The Shifting Sands Algorithm

Andrew McGregor* Paul Valiant†

Abstract

We resolve the problem of small-space approximate selection in random-order streams. Specifically, we present an algorithm that reads the n elements of a set in random order and returns an element whose rank differs from the true median by at most $n^{1/3+o(1)}$ while storing a constant number of elements and counters at any one time. This is optimal: it was previously shown that achieving better accuracy required poly(n) memory. However, it was conjectured that the lower bound was not tight and that a previous algorithm achieving an $n^{1/2+o(1)}$ approximation was optimal. We therefore consider the new result a surprising resolution to a natural and basic question.

1 Introduction

The Problem. This paper considers a simple computational problem about a random process. The elements of a set S of n integers are presented in random order, i.e., at time step t you are shown a value $a_t \in S$ that is chosen uniformly at random from the set of remaining elements, $S \setminus \{a_1, a_2, \ldots, a_{t-1}\}$. You wish to determine the median of S. Unfortunately you only have limited memory to remember the values you have observed. For example, you can only afford to maintain a constant number of elements and a few additional counters. Given this severe constraint, how well can you estimate the median of S? Can you return a n^γ-approximate median with high probability, i.e., an element whose rank is in the range $n/2 \pm n^\gamma$, for some small value of γ?

Prior Work. In the late seventies, Munro and Paterson [10] asked the related question of how much memory is required to return the exact median of S. They presented a simple algorithm that maintains a contiguous set of elements and they showed, via a random-walk analysis, that maintaining two counters and $O(\sqrt{n})$ elements from the stream was sufficient. It can be shown that their algorithm is essentially optimal [9, 10].

The approximate version of the problem has a long history in the data streams community. However, the vast majority of this work assumes that the elements of S are presented in an order that is chosen adversarially. The state of the art results in this setting are due to Greenwald and Khanna [6] and Shrivastava et al. [12]. Ignoring logarithmic factors, the basic trade-off is that $\Theta(n^{1-\gamma})$ space is necessary and sufficient for finding an n^γ-approximate median. Naturally, if the elements are read in random order, we should hope to need less space to achieve the same accuracy with high probability.

The question in the random order setting was first considered by Guha and McGregor [9]. They showed that less space was indeed sufficient. Specifically, they showed that:

- For $\gamma = \frac{1}{2} + o(1)$, it suffices to use $O(\log n)$ space[1].
- For $\gamma < \frac{1}{3}$, it is necessary to use $\Omega(n^{1/2-3\gamma/2})$ space.

Clearly there is a gap: if the lower bound is tight, there should exist an algorithm that finds an $O(n^{1/3})$-approximate median using only polylogarithmic space. Resolving whether or not this is the case is the main question addressed in this paper. Does such an algorithm exist or can the lower bound be improved?

Other related work includes finding the exact median given multiple-passes [1, 2]. It has been shown that for polylog(n)-space algorithms, $\Theta(\log n / \log\log n)$ passes is necessary and sufficient when the stream is adversarially ordered, whereas only $\Theta(\log\log n)$ passes are required with high probability given a random ordering. Other problems considered in the random-order stream model include frequency moments [1, 7], distinct elements [13], minimum enclosing balls [11], frequent items [4], robust statistics [3], and histogram construction [8]. The motivation for studying random order streams is twofold. First, when the stream isn't adversarially ordered it makes sense to design algorithms that take advantage of this, e.g., when processing a sequence of i.i.d. samples from an unknown distribution or when the stream order can explicitly be randomized (see, for example, the "backing sample" architecture proposed by Gibbons et al. [5]). Second, space lower bounds in the random-order setting (rather than the adversarial setting) are a more powerful indication that a problem is hard in practice.

*University of Massachusetts, Amherst. Supported by NSF CAREER Award CCF-0953754.

†University of California, Berkeley.

[1] We will assume throughout the paper that the integers in the stream have magnitude poly(n).

What's the Answer? There are numerous pieces of anecdotal evidence to suggest that it is the lower bound that needs to be improved and that the upper bound of $n^{1/2+o(1)}$-approximation is optimal. For example, suppose that rather than trying to estimate the median of a set, we are trying to estimate the median of a distribution μ on $\mathbb{R}$ of bounded variance, e.g., we want to return a value $x \in \mathbb{R}$ such that

$$1/2 - \epsilon \leq \int_{-\infty}^{x} \mu(x)dx \leq 1/2 + \epsilon.$$

It is straightforward to show that in general $\epsilon = \Theta(\frac{1}{\sqrt{n}})$ is the best one can do. The models of random sampling from a distribution versus receiving elements from a randomly ordered stream seem similar enough, that it seems intuitively unlikely to do much better on one than on the other *using only constant memory.*

More concretely, suppose we have observed the first half of the stream and have computed a $n^{1/2-\epsilon}$-approximation of the median, m_1, of the data seen thus far. It is straightforward to show that m_1 is only an $O(n^{1/2})$-approximation to the median, m, of the entire stream. Hence, it may not seem possible to leverage the fact we have a good approximation to m_1 since m_1 is itself only a weak approximation to m. Consequently it may seem that we can't improve upon $O(n^{1/2})$ accuracy.

However, it turns out that this intuition is totally wrong. In this paper we show that it is possible to $n^{1/3+o(1)}$-approximate the median in logarithmic space. Specifically, we present an algorithm that maintains only a constant number of elements from the stream at any one time, in addition to a constant number of counters. We consider this a surprising result. Furthermore, while the algorithm is rather subtle, the analysis requires nothing beyond standard tail inequalities.

2 Preliminaries and Notation

Throughout the paper we use the following notation. For numbers a, b, c we write $a = b \pm c$ to denote $b - c \leq a \leq b + c$.

DEFINITION 2.1. *Given a set $X \subset \mathbb{Z}$ and a number p, the* rank *of p with respect to X, denoted $R_X(p)$, equals the number of elements of X that are less than or equal to p. We will also find it convenient to denote the difference between the ranks of two elements a and b by*

$$R_X(a, b) := R_X(a) - R_X(b) \ .$$

We then say an element p is a t-approximate median of the set S if $R_S(p) = n/2 \pm t$.

Distinct Values. We next observe that we may assume without loss of generality that the stream contains distinct values. This can easily be achieved with probability at least $1 - \delta$ by attaching a secondary value $y_i \in_R [n^2\delta^{-1}]$ to each item x_i in the stream. We say $(x_i, y_i) < (x_j, y_j)$ iff $x_i < x_j$ or ($x_i = x_j$ and $y_i < y_j$). Note that breaking the ties arbitrarily results in a stream whose order is not random.

Conditioning the Ordering. Rather than analyze some rather intricate dependencies that could arise in the analysis of our algorithm, we will instead condition the uniformly random ordering of the stream on a specific event $\mathcal{A}$ which we now define.

$\mathcal{A}$: *The "Always a Good Estimate" Event.* For every two segments X and Y of the stream S and any two elements $a, b \in S$,

$$R_X(b, a) = (R_Y(b, a) \pm (c_1 \log n)\sqrt{1 + R_Y(b, a)}) \cdot \frac{|X|}{|Y|}$$

where c_1 is some suitably large constant.

The fact that $\mathcal{A}$ is a high probability event, i.e., $\Pr[\mathcal{A}] \geq 1 - 1/n^2$, follows from an application of the Hoeffding-Chernoff bound and n^6 applications of the union bound corresponding to the n^2 possible choices for a, b and the n^2 choices for each of X and Y. *Every statement in the rest of the paper will be conditioned on the event $\mathcal{A}$.*

3 Approximate Median Finding

In the introduction we questioned the value of finding a good approximation to the median of the first half of the stream, m_1, based on the observation that m_1 is itself only an $O(n^{1/2})$ approximation to the final median. Explicitly, given a randomly ordered sequence S with n elements, and letting m_1 be the median of the first half of the sequence, then $E[|R_S(m_1) - n/2|] = \Theta(n^{1/2})$. Further, it can be argued that an approximation to m_1 is essentially the only useful information that can be gleaned from the first half of the sequence—any other statistics are arguably either irrelevant, or may be adequately approximated instead from the second half of the sequence. This seems to lead to an impasse: the (approximate) median of the first half of the sequence is the only useful statistic to ask of the first half of the sequence, but the median of the first half of the sequence seems to yield errors on the order of $n^{1/2}$ when leveraged on the rest of the sequence.

However, this observation is only half (in an almost literal sense) of the story. Specifically, let m_2 be the median of the second half of the stream. Considering m_1 and m_2 separately yields no fruits: the rank of m_1 will typically differ from $n/2$ by some constant multiple

of $n^{1/2}$, and by symmetry, the corresponding statement holds for m_2: $E[|R_S(m_2) - n/2|] = \Theta(n^{1/2})$. However, adding the two ranks together yields a surprising observation that serves to motivate the rest of this paper:

> *The sum of the two ranks, $R_S(m_1) + R_S(m_2)$, has expectation $n+1$ and its expected difference from this expectation is* not $\Theta(n^{1/2})$ *as we might naively suspect, but instead the much smaller $\Theta(n^{1/4})$.*

One way of looking at this observation is that if the first half of the sequence is "bigger than expected," then the second half of the sequence has to be correspondingly "smaller than expected," and, further, that in some sense the median of the whole sequence lies almost exactly halfway in between the medians of the first and second half respectively. The crucial importance of this observation is that it relates the median of the first half of the sequence to some global rank properties of the sequence, to within error $\Theta(n^{1/4})$ instead of the more typical $\Theta(n^{1/2})$. We prove this observation now and then detail how to leverage it.

LEMMA 3.1. *Let S_1, S_2 denote the first and second half of the stream S and let m_1, m_2 be the medians of S_1, S_2. Then $R_S(m_1) + R_S(m_2) = n \pm O(n^{1/4} \cdot \text{polylog}\, n)$*

Proof. Let m be the median of $S_1 \cup S_2$. Given event $\mathcal{A}$, $R_{S_1}(m) = n/4 + O(n^{1/4} \cdot \text{polylog}\, n)$ and therefore

$$R_S(m_1, m) = 2R_{S_1}(m_1, m) \pm O(n^{1/4} \cdot \text{polylog}\, n)$$

and correspondingly for S_2, m_2. Adding these equations and rearranging yields the desired result:

$$\begin{aligned} & R_S(m_1) + R_S(m_2) \\ = \;& 2R_S(m) + 2R_{S_1}(m_1, m) + 2R_{S_2}(m_2, m) \\ & \quad \pm O(n^{1/4} \cdot \text{polylog}\, n) \\ = \;& 2R_{S_1}(m_1) + 2R_{S_2}(m_2) \pm O(n^{1/4} \cdot \text{polylog}\, n) \\ = \;& n \pm O(n^{1/4} \cdot \text{polylog}\, n)\ . \end{aligned}$$

where the second equality follows because $R_{S_1}(m) + R_{S_2}(m) = R_S(m)$ and the last equality follows because $R_{S_1}(m_1) = (n/2+1)/2$ and $R_{S_2}(m_2) = (n/2+1)/2$. □

It follows from the above lemma that we should aim to return the median of the set of elements in $S_1 \cup S_2$ whose values lie between m_1 and m_2. Unfortunately, once we are halfway through the stream, we don't know m_2 and only elements in S_2 remain. However, by appealing to the conditioning on $\mathcal{A}$ we can argue that it is sufficient to approximate this median based only on S_2. Specifically, event $\mathcal{A}$ implies that

$$\left(R_{S_2}(m_1) + R_{S_2}(m_2)\right)/2 = R_{S_2}(m) \pm O(\sqrt[4]{n} \cdot \log n)\ .$$

Since $R_{S_2}(m_2) = (n/2+1)/2$ by definition of m_2, this leads to the tantalizing fact that the median of the whole sequence is (to within $O(\sqrt[4]{n})$) the element of the second half of the sequence whose rank within the second half of the sequence is $R_{S_2}(m_1)/2 + n/8$. If we could find an element of the second half of the sequence that approximates this well, we may thus approximate the median. The heart of this paper is a proof of the following lemma. (The lemma should be considered as applying to the second half of the sequence, as discussed so far, and hence $n/2$ in the above discussion becomes n in the lemma below.)

LEMMA 3.2. (SHIFTING SANDS ALGORITHM) *Given a randomly ordered sequence S of length n, and a value p with*

$$R_S(p) = n/2 \pm O(\sqrt{n} \cdot \log n)\ ,$$

there is a streaming algorithm that, without advance knowledge of $R_S(p)$, uses constant memory and returns an element $q \in S$ such that

$$R_S(q) = n/4 + R_S(p)/2 \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n)\ .$$

We defer the proof until the next section. The basic idea is as follows: while we process the elements in the second half of the sequence, the algorithm in Lemma 3.2 *simultaneously* evaluates the rank of m_1 relative to the elements in the second half as they arrive—this yields an increasingly accurate estimate of $R_{S_2}(m_1)$—and determines a sequence of candidates whose rank relative to the second half of the sequence we expect to approach $n/4$ plus our current estimate of the rank of m_1.

We call this the *Shifting Sands Algorithm* because the target of our algorithm does not lie passive, but rather is constantly changing with each new sequence element we examine: the ground is shifting under us. And it is a race between the shifting of the target, and our ability to track it with increasingly better candidates. We cannot simply wait until most of the "sand has shifted" to start finding candidates because then we might have too much distance to cover, and not enough samples remaining in the sequence to let us accurately propose and evaluate candidates to make up the distance. There is a delicate balance between the rate at which our target shifts, and the rate at which we must constantly pursue it. This balance yields the compromise error of $\sqrt[3]{n}$.

The following corollary shows that by applying the algorithm in Lemma 3.2 recursively we can find an approximate median to the desired accuracy.

COROLLARY 3.1. *Given a randomly ordered sequence S of length n there is a streaming algorithm that uses constant memory and returns an element of S of rank $n/2 \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n)$.*

Proof. The algorithm is recursive: first use it to find the approximate median of the first half of the sequence, then apply the algorithm of Lemma 3.2 and return its answer. Thus the total memory used is just that needed by the algorithm of Lemma 3.2 since the recursive portion happens first and thus does not incur any memory overhead.

Denote the first half of the sequence by S_1 and the second half by S_2, and assume the recursive invocation of the algorithm returns a number m_1 of rank $R_{S_1}(m_1) = n/4 \pm e$ for some bound $e = o(\sqrt{n})$. We note that $R_{S_1}(m_1) + R_{S_2}(m_1) = R_S(m_1)$. Let m_2 be the median of S_2. Conditioned on event $\mathcal{A}$, $R_{S_2}(m_1) = n/4 \pm O(\sqrt{n} \cdot \log n)$. Therefore, using $p = m_1$, the algorithm of Lemma 3.2 picks an element q in S_2 such that

$$R_{S_2}(q) = R_{S_2}(m_1)/2 + R_{S_2}(m_2)/2 \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n) \ .$$

Conditioned on event $\mathcal{A}$, q will also satisfy

$$R_S(q) = R_S(m_1)/2 + R_S(m_2)/2 \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n) \ .$$

This follows because $|R_S(m_1, m_2)| = O(\sqrt{n} \cdot \log n)$. Further, Lemma 3.1 and event $\mathcal{A}$ implies that

$$R_S(m_2) + R_S(m_1) = n \pm 2e \pm O(\sqrt[4]{n} \cdot \log n) \ ,$$

and therefore the average of the ranks of m_1, m_2 is $n/2 \pm e \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n)$.

Thus if the invocation of the algorithm on sequences of length $n/2$ guarantees error e, then for sequences of length n we have error $e + O(\sqrt[3]{n} \cdot \text{polylog}\, n)$. Solving the recursion yields that in general, this algorithm has error $O(\sqrt[3]{n} \cdot \text{polylog}\, n)$. □

3.1 Proof of Lemma 3.2. In this section we present and analyze the Shifting Sands Algorithm. The analysis of this algorithm establishes Lemma 3.2. We first outline the main steps of algorithm in Section 3.1.1. In Section 3.1.2, we describe the main sub-routine of the algorithm and in Section 3.1.3, we complete all the details and prove the necessary properties of the algorithm.

3.1.1 Outline. We divide the stream into t segments

$$S = \langle S_t \mid S_{t-1} \mid \ \ldots \ \mid S_1 \rangle \ ,$$

where $t = \log_2 n^{1/6}$ and $|S_i| = 0.75 \cdot 4^i \cdot n^{2/3}$ with S_t expanded slightly such that the segments over the entire stream. Before we process each S_j we will have defined the following three quantities:

$$\begin{aligned} g_j &: \text{a current ``proposal''} \\ s_j &: \text{an estimate of } R_S(g_j) \\ r_j &: \text{an estimate of } R_S(p) \end{aligned}$$

where we initially set $g_t = p$ and $s_t = r_t = n/2$. The proposal g_{t_0} for some $t_0 = \Theta(\log \log n)$ will be the value q returned by the algorithm.

As we process S_j we construct g_{j-1}, s_{j-1}, and r_{j-1}. The new estimate of $R_S(p)$ is simply

$$r_{j-1} := \frac{|S|}{|S_t| + |S_{t-1}| + \ldots + |S_j|} \cdot R_{S_t \cup S_{t-1} \cup \ldots \cup S_j}(p) \ .$$

In the next section we will describe how we generate the new proposal g_{j-1} and an estimate δ_{j-1} for the difference between the ranks of g_{j-1} and g_j, i.e., $R_S(g_{j-1}, g_j)$. Our estimate for $R_S(g_{j-1})$ is then defined as

$$s_{j-1} := r_{j-1} + \delta_{t-1} + \delta_{t-2} + \ldots + \delta_{j-1} \ ,$$

whereas the true rank is $R_S(p) + R_S(g_{t-1}, g_t) + \ldots + R_S(g_{j-1}, g_j)$.

The goal of the proposal generation mechanism is to ensure that,

$$s_{j-1} = \frac{n}{4} + \frac{r_{j-1}}{2} \pm 3 \cdot \tau_{j-1}$$

where $\tau_{j-1} := O(2^{j-1} \cdot \log n \cdot \sqrt[3]{n})$. In other words, our estimate for the rank of the new proposal should be approximate halfway between $n/2$ and our current estimate of $R_S(p)$. Furthermore, the extent of this approximation should halve with each segment. Then, for $j = t_0$ this would guarantee that

$$s_{t_0} = \frac{n}{4} + \frac{r_{t_0}}{2} \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n) \ .$$

If we can additionally guarantee that

$$r_{t_0} = R_S(p) \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n)$$

and

$$s_{t_0} = R_S(g_{t_0}) \pm O(\sqrt[3]{n} \cdot \text{polylog}\, n)$$

then we have the desired result with $q = g_{t_0}$.

3.1.2 Finding New Proposals. Ideally we would want the next proposal to have rank $n/4 + R_S(p)/2$ but since we do not know $R_S(p)$ we aim for $n/4 + r_j/2$. Equivalently, we want to find an element whose rank differs from s_j by

$$\Delta_{j-1} := n/4 + r_j/2 - s_j$$

To do this, we further partition S_j into sub-segments:

$$S_j = \langle G_{1,j} \mid E_{1,j} \mid G_{2,j} \mid E_{2,j} \mid \ \ldots \ \mid G_{r,j} \mid E_{r,j} \mid \text{suffix} \rangle$$

where $r = O(\text{polylog}\, n)$ and

$$\begin{aligned} |G_{i,j}| &= O(2^{-j} \cdot n^{1/3} \cdot \text{polylog}\, n) \quad \text{and} \\ |E_{i,j}| &= O(3^j \cdot n^{2/3} \cdot \text{polylog}\, n) \ . \end{aligned}$$

This is well-defined since

$$\sum_{i=1}^{r} (|E_{i,j}| + |G_{i,j}|) = O(3^j \cdot n^{2/3} \cdot \text{polylog}\, n)$$

is less than $0.75 \cdot 4^j \cdot n^{2/3} = |S_j|$ for $j > t_0$ if t_0 is a suitably large multiple of $\log \log n$.

We determine g_{j-1} by generating a sequence of possible candidates until we find one whose rank appears to be suitable. The ith candidate is generated from $G_{i,j}$ and we use the elements in $E_{i,j}$ to estimate the rank. Specifically, we repeat the following two-step process:

1. *Generate Candidate:* Let $G'_{\Delta_{j-1}}$ be the first $\frac{n}{2|\Delta_{j-1}|}$ elements of $G_{i,j}$. Then define the candidate:

$$\alpha = \begin{cases} \min(x > g_j : x \in G'_{\Delta_{j-1}}) & \text{if } \Delta_{j-1} > 0 \\ \max(x < g_j : x \in G'_{\Delta_{j-1}}) & \text{if } \Delta_{j-1} < 0 \end{cases}$$

   if $n/(2|\Delta_{j-1}|) \geq |G_{i,j}|$ we set $\alpha = g_j$.

2. *Estimate Relative Rank:* Estimate $R_S(\alpha, g_j)$ by

$$\beta = R_{E_{i,j}}(\alpha, g_j) \cdot \frac{n}{|E_{i,j}|} \ .$$

   If $|\beta - \Delta_{j-1}| \leq \frac{\tau_{j-1}}{2}$, set $g_{j-1} \leftarrow \alpha$ and $\delta_{j-1} \leftarrow \beta$.

To avoid dealing with the dependencies that arise between different phrases, we define a single event $\mathcal{B}$ such that if we condition on $\mathcal{A}$ and $\mathcal{B}$, we ensure this process generates a suitable g_{j-1} and δ_{j-1}.

$\mathcal{B}$: *The "Always a Good Candidate" Event.* For every $(G_{1,j}, \ldots, G_{r,j})$, and every $g \in S, |\Delta| \geq n/(2|G_{i,j}|)$, there exists α such that $R_S(\alpha, g) = \Delta \pm \Delta/8$ and for some $i \in [r]$

$$\alpha = \begin{cases} \min(x > g : x \in G'_{\Delta}) & \text{if } \Delta > 0 \\ \max(x < g : x \in G'_{\Delta}) & \text{if } \Delta < 0 \end{cases}$$

where G'_{Δ} is the first $n/(2|\Delta|)$ elements of $G_{i,j}$.

To argue that $\mathcal{B}$ is a high probability event note that there are at most $O(\log n)$ choices for $j \in [t]$ and n^2 choices for g and Δ. It therefore suffices to prove for any specific g, Δ, and j that the event holds with high probability since the bound will then follow by using $O(n^2 \log n)$ applications of the union bound. The analysis for both cases is similar so assume $\Delta > 0$. Consider a specific $G_{i,j}$ and let G'_{Δ} be the first $n/(2\Delta)$ elements in $G_{i,j}$. Let A be the event that no element $y \in G'_{\Delta}$ satisfies $0 < R_S(y, g) \leq \Delta$. This happens with probability at least $1/2$ by Markov's inequality. Let B be the event that an element $y \in G'_{\Delta}$ satisfies $\Delta < R_S(y, g) \leq \Delta + \Delta/8$. This happens with probability at least $1 - (1 - \Delta/(8n))^{n/2\Delta} \geq 1 - e^{-1/16}$. Since A and B are positively correlated, the event $A \cap B$ has probability at least $0.5 \cdot (1 - e^{-1/16})$ and therefore α is suitable with constant probability. Hence, repeating over the $O(\text{polylog}\, n)$ choices of $i \in [r]$ gives that $\mathcal{B}$ occurs with probability at least $1 - 1/n$.

3.1.3 Analysis. The first claim establishes that the error in our estimate of $R_S(p)$ halves with each stage.

Claim 1. *For all stages j, $|r_j - R_S(p)| \leq \tau_j$.*

For $j \neq t$ this follows directly from the conditioning on event $\mathcal{A}$ since there are $O(4^j \cdot n^{2/3}) \leq \tau_j^2$ unseen elements at the start of stage j; the case $j = t$ follows directly from the assumptions in the setup of the algorithm.

We next establish that for each $j \in \{t, t-1, t-2, \ldots, t_0\}$,

$$s_j = n/4 + r_j/2 \pm 3\tau_j \tag{3.1}$$

or equivalently $|\Delta_{j-1}| \leq 3\tau_j$. We prove this by induction on decreasing j. For $j = t$, we have $s_t = r_t = n/2 = n/4 + n/4$ and hence the base case is satisfied. For the induction hypothesis, assume Eq. (3.1) holds for j.

Claim 2. *At the end of the jth phase, the algorithm has identified an element g_{j-1} such that*

$$s_{j-1} = s_j + \Delta_{j-1} \pm \tau_{j-1} \ .$$

together with an estimate δ_{j-1} of $R_S(g_{j-1}, g_j)$ such that

$$|R_S(g_{j-1}, g_j) - \delta_{j-1}| \leq O(\sqrt[3]{n}) \ .$$

Proof. First note that $|\beta - \Delta_{j-1}| \leq \tau_{j-1}/2$ implies that

$$R_{E_{i,j}}(\alpha, g_j) \leq \frac{|E_{i,j}|}{n} \cdot \left(\frac{\tau_{j-1}}{2} + \Delta_{j-1}\right) \leq \frac{4\tau_{j-1} \cdot |E_{i,j}|}{n}$$

where the last inequality follows from the induction hypothesis. Therefore, given the conditioning on event $\mathcal{A}$, we deduce that the error in our estimate of $R_S(\alpha, g_j)$ is less than $\tau_{j-1}/2$ since:

$$O\left(\sqrt{\frac{\tau_{j-1} \cdot |E_{i,j}|}{n}} \cdot \frac{n}{|E_{i,j}|}\right) = O\left(\sqrt{\frac{\tau_{j-1} \cdot n}{|E_{i,j}|}}\right) \leq O(\sqrt[3]{n})$$

where the last step follows by substituting the values for $|E_{i,j}|$ and τ_{j-1}. Therefore, if we ever accept an α we know that

$$R_S(\alpha, g_j) = \Delta_{j-1} \pm (\tau_{j-1}/2 + \tau_{j-1}/2) = \Delta_{j-1} \pm \tau_{j-1}$$

and that $|R_S(g_{j-1}, g_j) - \delta_{j-1}| \leq \tau_{j-1}/2$. Conditioned on event $\mathcal{B}$, there exists i such that the candidate α returned from $G_{i,j}$ satisfies

$$R_S(\alpha, g_j) = \Delta_{j-1} \pm \Delta_{j-1}/8 = \Delta_{j-1} \pm 3\tau_j/8$$

by the induction hypothesis. Conditioned on the event $\mathcal{A}$, for this α, β satisfies $|\beta - \Delta_{j-1}| \leq \tau_{j-1}/2$ and hence at least one α gets accepted. □

Therefore, at the end of stage j we have found a g_{j-1} such that,

$$\begin{aligned} s_{j-1} &= s_j + \Delta_{j-1} \pm \tau_{j-1} \\ &= n/4 + r_j/2 \pm \tau_{j-1} \\ &= n/4 + r_{j-1}/2 \pm 3 \cdot \tau_{j-1} \end{aligned}$$

where the last equality follows from applying the triangle inequality to Claim 1. We can therefore conclude from the induction that,

$$s_{t_0} = n/4 + r_{t_0}/2 \pm O(\sqrt[3]{n} \cdot \operatorname{polylog} n) \ . \tag{3.2}$$

Appealing to Claim 1 with $t_0 = O(\log \log n)$, we deduce

$$s_{t_0} = n/4 + R_S(p)/2 \pm O(\sqrt[3]{n} \cdot \operatorname{polylog} n) \ .$$

Combining this with the following claim completes the proof of Lemma 3.2.

Claim 3. $s_{t_0} = R_S(g_{t_0}) \pm O(\sqrt[3]{n} \cdot \operatorname{polylog} n)$.

Proof. The claim follows from Claims 1 and 2 because

$$|s_{t_0} - R_S(g_{t_0})| \leq |r_{t_0} - R_S(p)| + \sum_{j=t_0}^{t-1} |R_S(g_j, g_{j+1}) - \delta_j|.$$

References

[1] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *STOC*, pages 641–650, 2008.

[2] A. Chakrabarti, T. S. Jayram, and M. Patrascu. Tight lower bounds for selection in randomly ordered streams. In *SODA*, pages 720–729, 2008.

[3] S. Chien, K. Ligett, and A. McGregor. Space-efficient estimation of robust statistics and distribution testing. In *ICS*, pages 251–265, 2010.

[4] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*, pages 348–360, 2002.

[5] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, 2002.

[6] M. Greenwald and S. Khanna. Efficient online computation of quantile summaries. In *ACM International Conference on Management of Data*, pages 58–66, 2001.

[7] S. Guha and Z. Huang. Revisiting the direct sum theorem and space lower bounds in random order streams. In *ICALP (1)*, pages 513–524, 2009.

[8] S. Guha and A. McGregor. Space-efficient sampling. In *AISTATS*, pages 169–176, 2007.

[9] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

[10] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[11] P. Rai, H. Daumé III, and S. Venkatasubramanian. Streamed learning: One-pass SVMs. In *IJCAI*, pages 1211–1216, 2009.

[12] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys*, pages 239–249, 2004.

[13] D. P. Woodruff. The average-case complexity of counting distinct elements. In *ICDT*, pages 284–295, 2009.