*Adv. Appl. Prob.* **43,** 524–544 (2011)
*Printed in Northern Ireland*


# ANALYSIS OF SWAPS IN RADIX SELECTION

AMR ELMASRY,* *University of Copenhagen*
HOSAM MAHMOUD,** *The George Washington University*

**Abstract**

Radix Sort is a sorting algorithm based on analyzing digital data. We study the number of swaps made by Radix Select (a one-sided version of Radix Sort) to find an element with a randomly selected rank. This kind of grand average provides a smoothing over all individual distributions for specific fixed-order statistics. We give an exact analysis for the grand mean and an asymptotic analysis for the grand variance, obtained by poissonization, the Mellin transform, and depoissonization. The digital data model considered is the Bernoulli($p$). The distributions involved in the swaps experience a phase change between the biased cases ($p \neq \frac{1}{2}$) and the unbiased case ($p = \frac{1}{2}$). In the biased cases, the grand distribution for the number of swaps (when suitably scaled) converges to that of a perpetuity built from a two-point distribution. The tool for this proof is contraction in the Wasserstein metric space, and identifying the limit as the fixed-point solution of a distributional equation. In the unbiased case the same scaling for the number of swaps gives a limiting constant in probability.

*Keywords:* Random structure; algorithm; order statistic; recurrence; perpetuity; phase change; digital data; digital sorting; selection; Mellin transform; poissonization; depoissonization

2010 Mathematics Subject Classification: Primary 60C05
Secondary 60F05; 68P10

## 1. Radix methods

Radix Sort is a sorting technique based on analyzing the digital composition of keys. Digits (possibly bits at the lowest machine level) are extracted from keys and used to classify the keys. Radix Sort dates back to the nineteenth century and can be found in the work of Hollerith (1894) on tabulating machines. It provides a good alternative for comparison-based sorting algorithms like Quick Sort and Merge Sort, where keys are compared according to an ordering relation; see Knuth (1998, pp. 116–119), Mahmoud (2000, pp. 148–151), and Mehlhorn (1984, pp. 42–68) for a broad discussion of these sorting algorithms.

Two different variants of Radix Sort are known, the least-significant-digit (LSD) Radix Sort and the most-significant-digit (MSD) Radix Sort. The LSD Radix Sort starts with the least-significant digit and orders the keys accordingly. It repeatedly orders the keys moving towards the most-significant digit and using one digit at a time; see, for example, Cormen *et al.* (2001, p. 178) for details. To guarantee the correctness of the LSD variant, every ordering round

Received 11 January 2010; revision received 12 February 2011.
* Postal address: Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark. Email address: elmasry@mpi-inf.mpg.de
Part of this work was carried out while the author was at Max-Planck-Institut für Informatik. The author is currently on leave from Alexandria University of Egypt.
** Postal address: Department of Statistics, The George Washington University, Washington, DC 20052, USA.
Email address: hosam@gwu.edu

should be stable (i.e. does not alter the order of the keys that have the same value of the digit under consideration). Alternatively, the MSD Radix Sort starts with the most-significant digit and partitions the keys accordingly. It then recursively sorts each partition moving towards the least-significant digit and using one digit at a time. Throughout the rest of this paper, we use Radix Sort for MSD Radix Sort.

It is useful to connect digital sorting algorithms to digital trees. Certain classes of digital trees are called 'tries'. They have numerous applications in storing computer files (see Knuth (1998, pp. 116–119)), detecting similarity of DNA strands (see Mahmoud and Ward (2008)), etc. Tries also provide models for the design and analysis of several important algorithms, such as extendible hashing (see Fagin *et al.* (1979)). In the context of the present paper, the trie is a backbone for Radix Sort and Radix Select. The trie was proposed independently in De La Briandais (1959) and Fredkin (1960) as a data structure for information re*trie*val.

A trie is a (digital) tree structure used to store a set of strings. All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string. Every root-to-leaf path represents one string. The keys in a branch of a node in a trie correspond to the partitioning of the keys carried out by Radix Sort. A search for a string in a trie can be done in time proportional to the length of the string.

The ideas of Radix Sort can be adopted to perform selection (identification of a key with a certain rank). The difference is that for selection we need to only recursively invoke the algorithm on one part of the partition, namely the one that has the sought key. The resulting algorithm is called Radix Select; this is in analogy to Quick Sort and Quick Select. In the terminology of tries, Radix Select constructs and traverses a root-to-leaf path.

Recently, there has been renewed interest in the overall number of bit comparisons of sorting algorithms, not only for radix-based sorting algorithms but also for comparison-based algorithms, such as Quick Sort. The concern is that comparing the rank of two keys depends critically on the nature of the data. For instance, if the data are short integers, made up of a few bytes, the number of bit comparisons is small, when comparing two keys. But, if we order data like DNA strands, typically of length in the order of hundreds of thousands of nucleotides, an excessive number of bit comparisons is needed to determine the order of two similar strands. So, it is not straightforward to compare Quick Sort to a radix-based algorithm such as Radix Sort without considering the number of bit comparisons in both. We refer the reader to the recent work of Fill and Janson (2004), Fill and Nakama (2008), and Vallée *et al.* (2009).

This renewed interest in the analysis of algorithms from a bits point of view provided the motivation for us to consider radix selection, and to perform an analysis of the number of swaps, to further put in perspective a meaningful distinction between radix-based methods and comparison-based methods.

## 2. Implementation issues

At the heart of both Radix Sort and Radix Select is a partitioning procedure in which the keys are separated into groups according to a specified digit. In this section we overview different alternative popular implementations of the partitioning. We assume $m$-ary keys (each digit may have one of $m$ different values, also called letters or symbols), and accordingly use $m$-ary partitioning.

*A linked-list implementation.* We start by initializing $m$ empty buckets, each will hold a portion of the list. We traverse the linked list while detaching pieces of it. More precisely, the following two steps are repeatedly performed until the list is empty. We traverse the list as long as the value of the specified digit is the same, and detach the traversed sublist. We then append the

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

detached sublist to the end of the list in the bucket that matches its value for the specified digit. Note that this implementation is stable.

*A two-array implementation.* We assume that the data is in one array, and partitioned in the other array. (The role of the two arrays is interchanged with every partitioning round.) We start by initializing a set of $m$ counters to 0. We then scan the data and count the number of keys for each of the $m$ values of the specified digit. The starting position in the second array for each part of the partition is initialized accordingly. We scan the data array one more time, and during this scan we move each key to the next available position of the part of the partition matching its specified digit. This implementation is also stable.

*A one-array (in-situ) implementation.* As in the two-array case, we scan the data and count the number of keys that should go to each part of the partition. We then initialize $m$ pointers to index the positions of the array where each of these parts should start. Next, we advance each pointer within its designated part as long as the current key has a matching value for the specified digit. If the pointer reaches the end of the portion allocated for this part of the partition, we exclude this part from all subsequent iterations.

We then arbitrarily choose one key for which the value $b$ of its specified digit does not match the part of the partition $a$ in which it resides. We subsequently move this key to the current pointer position for the nonmatching key of the $b$th part of the partition, eject this nonmatching key, and advance the pointer of the $b$th part as above. We then move the ejected key to its matching part, and so on. The aforementioned steps are repeated until a key is moved to the $a$th part of the partition; this can be thought of as permuting nonmatching keys to their matching parts of the partition. As long as there exist nonmatching keys, we arbitrarily select one to start a new permutation. By the end of the algorithm, every key lands in its correct destination.

It is clear that algorithmically handling the case $m = 2$ is easier and more efficient for all implementations. We thus opt for analyzing this case. We discuss extensions of the results to other MSD implementations at the end. The in-situ array implementation can be done even more efficiently for the binary case, where one scan through the array suffices. In Sections 3–9 Radix Select refers to the binary in-situ MSD radix selection.

## 3. The binary case

We assume the data to be binary strings on $\{0, 1\}$ residing in an array $A[1\,..\,n]$. For binary data, Radix Sort is a two-sided algorithm: it examines the first bit of each key and classifies the data into two groups according to their first bit. Keys starting with 0 go in one group, the rest are placed in the other group. The two groups are then handled recursively, until arrays of size 1 or 0 are reached, and that is when the recursion is stopped, but at the $b$th call to the algorithm, the $b$th bit in the keys are used for grouping. The corresponding tree in such cases is a binary trie.

The operation of Radix Sort assumes the presence of Partition, a digital partition procedure that classifies the keys according to some bit position into two groups. When called to divide the stretch $A[\ell\,..\,u]$ according to the $b$th bit, Partition comes up with a *splitting position* $k$ and places all the keys having 0 at the $b$th position in $A[\ell\,..\,k-1]$ and places all the keys having 1 at the $b$th position in $A[k\,..\,u]$.

Radix Sort can be easily adapted to deliver a certain order statistic $r$ (the $r$th smallest data item). Once the splitting position $k$ is determined, we know whether $r < k$ or $r \geq k$. If $r < k$, the algorithm recursively seeks the $r$th-order statistic in $A[1\,..\,k-1]$, and if $r \geq k$, the algorithm

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

seeks the $(r-k+1)$th order statistic among the elements of $A[k\,..\,n]$. This $(r-k+1)$th element is, of course, ranked $r$th among the entire data set. Thus, only the segment containing the desired order statistic is searched and the other is truncated. The recursion is continued until equal values of $\ell$ and $u$ are passed to it, and that is when it knows that the search has been narrowed down to one key, which must be the order statistic it is seeking. Using the trie lingo, the sum of the sizes of the parts of the partitions handled by Radix Select equals the sum of the sizes of the subtrees of the nodes on the path constructed by Radix Select in the corresponding trie.

## 4. Partition

We assume a digital partition algorithm (Partition) that is an adaptation of a well-known comparison-based algorithm to appeal to the nature of digital data rather than their order statistics. The comparison-based analog first appeared in Hoare (1962) and was widely disseminated in the work of Sedgwick (see, for example, Sedgwick (1977), (1978), (1980, p. 25), (1998, pp. 305–308)).

Suppose that Partition is invoked at the $b$th call to Radix Select to partition $A[\ell\,..\,u]$. The partition is carried out according to the $b$th bit. Partition sets up two pointers, one at the lower end $i$ which moves forward, and one at the upper end $j$ which moves backward. Initially, $i$ is set at the value $\ell$, and $j$ is set at the value $u$. We keep advancing $i$ so long as it falls in range and we see keys with 0 at the $b$th bit. (The case of falling out of range is handled with an artificial sentinel starting with 1 placed at the end of the array.) When this iterative operation is stopped, $i$ is pointing at a key with 1 as its $b$th bit. At this point we retract $j$ so long as it falls in range and we see keys that have 1 at their $b$th position. When this iterative operation is stopped, $j$ is pointing at a key with 0 as its $b$th bit. (The case of falling out of range is handled with an artificial sentinel starting with 0 placed at the beginning of the array.) If the two pointers $i$ and $j$ are not crossed, the two keys under $i$ and $j$ are both at the 'wrong side' of the splitting position. We swap them and continue the process on $A[i+1\,..\,j-1]$.

While carrying out its mechanics, Radix Select performs bit extractions and comparisons as well as swaps of keys (moves and exchanges). For unbiased data, the average number of bit comparisons has been known for a long time (see Knuth (1998, pp. 482–486)), and its distribution is studied in Mahmoud *et al.* (2000). Our purpose here is to analyze the number of key swaps of Radix Select for biased and unbiased data. Studies of the influence of swaps on the performance of an algorithm are in the vogue; see Hwang and Tsai (2002), Mahmoud (2010), and Martínez and Prodinger (2009) for a similar undertaking in Quick Sort.

## 5. The data model

We assume the Bernoulli($p$) model of randomness, according to which the bits within a key are independent with probability $p \in (0, 1)$ of a bit being 1, and probability $q$ of a bit being 0 (with $p + q = 1$), and the keys themselves are independent.

In this data model, keys are strings over a finite alphabet. In particular, numbers from the interval $(0, 1]$ may be thought of as infinite expansions. For rational numbers, there are two representations: we choose the infinite. In practice, real numbers are approximated by cutting off the expansion at some finite precision.

The ideal unbiased Bernoulli ($\frac{1}{2}$) model is equivalent to sampling from a uniform distribution, a realistic assumption under hashing schemes, where the primary goal is to achieve uniformity; see Fagin *et al.* (1979). However, a data source may degrade over time and $p$ may shift from an ideal $\frac{1}{2}$ to another value, which makes the analysis for general $p$ relevant.

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

## 6. Scope

Let $S_n^{(r)}$ be the number of swaps made by Radix Select on a random input of size $n$ to find the $r$th-order statistic. This variable is easy to analyze for relatively small or relatively large $r$. It is harder to analyze this variable for intermediate values of $r$, such as when $r = \lfloor 0.395n \rfloor$.

Analyzing $S_n^{(r)}$ when $r$ itself is random (chosen uniformly from $\{1, \ldots, n\}$) provides smoothing over all possible values of $r$. So, we let $r$ be a random variable $R_n$ distributed like Uniform$[1\,..\,n]$ (the discrete uniform random variable on the set $\{1, \ldots, n\}$). This rank randomization introduces a smoothing operation over the easy and hard cases that makes the problem of moderate complexity and amenable to analysis. In this case we can use the simplified notation $S_n := S_n^{(R_n)}$. We seek a grand average of all averages, a grand variance of all variances, and a grand (average) distribution of all distributions with a fixed $r$ as a global measure over all possible order statistics. This smoothing technique was introduced in Mahmoud *et al.* (1995), and was used successfully in Lent and Mahmoud (1996) and Prodinger (1995).

We will analyze the number of swaps made by Radix Select to select a key with a random rank under biased and unbiased data models, and we will see that there is a phase change in the distribution of the number of swaps between the unbiased case and all the biased cases. In the unbiased case, when suitably scaled, $S_n$ converges in probability to a constant, whereas the same scaling gives a perpetuity in all the biased cases.

## 7. Organization

The sections of the sequel are organized as follows. Section 8 lays out the basic probability notation. The technical analysis is given in Section 9, which is divided into subsections. Subsection 9.1 is devoted to the analysis of the number of swaps in the first round of partitioning. Subsection 9.2 is for an exact computation of the overall mean (by induction; biased and unbiased cases have the same mean). Subsection 9.3 is for an asymptotic analysis of the overall variance, done by poissonization, the Mellin transform and its inverse, followed by depoissonization. Here we see a phase change in the variance, being quadratic in all the biased cases, but turns linear in the unbiased case. A word about the significance of the Mellin transform in the analysis of algorithms is mentioned in this subsection. The details of depoissonization are relegated to Appendix A. Finally, Subsection 9.4 is where the main result is formally stated and proved (convergence of the scaled number of swaps to a perpetuity in the biased cases, and to a constant in the unbiased case). The method used in the biased cases is contraction in the metric space of distribution functions (endowed with the Wasserstein distance) to a fixed point. Technical details of the contraction proof are given in Appendix B. In Section 10 we sketch how the results can be extended to a number of other MSD variants of the algorithm.

## 8. Notation

We will use the following standard notation: $\mathbf{1}_{\mathcal{E}}$ is the indicator function of the event $\mathcal{E}$ that assumes the value 1 when $\mathcal{E}$ occurs, and assumes the value 0 otherwise.

The notation $\mathrm{Bin}(n, p)$ stands for a binomially distributed random variable counting the number of successes in $n$ independent, identically distributed experiments, with rate of success $p$ per experiment.

We use the notation '$\stackrel{\mathrm{D}}{=}$' to mean (exact) equality in distribution, and '$\stackrel{\mathrm{D}}{\to}$' to mean weak convergence (convergence in distribution). The notation '$\stackrel{\mathrm{P}}{\to}$' stands for convergence in probability, '$\stackrel{\mathrm{a.s.}}{\longrightarrow}$' stands for convergence almost surely, and '$\stackrel{\mathcal{L}_k}{\longrightarrow}$' stands for convergence in $\mathcal{L}_k$.

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

The notation $O_{\mathcal{L}_k}(g(n))$ stands for a sequence of random variables that is $O(g(n))$ in the $\mathcal{L}_k$ norm, that is, when we describe a sequence of random variables $Y_n$ to be $O_{\mathcal{L}_k}(g(n))$, we mean there exist a positive constant $C$ and a positive integer $n_0$, such that $\mathrm{E}[|Y_n|^k] \le C|g(n)|$ for all $n \ge n_0$. Unless otherwise stated, all asymptotics mean asymptotic equivalents and bounds as $n \to \infty$.

Let Hypergeo$(n, s, w)$ be a hypergeometric random variable that denotes the number of white balls in a sample of size $s$ balls taken at random (all subsets of size $s$ being equally likely) from an urn containing a total of $n$ white and black balls of which $w$ are white. The mean and variance for this standard distribution are given by the formulae

$$\mathrm{E}[\text{Hypergeo}(n, s, w)] = \frac{ws}{n}, \tag{1}$$

$$\mathrm{var}[\text{Hypergeo}(n, s, w)] = \frac{ws(n-w)(n-s)}{n^2(n-1)}; \tag{2}$$

see Stuart and Ord (1987, Article 5.14).

## 9. Analysis of in-situ binary radix selection

Let $M_n$ be the number of swaps exercised in the first call to Partition; we refer to this invocation as the *first round*.

Let $k = J_n + 1 = \mathrm{Bin}(n, q) + 1$ be the splitting position. Note that $R_n$ and $J_n$ are independent. Immediately after the first round of partitioning, the array is split into two segments: $A[1\,..\,J_n]$ containing keys starting with 0, and $A[J_n + 1\,..\,n]$ containing keys that start with 1. We have a dichotomy:

$$S_n = \begin{cases} M_n + S_{J_n} & \text{if } R_n \le J_n, \\ M_n + \tilde{S}_{n-J_n} & \text{if } R_n > J_n, \end{cases}$$

or, equivalently,

$$S_n \stackrel{\mathrm{D}}{=} M_n + S_{J_n}\,\mathbf{1}_{\{R_n \le J_n\}} + \tilde{S}_{n-J_n}\,\mathbf{1}_{\{R_n > J_n\}}\,. \tag{3}$$

Here $\tilde{S}_{n-J_n} \stackrel{\mathrm{D}}{=} S_{n-J_n}$, and $\tilde{S}_{n-J_n}$ and $S_{J_n}$ are conditionally independent (in the sense that, given $J_n = j$, $\tilde{S}_{n-j}$ and $S_j$ are independent).

### 9.1. Analysis of swaps in the first round

It is evident from the stochastic recurrence (3) that an analysis of Radix Select falls back on a clear understanding of the probabilistic behavior of the number of swaps in the first round.

**Lemma 1.** *The random variable $M_n$ is distributed like* Hypergeo$(n, J_n, n - J_n)$.

*Proof.* Let $J_n$ be the number of keys starting with 0. Suppose that the raw data (unsorted array) contain $m$ keys starting with 0 placed after position $J_n$, with $0 \le m \le \min\{J_n, n - J_n\}$. Each of these keys induces one swap during the first call to partition. Given $J_n = j$, we have $\binom{n}{j}$ equally likely arrangements of keys (each occurring with unconditional probability $p^{n-j}q^j$). For $M_n$ to be $m$, we must have exactly $j - m$ keys starting with 0 appear among the first $j$ keys (which can be done by choosing $j - m$ positions among the first $j$ positions in $\binom{j}{j-m}$ ways), and we must have exactly $m$ keys starting with 0 appear among the last $n - j$ keys (which can be done by choosing $m$ positions among the last $n - j$ positions in $\binom{n-j}{m}$ ways). Therefore,

$$\mathrm{P}(M_n = m \mid J_n = j) = \binom{n-j}{m}\binom{j}{j-m}\Big/\binom{n}{j}.$$

Whence, $M_n \mid J_n = j$ has the distribution of Hypergeo$(n, j, n - j)$.

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

**Corollary 1.** *We have* $\mathrm{E}[M_n] = pq(n-1)$.

*Proof.* We appeal to standard double expectation and (1) to obtain

$$\begin{aligned}\mathrm{E}[M_n] &= \mathrm{E}[\mathrm{Hypergeo}(n, J_n, n - J_n)]\\ &= \mathrm{E}[\mathrm{E}[\mathrm{Hypergeo}(n, J_n, n - J_n) \mid J_n]]\\ &= \mathrm{E}\left[\frac{J_n(n - J_n)}{n}\right].\end{aligned}$$

Recall that $J_n \stackrel{\mathrm{D}}{=} \mathrm{Bin}(n, q)$, and the result follows.

**Corollary 2.** *We have*

$$\mathrm{var}[M_n] = pq\left((1 - 4pq)n - 2(1 - 5pq) + \frac{2}{n}(1 - 3pq)\right).$$

*Proof.* We compute $\mathrm{var}[M_n]$ via the conditional variance formula

$$\mathrm{var}[M_n] = \mathrm{var}[\mathrm{E}[M_n \mid J_n]] + \mathrm{E}[\mathrm{var}[M_n \mid J_n]].$$

Using (1) and (2), we write this as

$$\begin{aligned}\mathrm{var}[M_n] &= \mathrm{var}\left[\frac{J_n(n - J_n)}{n}\right] + \mathrm{E}\left[\frac{J_n^2(n - J_n)^2}{n^2(n-1)}\right]\\ &= \frac{1}{n^2}(\mathrm{E}[J_n^2(n - J_n)^2] - \mathrm{E}^2[J_n(n - J_n)]) + \mathrm{E}\left[\frac{J_n^2(n - J_n)^2}{n^2(n-1)}\right].\end{aligned}$$

Recall that $J_n \stackrel{\mathrm{D}}{=} \mathrm{Bin}(n, q)$. Hence, the variance of $M_n$ requires the first four moments of $J_n$. These can be obtained from reductions of

$$\sum_{j=0}^{n} j^{\nu}\binom{n}{j}q^j p^{n-j} \quad \text{for } \nu = 1, 2, 3, 4,$$

by standard combinatorial identities (we can also get the first four moments of the binomial distribution from a standard book like Stuart and Ord (1987, Article 5.4)). The corollary follows after simplification.

Forthcoming proofs need finer properties of $M_n$ that go beyond the first two moments, such as the concentration law given in the next lemma.

**Lemma 2.** *The random variable* $M_n$ *follows the concentration laws:*

(a) *in the biased cases*

$$\frac{M_n}{n} \stackrel{\mathrm{P}}{\longrightarrow} pq,$$

(b) *in the unbiased case*

$$\frac{M_n}{n} \stackrel{\text{a.s.}}{\longrightarrow} \frac{1}{4}.$$

*Proof.* Consider $n \geq 2$. By Chebyshev's inequality, for any fixed $\varepsilon > 0$, we write

$$\mathrm{P}(|M_n - pq(n-1)| > \varepsilon) \leq \frac{\mathrm{var}[M_n]}{\varepsilon^2}.$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

Replace $\varepsilon$ by $\varepsilon(n-1)$ to obtain

$$\mathrm{P}\left(\left|\frac{M_n}{n-1} - pq\right| > \varepsilon\right) \le \frac{pq((1-4pq)n - 2(1-5pq) + 2(1-3pq)/n)}{\varepsilon^2(n-1)^2} \to 0. \tag{4}$$

Therefore, $M_n/n \xrightarrow{\mathrm{P}} pq$ in both the unbiased and biased cases.

In the unbiased case, we get a sharper result via a faster rate of convergence to 0 for the probability of a deviation. The bound in (4) becomes

$$\mathrm{P}\left(\left|\frac{M_n}{n-1} - \frac{1}{4}\right| > \varepsilon\right) \le \frac{1/8 + 1/8n}{\varepsilon^2(n-1)^2} \le \frac{1}{2\varepsilon^2 n^2}$$

for $n \ge 2$. This fast rate of decrease in the probabilities of a deviation renders the series of the probabilities summable:

$$\sum_{n=2}^{\infty} \mathrm{P}\left(\left|\frac{M_n}{n-1} - \frac{1}{4}\right| > \varepsilon\right) \le \frac{\pi^2}{12\varepsilon^2} < \infty.$$

According to the Borel–Cantelli lemma,

$$\frac{M_n}{n} \xrightarrow{\text{a.s.}} \frac{1}{4};$$

thus, case (b) is established.

### 9.2. The mean of the overall number of swaps

Equation (3) yields a recurrence for the average, which can be solved exactly.

**Proposition 1.** *For $n \ge 1$,*

$$\mathrm{E}[S_n] = \tfrac{1}{2}(n-1).$$

*Proof.* Equation (3) gives a recurrence for the average:

$$\mathrm{E}[S_n] = \mathrm{E}[M_n] + \mathrm{E}[S_{J_n}\mathbf{1}_{\{R_n \le J_n\}}] + \mathrm{E}[S_{n-J_n}\mathbf{1}_{\{R_n > J_n\}}].$$

By conditioning on $J_n$ and $R_n$ (which are independent) and Corollary 1, we obtain

$$\begin{aligned}
\mathrm{E}[S_n] &= \mathrm{E}[M_n] + \sum_{j=0}^{n}\sum_{r=1}^{n} \mathrm{E}[S_{J_n}\mathbf{1}_{\{R_n \le J_n\}} \mid J_n = j,\ R_n = r]\,\mathrm{P}(J_n = j,\ R_n = r) \\
&\quad + \sum_{j=0}^{n}\sum_{r=1}^{n} \mathrm{E}[S_{n-J_n}\mathbf{1}_{\{R_n > J_n\}} \mid J_n = j,\ R_n = r]\,\mathrm{P}(J_n = j,\ R_n = r) \\
&= pq(n-1) + \sum_{j=0}^{n}\sum_{r=1}^{n} \mathrm{E}[S_j\mathbf{1}_{\{r \le j\}}]\,\mathrm{P}(J_n = j)\,\mathrm{P}(R_n = r) \\
&\quad + \sum_{j=0}^{n}\sum_{r=1}^{n} \mathrm{E}[S_{n-j}\mathbf{1}_{\{r > j\}}]\,\mathrm{P}(J_n = j)\,\mathrm{P}(R_n = r) \\
&= pq(n-1) + \frac{1}{n}\sum_{j=0}^{n} j\,\mathrm{E}[S_j]\binom{n}{j}q^j p^{n-j} + \frac{1}{n}\sum_{j=0}^{n}(n-j)\,\mathrm{E}[S_{n-j}]\binom{n}{j}q^j p^{n-j}.
\end{aligned}$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

We next show by induction on $n$ that $\mathrm{E}[S_n] = \frac{1}{2}(n-1)$ is an exact solution to the latter recurrence. At $n = 1$, we have $\mathrm{E}[S_1] = 0 = \frac{1}{2}(1-1)$, providing a basis. Assuming that the result holds up to $n - 1 \geq 0$, we then have

$$\begin{aligned}(1 - p^n - q^n)\,\mathrm{E}[S_n] &= pq(n-1) + \frac{1}{2n}\sum_{j=0}^{n-1} j(j-1)\binom{n}{j} q^j p^{n-j} \\ &\quad + \frac{1}{2n}\sum_{j=1}^{n}(n-j)(n-j-1)\binom{n}{j} q^j p^{n-j} \\ &= \tfrac{1}{2}(n-1)(1-p^n-q^n),\end{aligned}$$

and the induction is complete.

### 9.3. The variance of the overall number of swaps

Equation (3) yields a recurrence for the second moment, which can be solved via the poissonization-Mellin transform and its inverse-depoissonization program, and the variance follows. Interestingly, there is a phase change in the unbiased case $p = q = \frac{1}{2}$. Except for the unbiased case, the variance is quadratic in $n$; in the unbiased case it is linear.

A main tool in the ensuing analysis is the Mellin transform. We give a brief sketch. The Mellin transform of a function $f(x)$ is

$$\int_0^\infty f(x) x^{s-1}\,\mathrm{d}x,$$

and is denoted by $f^*(s)$. The Mellin transform usually exists in vertical strips, in the $s$-complex plane, of the form

$$a < \mathrm{Re}(s) < b$$

for real numbers $a < b$. We will denote this strip by $\langle a, b\rangle$. The function $f(x)$ can be recovered from its transform by a line integral

$$f(x) = \frac{1}{2\pi \mathrm{i}} \int_{c-\mathrm{i}\infty}^{c+\mathrm{i}\infty} f^*(s) x^{-s}\,\mathrm{d}s$$

for any $c \in (a, b)$. Usually, such an integral is computed asymptotically (as $x \to \infty$) by shifting the line of integration an arbitrary distance to the right of the existence strip, and compensating for the shift by the residues of the poles between the two lines of integration. There often is a small residual error of the form $O(x^{-\theta})$ for an arbitrary large positive number $\theta$. For a survey of the uses of the Mellin transform in the analysis of algorithms, see Flajolet *et al.* (1995).

**Proposition 2.** *We have*

$$\mathrm{var}[S_n] = \begin{cases} \frac{1}{12}(1-4pq)n^2 + O(n) & \text{if } p \neq q, \\ \frac{1}{4}n + O(1) & \text{if } p = q = \frac{1}{2}. \end{cases}$$

*Proof.* Let $H_{n,j} \overset{\mathrm{D}}{=} \mathrm{Hypergeo}(n, j, n-j)$. Starting with (3) in the squared form, i.e.

$$S_n^2 \overset{\mathrm{D}}{=} M_n^2 + S_{J_n}^2\,\mathbf{1}_{\{R_n \leq J_n\}} + \tilde{S}_{n-J_n}^2\,\mathbf{1}_{\{R_n > J_n\}} + 2M_n S_{J_n}\,\mathbf{1}_{\{R_n \leq J_n\}} + 2M_n \tilde{S}_{n-J_n}\,\mathbf{1}_{\{R_n > J_n\}},$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

we take expectations to get a recurrence for the second moment:

$$\mathrm{E}[S_n^2] = \frac{1}{n}\Bigg(\sum_{j=0}^{n}(n\,\mathrm{E}[H_{n,j}^2] + j\,\mathrm{E}[S_j^2] + (n-j)\,\mathrm{E}[S_{n-j}^2] + 2j\,\mathrm{E}[H_{n,j}]\,\mathrm{E}[S_j] + 2(n-j)\,\mathrm{E}[H_{n,j}]\,\mathrm{E}[S_{n-j}])\binom{n}{j}p^{n-j}q^j\Bigg).$$

For $\mathrm{E}[S_j]$, we plug in the simple result of Proposition 1 and, for the first two moments of $H_{n,j}$, we plug in (1) and (2). The recurrence takes the form

$$\mathrm{E}[S_n^2] = \frac{1}{n}\Bigg(\sum_{j=0}^{n} j\,\mathrm{E}[S_j^2]\binom{n}{j}p^{n-j}q^j + \sum_{j=0}^{n} j\,\mathrm{E}[S_j^2]\binom{n}{j}p^{j}q^{n-j}\Bigg) + f_n(p,q), \tag{5}$$

where

$$\begin{aligned} f_n(p,q) = {} & \sum_{j=0}^{n}\left(\frac{j^2(n-j)^2}{n^2(n-1)} + \frac{j^2(n-j)^2}{n^2}\right)\binom{n}{j}p^{n-j}q^j \\ & + \frac{2}{n}\sum_{j=2}^{n} j\frac{j(n-j)}{n}\frac{1}{2}(j-1)\binom{n}{j}p^{n-j}q^j \\ & + \frac{2}{n}\sum_{j=0}^{n-2}(n-j)\frac{j(n-j)}{n}\frac{1}{2}(n-j-1)\binom{n}{j}p^{n-j}q^j. \end{aligned}$$

All the sums, except those involving second moments of $S_j$, $j = 0, \ldots, n$, are then reduced by standard binomial identities. We obtain

$$f_n(p,q) = pq(1-pq)n^3 - pq(3-7pq)n^2 + 4(1-4pq)n - 2pq(1-6pq).$$

We then poissonize by introducing the generating function

$$\tilde{A}(z) = \mathrm{e}^{-z}A(z) := \mathrm{e}^{-z}\sum_{n=0}^{\infty} n\,\mathrm{E}[S_n^2]\frac{z^n}{n!}.$$

In this context poissonization means considering an analogous problem, but with a Poisson random number of keys, instead of fixed $n$. The number of keys is taken to be a Poisson random variable with parameter $z$. Indeed, $A(z)$ has a poissonization interpretation. For, if $N(z)$ is distributed like a Poisson random variable with mean $z$ then

$$\begin{aligned} \tilde{A}(z) &= \sum_{n=0}^{\infty} n\,\mathrm{E}[S_n^2]\,\mathrm{P}(N(z)=n) \\ &= \sum_{n=0}^{\infty} \mathrm{E}[N(z)S_{N(z)}^2 \mid N(z)=n]\,\mathrm{P}(N(z)=n) \\ &= \mathrm{E}[N(z)S_{N(z)}^2]. \end{aligned}$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

After some lengthy algebraic operations we arrive at

$$A(z) = \mathrm{e}^{qz}A(pz) + \mathrm{e}^{pz}A(qz) + 2pq(1-6pq)(z+1-\mathrm{e}^z) + 2pq(1-5pq)z(\mathrm{e}^z-1) + pq(1-pq)z^3\mathrm{e}^z + 4p^2q^2z^2\mathrm{e}^z.$$

To complete the poissonization, multiply by $\mathrm{e}^{-z}$ to write the linearized functional equation

$$\tilde{A}(z) = \tilde{A}(pz) + \tilde{A}(qz) + 2pq(1-6pq)((z+1)\mathrm{e}^{-z}-1) + 2pq(1-5pq)z(1-\mathrm{e}^{-z}) + pq(1-pq)z^3 + 4p^2q^2z^2.$$

The trailing cubic polynomial does not have a domain of existence for the Mellin transform. However, a transformation that puts the functional equation in the form

$$B(z) = B(pz) + B(qz) + \psi(z),$$

where $\psi(z)$ is a function that has a Mellin transform and enables us to asymptotically solve the functional equation for $A(z)$. One good form for $\psi(z)$ is $\alpha z(\mathrm{e}^{-z}-1) + \beta(\mathrm{e}^{-z}-1+z)$ for constants $\alpha$ and $\beta$, which has a Mellin transform in $\langle -2, -1\rangle$. This is accomplished by subtracting an appropriate polynomial in $z$ of degree 3 from $A(z)$.

We seek

$$Q(z) = a_3z^3 + a_2z^2 + a_1z + a_0,$$

such that

$$B(z) := \tilde{A}(z) - Q(z) = B(pz) + B(qz) + \alpha(\mathrm{e}^{-z}-1) + \beta(\mathrm{e}^{-z}-1+z).$$

We choose $a_3$ to eliminate third powers of $z$, that is,

$$a_3 = \frac{1-pq}{3}.$$

Similarly, we eliminate quadratic powers by setting $a_2 = 2pq$. It is clear that $a_1$ can be chosen arbitrarily, and we take it to be 0. If we take $a_0 = 0$, we can organize what is left as $\alpha z(\mathrm{e}^{-z}-1) + \beta(\mathrm{e}^{-z}-1+z)$, then solve for $\alpha$ and $\beta$ to obtain

$$\alpha = -2p^2q^2, \qquad \beta = 2pq(1-6pq).$$

Hence,

$$B(z) = \tilde{A}(z) - \frac{1-pq}{3}z^3 - 2pqz^2 - 2pq + 12p^2q^2$$

is the transformation that puts the functional equation in the required form:

$$B(z) = B(pz) + B(qz) - 2p^2q^2z(\mathrm{e}^{-z}-1) + 2pq(1-6pq)(\mathrm{e}^{-z}-1+z).$$

This functional equation has the Mellin transform

$$B^*(s) = \frac{2pq(1-pq(s+6))}{1-p^{-s}-q^{-s}}\Gamma(s),$$

existing in $\langle -2, -1\rangle$, with inverse

$$B(z) = 2pq(1-5pq)z\ln z + O(z).$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

The $O(z)$ term has oscillations. We obtain the asymptotic expansion

$$A(z) = \mathrm{E}[N(z)S_{N(z)}^2] = \frac{1-pq}{3}z^3 + 2pqz^2 + 2pq(1-5pq)z\ln z + O(z);$$

the $O(z)$ term contains oscillations.

Consider first the biased case, where the coefficient of $z^3$ is a positive constant. By depoissonization (see Appendix A) we obtain

$$\mathrm{E}[nS_n^2] = \frac{1-pq}{3}n^3 + O(n^2).$$

Note that we cannot identify a more accurate error term, because depoissonization adds an error of order $n^2$. Hence,

$$\begin{aligned}\mathrm{var}[S_n] &= \mathrm{E}[S_n^2] - \mathrm{E}^2[S_n] \\ &= \frac{1-pq}{3}n^2 + O(n) - \left(\frac{n-1}{2}\right)^2 \\ &= \tfrac{1}{12}(1-4pq)n^2 + O(n).\end{aligned}$$

In the unbiased case the leading term of order $n^2$ disappears, leaving $O(n)$ variance, which cannot be determined more explicitly by this method, because the depoissonization error cannot be improved beyond $O(n)$. However, a direct induction gives us asymptotics accurate enough for our purpose.

We show by induction on $n$ that in the unbiased case, for all $n \geq 1$,

$$\frac{n^2}{4} - \frac{n}{4} - \frac{1}{4} \leq \mathrm{E}[S_n^2] \leq \frac{n^2}{4} - \frac{n}{4}. \tag{6}$$

In the unbiased case recurrence (5) takes the symmetrical form

$$n\,\mathrm{E}[S_n^2] = \frac{2}{2^n}\sum_{j=0}^{n} j\,\mathrm{E}[S_j^2]\binom{n}{j} + \frac{3}{16}n^3 - \frac{5}{16}n^2 + \frac{1}{4}$$

for $n \geq 2$. If we let $y_n = n\,\mathrm{E}[S_n^2]$, we have

$$y_n = \frac{2}{2^n}\sum_{j=0}^{n} y_j\binom{n}{j} + \frac{3}{16}n^3 - \frac{5}{16}n^2 + \frac{1}{4}$$

for $n \geq 2$, with $y_0 = y_1 = 0$.

We give a proof by induction to show that $y_n \leq n^3/4 - n^2/4$ for all $n \geq 0$. We have $y_0 = 0 = \frac{1}{4}(0^3 - 0^2)$, and $y_1 = 0 = \frac{1}{4}(1^3 - 1^2)$. Suppose that the assertion holds up to $n-1$ for some $n \geq 2$. Then,

$$\begin{aligned}y_n\left(1 - \frac{2}{2^n}\right) &\leq \frac{2}{2^n}\sum_{j=0}^{n-1}\frac{1}{4}(j^3 - j^2)\binom{n}{j} + \frac{3}{16}n^3 - \frac{5}{16}n^2 + \frac{1}{4} \\ &= \frac{1}{4}(n^3 - n^2)\left(1 - \frac{2}{2^n}\right) + \frac{1}{4} - \frac{n}{8}.\end{aligned}$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

The quantity $\frac{1}{4} - \frac{1}{8}n$ is nonpositive for all $n \geq 2$, yielding

$$y_n\left(1 - \frac{2}{2^n}\right) \leq \frac{1}{4}(n^3 - n^2)\left(1 - \frac{2}{2^n}\right),$$

or

$$y_n \leq \tfrac{1}{4}(n^3 - n^2),$$

and the induction is complete. The upper bound stated in (6) follows. The proof for the lower bound is similar, and we omit it.

As an immediate consequence of (6) and Proposition 1, we have

$$\begin{aligned}\operatorname{var}[S_n] &= \mathrm{E}[S_n^2] - \mathrm{E}^2[S_n] \\ &= \frac{n^2}{4} - \frac{n}{4} + O(1) - \left(\frac{n-1}{2}\right)^2 \\ &= \tfrac{1}{4}n + O(1).\end{aligned}$$

### 9.4. The asymptotic distribution of the overall number of swaps

We now present the main result of this paper.

**Theorem 1.** *Let $S_n$ be the number of swaps Radix Select executes when it searches for a key of a rank selected uniformly at random, among n keys following the* Bernoulli*(p) model. We then have*

(a) *in the unbiased case*

$$\frac{S_n}{n} \xrightarrow{\mathrm{P}} \frac{1}{2},$$

(b) *in the biased case*

$$\frac{S_n}{n} \xrightarrow{\mathrm{D}} S^*,$$

*where $S^*$ is a perpetuity given by*

$$S^* = pq \sum_{n=0}^{\infty} \prod_{j=1}^{n} V_j,$$

*and the variables $\{V_j\}_{j=1}^{\infty}$ are independent and identically distributed like the two-point random variable*

$$V = \begin{cases} p & \text{with probability } p, \\ q & \text{with probability } q. \end{cases}$$

*Proof.* In the unbiased case, by Proposition 2 and Chebyshev's inequality, for any fixed $\varepsilon > 0$,

$$\mathrm{P}\left(\left|S_n - \frac{1}{2}(n-1)\right| > \varepsilon\right) \leq \frac{\operatorname{var}[S_n]}{\varepsilon^2}.$$

Replace $\varepsilon$ by $\varepsilon(n-1)$ to obtain

$$\mathrm{P}\left(\left|\frac{S_n}{n-1} - \frac{1}{2}\right| > \varepsilon\right) \leq \frac{\operatorname{var}[S_n]}{\varepsilon^2(n-1)^2} = O\left(\frac{1}{n}\right) \to 0.$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

Therefore,

$$\frac{S_n}{n} \xrightarrow{\text{P}} \frac{1}{2}.$$

We next take up the biased cases. Introduce

$$S_n^* = \frac{S_n}{n}.$$

We normalize (3) by writing it in the form

$$\frac{S_n}{n} \stackrel{\text{D}}{=} \frac{S_{J_n}}{J_n}\frac{J_n}{n}\mathbf{1}_{\{R_n \le J_n\}} + \frac{\tilde{S}_{n-J_n}}{n-J_n}\frac{n-J_n}{n}\mathbf{1}_{\{R_n > J_n\}} + \frac{M_n}{n}.$$

Expressed in terms of the normalized random variables, this is

$$S_n^* \stackrel{\text{D}}{=} S_{J_n}^*\frac{J_n}{n}\mathbf{1}_{\{R_n \le J_n\}} + \tilde{S}_{n-J_n}^*\frac{n-J_n}{n}\mathbf{1}_{\{R_n > J_n\}} + \frac{M_n}{n}, \tag{7}$$

where, for each $j$, $\tilde{S}_j \stackrel{\text{D}}{=} S_j$ and the families $\{S_j^*\}$, $\{\tilde{S}_j^*\}$, $\{J_j\}$, and $\{R_j\}$ are totally independent (for the usual definition of total independence, see any classic book on probability, such as Chung (1974), for example). This representation suggests a limiting functional equation as follows. We first observe that

$$\frac{J_n}{n} \xrightarrow{\text{P}} q \quad \text{and} \quad \frac{n-J_n}{n} \xrightarrow{\text{P}} p.$$

Also,

$$\mathbf{1}_{\{R_n \le J_n\}} \xrightarrow{\text{P}} B^* \quad \text{and} \quad \mathbf{1}_{\{R_n > J_n\}} \xrightarrow{\text{P}} 1 - B^*,$$

where $B^*$ is a Bernoulli random variable with success probability $q$. Recall from Lemma 2 that $M_n/n \xrightarrow{\text{P}} pq$.

To summarize, if $S_n^*$ converges in distribution to a limiting random variable $S^*$, so will $S_{J_n}^*$ and $\tilde{S}_{n-J_n}^*$, as both $J_n$ and $n-J_n$ go to $+\infty$ almost surely. We can surmise that the limit satisfies the distributional equation

$$S^* \stackrel{\text{D}}{=} qB^*S^* + p(1-B^*)\tilde{S}^* + pq, \tag{8}$$

with $S^*$, $\tilde{S}^*$, and $B^*$ being independent. To formally justify this guessed limit equation, we use a technical lemma presented in Appendix B.

Equation (8) is equivalent to

$$S^* \stackrel{\text{D}}{=} VS^* + pq, \tag{9}$$

where $S^*$ and $V$ are independent, and $V$ is a two-point random variable with distribution

$$V = \begin{cases} p & \text{with probability } p, \\ q & \text{with probability } q. \end{cases}$$

A random variable satisfying a distributional recurrence of the type (9) is called a *perpetuity*; see Knape and Neininger (2008). The perpetuity representation (9) allows us to obtain an expression for $S^*$ as a sum of products of independent random variables. Toward this end, let $S_1^*, S_2^*, \ldots$ be independent copies of $S^*$, and let $V_1, V_2, \ldots$ be independent copies of $V$. Then

$$S^* \stackrel{\text{D}}{=} pq + V_1S_1^* \stackrel{\text{D}}{=} pq + V_1(pq + V_2S_2^*).$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

Note that because $S_1^*$ is independent of $V_1$, the $S^*$ and $V$ introduced in the next iteration must be independent copies of $S_1^*$ and $V_1$. Continuing the iterations (always introducing new independent random variables), we arrive at

$$\begin{aligned} S^* &\stackrel{\mathrm{D}}{=} pq + pqV_1 + V_1V_2(pq + V_3S_3^*) \\ &\stackrel{\mathrm{D}}{=} pq\sum_{n=0}^{L}\left(\prod_{j=1}^{n} V_j\right) + V_1V_2\cdots V_{L+1}S^* \end{aligned} \tag{10}$$

for any integer $L \geq 0$.

We demonstrate next that the tail term converges to 0, allowing us passage to a limit representation as an infinite series. Let $\rho = \max\{p, q\}$. We have $V \leq \rho$. It follows that the tail term

$$V_1V_2\cdots V_{L+1} \leq \rho^{L+1} \to 0 \quad \text{as } L \to \infty.$$

By Slutsky's theorem, the product $V_1V_2\cdots V_{L+1}S^*$ converges to 0 in probability. We can proceed with the limit of (10) and write

$$S^* \stackrel{\mathrm{D}}{=} pq(1 + V_1 + V_1V_2 + V_1V_2V_3 + \cdots), \tag{11}$$

which establishes the result.

**Remark.** We can still interpret the result in the unbiased case as a 'perpetuity' built from constants—in this case the two points of the distribution of $V$ coincide in the center and $V$ becomes degenerate—the two probabilities fold up, and $\mathrm{P}(V = \frac{1}{2}) = 1$; then we can say that (11) holds in the form

$$S^* \stackrel{\mathrm{D}}{=} \tfrac{1}{4}\big(1 + \tfrac{1}{2} + \tfrac{1}{4} + \cdots\big) = \tfrac{1}{2}.$$

## 10. Extensions to some other MSD variants

The issues with other binary variants of MSD Radix Select are only algorithmic, and so are the extensions to $m$-ary data. However, the analysis of the underlying parameters remains essentially the same. Other variants are more complicated and some variants are wasteful of some resources, such as the amount of space allocated to the execution of the algorithm. A practitioner might ask for advice on how these variants compare. For the variants we discussed in Section 2, some execute the algorithm through a sequence of moves rather than swaps. To have a uniform base for comparing different variants, we should translate swaps-based analysis into moves-based analysis. We let $X_n$ denote the total number of moves exercised by any version.

In the binary version we analyzed, the average number of swaps is asymptotic to $\frac{1}{2}n$. Each swap is essentially three cyclic moves (via a temporary container). We thus have an asymptotic average of $\frac{3}{2}n$ moves. The exact distributional equation (7) gives rise to the limiting perpetuity equation

$$S^* \stackrel{\mathrm{D}}{=} VS^* + pq,$$

where $V$ is a 'selector' with a distribution on the two points $p$ and $q$ (with probabilities $p$ and $q$, respectively). In terms of scaled moves ($X_n^* = X_n/n$ with $\lim_{n\to\infty} X_n^* = X^*$), the perpetuity equation becomes

$$X^* \stackrel{\mathrm{D}}{=} VX^* + 3pq.$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

If instead we used the two-array implementation, we move the $n$ keys at the first round of partitioning in such a way to separate the keys beginning with 0 on the left-hand side of an auxiliary array, and the keys beginning with 1 on the right-hand side of the same auxiliary array. The equivalent of (7) then becomes

$$X_n^* \stackrel{\mathrm{D}}{=} X_{J_n}^* \frac{J_n}{n} \mathbf{1}_{\{R_n \le J_n\}} + \tilde{X}_{n-J_n}^* \frac{n-J_n}{n} \mathbf{1}_{\{R_n > J_n\}} + 1,$$

giving rise to the limiting perpetuity equation

$$X^* \stackrel{\mathrm{D}}{=} V X^* + 1;$$

the proofs are quite similar to those already presented, and we omit them. The average number of moves in this implementation is asymptotic to $n/(2pq)$. The best case is the unbiased case, with an asymptotic average of $2n$ moves. In the unbiased case, this version of the algorithm consumes twice the amount of space and executes $\frac{4}{3}$ as many moves as the in-situ version on average. The situation gets worse as $p$ and $q$ deviate further away from the center.

In the $m$-ary case, we have $m$ symbols in the alphabet, occurring with probabilities $p_1, \ldots, p_m$. (Let $q_i := 1 - p_i$ for $i = 1, \ldots, m$.) The two-array version still moves $n$ keys at the first round, and has the limiting perpetuity equation

$$X^* \stackrel{\mathrm{D}}{=} V_m X^* + 1;$$

here $V_m$ is an $m$-point distribution, on the set $\{p_1, \ldots, p_m\}$, with probabilities $p_1, \ldots, p_m$. Asymptotically, as $n \to \infty$, the average number of moves is

$$\mathrm{E}[X_n] \sim \frac{n}{1 - \sum_{i=1}^m p_i^2}.$$

Just like the binary in-situ case, any of these perpetuity equations can be iterated to give us sums of random variables. For example, the two-array implementation for binary data admits the explicit representation

$$X^* \stackrel{\mathrm{D}}{=} 1 + V_1 + V_1 V_2 + V_1 V_2 V_3 + \cdots.$$

The linked-list implementation must be viewed in a different light. This algorithm makes no moves per se. It performs the selection via a series of linkage changes. If we let $X_n$ denote the number of such operations, the equivalent of (7) becomes

$$X_n^* \stackrel{\mathrm{D}}{=} X_{J_n}^* \frac{J_n}{n} \mathbf{1}_{\{R_n \le J_n\}} + \tilde{X}_{n-J_n}^* \frac{n-J_n}{n} \mathbf{1}_{\{R_n > J_n\}} + \frac{H(n)}{n},$$

where $H(n)$ is the number of linkage changes in the first round of key separation. Let us discuss the $m$-ary case. Let $Y_i$ be the first letter in the $i$th key, and let us form the word $w_n$ by concatenating the $Y_i$s. As discussed, in this algorithm we detach a stretch of keys all beginning with the same letter (a sublist) all at once, whenever we detect two consecutive letters in $w_n$ that are different. Let $\tilde{H}(n)$ be the number of such letter changes. The letters $Y_i$ occur independently and are identically distributed. It is clear that $H(n) = \tilde{H}(n) + O(1)$, where the $O(1)$ term accounts for the additional overhead for bookkeeping, and

$$\tilde{H}(n) = \sum_{i=2}^{n} Z_i,$$

where $Z_i = \mathbf{1}_{\{Y_i \neq Y_{i-1}\}}$. The variables $Z_i$ are identically distributed Bernoulli $(2\sum_{i=1}^m p_i q_i)$.

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

Note that the random variables $Z_i$ and $Z_{i+1}$ are dependent with covariance $\sum_{i=1}^{m} p_i q_i^2 - (2\sum_{i=1}^{m} p_i q_i)^2$. However, $Z_i$ and $Z_j$ are independent when the indices are at least two apart, and thus have 0 covariance. We have

$$\mathrm{E}[\tilde{H}(n)] = (n-1)\,\mathrm{E}[Z_2] = (n-1)\left(1 - \sum_{i=1}^{m} p_i^2\right),$$

and

$$\begin{aligned}
\mathrm{var}[\tilde{H}(n)] &= \sum_{i=2}^{n} \mathrm{var}[Z_i] + 2 \sum_{2\le i<j\le n}^{n} \mathrm{cov}[Z_i, Z_j] \\
&= (n-1)\,\mathrm{var}[Z_2] + 2\sum_{i=2}^{n-1} \mathrm{cov}[Z_2, Z_3] \\
&= (n-1)\left(\sum_{i=1}^{m} p_i^2\right)\left(1 - \sum_{i=1}^{m} p_i^2\right) \\
&\quad + 2(n-2)\left(\sum_{i=1}^{m} p_i q_i^2 - 4\left(\sum_{i=1}^{m} p_i q_i\right)^2\right).
\end{aligned}$$

These orders of mean and variance (via Chebyshev's inequality) assert that $n^{-1}H(n) \xrightarrow{\mathrm{P}} 1 - \sum_{i=1}^{m} p_i^2$ . The perpetuity equation is

$$X^* \stackrel{\mathrm{D}}{=} V X^* + 1 - \sum_{i=1}^{m} p_i^2.$$

For any sequence $p_1, \ldots, p_m$ of probabilities whatsoever, the linked-list implementation performs an asymptotic average of $n$ linkage changes to select an element with random rank from the list. The independence of this asymptotic average from the frequency of the letters is remarkable.

We have just shown how the analysis in the binary in-situ case can be extended to other MSD variants of Radix Sort. We covered all the variants mentioned in Section 2, except for the in situ with $m$-ary data; this is not one unique well-defined algorithm—it depends greatly on how the partition and the movement of keys are carried out. In principle, whichever choice for the implementation of the partitioning will lead essentially to the same type of analysis, i.e. some associated perpetuity.

## Appendix A. Depoissonization

**Theorem 2.** (Jacquet and Szpankowski (1998).) *Let $p, q > 0$, and let $p + q = 1$. Suppose that $g_n$ is a sequence of numbers with Poisson transform $G(z)$ satisfying the functional equation*

$$G(z) = u_1(z)G(pz) + u_2(z)G(qz) + T(z),$$

*which is stipulated to have an entire solution. Suppose further that, for some positive constants $C$, $\beta$, $z_0$, $0 < \theta < \pi/2$, and $0 < \eta < 1$, the following conditions hold.*

(i) *For all $z$ with $|\mathrm{Arg}(z)| < \theta$ and $|z| \ge z_0$,*

$$|u_1(z)|p^\beta + |u_2(z)|q^\beta \le 1 - \eta, \qquad |T(z)| \le C\eta|z|^\beta.$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

(ii) *For all $z$ with $|\mathrm{Arg}(z)| \geq \theta$ and $|z| \geq z_0$, and some $\alpha < 1$,*

$$|u_1(z)\mathrm{e}^{q\,\mathrm{Re}(z)}| \leq \tfrac{1}{3}\mathrm{e}^{\alpha|z|^q}, \qquad |u_2(z)\mathrm{e}^{p\,\mathrm{Re}(z)}| \leq \tfrac{1}{3}\mathrm{e}^{\alpha|z|^p}, \qquad |T(z)\mathrm{e}^{\mathrm{Re}(z)}| \leq \tfrac{1}{3}\mathrm{e}^{\alpha|z|}.$$

*Then, for large $n$,*

$$g_n = G(n) + O(n^{\beta-1});$$

*the $O$ term can be taken to depend only on $p$, $q$, and the constants that appear in the conditions.*

This depoissonization applies to the recurrence

$$\begin{aligned}\tilde{A}(z) = \tilde{A}(pz) + \tilde{A}(qz) &+ 2pq(1-6pq)((z+1)\mathrm{e}^{-z}-1) + 2pq(1-5pq)z(1-\mathrm{e}^{-z}) \\ &+ pq(1-pq)z^3 + 4p^2q^2z^2,\end{aligned}$$

where $u_1(z) = u_2(z) \equiv 1$, and the toll function is

$$\begin{aligned}T(z) = &\,2pq(1-6pq)((z+1)\mathrm{e}^{-z}-1) + 2pq(1-5pq)z(1-\mathrm{e}^{-z}) \\ &+ pq(1-pq)z^3 + 4p^2q^2z^2.\end{aligned}$$

We can take $\beta = 3$, $\eta = 3pq$, $C = 17/48pq$, and certainly for large $|z| \geq z_0 > 1$ inside the depoissonization cone the conditions

$$p^3 + q^3 = 1 - (3p^2q + 3q^2p) = 1 - 3pq = 1 - \eta$$

and

$$\begin{aligned}|T(z)| &= |2pq(1-6pq)(z\mathrm{e}^{-z} + (\mathrm{e}^{-z}-1)) + 2pq(1-5pq)z(1-\mathrm{e}^{-z}) \\ &\quad + pq(1-pq)z^3 + 4p^2q^2z^2| \\ &\leq \tfrac{1}{4}(|z|^3 + |z|^3) + \tfrac{1}{8}|z|^3 + \tfrac{3}{16}|z|^3 + \tfrac{1}{4}|z|^3 \\ &= \tfrac{17}{16}|z|^3 \\ &= C\eta|z|^3\end{aligned}$$

hold, and outside the cone so do the conditions

$$\mathrm{e}^{q\,\mathrm{Re}(z)} \leq \tfrac{1}{3}\mathrm{e}^{q|z|^\alpha}, \qquad \mathrm{e}^{p\,\mathrm{Re}(z)} \leq \tfrac{1}{3}\mathrm{e}^{p|z|^\alpha}, \qquad T(z)\mathrm{e}^{\mathrm{Re}(z)} \leq \tfrac{1}{3}\mathrm{e}^{|z|^\alpha},$$

for some $\alpha \in (\cos\theta, 1)$.

## Appendix B. Contraction

The *Wasserstein distance of order $k$* between two distribution functions $F$ and $G$ is defined by

$$d_k(F, G) = \inf ||W - Z||_k,$$

where the infimum is taken over all random variables $W$ and $Z$ having the respective distribution functions $F$ and $G$ (with $||\cdot||_k$ being the usual $\mathcal{L}_k$ norm). It is known (see Barbour *et al.* (1992)) that convergence in the second-order Wasserstein distance implies weak convergence, as well as convergence of the first two moments.

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

**Lemma 3.** *There exists a limiting random variable $S^*$ such that $S_n^* \xrightarrow{\mathrm{D}} S^*$. The limit $S^*$ satisfies the distributional functional equation*

$$S^* \stackrel{\mathrm{D}}{=} q B^* S^* + p \tilde{B}^* \tilde{S}^* + pq,$$

*where $\tilde{S}^* \stackrel{\mathrm{D}}{=} S^*$, $\tilde{B}^* = 1 - B^*$, $B^* \stackrel{\mathrm{D}}{=}$ Bernoulli$(q)$, and $S^*$, $\tilde{S}^*$, and $B^*$ are independent.*

*Proof.* Let $F_n^*(x)$ be the distribution function of $S_n^*$, and let $F^*(x)$ be the distribution function of $S^*$. We will show that the second-order Wasserstein distance between $F_n^*(x)$ and $F^*(x)$ converges to 0, and, consequently, $S_n^* \xrightarrow{\mathrm{D}} S^*$.

As $d_2(F_n^*, F^*)$ is an infimum over all $||W_n - W||_2$ for any random variables $W_n \stackrel{\mathrm{D}}{=} S_n^*$ and $W \stackrel{\mathrm{D}}{=} S^*$, $d_2(F_n^*, F^*) \le ||Z_n - Z||_2$ for any *particular* choice of $Z_n \stackrel{\mathrm{D}}{=} S_n^*$ and $Z \stackrel{\mathrm{D}}{=} S^*$. Furthermore, if we manage to show that $||Z_n - Z||_2 \to 0$ then certainly $d_2(F_n^*, F^*) \to 0$.

The variables $B_n = \mathbf{1}_{\{R_n \le J_n\}}$ all have the same distribution as a Bernoulli random variable with success probability $q$; this follows from the calculation

$$\mathrm{P}(B_n = 1) = \frac{1}{n} \sum_{j=0}^{n} \sum_{r=1}^{n} \mathrm{P}(\mathbf{1}_{\{r \le j\}} = 1) q^j p^{n-j} \binom{n}{j} = \frac{1}{n} \sum_{j=0}^{n} j q^j p^{n-j} \binom{n}{j} = q.$$

We choose to work with the realization

$$Q \stackrel{\mathrm{D}}{=} q B_n Q + q \tilde{B}_n \tilde{Q} + pq,$$

which employs the same $B_n$ and $\tilde{B}_n = 1 - B_n$ that appear in the recurrence for $S_n$ (with $Q$ and $\tilde{Q}$ being distributed like $S^*$, and $Q$, $\tilde{Q}$, and $B_n$ being independent).

Define

$$\begin{aligned} b_n &= \mathrm{E}[(S_n - Q)^2] \\ &= \mathrm{E}\left[\left(\left(S^*_{J_n} \frac{J_n}{n} \mathbf{1}_{\{R_n \le J_n\}} + \tilde{S}^*_{n-J_n} \frac{n - J_n}{n} \mathbf{1}_{\{R_n > J_n\}} + \frac{M_n}{n}\right) - (q B_n Q + p \tilde{B}_n \tilde{Q} + pq)\right)^2\right]. \end{aligned}$$

Toward a quick calculation, we replace several quantities with their limits and compensate by correction terms—namely, we replace $M_n/n$ with $pq + O_{\mathcal{L}_1}(1/\sqrt{n})$, according to a known approximation of hypergeometric random variables by normal variates (see Feller (1968, p. 194)) and uniform integrability; likewise, we replace $J_n$ with $qn + O_{\mathcal{L}_1}(\sqrt{n})$ (following from the uniform integrability of binomial random variables and their usual approximation by normal random variates). We can write

$$\begin{aligned} b_n = \mathrm{E}\Bigg[\Bigg(& B_n S^*_{J_n}\left(q + O_{\mathcal{L}_1}\left(\frac{1}{\sqrt{n}}\right)\right) + \tilde{B}_n \tilde{S}^*_{n-J_n}\left(p + O_{\mathcal{L}_1}\left(\frac{1}{\sqrt{n}}\right)\right) \\ &+ \left(pq + O_{\mathcal{L}_1}\left(\frac{1}{\sqrt{n}}\right)\right) - (q B_n Q + p \tilde{B}_n \tilde{Q} + pq)\Bigg)^2\Bigg]. \end{aligned}$$

When we group these terms appropriately, square out, and then take expectations, we get quadratic terms giving rise to a recurrence for $b_n$, and all the other terms are subsumed in a small error:

$$\begin{aligned} b_n &= q^2 \mathrm{E}[B_n (S^*_{J_n} - Q)^2] + p^2 \mathrm{E}[\tilde{B}_n (\tilde{S}^*_{n-J_n} - \tilde{Q})^2] + O\left(\frac{1}{\sqrt{n}}\right) \\ &= \frac{q^2}{n} \sum_{j=0}^{n} j b_j p^{n-j} q^j \binom{n}{j} + \frac{p^2}{n} \sum_{j=0}^{n} b_j p^j q^{n-j} + O\left(\frac{1}{\sqrt{n}}\right). \end{aligned}$$

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

We show from the last recurrence that $b_n$ is bounded by $K/\sqrt{n}$ for some positive constant $K$. We show this by induction on $n$. The $O$ in the recurrence means that the last term in the recurrence is bounded from above by $C/\sqrt{n}$ for some positive constant $C$, and all $n \geq n_0'$ for some $n_0' \geq 1$.

Also, for any fixed $p$ (and, hence, $q$), the function $1 - x(p^{x+2} + q^{x+2})$ approaches 1 (from below) as $x \to \infty$. As a function of $p$, the function $(p^{5/4} + q^{5/4})^2$ is convex and reaches the value 1 at $p = 0$ and $p = 1$; at all intermediate values of $p$ it is strictly less than 1. It then follows that, for large enough $n$, say $n \geq n_0'' \geq 1$,

$$1 - n(p^{n+2} + q^{n+2}) \geq (p^{5/4} + q^{5/4})^2. \tag{12}$$

Take $n_0 = \max\{n_0', n_0''\}$. Note that

$$b_j \leq \max\{b_1, b_2, \ldots, b_{n_0}\} \leq \frac{\max\{b_1, b_2, \ldots, b_{n_0}\}\sqrt{n_0}}{\sqrt{j}} \quad \text{for } j = 1, 2, \ldots, n_0.$$

This guarantees the bound $b_n \leq K/\sqrt{n}$ at $n = 1$, up to $n_0$, if $K > \max\{b_1, b_2, \ldots, b_{n_0}\}\sqrt{n_0}$. We take

$$K > \max\left\{\frac{C}{2p^{5/4}q^{5/4}}, \max\{b_1, b_2, \ldots, b_{n_0}\}\sqrt{n_0}\right\}.$$

Assume that the induction hypothesis holds from 0 up to $n - 1 \geq n_0$. Then

$$b_n(1 - n(p^{n+2} + q^{n+2})) \leq \frac{q^2}{n}\sum_{j=1}^{n-1} j\frac{K}{\sqrt{j}}p^{n-j}q^j\binom{n}{j} + \frac{p^2}{n}\sum_{j=1}^{n-1} j\frac{K}{\sqrt{j}}p^j q^{n-j}\binom{n}{j} + \frac{C}{\sqrt{n}}.$$

Note that the remaining sums have an interpretation as averages of square roots of binomial random variables, namely,

$$\mathrm{E}[\sqrt{J_n}] = \sum_{j=0}^{n}\sqrt{j}\,p^{n-j}q^j\binom{n}{j},$$

and, by Jensen's inequality,

$$\sum_{j=1}^{n-1}\sqrt{j}\,p^{n-j}q^j\binom{n}{j} \leq \sqrt{\mathrm{E}[J_n]} = \sqrt{qn}.$$

Likewise,

$$\sum_{j=1}^{n-1}\sqrt{j}\,p^j q^{n-j}\binom{n}{j} = \mathrm{E}[\sqrt{n - J_n}] \leq \sqrt{\mathrm{E}[n - J_n]} = \sqrt{pn}.$$

It then follows from (12) that, for all $n \geq 1$,

$$b_n \leq \frac{K(p^{5/2} + q^{5/2}) + 2Kp^{5/4}q^{5/4}}{\sqrt{n}(1 - n(p^{n+2} + q^{n+2}))} \leq \frac{K(p^{5/4} + q^{5/4})^2}{\sqrt{n}(1 - n(p^{n+2} + q^{n+2}))} \leq \frac{K}{\sqrt{n}},$$

completing the induction.

This induction demonstrates that

$$d_2^2(S_n^*, S^*) \leq b_n \to 0 \quad \text{as } n \to \infty,$$

which is sufficient to establish convergence of $S_n^*$ to $S^*$ in distribution and in the first two moments.

https://doi.org/10.1239/aap/1308662491 Published online by Cambridge University Press

## Acknowledgements

The second author is grateful to the Max-Planck-Institut für Informatik for an invitation that made this collaboration possible. The authors also thank the anonymous referee for constructive comments that improved the presentation.

## References

Barbour, A. D., Holst, L. and Janson, S. (1992). *Poisson Approximation*. Oxford University Press.

Chung, K. L. (1974). *A Course in Probability Theory*, 2dn edn. Academic Press, New York.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001). *Introduction to Algorithms*, 2nd edn. MIT Press, Cambridge, MA.

De La Briandais, R. (1959). File searching using variable length keys. In *Proc. Western Joint Computer Conf.*, AFIPS, San Francisco, CA, pp. 295–298.

Fagin, R., Nievergelt, J., Pippenger, N. and Strong, H. (1979). Extendible hashing—a fast access method for dynamic files. *ACM Trans. Database Systems* **4,** 315–344.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd edn. John Wiley, New York.

Fill, J. A. and Janson, S. (2004). The number of bit comparisons used by Quicksort: an average-case analysis. In *Proc. ACM-SIAM Symp. Discrete Algorithms*, Association for Computing Machinery, New York, pp. 300–307.

Fill, J. A. and Nakama, T. (2008). Analysis of the expected number of bit comparisons required by Quickselect. In *Proc. 10th Workshop Algorithm Engineering and Experiments and the 5th Workshop on Analytic Algorithmics and Combinatorics*, Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 249–256.

Flajolet, P., Gourdon, X., and Dumas, P. (1995). Mellin transforms and asymptotics: harmonic sums. *Theoret. Comput. Sci.* **144,** 3–58.

Fredkin, E. (1960). Trie memory. *Commun. ACM* **3,** 490–499.

Hoare, C. A. R. (1962). Quicksort. *Comput. J.* **5,** 10–15.

Hollerith, H. (1894). The electric tabulating machine. *J. Roy. Statist. Soc.* **57,** 678–682.

Hwang, H.-K. and Tsai, T.-H. (2002). Quickselect and the Dickman function. *Combinatorics Prob. Comput.* **11,** 353–371.

Jacquet, P. and Szpankowski, W. (1998). Analytical depoissonization and its applications. *Theoret. Comput. Sci.* **201,** 1–62.

Knape, M. and Neininger, R. (2008). Approximating perpetuities. *Methodology Comput. Appl. Prob.* **10,** 507–529.

Knuth, D. E. (1998). *The Art of Computer Programming*, Vol. 3, 2nd edn. Addison-Wesley, Reading, MA.

Lent, J. and Mahmoud, H. M. (1996). Average-case analysis of multiple Quickselect: an algorithm for finding order statistics. *Statist. Prob. Lett.* **28,** 299–310.

Mahmoud, H. M. (2000). *Sorting*. Wiley-Interscience, New York.

Mahmoud, H. M. (2010). Distributional analysis of swaps in Quick Select. *Theoret. Comput. Sci.* **411,** 1763–1769.

Mahmoud, H. M. and Ward, M. D. (2008). Average-case analysis of cousins in $m$-ary tries. *J. Appl. Prob.* **45,** 888–900.

Mahmoud, H. M., Modarres, R. and Smythe, R. T. (1995). Analysis of QUICKSELECT: an algorithm for order statistics. *RAIRO Inf. Théor. Appl.* **29,** 255–276.

Mahmoud, H., Flajolet, P., Jacquet, P. and Régnier, M. (2000). Analytic variations on bucket selection and sorting. *Acta Informatica* **36,** 735–760.

Martínez, C. and Prodinger, H. (2009). Moves and displacements of particular elements in quicksort. *Theoret. Comput. Sci.* **410,** 2279–2284.

Mehlhorn, K. (1984). *Data Structures and Algorithms*. Springer, Berlin.

Prodinger, H. (1995). Multiple Quickselect–Hoare's Find algorithm for several elements. *Inform. Process. Lett.* **56,** 123–129.

Sedgewick, R. (1977). The analysis of Quicksort programs. *Acta Informatica* **7,** 327–355.

Sedgewick, R. (1978). Implementing quicksort programs. *Commun. ACM* **21,** 847–857.

Sedgewick, R. (1980). *Quicksort*. Garland, New York.

Sedgewick, R. (1998). *Algorithms in C, Parts 1–4*, 3nd edn. Addison-Wesley, Reading, MA.

Stuart, A. and Ord, J. (1987). *Kendall's Advanced Theory of Statistics*, Vol. 1, 5th edn. Oxford University Press, New York.

Vallée, B., Clément, J., Fill, J. A. and Flajolet, P. (2009). The number of symbol comparisons in quicksort and quickselect. In *Automata, Languages and Programming* (Lecture Notes Comput. Sci. **5555**), eds S. Albers *et al.*, Springer, Berlin, pp. 750–763.