The Number of Bit Comparisons Used by Quicksort: An Average-case Analysis

James Allen Fill [*][†] Svante Janson [‡]

Abstract

The analyses of many algorithms and data structures (such as digital search trees) for searching and sorting are based on the representation of the keys involved as bit strings and so count the number of bit comparisons. On the other hand, the standard analyses of many other algorithms (such as `Quicksort`) are performed in terms of the number of key comparisons. We introduce the prospect of a fair comparison between algorithms of the two types by providing an average-case analysis of the number of bit comparisons required by `Quicksort`. Counting bit comparisons rather than key comparisons introduces an extra logarithmic factor to the asymptotic average total. We also provide a new algorithm, "`BitsQuick`", that reduces this factor to constant order by eliminating needless bit comparisons.

1 Introduction and summary

Algorithms for sorting and searching (together with their accompanying analyses) generally fall into one of two categories: either the algorithm is regarded as comparing items pairwise irrespective of their internal structure (and so the analysis focuses on the number of comparisons), or else it is recognized that the items (typically numbers) are represented as bit strings and that the algorithm operates on the individual bits. Typical examples of the two types are `Quicksort` and digital search trees, respectively; see [7].

In this extended abstract we take a first step towards bridging the gap between the two points of view, in order to facilitate run-time comparisons across the gap, by answering the following question posed many years ago by Bob Sedgewick [personal communication]: What is the bit complexity of `Quicksort`?

More precisely, we consider `Quicksort` (see Section 2 for a review) applied to n distinct keys (numbers) from the interval $(0, 1)$. Many authors (Knuth [7], Régnier [10], Rösler [11], Knessl and Szpankowski [6], Fill and Janson [3] [4], Neininger and Ruschendorff [9], and others) have studied K_n, the (random) number of key comparisons performed by the algorithm. This is a natural measure of the cost (run-time) of the algorithm, if each comparison has the same cost. On the other hand, if comparisons are done by scanning the bit representations of the numbers, comparing their bits one by one, then the cost of comparing two keys is determined by the number of bits compared until a difference is found. We call this number the number of *bit comparisons* for the key comparison, and let B_n denote the total number of bit comparisons when n keys are sorted by `Quicksort`.

We assume that the keys $X_1, \dots, X_n$ to be sorted are independent random variables with a common continuous distribution F over $(0, 1)$. It is well known that the distribution of the number K_n of key comparisons does not depend on F. This invariance clearly fails to extend to the number B_n of bit comparisons, and so we need to specify F.

For simplicity, we study mainly the case that F is the uniform distribution, and, throughout, the reader should assume this as the default. But we also give a result valid for a general absolutely continuous distribution F over $(0, 1)$ (subject to a mild integrability condition on the density).

In this extended abstract we focus on the mean of B_n. One of our main results is the following Theorem 1.1, the concise version of which is the asymptotic equivalence

$$\mathbf{E}\, B_n \sim n(\ln n)(\lg n) \text{ as } n \to \infty.$$

Throughout, we use ln (respectively, lg) to denote natural (resp., binary) logarithm, and use log when the base doesn't matter (for example, in remainder estimates). The symbol $\doteq$ is used to denote approximate equality, and $\gamma \doteq 0.57722$ is Euler's constant.

Theorem 1.1. *If the keys $X_1, \dots, X_n$ are independent and uniformly distributed on $(0, 1)$, then the number B_n of bit comparisons required to sort these keys using*

[*]Department of Mathematical Sciences, The Johns Hopkins University, 34th and Charles Streets, Baltimore, MD 21218-2682 USA. `jimfill@jhu.edu`

[†]Research supported by NSF grant DMS-0104167 and by the Acheson J. Duncan Fund for the Advancement of Research in Statistics.

[‡]Department of Mathematics, Uppsala University, P. O. Box 480, SE-751 06 Uppsala, Sweden. `svante.janson@math.uu.se`

Quicksort *has expectation given by the following exact and asymptotic expressions:*

$$\mathbf{E}\, B_n = 2\sum_{k=2}^{n}(-1)^k\binom{n}{k}\frac{1}{(k-1)k[1-2^{-(k-1)}]} \tag{1.1}$$

$$= n(\ln n)(\lg n) - c_1 n\ln n + c_2 n + \pi_n n + O(\log n), \tag{1.2}$$

where, with $\beta := 2\pi/\ln 2$,

$$c_1 := \frac{1}{\ln 2}(4 - 2\gamma - \ln 2) \doteq 3.105,$$

$$c_2 := \frac{1}{\ln 2}\left[\frac{1}{6}(6-\ln 2)^2 - (4-\ln 2)\gamma + \frac{\pi^2}{6} + \gamma^2\right] \doteq 6.872,$$

and

$$\pi_n := \sum_{k\in\mathbf{Z}:\, k\neq 0}\frac{i}{\pi k(-1-i\beta k)}\Gamma(-1-i\beta k)n^{i\beta k}$$

is periodic in $\lg n$ *with period* 1 *and amplitude smaller than* 5×10^{-9}.

Small periodic fluctuations as in Theorem 1.1 come as a surprise to newcomers to the analysis of algorithms but in fact are quite common in the analysis of digital structures and algorithms; see, for example, Chapter 6 in [8].

For our further results, it is technically convenient to assume that the number of keys is no longer fixed at n, but rather Poisson distributed with mean λ and independent of the values of the keys. (In this extended abstract, we shall not deal with the "de-Poissonization" needed to transfer results back to the fixed-n model; for the results here we can simply compare the fixed-n case to Poisson cases with slightly smaller and larger means, say $n \pm n^{2/3}$.) In obvious notation, the Poissonized version of (1.2) is

$$\mathbf{E}\, B(\lambda) = \lambda(\ln\lambda)(\lg\lambda) - c_1\lambda\ln\lambda + c_2\lambda + \pi_\lambda\lambda + O(\log\lambda), \tag{1.3}$$

whose proof is indicated in Section 5 [following (5.2)]. We will also see (Proposition 5.1) that $\mathbf{Var}\, B(\lambda) = O(\lambda^2)$, so $B(\lambda)$ is concentrated about its mean. Since the number $K(\lambda)$ of key comparisons is likewise concentrated about its mean $\mathbf{E}\, K(\lambda) \sim 2\lambda\ln\lambda$ for large λ (see Lemmas 5.1 and 5.2), it follows that

$$\frac{2}{\lg\lambda}\times\frac{B(\lambda)}{K(\lambda)} \to 1 \text{ in probability as } \lambda\to\infty. \tag{1.4}$$

In other words, about $\frac{1}{2}\lg\lambda$ bits are compared per key comparison.

For non-uniform distribution F, we have the same leading term for the asymptotic expansion of $\mathbf{E}\, B(\lambda)$, but the second-order term is larger. (Throughout, $\ln_+$ denotes the positive part of the natural logarithm function. We denote the uniform distribution by unif.)

Theorem 1.2. *Let* $X_1, X_2, \ldots$ *be independent with a common distribution* F *over* $(0,1)$ *having density* f, *and let* N *be independent and Poisson with mean* λ. *If* $\int_0^1 f(\ln_+ f)^4 < \infty$, *then the expected number of bit comparisons, call it* $\mu_f(\lambda)$, *required to sort the keys* $X_1, \ldots, X_N$ *using* Quicksort *satisfies*

$$\mu_f(\lambda) = \mu_{\text{unif}}(\lambda) + 2H(f)\lambda\ln\lambda + o(\lambda\log\lambda)$$

as $\lambda\to\infty$, *where* $H(f) := \int_0^1 f\lg f \geq 0$ *is the entropy (in bits) of the density* f.

In applications, it may be unrealistic to assume that a specific density f is known. Nevertheless, even in such cases, Theorem 1.2 may be useful since it provides a measure of the robustness of the asymptotic estimate in Theorem 1.1.

Bob Sedgewick (on June 23, 2003, and among others who have heard us speak on the present material) has suggested that the number of bit comparisons for Quicksort might be reduced substantially by not comparing bits that have to be equal according to the results of earlier steps in the algorithm. In the final section, we note that this is indeed the case: for a fixed number n of keys, the average number of bit comparisons in the improved algorithm (which we dub "BitsQuick") is asymptotically $2(1+\frac{3}{2\ln 2})n\ln n$, only a constant ($\doteq 3.2$) times the average number of key comparisons [see (2.2)]. A related algorithm is the digital version of Quicksort by Roura [12]; it too requires $\Theta(n\log n)$ bit comparisons (we do not know the exact constant factor).

We may compare our results to those obtained for radix-based methods, for example radix exchange sorting, see [7, Section 5.2.2]. This method works by bit inspections, that is comparisons to constant bits, rather than pairwise comparisons. In the case of n uniformly distributed keys, radix exchange sorting uses asymptotically $n\lg n$ bit inspections. Since radix exchange sorting is designed so that the number of bit inspections is minimal, it is not surprising that our results show that Quicksort uses more bit comparisons. More precisely, Theorem 1.1 shows that Quicksort uses about $\ln n$ times as many bit comparisons as radix exchange sorting. For BitsQuick, this is reduced to a small constant factor. This gives us a measure of the cost in bit comparisons of using these algorithms; Quicksort is often used because of other advantages, and our results opens the possibility to see when they out-weight the increase in bit comparisons.

In Section 2 we review `Quicksort` itself and basic facts about the number K_n of key comparisons. In Section 3 we derive the exact formula (1.1) for $\mathbf{E}\,B_n$, and in Section 4 we derive the asymptotic expansion (1.2) from an alternative exact formula that is somewhat less elementary than (1.1) but much more transparent for asymptotics. In the transitional Section 5 we establish certain basic facts about the moments of $K(\lambda)$ and $B(\lambda)$ in the Poisson case with uniformly distributed keys, and in Section 6 we use martingale arguments to establish Theorem 1.2 for the expected number of bit comparisons for Poisson(λ) draws from a general density f. Finally, in Section 7 we study the improved `BitsQuick` algorithm discussed in the preceding paragraph.

Remark 1.1. The results can be generalized to bases other than 2. For example, base 256 would give corresponding results on the "byte complexity".

Remark 1.2. Cutting off and sorting small subfiles differently would affect the results in Theorems 1.1 and 1.2 by $O(n \log n)$ and $O(\lambda \log \lambda)$ only. In particular, the leading terms would remain the same.

2 Review: number of key comparisons used by Quicksort

In this section we briefly review certain basic known results concerning the number K_n of key comparisons required by `Quicksort` for a fixed number n of keys uniformly distributed on $(0, 1)$. (See, for example, [4] and the references therein for further details.)

`Quicksort`, invented by Hoare [5], is the standard sorting procedure in `Unix` systems, and has been cited [2] as one of the ten algorithms "with the greatest influence on the development and practice of science and engineering in the 20th century." The `Quicksort` algorithm for sorting an array of n distinct keys is very simple to describe. If $n = 0$ or $n = 1$, there is nothing to do. If $n \geq 2$, pick a key uniformly at random from the given array and call it the "pivot". Compare the other keys to the pivot to partition the remaining keys into two subarrays. Then recursively invoke `Quicksort` on each of the two subarrays.

With $K_0 := 0$ as initial condition, K_n satisfies the distributional recurrence relation

$$K_n \overset{\mathcal{L}}{=} K_{U_n-1} + K^*_{n-U_n} + n - 1, \qquad n \geq 1,$$

where $\overset{\mathcal{L}}{=}$ denotes equality in law (i.e., in distribution), and where, on the right, U_n is distributed uniformly over the set $\{1, \dots, n\}$, $K^*_j \overset{\mathcal{L}}{=} K_j$, and

$$U_n;\ K_0, \dots, K_{n-1};\ K^*_0, \dots, K^*_{n-1}$$

are all independent.

Passing to expectations we obtain the "divide-and-conquer" recurrence relation

$$\mathbf{E}\,K_n = \frac{2}{n} \sum_{j=0}^{n-1} \mathbf{E}\,K_j + n - 1,$$

which is easily solved to give

$$\begin{aligned} &(2.1) \quad \mathbf{E}\,K_n = 2(n+1)H_n - 4n \\ &(2.2) \qquad\qquad = 2n \ln n + (2\gamma - 4)n + 2 \ln n + (2\gamma + 1) \\ &\qquad\qquad\qquad + O(1/n). \end{aligned}$$

It is also routine to use a recurrence to compute explicitly the exact variance of K_n. In particular, the asymptotics are

$$\mathbf{Var}\,K_n = \sigma^2 n^2 - 2n \ln n + O(n)$$

where $\sigma^2 := 7 - \frac{2}{3}\pi^2 \doteq 0.4203$. Higher moments can be handled similarly. Further, the normalized sequence

$$\widehat{K}_n := (K_n - \mu_n)/n, \qquad n \geq 1,$$

converges in distribution to $\widehat{K}$, where the law of $\widehat{K}$ is characterized as the unique distribution over the real line with vanishing mean that satisfies a certain distributional identity; and the moment generating functions of $\widehat{K}_n$ converge to that of $\widehat{K}$.

3 Exact mean number of bit comparisons

In this section we establish the exact formula (1.1), repeated here as (3.1) for convenience, for the expected number of bit comparisons required by `Quicksort` for a fixed number n of keys uniformly distributed on $(0, 1)$:

$$(3.1) \quad \mathbf{E}\,B_n = 2 \sum_{k=2}^{n} (-1)^k \binom{n}{k} \frac{1}{(k-1)k[1 - 2^{-(k-1)}]}.$$

Let $X_1, \dots, X_n$ denote the keys, and $X_{(1)} < \cdots < X_{(n)}$ their order statistics. Consider ranks $1 \leq i < j \leq n$. Formula (3.1) follows readily from the following three facts, all either obvious or very well known:

- The event $C_{ij} := \{\text{keys } X_{(i)} \text{ and } X_{(j)} \text{ are compared}\}$ and the random vector $(X_{(i)}, X_{(j)})$ are independent.
- $\mathbf{P}(C_{ij}) = 2/(j - i + 1)$. [Indeed, C_{ij} equals the event that the first pivot chosen from among $X_{(i)}, \dots, X_{(j)}$ is either $X_{(i)}$ or $X_{(j)}$.]

- The joint density $g_{n,i,j}$ of $(X_{(i)}, X_{(j)})$ is given by

$$g_{n,i,j}(x,y) = \binom{n}{i-1,1,j-i-1,1,n-j} \times x^{i-1}(y-x)^{j-i-1}(1-y)^{n-j}.$$

Let $b(x,y)$ denote the index of the first bit at which the numbers $x, y \in (0,1)$ differ, where for definiteness we take the non-terminating expansion for terminating rationals. Then

(3.2)
$$\begin{aligned}\mathbf{E}\, B_n &= \sum_{1 \le i < j \le n} \mathbf{P}(C_{ij}) \int_0^1 \int_x^1 b(x,y)\, g_{n,i,j}(x,y)\, dy\, dx \\ &= \int_0^1 \int_x^1 b(x,y)\, p_n(x,y)\, dy\, dx,\end{aligned}$$

where $p_n(x,y)$ has the definition and interpretation

$$\begin{aligned}p_n(x,y) &:= \sum_{1 \le i < j \le n} \mathbf{P}(C_{ij}) g_{n,i,j}(x,y)\, dy\, dx \\ &= \frac{\mathbf{P}(\text{keys in } (x, x+dx) \text{ and } (y, y+dy) \text{ are compared})}{dx\, dy}.\end{aligned}$$

By a routine calculation,

(3.3)
$$\begin{aligned}p_n(x,y) &= \frac{2}{(y-x)^2}\left[(1-(y-x))^n - 1 + n(y-x)\right] \\ &= 2\sum_{k=2}^{n} (-1)^k \binom{n}{k} (y-x)^{k-2},\end{aligned}$$

which depends on x and y only through the difference $y - x$. Plugging (3.3) into (3.2), we find

$$\mathbf{E}\, B_n = 2\sum_{k=2}^{n} (-1)^k \binom{n}{k} \int_0^1 \int_x^1 b(x,y)(y-x)^{k-2}\, dy\, dx.$$

But, by routine (if somewhat lengthy) calculation,

$$\begin{aligned}&\int_0^1 \int_x^1 b(x,y)(y-x)^{k-2}\, dy\, dx \\ &= \sum_{\ell=0}^{\infty} (\ell+1) \iint_{0<x<y<1:\, b(x,y)=\ell+1} (y-x)^{k-2}\, dx\, dy \\ &= \sum_{\ell=0}^{\infty} (\ell+1) 2^{\ell} \int_0^{2^{-(\ell+1)}} \int_{2^{-(\ell+1)}}^{2^{-\ell}} (y-x)^{k-2}\, dy\, dx \\ &= \frac{1}{(k-1)k[1-2^{-(k-1)}]}.\end{aligned}$$

This now leads immediately to the desired (3.1).

4 Asymptotic mean number of bit comparisons

Formula (1.1), repeated at (3.1), is hardly suitable for numerical calculations or asymptotic treatment, due to excessive cancellations in the alternating sum. Indeed, if (say) $n = 100$, then the terms (including the factor 2, for definiteness) alternate in sign, with magnitude as large as 10^{25}, and yet $\mathbf{E}\, B_n \doteq 2295$. Fortunately, there is a complex-analytic technique designed for precisely our situation (alternating binomial sums), namely, *Rice's method.* Because of space limitations, we will not review the idea behind the method here, but rather refer the reader to (for example) Section 6.4 of [8]. Let

$$h(z) := \frac{2}{(z-1)z[1-2^{-(z-1)}]}$$

and let $B(z,w) := \Gamma(z)\Gamma(w)/\Gamma(z+w)$ denote the (meromorphic continuation) of the classical beta function. According to Rice's method, $\mathbf{E}\, B_n$ equals the sum of the residues of the function $B(n+1, -z)h(z)$ at

- the triple pole at $z = 1$;
- the simple poles at $z = 1 + i\beta k$, for $k \in \mathbf{Z} \setminus \{0\}$;
- the double pole at $z = 0$.

The residues are easily calculated, especially with the aid of such symbolic-manipulation software as `Mathematica` or `Maple`. Corresponding to the above list, the residues equal

- $\frac{n}{\ln 2}\left[H_{n-1}^2 - (4 - \ln 2)H_{n-1} + \frac{1}{6}(6 - \ln 2)^2 + H_{n-1}^{(2)}\right]$;
- $\frac{i}{\pi k(-1-i\beta k)}\Gamma(-1-i\beta k)\frac{n!}{\Gamma(n-i\beta k)}$;
- $-2(H_n + 2\ln 2 + 1)$,

where $H_n^{(r)} := \sum_{j=1}^{n} j^{-r}$ denotes the nth harmonic number of order r and $H_n := H_n^{(1)}$. Summing the residue contributions gives an alternative exact formula for $\mathbf{E}\, B_n$, from which the asymptotic expansion (1.2) (as well as higher-order terms) can be read off easily using standard asymptotics for $H_n^{(r)}$ and Stirling's formula; we omit the details.

This completes the proof of Theorem 1.1.

REMARK 4.1. We can calculate $\mathbf{E}\, K_n$ in the same fashion (and somewhat more easily), by replacing the bit-index function b by the constant function 1. Following this approach, we obtain first the following analogue of (3.1):

$$\mathbf{E}\, K_n = 2\sum_{k=2}^{n} (-1)^k \binom{n}{k} \frac{1}{(k-1)k}.$$

Then the residue contributions using Rice's method are

- $2n(H_n - 2 - \frac{1}{n})$, at the double pole at $z = 1$;
- $2(H_n + 1)$, at the double pole at $z = 0$.

Summing the two contributions gives an alternative derivation of (2.1).

5 Poissonized model for uniform draws

As a warm-up for Section 6, we now suppose that the number of keys (throughout this section still assumed to be uniformly distributed) is Poisson with mean λ. We begin with a lemma which provides both the analogue of (2.1)–(2.2) and two other facts we will need in Section 6.

LEMMA 5.1. *In the setting of Theorem 1.2 with F uniform, the expected number of key comparisons is a strictly convex function of λ given by*

$$\mathbf{E}\,K(\lambda) = 2\int_0^\lambda (\lambda - y)(e^{-y} - 1 + y)y^{-2}\,dy.$$

Asymptotically, as $\lambda \to \infty$ we have

$$\mathbf{E}\,K(\lambda) = 2\lambda\ln\lambda + (2\gamma - 4)\lambda + 2\ln\lambda + 2\gamma + 2 + O(e^{-\lambda})$$

and as $\lambda \to 0$ we have

$$\mathbf{E}\,K(\lambda) = \tfrac{1}{2}\lambda^2 + O(\lambda^3).$$

Proof. To obtain the exact formula, begin with

$$\mathbf{E}\,K_n = \int_0^1 \int_x^1 p_n(x, y)\,dy\,dx;$$

cf. (3.2) and recall Remark 4.1. Then multiply both sides by $e^{-\lambda}\lambda^n/n!$ and sum, using the middle expression in (3.3); we omit the simple computation. Strict convexity then follows from the calculation $\frac{d^2}{d\lambda^2}\mathbf{E}\,K(\lambda) = 2(e^{-\lambda} - 1 + \lambda)/\lambda^2 > 0$, and asymptotics as $\lambda \to 0$ are trivial: $\mathbf{E}\,K(\lambda) = 2\int_0^\lambda(\lambda - y)[\frac{1}{2} + O(y)]\,dy = \frac{1}{2}\lambda^2 + O(\lambda^3)$. We omit the proof of the result for $\lambda \to \infty$, but plan to include it in our full-length paper; comparing the fixed-n and Poisson(λ) expansions, note the difference in constant terms and the much smaller error term in the Poisson case.

To handle the number of bit comparisons, we will also need the following bounds on the moments of $K(\lambda)$. Together with Lemma 5.1, these bounds also establish concentration of $K(\lambda)$ about its mean when λ is large. For real $1 \le p < \infty$, we let $\|W\|_p := (\mathbf{E}\,|W|^p)^{1/p}$ denote L^p-norm and use $\mathbf{E}(W; A)$ as shorthand for the expectation of the product of W and the indicator of the event A.

LEMMA 5.2. *For every integer $p \ge 1$, there exists a constant $c_p < \infty$ such that*

$$\|K(\lambda) - \mathbf{E}\,K(\lambda)\|_p \le c_p\lambda \qquad \text{for } \lambda \ge 1,$$
$$\|K(\lambda)\|_p \le c_p\lambda^{2/p} \qquad \text{for } \lambda \le 1.$$

Proof (sketch). The first result is certainly true for $\lambda \ge 1$ bounded away from ∞. For $\lambda \to \infty$ we need only Poissonize standard `Quicksort` moment calculations; cf. the very end of Section 2 for the fixed-n case. For $\lambda \le 1$ we use

$$\mathbf{E}\,K^p(\lambda) \le \mathbf{E}\left[\binom{N}{2}^p; N \ge 2\right] \le 2^{-p}\mathbf{E}\,[N^{2p}; N \ge 2]$$
$$= 2^{-p}\lambda^2\sum_{n=2}^\infty e^{-\lambda}\frac{\lambda^{n-2}}{n!}n^{2p} \le c_p^p\lambda^2$$

where N is Poisson(λ) and c_p is taken to be at least the finite value $\frac{1}{2}\left[\sum_{n=2}^\infty(n^{2p}/n!)\right]^{1/p}$.

We now turn our attention from $K(\lambda)$ to the more interesting random variable $B(\lambda)$, the total number of bit comparisons. First, let

$$I_{k,j} := [(j-1)2^{-k}, j2^{-k})$$

be the jth dyadic rational interval of rank k, and

$$B_k(\lambda) := \text{number of comparisons of } (k+1)\text{st bits},$$
$$B_{k,j}(\lambda) := \text{number of comparisons of } (k+1)\text{st bits between keys in } I_{k,j}.$$

Observe that

$$B(\lambda) = \sum_{k=0}^\infty B_k(\lambda) = \sum_{k=0}^\infty\sum_{j=1}^{2^k} B_{k,j}(\lambda). \tag{5.1}$$

A simplification provided by our Poissonization is that, for each fixed k, the variables $B_{k,j}(\lambda)$ are independent. Further, the marginal distribution of $B_{k,j}(\lambda)$ is simply that of $K(2^{-k}\lambda)$. Taking expectations in (5.1), we find

$$\mu_{\text{unif}}(\lambda) = \mathbf{E}\,B(\lambda) = \sum_{k=0}^\infty 2^k\mathbf{E}\,K(2^{-k}\lambda). \tag{5.2}$$

In the full-length paper we (tentatively) plan to include the details of a proof we have written showing how the two asymptotic estimates in Lemma 5.1 can be used in conjunction with (5.2) to establish the asymptotic estimate (1.3) for $\mu_{\text{unif}}(\lambda)$ as $\lambda \to \infty$. [An alternative approach is to Poissonize (1.2).] Moreover, we are now in position to establish the concentration of $B(\lambda)$ about $\mu_{\text{unif}}(\lambda)$ promised just prior to (1.4).

Proposition 5.1. *There exists a constant c such that $\mathbf{Var}\,B(\lambda) \le c^2\lambda^2$ for $0 < \lambda < \infty$.*

Proof. For $0 < \lambda < \infty$, we have by the triangle inequality for $\|\cdot\|_2$, independence and $B_{k,j}(\lambda) \overset{\mathcal{L}}{=} K(2^{-k}\lambda)$, and Lemma 5.2, with $c := c_2 \sum_{k=0}^{\infty} 2^{-k/2}$,

$$\begin{aligned}
[\mathbf{Var} B(\lambda)]^{1/2} &\le \sum_{k=0}^{\infty} [\mathbf{Var}\,B_k(\lambda)]^{1/2} \\
&\le \sum_{k=0}^{\infty} [2^k \mathbf{Var}\,K(2^{-k}\lambda)]^{1/2} \\
&\le c\lambda.
\end{aligned}$$

Remark 5.1. (a) In the full-length paper we will show that Proposition 5.1 can be extended to

$$\|B(\lambda) - \mathbf{E}\,B(\lambda)\|_p \le c'_p \lambda$$

for any real $1 \le p < \infty$ (and some finite c'_p) and all $\lambda \ge 1$.

(b) It is quite plausible that the variables $B_k(\lambda)$ are positively correlated [because, for $k \ge 1$, the $(k+1)$st bits of two keys can't be compared unless the kth bits are], in which case it is easy to check that $\mathbf{Var}\,B(\lambda) = \Omega(\lambda^2)$ for $\lambda \ge 1$, but we do not know a proof. We would then have $\|B(\lambda) - \mathbf{E}\,B(\lambda)\|_p = \Theta(\lambda)$ for each real $2 \le p < \infty$. Perhaps it is even true that $[B(\lambda) - \mathbf{E}\,B(\lambda)]/\lambda$ has a limiting distribution, but we have no conjecture as to its form.

6 Mean number of bit comparisons for keys drawn from an arbitrary density f

In this section we outline martingale arguments for proving Theorem 1.2 for the expected number of bit comparisons for Poisson(λ) draws from a rather general density f. (For background on martingales, see any standard measure-theoretic probability text, e.g., [1].) In addition to the notation above, we will use the following:

$$\begin{aligned}
p_{k,j} &:= \int_{I_{k,j}} f, \\
f_{k,j} &:= (\text{average value of } f \text{ over } I_{k,j}) = 2^k p_{k,j}, \\
f_k(x) &:= f_{k,j} \quad \text{for all } x \in I_{k,j}, \\
f^*(\cdot) &:= \sup_k f_k(\cdot).
\end{aligned}$$

Note for each $k \ge 0$ that $\sum_j p_{k,j} = 1$ and that $f_k : (0,1) \to [0,\infty)$ is the smoothing of f to the rank-k dyadic rational intervals. From basic martingale theory we have immediately the following simple but key observation.

Lemma 6.1. *With $f_\infty := f$,*

$$(f_k)_{0 \le k \le \infty} \text{ is a martingale,}$$

and $f_k \to f$ almost surely (and in L^1).

Before we begin the proof of Theorem 1.2 we remark that the asymptotic inequality $\mu_f(\lambda) \ge \mu_{\text{unif}}(\lambda)$ observed there in fact holds for every $0 < \lambda < \infty$. Indeed,

$$\begin{aligned}
\mu_f(\lambda) &= \sum_{k=0}^{\infty} \sum_{j=1}^{2^k} \mathbf{E}\,K(\lambda p_{k,j}) \\
&\ge \sum_{k=0}^{\infty} 2^k \mathbf{E}\,K(\lambda 2^{-k}) = \mu_{\text{unif}}(\lambda),
\end{aligned} \tag{6.1}$$

where the first equality appropriately generalizes (5.2), the inequality follows by the convexity of $\mathbf{E}\,K(\lambda)$ (recall Lemma 5.1), and the second equality follows by (5.2).

Proof (sketch) of Theorem 1.2. Assume $\lambda \ge 1$ and, with $m \equiv m(\lambda) := \lceil \lg \lambda \rceil$, split the double sum in (6.1) as

$$\mu_f(\lambda) = \sum_{k=0}^{m} \sum_{j=1}^{2^k} \mathbf{E}\,K(\lambda p_{k,j}) + R(\lambda), \tag{6.2}$$

with $R(\lambda)$ a remainder term. Assuming

$$\int f^*(\ln_+ f^*)^3 < \infty \tag{6.3}$$

(to be proved in the full-length paper from the assumption on f in the statement of the theorem, using the maximal inequality for nonnegative submartingales) and using the estimates of Lemma 5.1, one can show $R(\lambda) = O(\lambda)$. Plugging this and the consequence

$$\mathbf{E}\,K(x) = 2x \ln x + (2\gamma - 4)x + O(x^{1/2}),$$

which holds uniformly in $0 \le x < \infty$, of Lemma 5.1 into (6.2), we find

$$\begin{aligned}
\mu_f(\lambda) &= \sum_{k=0}^{m} \sum_{j=1}^{2^k} \Big[2\lambda p_{k,j}(\ln \lambda + \ln p_{k,j}) + (2\gamma - 4)\lambda p_{k,j} \\
&\qquad + O\left((\lambda p_{k,j})^{1/2} \right) \Big] + O(\lambda) \\
&= \sum_{k=0}^{m} \Bigg[2\lambda \ln \lambda + 2\lambda \sum_{j=1}^{2^k} p_{k,j} \ln p_{k,j} + (2\gamma - 4)\lambda \\
&\qquad + O\left(\lambda^{1/2} 2^{k/2} \right) \Bigg] + O(\lambda) \\
&= \mu_{\text{unif}}(\lambda) + 2\lambda \sum_{k=0}^{m} \int f_k \ln f_k + O(\lambda),
\end{aligned}$$

where we have used the Cauchy–Schwarz inequality at the second equality and comparison with the uniform case ($f \equiv 1$) at the third.

But, by Lemma 6.1, (6.3), and the dominated convergence theorem,

$$\int f_k \ln f_k \longrightarrow \int f \ln f \text{ as } k \to \infty, \tag{6.4}$$

from which follows

$$\begin{aligned}\mu_f(\lambda) &= \mu_{\text{unif}}(\lambda) + 2\lambda(\lg \lambda) \int f \ln f + o(\lambda \log \lambda) \\ &= \mu_{\text{unif}}(\lambda) + 2\lambda(\ln \lambda) \int f \lg f + o(\lambda \log \lambda),\end{aligned}$$

as desired.

Remark 6.1. If we make the stronger assumption that

f is Hölder(α) continuous on $[0, 1]$ for some $\alpha > 0$,

then we can quantify (6.4) and improve the $o(\lambda \log \lambda)$ remainder in the statement of Theorem 1.2 to $O(\lambda)$.

7 An improvement: BitsQuick

Recall the operation of `Quicksort` described in Section 2. Suppose that the pivot [call it $x = 0.x(1)\,x(2)\,\ldots$] has first bit $x(1)$ equal to 0, say. Then the subarray of keys smaller than x all have first bit equal to 0 as well, and it wastes time to compare first bits when `Quicksort` is called recursively on this subarray.

We call `BitsQuick` the obvious recursive algorithm that does away with this waste. We give one possible implementation in the boxed pseudocode, where $L(y)$ denotes the result of rotating the register containing key y to the left —i.e., replacing $y = .y(1)\,y(2)\,\ldots\,y(m)$ by $.y(2)\,\ldots\,y(m)\,y(1)$ [and similarly $R(A)$ for rotation of every element of array A to the right]. The input bit b indicates whether or not the array elements need to be rotated to the right before the routine terminates. The symbol $\|$ denotes concatenation (of sorted arrays). (We omit minor implementational details, such as how to do sorting in place and to maintain random ordering for the generated subarrays, that are the same as for `Quicksort` and very well known.) The routine `BitsQuick`(A, b) receives a (randomly ordered) array A and a single bit b, and returns the sorted version of A. The initial call is to `BitsQuick`$(A_0, 0)$, where A_0 is the full array to be sorted.

A related but somewhat more complicated algorithm has been considered by Roura [12, Section 5].

In the full-length paper we plan to present arguments justifying the number of bit comparisons used by `BitsQuick` as a fair measure of its overall cost. In the full-length paper we will also prove the following analogue of Theorem 1.1 (and establish further terms in the asymptotic expansion), wherein

```
The routine BitsQuick($A, b$)

If $|A| \le 1$
  Return $A$
Else
  Set $A_- \leftarrow \emptyset$ and $A_+ \leftarrow \emptyset$
  Choose a random pivot key $x = 0.x(1)\,x(2)\,\ldots$
  If $x(1) = 0$
    For $y \in A$ with $y \ne x$
      If $y < x$
        Set $y \leftarrow L(y)$ and then $A_- \leftarrow A_- \cup \{y\}$
      Else
        Set $A_+ \leftarrow A_+ \cup \{y\}$
    Set $A_- \leftarrow$ BitsQuick($A_-, 1$) and
        $A_+ \leftarrow$ BitsQuick($A_+, 0$)
    Set $A \leftarrow A_- \,\|\, \{x\} \,\|\, A_+$
  Else
    For $y \in A$ with $y \ne x$
      If $y < x$
        Set $A_- \leftarrow A_- \cup \{y\}$
      Else
        Set $y \leftarrow L(y)$ and then $A_+ \leftarrow A_+ \cup \{y\}$
    Set $A_- \leftarrow$ BitsQuick($A_-, 0$) and
        $A_+ \leftarrow$ BitsQuick($A_+, 1$)
    Set $A \leftarrow A_- \,\|\, \{x\} \,\|\, A_+$
  If $b = 0$
    Return $A$
  Else
    Return $R(A)$
```

$$\tilde{c}_1 := \frac{7}{\ln 2} + \frac{15}{2} - \Big(\frac{3}{\ln 2} + 2\Big)\gamma \doteq 13.9$$

and

$$\tilde{\pi}_n := \frac{1}{\ln 2} \sum_{k \in \mathbf{Z}\colon k \ne 0} \frac{3 - i\beta k}{1 + i\beta k} \Gamma(-1 - i\beta k)\, n^{i\beta k}$$

is periodic in $\lg n$ with period 1 and amplitude smaller than 2×10^{-7}:

$$\begin{aligned}\mathbf{E}\, Q_n &= \sum_{k=2}^{n} (-1)^k \binom{n}{k} k^{-1} \left[\frac{2(k-2)}{1 - 2^{-k}} - \frac{k-4}{1 - 2^{-(k-1)}}\right] \\ &\qquad + 2nH_n - 5n + 2H_n + 1 \\ &= \Big(2 + \frac{3}{\ln 2}\Big) n \ln n - \tilde{c}_1 n + \tilde{\pi}_n n + O(\log^2 n).\end{aligned}$$

We have not yet had the opportunity to consider the variability of Q_n.

References

[1] Kai Lai Chung. *A course in probability theory.* Second edition, Probability and Mathematical Statistics, Vol. 21, Academic Press, New York–London, 1974.

[2] J. Dongarra and F. Sullivan. Guest editors' introduction: the top 10 algorithms. *Computing in Science & Engineering*, 2(1):22–23, 2000.

[3] James Allen Fill and Svante Janson. Smoothness and decay properties of the limiting Quicksort density function. *Mathematics and Computer Science (Versailles, 2000)*, pp. 53–64. Trends Math., Birkhäuser, Basel, 2000.

[4] James Allen Fill and Svante Janson. Quicksort asymptotics. *J. Algorithms*, 44(1):4–28, 2002.

[5] C. A. R. Hoare. Quicksort. *Comput. J.*, 5:10–15, 1962.

[6] Charles Knessl and Wojciech Szpankowski. Quicksort algorithm again revisited. *Discrete Math. Theor. Comput. Sci.*, 3(2):43–63 (electronic), 1999.

[7] Donald E. Knuth. *The Art of Computer Programming. Vol. 3: Sorting and Searching.* 2nd ed., Addison-Wesley, Reading, Mass., 1998.

[8] Hosam M. Mahmoud. *Evolution of random search trees.* John Wiley & Sons Inc., New York, 1992.

[9] Ralph Neininger and Ludger Rüschendorf. Rates of convergence for Quicksort. *J. Algorithms* 44(1):51–62, 2002.

[10] Mireille Régnier. A limiting distribution for quicksort. *RAIRO Inform. Théor. Appl.*, 23(3):335–343, 1989.

[11] Uwe Rösler. A limit theorem for "Quicksort". *RAIRO Inform. Théor. Appl.*, 25(1):85–100, 1991.

[12] Salvador Roura. Digital access to comparison-based tree data structures and algorithms. *J. Algorithms*, 40(1):1–23, 2001.