\; H_k - H_t \,.$$

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

Proof. Let ε with $c = c_\varepsilon \ge 0$ be given; then also $d = \varepsilon \ge 0$ holds. 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 (6.2). 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 (6) and conditioning on the pivot value P:

(G.52)
$$\mathbb{E}[A_{\boldsymbol{q}}] \;=\; 1 \;+\; \mathbb{E}_P\left[\sum_{r=1}^{2} \mathbb{E}[V_r A_{Z_r} \mid P]\right]$$

using the inductive hypothesis

(G.53)
$$\le\; 1 \;+\; \mathbb{E}_P\left[\sum_{r=1}^{2} V_r\,(c\mathcal{H}_{\ln}(Z_r) + d)\right]$$

(G.54)
$$=\; 1 \;+\; c \cdot \mathbb{E}\left[\sum_{r=1}^{2} V_r\,\mathcal{H}_{\ln}(Z_r)\right] \;+\; d \cdot \mathbb{E}[\Sigma\boldsymbol{V}]$$

$$\underset{\text{(G.47)}}{=}\; c \cdot \mathcal{H}_{\ln}(\boldsymbol{q})$$

(G.55)
$$+\; \underbrace{1 \;-\; c \cdot \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)] \;+\; d \cdot \mathbb{E}[\Sigma\boldsymbol{V}]}_{\varpi}$$

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

1. Case $\mu < \varepsilon$:
   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 3 (page 6)) we have

   (G.56)
   $$V_r \;=\; D_r \;+\; (-2\varepsilon, 0), \qquad (r = 1, 2)$$

   using interval-arithmetic notation. For the expected partitioning entropy, this means

   (G.57)
   $$\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$$

   (G.58)
   $$\ge\; \sum_{r=1}^{2} \mathbb{E}[V_r \ln(1/V_r)]$$

   using Equation (G.56) and $x\log(1/x) \ge (x - \varepsilon)\log(1/(x + \varepsilon'))$ for $\varepsilon, \varepsilon' \ge 0$ and $x \in [0,1]$, this is

   (G.59)
   $$\ge\; \sum_{r=1}^{2} \mathbb{E}[(D_r - 2\varepsilon)\ln(1/D_r)]$$

   (G.60)
   $$=\; \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{D})] \;+\; 2\varepsilon \sum_{r=1}^{2} \mathbb{E}[\ln(D_r)]$$

   $$=\; 1 \;+\; \mathbb{E}_P\left[\sum_{r=1}^{2} \mathbb{E}[V_r A_{Z_r} \mid P]\right]$$

   (G.61)
   $$\underset{\text{(D.14), (D.15)}}{=}\; \tilde{H} \;+\; 2\varepsilon \sum_{r=1}^{2} \bigl(\psi(t+1) - \psi(k+1)\bigr)$$

   (G.62)
   $$=\; \tilde{H} \;-\; 4\varepsilon(H_k - H_t)$$

   (G.63)
   $$=\; \tilde{H} \;-\; 4\varepsilon\tilde{h} \,.$$

With this, c satisfies by assumption $c \geq 1/\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)] \geq 0$, which implies

$$\varpi \;\leq\; 1 \;-\; \frac{1}{\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]} \cdot \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$$

$$\text{(G.64)} \qquad +\; d \cdot \underbrace{\mathbb{E}[\Sigma \boldsymbol{V}]}_{\leq 1}$$

$$\text{(G.65)} \qquad \leq\; d.$$

The inductive step is proven in this case.

2. Case $\mu \geq \varepsilon$:
In the second case, there is a *likely* value $v \in [u]$ with $q_v \geq \varepsilon$. 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 2), 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:

$$\text{(G.66)} \qquad \mathbb{P}[P_r = v] \;\geq\; \min_{0 \leq \zeta \leq 1-\varepsilon} \int_{z=\zeta}^{\zeta+\varepsilon} f_{\Pi_r}(z)\, dz$$

$$\text{(G.67)} \qquad =\; \min_{0 \leq \zeta \leq 1-\varepsilon} \int_{z=\zeta}^{\zeta+\varepsilon} \frac{z^t(1-z)^t}{\mathrm{B}(t+1,t+1)}\, dz$$

$$\text{(G.68)} \qquad =\; \int_{z=0}^{\varepsilon} \frac{z^t(1-z)^t}{\mathrm{B}(t+1,t+1)}\, dz\,,$$

$$\text{(G.69)} \qquad \geq\; \int_{0}^{\varepsilon} \frac{z^t(1-\varepsilon)^t}{\mathrm{B}(t+1,t+1)}\, dz$$

$$\text{(G.70)} \qquad =\; \frac{\varepsilon^{t+1}(1-\varepsilon)^t}{(t+1)\,\mathrm{B}(t+1,t+1)}\,.$$

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

$$\text{(G.71)} \qquad \mathbb{E}[H] \;\geq\; q_v \cdot \mathbb{P}[P = v]$$

$$\text{(G.72)} \qquad \underset{\text{(G.70)}}{\geq} \frac{\varepsilon^{t+2}(1-\varepsilon)^t}{(t+1)\,\mathrm{B}(t+1,t+1)}.$$

This means that $d \geq 1/\mathbb{E}[H]$, which implies

$$\varpi - d \;\leq\; 1 \;-\; c \cdot \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$$

$$\text{(G.73)} \qquad + d \cdot \mathbb{E}[\Sigma \boldsymbol{V}] \;-\; d$$

$$\text{(G.74)} \qquad \leq\; 1 - (1 - \mathbb{E}[\Sigma \boldsymbol{V}]) \cdot \frac{1}{\mathbb{E}[H]}$$

$$\text{(G.75)} \qquad =\; 0.$$

This concludes the inductive step also in the second case.

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

G.2 Lower Bound. The lower bound on $\mathbb{E}[A_{\boldsymbol{q}}]$ uses basically the same techniques; only a few details differ.

Proposition G.2. *Let $A_{\boldsymbol{q}}$ satisfy Equation* (6) *on page 6, and let $\varepsilon \in (0, 1/e)$ be given. Define*

$$\text{(G.76)} \qquad c \;=\; c_\varepsilon \;=\; \frac{1}{\tilde{H} + 4\varepsilon + \varepsilon \ln(1/\varepsilon)}\,,$$

$$\text{(G.77)} \qquad d \;=\; d_\varepsilon \;=\; (c_\varepsilon \ln(3) \;-\; 1) \frac{(t+1)\,\mathrm{B}(t+1,t+1)}{\varepsilon^{t+2}(1-\varepsilon)^t}\,.$$

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

Proof. Let $\varepsilon \in (0, 1/e)$ with $d = d_\varepsilon \geq 0$ be given; then also $c = \varepsilon \geq 0$ holds. The proof is similar to that for Proposition G.1, so we emphasize the differences and skip identical parts. If $u = 0$, i.e., $\boldsymbol{q} = ()$, the claim holds since $d \geq 0$.

Now assume $u \geq 1$ 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

$$\text{(G.78)} \qquad \mathbb{E}[A_{\boldsymbol{q}}] \;\geq\; 1 \;+\; \mathbb{E}_P\left[\sum_{r=1}^{s} V_r \left(c\mathcal{H}_{\ln}(Z_r) - d\right)\right]$$

$$\underset{\text{(G.47)}}{=} c \cdot \mathcal{H}_{\ln}(\boldsymbol{q})$$

$$\text{(G.79)} \qquad + \underbrace{1 \;-\; c \cdot \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)] \;-\; d \cdot \mathbb{E}[\Sigma \boldsymbol{V}]}_{\varpi},$$

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

1. Case $\mu < \varepsilon$:
   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

   (G.80)

   $$\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$$

   (G.81)
   $$= \sum_{r=1}^{2} \mathbb{E}\big[V_r \ln(1/V_r)\big] \;+\; \mathbb{E}\big[H \ln(1/H)\big]$$

   using Equation (G.56) and $(x-\varepsilon)\log(1/(x-\varepsilon)) \le x\ln(1/x)+\varepsilon'$ for $0 \le \varepsilon \le \varepsilon'$ and $x \in [0,1]$, this is

   (G.82)
   $$\le \sum_{r=1}^{2} \mathbb{E}\big[D_r \ln(1/D_r) + 2\varepsilon\big] \;+\; \varepsilon \ln(1/\varepsilon)$$

   (G.83)
   $$\underset{(\text{D.14})}{=} \tilde{H} \;+\; 4\varepsilon \;+\; \varepsilon\ln(1/\varepsilon)$$

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

   $$\varpi \;\ge\; 1 \;-\; \frac{1}{\mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]} \cdot \mathbb{E}[\mathcal{H}_{\ln}(\boldsymbol{V}, H)]$$

   (G.84)
   $$- \,d \cdot \underbrace{\mathbb{E}[\varSigma \boldsymbol{V}]}_{\le 1}$$

   (G.85)
   $$\ge \; -d.$$

   The inductive step is proven in this case.

2. Case $\mu \ge \varepsilon$:
   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 Proposition G.1, we find

   $$\text{(G.86)} \qquad \mathbb{E}[H] \underset{(\text{G.72})}{\ge} \frac{\varepsilon^{t+2}(1-\varepsilon)^t}{(t+1)\,\mathrm{B}(t+1, t+1)},$$

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

   (G.87)

   $$\varpi + d$$

   (G.88)
   $$= \; 1 \;-\; c \cdot \mathbb{E}\big[\underbrace{\mathcal{H}_{\ln}(\boldsymbol{V}, H)}_{\le \ln(3)}\big] \;+\; d\big(1 - \mathbb{E}[\varSigma\boldsymbol{V}]\big)$$

   (G.89)
   $$\ge \; 1 \;-\; c\cdot\ln(3) \;+\; \frac{c\ln(3) - 1}{\mathbb{E}[H]} \cdot \mathbb{E}[H]$$

   (G.90)
   $$= \; 0.$$

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

$\square$

G.3 Asymptotic Approximations. 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. So 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 Proposition G.1. As functions in ε, they satisfy for $\varepsilon \to 0$

$$\text{(G.91)} \quad c_\varepsilon \;=\; \frac{1}{\tilde{H}} \pm O(\varepsilon)\,, \quad d_\varepsilon \;=\; O(\varepsilon^{-t-2})$$

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

$$\text{(G.92)} \qquad \varepsilon \;=\; \varepsilon_i \;=\; \mathcal{H}_i^{-\frac{1}{t+3}}$$

we have $\varepsilon_i \to 0$ as $i \to \infty$ and so $c_{\varepsilon_i} > 0$ for large enough i. Then we have by Proposition G.1

$$\text{(G.93)} \qquad \mathbb{E}[A_{\boldsymbol{q}^{(i)}}] \;\le\; c_{\varepsilon_i}\mathcal{H}_i \;+\; d_{\varepsilon_i}$$

$$\text{(G.94)} \qquad \underset{(\text{G.91})}{=} \frac{\mathcal{H}_i}{\tilde{H}} \;\pm\; O\Big(\mathcal{H}_i^{\frac{t+2}{t+3}}\Big).$$

Now consider the lower bound. For c_ε and d_ε from Proposition G.2 we similarly find as $\varepsilon \to 0$ that

(G.95)
$$c_\varepsilon \;=\; \frac{1}{\tilde{H}} \pm O(\varepsilon\log\varepsilon)\,, \quad d_\varepsilon \;=\; O(\varepsilon^{-t-2}).$$

With the same ε_i as above (Equation (G.92)) we have $d_{\varepsilon_i} \geq 0$ for large enough i, so by Proposition G.2 holds

$$\mathbb{E}[A_{\boldsymbol{q}^{(i)}}] \;\geq\; \frac{\mathcal{H}_i}{\tilde{H}} \;\pm\; O\Big(\mathcal{H}_i^{\frac{t+2}{t+3}} \log \mathcal{H}_i\Big). \tag{G.96}$$

Together with Equation (G.94), this proves Theorem 8.1. $\square$