Comparison-Based Time–Space Lower Bounds for Selection*

Timothy M. Chan†

March 10, 2009

Abstract

We establish the first nontrivial lower bounds on time-space tradeoffs for the selection problem. We prove that any comparison-based randomized algorithm for finding the median requires $\Omega(n \log\log_S n)$ expected time in the RAM model (or more generally in the comparison branching program model), if we have S bits of extra space besides the read-only input array. This bound is tight for all $S \gg \log n$, and remains true even if the array is given in a random order. Our result thus answers a 16-year-old question of Munro and Raman, and also complements recent lower bounds that are restricted to sequential access, as in the multi-pass streaming model [Chakrabarti *et al.*, SODA 2008].

We also prove that any comparison-based, deterministic, multi-pass streaming algorithm for finding the median requires $\Omega(n \log^*(n/s) + n \log_s n)$ worst-case time (in scanning plus comparisons), if we have s cells of space. This bound is also tight for all $s \gg \log^2 n$. We get deterministic lower bounds for I/O-efficient algorithms as well.

The proofs in this paper are self-contained and do not rely on communication complexity techniques.

1 Introduction

Time–space lower bounds. Lower bounds on time–space tradeoffs have been the subject of extensive study for over four decades [15]. The earliest work of which the present paper is a direct descendant is perhaps Borodin *et al.*'s paper [11], which considered the sorting problem in a comparison RAM model (or more generally in the "comparison branching program" model), where the n input elements reside in a read-only array permitting random access, and the algorithm can inspect the input elements only through comparisons of pairs of elements. They proved that if an algorithm is allowed S bits of extra space, then it must require $\Omega(n^2/S)$ time. A matching upper bound on the comparison RAM, for all $\log n < S < n/\log n$, was only discovered in 1998 by Pagter and Rauhe [27]. Refining a further result by Borodin and Cook [9], Beame [4] proved that the $\Omega(n^2/S)$-time lower bound holds for sorting $O(\log n)$-bit integers, in a very general model of computation that goes beyond comparison-based algorithms (namely, in the "R-way branching program" model). This result applies to randomized algorithms as well, and to the weaker problem of outputting the unique elements of the array in arbitrary order.

Lower bounds for problems with smaller output size are even more challenging. For the problem of simply deciding whether all elements are distinct, Borodin *et al.* [10] obtained the first lower bound

*A preliminary version of this paper appeared in *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms*, 2009.

†School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (tmchan@uwaterloo.ca).
Work supported by NSERC.

of $\Omega(n^{3/2}/\sqrt{S})$ time for any $S \geq \log n$ in the comparison RAM (or comparison branching program) model. This lower bound was subsequently improved by Yao [30] to $\Omega(n^{2-O(1/\sqrt{\log n})}/S)$.

Since these classical results, new communication-complexity-style proof techniques have emerged that led to the the nontrivial time–space lower bounds for decision problems, like element distinctness, for the first time under general computational models (R-way branching programs) beyond comparison-based algorithms [1, 2, 5, 6]. These impressive "modern" techniques pave the way for proving general lower bounds for other problems. However, we remark that time–space lower bounds in the traditional comparison-based setting are still far from completely understood. As one open question, for example, can Yao's lower bound on the element distinctness problem be improved to $\Omega(n^2/S)$ in the comparison RAM model? The present paper will return to these original types of lower bound questions, but for one very basic problem that has (surprisingly) not been previously examined in this context—namely, the selection problem.

Selection. There have actually been several papers exploring upper bounds, i.e., algorithms, for the selection problem with space restrictions. To be consistent with the literature, we use s to denote the number of extra storage cells in addition to the read-only input array, where each cell can hold $O(\log n)$ bits or (a pointer to) an input element; in other words, think of S as $s \log n$. Building on Munro and Paterson's work [25], Frederickson [20] gave a deterministic selection algorithm on the comparison RAM that runs in $O(n \log^* n + n \log_s n)$ time for any $s = \Omega(\log^2 n)$. Though not stated in the paper, it is not hard to reduce the first term to $O(n \log^*(n/s))$; in particular, this meets the standard linear time bound for selection when there is no space restriction, i.e., when $s = \Theta(n)$. The iterated-log term dominates if $s = \Omega(n^\varepsilon)$, for example. Munro and Raman [26] and Raman and Ramnath [28] have also given nontrivial upper bounds for the case of very small s: e.g., $O(n^{1+\varepsilon})$ time for a constant s, and $O(n \log^2 n)$ time for $s = \Theta(\log n)$. (We emphasize that throughout our discussion, the input is read-only; if swaps of array elements are allowed, then there are *in-place* selection algorithms that run in linear time using $O(1)$ extra cells [22].)

If each permutation of the input is equally likely, Munro and Raman [26] have shown a much better upper bound of $O(n \log \log_s n)$ average time for all s exceeding a constant. For example, if $s = \Omega(n^\varepsilon)$, the bound is linear. In the appendix, we observe that the random-order input assumption can be eliminated by using randomization within the algorithm; the resulting randomized algorithm still achieves $O(n \log \log_s n)$ expected time. Munro and Raman wondered (naturally) about the optimality of their results at the end of their paper.

To the author's knowledge, investigations into time–space tradeoff lower bounds for the selection problem have been overlooked in the past. Perhaps a reason is that the dependence on S is less dramatic here than in, say, the sorting problem. On the other hand, one may argue that the tradeoff results for selection are more practically relevant than those for sorting, because we can get close-to-linear running time.

Main result. We prove that any randomized algorithm for the selection problem requires $\Omega(n \log \log_S n)$ expected time for the comparison RAM (or comparison branching programs). Thus, this completely resolves the randomized, comparison-based complexity of the problem, since it matches the above-mentioned upper bound, for all $S \gg \log n$. In particular, the result rules out the possibility of a linear-time (randomized or deterministic) selection algorithm with polylogarithmic (or more generously $2^{O(\log^{1-\varepsilon} n)}$) space. The lower bound holds for randomly ordered input.

We believe the result is interesting, despite its restriction to comparison-based algorithms, because (i) tight time–space lower bounds for the RAM are few and far between, and (ii) historically, the sorting and selection problems, as well as extensions to problems in computational geometry, have been more popularly studied from the comparison-based perspective; e.g., see all the often-cited papers on selection [8, 29, 7, 16, 17, 18].

Multi-pass streaming. Though our main result might seem "old-fashioned" (it could well have been discovered in the 80s or early 90s), the original motivation of the present paper actually stems from more topical concerns, on streaming models for massive data sets. Specifically, we are interested in *multi-pass streaming* algorithms, i.e., algorithms that scan through the entire read-only input, from beginning to end, in a multiple number of passes; the algorithm can at any time keep only a small amount of extra space.

Recently, Chakrabarti *et al.* [12] (see also [13]) answered a 30-year-old question from Munro and Paterson's seminal paper on streaming [25], by proving that the selection problem for a randomly ordered stream requires $\Omega(\log\log_s n)$ passes with high probability. Our proof can be easily modified to yield the same lower bound in the comparison-based setting. So, in a sense, our proof is stronger and demonstrates that the reason for the $\log\log$ behavior is due not to limitation of sequential access but solely to time–space relationship. On the other hand, Chakrabarti *et al.*'s proof, of the communication-complexity variety, applies to the most general model that goes beyond comparison-based algorithms, and holds for $O(\log n)$-bit integer input. (Still, a look at Munro and Paterson's paper reveals that the interest at the time was entirely on comparison-based algorithms.)

Actually, the real starting point of this research is on another question inspired by Munro and Paterson's paper, this time, about their deterministic multi-pass selection algorithm. In streaming, analyses have (rightly) focused on two parameters, the number of passes and space, but what about the total processing time (which includes the time for not just the scans but also other operations such as comparisons)? For example, with $s = \Omega(n^\varepsilon)$ space, Munro and Paterson's algorithm takes only $O(1)$ passes, but on closer inspection requires more than linear amount of work. Frederickson's improvement [20] is basically an $O(p)$-pass algorithm that runs in almost-linear $O(n\log^{(p)}(n/s))$ time, for any p up to a constant times $\log^*(n/s)$, when $s = \Omega(n^\varepsilon)$. Chan and Chen [14], unaware of Frederickson's work, obtained a similar deterministic multi-pass result for the 2-d linear programming problem. Can this almost-linear time bound be reduced to $O(n)$?

We answer this question in the negative: among deterministic multi-pass algorithms, Frederickson's is basically optimal. Although we are unable to prove a better deterministic lower bound for the selection problem in the comparison RAM model, we can get such a lower bound in the multi-pass model: any selection algorithm must either take $\Omega(p)$ passes or perform $\Omega(n\log^{(p)}(n/s))$ comparisons, for any p. Since Munro and Paterson already showed a lower bound of $\Omega(\log_s n)$ passes for comparison-based algorithms when $s = o(n^\varepsilon)$, we can conclude that any multi-pass deterministic selection algorithm must take $\Omega(n\log^*(n/s) + n\log_s n)$ time in scanning or in comparisons. This matches Frederickson's upper bound for all $s \gg \log^2 n$. In particular, the result rules out the possibility of a linear-time, deterministic, multi-pass algorithm even with space up to $s = O(n/\log^{(c)} n)$ for any constant c. A similar result holds for 2-d linear programming but will be reported elsewhere.

Techniques. Standard techniques for proving comparison-based lower bounds, such as decision trees and adversary arguments, are usually not sufficient to get lower bounds for space-restricted RAM algorithms. The elegant proofs by Beame [4] and Borodin *et al.* [10] make use of probabilistic/counting arguments, by considering a uniformly distributed input (and as a byproduct their lower bounds hold for randomized algorithms by Yao's principle). The proof of our randomized lower bound on the RAM is of the same kind, though more technical effort and new ideas are required.

Our deterministic lower bound on the multi-pass model in contrast uses a direct adversary argument to construct a bad input and is short. For this reason, we describe this proof first, in Section 2. This deterministic proof is simple enough to be presentable to undergraduate students, for instance, yet nontrivial enough to yield the iterated-log bound.

In Section 3, we also present an adaptation of this proof for an intermediate model that is more powerful than the multi-pass model and allows random access but only in blocks, as in I/O-efficient algorithms. The iterated-log behavior must still occur in the number of block accesses (I/Os) or in the number of comparisons, provided the block size is sufficiently large (superpolylogarithmic). This proof combines both uniformly distributed input and adversarially chosen input strategies in an interesting way, and is a good preparation of the proof of our final main result on the RAM in Section 4. If one is only interested in the main randomized RAM proof, though, one can go directly to Section 4 after reading Sections 3.1–3.2.

In the rest of the paper, we switch to using N for the size of the given array. We ignore floors and ceilings for simplicity.

2 A Deterministic Iterated-Log Lower Bound in the Multi-Pass Model

As noted, our deterministic multi-pass lower bound has a short, neat proof (which would make a good introduction to adversary arguments). Munro and Paterson [25] already gave a simple, adversary-based proof of an $\Omega(\log_s N)$-pass deterministic lower bound. Our proof is in a similar spirit, but is a little more challenging, taking into account the number of comparisons made:

Theorem 2.1 *Any deterministic comparison-based median finding algorithm in the multi-pass streaming model that uses s storage cells must require $\Omega(p)$ passes or $\Omega(N \log^{(p)}(N/s))$ comparisons in the worst case, for any p.*

Proof: Let $\mathcal{A}$ denote the given algorithm, which takes at most T comparisons. We describe an adversary strategy to construct a bad real-valued input for $\mathcal{A}$. The adversary will simulate $\mathcal{A}$ and, along the way, maintain an interval I_x for each element x, with the property that we could set x to *any* value in I_x without affecting the flow of $\mathcal{A}$ up to this point. Some elements x may be fixed, in which case the interval I_x associated with x degenerates to a single point.

At the beginning of a pass, assume that we have n elements having the same interval I, i.e., each such element x has $I_x = I$, where $n > 8s$. Assume that the remaining elements have been fixed to values outside I, so that the median of the elements in I gives the overall median. Let $|I|$ denote the length of an interval I.

(1) We first simulate the pass of $\mathcal{A}$ until $n/2$ elements in I are encountered. Each time $\mathcal{A}$ performs a comparison, say, between x and y, the adversary resolves the comparison as follows: if the midpoint of I_x is at most the midpoint of I_y, then set I_x to the left half of I_x, set I_y to the right half of I_y and declare "less than"; the other case is symmetric. As soon as $|I_x|$ drops below $|I|/2^{8T/n}$, the adversary fixes x to an arbitrary value in I_x. The adversary also fixes the values of the at most s elements remaining in memory at the end of this step.

(2) Observe that the number of elements that can participate in more than $8T/n$ comparisons is at most $n/4$, since the total number of comparisons is at most T. It follows that the number of elements just fixed is at most $n/4 + s$, so at least $n/4 - s > n/8$ of the $n/2$ elements in I encountered are still unset. For each of these unset elements x, shrink I_x so that it equals one of the $2^{8T/n}$ subintervals of I of length $|I|/2^{8T/n}$. Pick the subinterval I' that is the most popular, occurring $n' \geq (n/8)/2^{8T/n}$ times. Further shrink all $I_x = I'$ to a common subinterval that avoids any of the values fixed previously.

(3) Now, the adversary can set the remaining $n/2$ elements in I to appropriate values and force the answer to be the median in I', by equalizing the number of elements left of I' and right of I'. We can then complete the simulation of the pass of $\mathcal{A}$.

Through this process, we have ensured that the number of passes satisfies

$$P(n) \;\geq\; P((n/8)/2^{8T/n}) + 1 \qquad \text{if } n > 8s.$$

Changing variables with $P'(m) := P(T/m)$, the recurrence $P'(m) \geq P'(8m2^{8m}) + 1$ for $m < T/(8s)$ implies $P'(m) = \Omega(\log^*(T/s) - \log^* m)$. So the number of passes must be $P(N) = \Omega(\log^*(T/s) - \log^*(T/N))$, which is $\Omega(p)$ if $T = O(N \log^{(p)}(N/s))$. □

3 The Iterated-Log Lower Bound in an I/O model

The preceding proof already fails to imply $\Omega(N \log^*(N/s))$ time if we extend the model slightly so that a pass can start or end at arbitrary positions of the array (since in the proof, the elements in I occur in prefixes of the array whose sizes decrease geometrically). In this section, we provide a proof that works in a more powerful *I/O model*. Here, the algorithm can access any block of B contiguous elements of the read-only array with a single I/O operation for a fixed $B < s$. For example, one scan over the entire array can be done in $O(N/B)$ I/Os. The algorithm can still keep at most s storage cells at any time. Some previous work has addressed lower bounds in various versions of the I/O model (e.g., see [3]). Here, we rule out the possibility of a selection algorithm that simultaneously uses $O(N/B)$ I/Os and makes $O(N)$ number of comparisons, if B is not too small.

As hinted at in the introduction, the proof requires more than just an adversary argument approach. In addition, we adopt the probabilistic method. While these two approaches may seem incompatible at first, our idea is to take a random input initially, and then alter the values of a subset of the elements by an adversary, yielding the final bad input.

Let $\mathcal{A}$ denote the given algorithm. For simplicity, assume N is a power of 2; the default base of logarithm is 2. From now on, an *interval* will refer only to an interval[1] of the form $[Nj/2^k, N(j+1)/2^k)$ where $k \in \{0, 1, \ldots, \log N\}$ and $j \in \{0, 1, \ldots, 2^k - 1\}$; we say the interval has *depth* k. Let n_I and d_I denote the length and depth of I respectively. Intervals in the usual sense will instead be referred to as *ranges*. The abbreviation "w.h.p." means "with probability $1 - O(1/N^{c_0})$" where the constant c_0 can be made arbitrarily large.

3.1 The advantages of random input

We start with an array of N random real-valued elements, independent and uniformly distributed from $[0, N)$.

[1] Such intervals are sometimes called *dyadic intervals*.

Lemma 3.1 *For random input, the following holds w.h.p.: for every interval I with $n_I > \log N$, the number of elements in I is $n_I \pm O(\sqrt{n_I \log N})$.*

Proof: The number of elements in I is a sum of independent 0-1 random variables, with overall mean n_I and variance $O(n_I)$. For a fixed I, the condition thus holds w.h.p. by the standard Chernoff bound. The lemma follows since there are only polynomially number ($O(N \log N)$) of different I's. $\square$

We say that an I/O operation *encounters* an element x if the block involved in the operation contains x. Consider the subset of all elements in an interval I. Intuitively, because the input is random, the subset is "scattered" throughout the array, and so one would need $\Omega(N/B)$ I/O operations in order to encounter a constant fraction of the elements in I. This fact is justified formally by the lemma below, assuming that the number of elements in I is sufficiently large.

Lemma 3.2 *For random input, the following holds w.h.p.: for every interval I with $n_I > (N/B)\log N$, one I/O operation can encounter at most $O(Bn_I/N)$ elements in I.*

Proof: For a fixed I and a fixed block, the number of elements in I from the block is at most $O(Bn_I/N)$ w.h.p. by the Chernoff bound (since $Bn_I/N = \Omega(\log N)$). The lemma follows since there are only polynomially number of different I's and different blocks. $\square$

Although deterministic constructions can easily guarantee the above two lemmas, we want the input to satisfy additional properties as explained in the next subsection.

3.2 A bit-revealing scheme

In the proof in Section 2, cutting an element's interval in half is essentially equivalent to revealing an additional bit of the element: a 0 bit corresponds to taking the left half, and a 1 bit corresponds to taking the right half. There, we resolve a comparison by adversarily setting at most one bit from each of the two elements involved. Here, we observe that a random setting of bits is just as good.

It is worth mentioning that our proof is a perfect illustration of the so-called "principle of deferred decision" [24]. We view the probability space differently at this point: instead of viewing the input elements as chosen uniformly at random from the beginning, we reveal the random bits of (the binary representation of) the elements one by one, in an order that is determined as the process unfolds.

Formally, assume that a prefix of each element's binary representation has previously been revealed. In simulating $\mathcal{A}$, suppose $\mathcal{A}$ performs a comparison between x and y. Write $x = x_1x_2\cdots$ and $y = y_1y_2\cdots$ in binary, and suppose the prefixes $x_1\cdots x_j$ and $y_1\cdots y_k$ have been revealed. If $x_\ell \neq y_\ell$ for some $\ell \leq \min\{j,k\}$, then the comparison has already been resolved. Otherwise, if $j < k$, we reveal the next bit x_{j+1} of x, and if $x_{j+1} \neq y_{j+1}$, the comparison is resolved. The case $j > k$ is symmetric. If $j = k$, we reveal both the next bits x_{j+1} and y_{k+1}, and if they are different, the comparison is resolved. If the comparison is not resolved, we repeat the above, moving on to the next bits.

The following lemma shows that a comparison can be resolved by revealing only a constant number of bits on average.

Lemma 3.3 *Assume that for each input element, some prefix of its binary representation has been fixed, but the remaining bits of the input are random, independent and uniformly distributed. Fix a* *current state of $\mathcal{A}$. Then the following holds w.h.p. for $T \geq \log N$: a sequence of T comparison steps of $\mathcal{A}$ can reveal at most $O(T)$ additional bits of the input.*

Proof: In the above, the probability that a random bit x_{j+1} is different from a fixed bit y_{j+1} (or another random bit y_{k+1}) is $1/2$. Thus, the number of bits revealed during a comparison is a geometrically distributed random variable with mean 2. The total number of bits revealed in T comparisons is a sum of independent such variables with overall mean $2T$. By a Chernoff bound, the number has to be $O(T)$ w.h.p. (Put another way, for a constant $c > 2$, a sequence of cT coin tosses has to produce at least T heads w.h.p.) $\square$

To prepare for the overall proof, we make a few definitions:

Definition 3.4

- We say that a bit of an element has *depth* k if it is the k-th most significant bit of the element. (Notice that all elements in the same depth-k interval have the same prefix of bits up to depth k.)
- A bit of an element is *in* the interval I if the element is in I and the bit has depth greater than d_I.
- An element in I is *constrained* in I if it has a revealed bit in I. Otherwise, the element is *free* in I.

Remark: There is another, less algorithmic way of describing which bits are revealed by the above scheme. Define the depth of a comparison between x and y as the depth of the most significant bits of x and y that differ. For an element x, the number of bits revealed in x is precisely the largest depth of all comparisons that x has participated in up to this moment.

3.3 The overall proof by adversarial alteration

Theorem 3.5 *Any deterministic comparison-based median-finding algorithm in the I/O model that uses s storage cells must require $\Omega(pN/B)$ I/O operations or $\Omega(N \log^{(p)}(N/s))$ comparisons in the worst case for any p, if $B > \log N \log^{(b)} N$ for some constant b.*

Proof: Let T be the worst-case number of comparisons made by $\mathcal{A}$. We work with a random input, whose bits are revealed in various phases as we simulate $\mathcal{A}$. At the beginning of a phase, assume that all bits up to depth d of all elements have been revealed, and let I be the length-n, depth-d interval that contains the answer, where $n > \max\{100s, N/\log^{(b)} N\}$ and $d = \log(N/n)$. If we have not quit yet, we ensure that the following conditions are true:

(I) there are at least $0.7n$ free elements in I, and

(II) the answer is the median of the elements in I.

The presence of free elements means that the answer is not yet determined and $\mathcal{A}$ has not yet finished. The adversary may alter the values of certain elements, but we ensure the invariant that at the beginning of the phase, all elements in I are unaltered, and the current subset of elements in I

is exactly the same as the original subset of elements in I, so that the lemmas in Section 3.1 are still applicable to the interval I. Furthermore, the adversary will only alter elements in a way so that comparisons made in the past still have the same outcomes.

During the phase, we proceed as follows:

(1) We simulate the next steps of $\mathcal{A}$ until $\delta N/B$ I/Os have been performed, revealing bits along the way according to the scheme in Section 3.2. We check that this simulation encounters at most $0.1n$ elements in I, including the at most $s < 0.01n$ elements in memory at the beginning of the phase. If false, we quit. By Lemma 3.2, we do not quit here w.h.p. if δ is a sufficiently small constant.

Let F be the elements that remain free in I at this point. Then $|F| \geq 0.6n$.

(2) We check that the total number of bits revealed in I so far is at most cT. If false, we quit. By Lemma 3.3, we do not quit here w.h.p. if c is a sufficiently large constant.

(3) Set $n' = n/2^{5cT/n}$ and $d' = d + 5cT/n$. For each constrained element in a depth-d' subinterval of I, the number of its bits revealed in I must be at least $d' - d = 5cT/n$. It follows by (2) that the number of such elements is at most $0.2n$. Therefore, there must exist a depth-d' subinterval I' of I that has at most $0.2n/(n/n') = 0.2n'$ constrained elements in I'.

(4) Next, we reveal all bits up to depth d' of all elements. We check that I has $n \pm o(n)$ elements and I' has $n' \pm o(n')$ elements. If false, we quit. By Lemma 3.1, we do not quit here w.h.p.

(5) By (3) and (4), the number of free elements in I' is at least $0.8n' - o(n')$, which exceeds $0.7n'$. So we have established (I) for the interval I'.

(6) Furthermore, by (1) and (4), the number of elements in $F - I'$ is at least $0.6n - n' - o(n')$, which exceeds half of the number of elements in I. The adversary can then alter the elements in $F - I'$ to appropriate values in $I - I'$ and force the answer to be the median in I', by equalizing the number of elements left of I' and right of I'. So, we have established (II) for I'. All invariants have now been maintained to begin the next phase.

W.h.p., the number of phases thus satisfies the recurrence

$$P(n) \;\geq\; P(n/2^{5cT/n}) + 1 \qquad \text{if } n > \max\{100s,\, N/\log^{(b)} N\}.$$

As in the proof of Theorem 2.1, w.h.p., the number of phases must be $\Omega(p)$ if $T = O(N \log^{(p)}(N/s))$. We conclude the existence of some input instance requiring that $\Omega(pN/B)$ I/Os if $T = O(N \log^{(p)}(N/s))$. $\square$

Remark: The result holds even if elements in a block are not necessarily contiguous, as long as the number of different possible blocks is polynomially bounded in N. In the case of blocks of contiguous elements, the extra log factor in the restriction on B can probably be reduced, by being more careful (in Lemma 3.2).

4 A Randomized $\Omega(N \log \log_S N)$ Lower Bound in the RAM Model

For the proof of our randomized lower bound on the RAM, we abandon the adversary altogether and work entirely with an independent, uniformly distributed input. Consequently, the result automatically applies to randomly ordered input. By Yao's principle, it suffices to consider the complexity of a deterministic algorithm $\mathcal{A}$ over this random input. Our proof can be restated in the *comparison branching program* model, but we will not elaborate, to avoid more definitions. Assume that $\mathcal{A}$ uses S bits of space and that $S = \omega(\log N)$, w.l.o.g.

4.1 An encounter lemma for the RAM

We first give an analog of Lemma 3.2 for the RAM. We say that a computational step *encounters* an element x if the algorithm reads the memory location holding x. The lemma below exploits the advantage of random input and the fact that small-space algorithms can have only a small number of possible states. The proof of the lemma is in a similar style as in some previous papers like [4, 10, 30].

Lemma 4.1 *For random input, the following holds w.h.p.: for every interval I and every state of $\mathcal{A}$, a sequence of SN/n_I steps of $\mathcal{A}$ can encounter at most $O(S)$ elements in I.*

Proof: Fix a state. Let $x^{(1)}, x^{(2)}, \ldots$ be the elements encountered by $\mathcal{A}$ starting at this state, in order of the times of their first encounters. Observe that $x^{(j)}$ is uniformly distributed and independent of $x^{(1)}, \ldots, x^{(j-1)}$. Thus, for a fixed I, the number of elements of $x^{(1)}, \ldots, x^{(SN/n_I)}$ in I is a sum of independent 0-1 random variables, with overall mean S. For a fixed state and a fixed I, the condition thus holds with probability at least $1 - 2^{-c_0 S}$ by the standard Chernoff bound where the constant c_0 can be made arbitrarily large. The lemma follows since there are $O(2^S)$ number of different states and polynomially number of different I's. $\square$

4.2 Another bit-revealing lemma

Our proof will again rely on the principle of deferred decision and the process of revealing bits as described in Section 3.2. However, Lemma 3.3 is not sufficient, and we now need a different bound on the number of bits revealed:

Lemma 4.2 *Assume that for each input element, some prefix of its binary representation has been fixed, but the remaining bits of the input are random, independent and uniformly distributed. Fix a current state of $\mathcal{A}$. Then the following holds w.h.p.: for every interval I, a sequence of SN/n_I steps of $\mathcal{A}$ can reveal at most $O(S \log S)$ additional bits in I, or the condition in Lemma 4.1 fails.*

Proof: Let (‡) denote the condition in Lemma 4.1. If (‡), we know that there can be at most $O(S^2)$ nonredundant comparisons between elements in I. Since only such comparisons can reveal bits in I, we can apply Lemma 3.3 to bound the number of additional revealed bits in I by $O(S^2)$ w.h.p., which unfortunately is not good enough for our purposes. However, if for the sake of intuition, we consider the scenario where initially no bits have been revealed and we know which of the $O(S)$ elements are to be encountered, it is not difficult to see that only $O(S \log S)$ bits are needed to determine the total order of these $O(S)$ elements w.h.p. For a formal proof of the $O(S \log S)$ bound, we adopt a "preemptive" strategy that attempts to make more comparisons early on, in order to make future comparisons redundant, as we now explain.

Fix I. Consider the following process we call $\mathcal{P}_I$: First reveal all bits of depth up to d_I of all elements, and reveal all elements not in I. We simulate the SN/n_I steps of $\mathcal{A}$ while maintaining a sorted list L of all elements in I encountered so far. Whenever we encounter an element x in I for the first time while running $\mathcal{A}$, we insert x to L by performing a binary search. This requires $O(\log|L|)$ comparisons, which are resolved by revealing bits as we proceed. When $\mathcal{A}$ actually makes a comparison, say, between x and y, the two elements x and y have already been inserted to L and the comparison can be resolved without revealing any more bits.

Since $\mathcal{P}_I$ resolves more comparisons than $\mathcal{A}$, the number of additional revealed bits in I by $\mathcal{A}$ is at most the number of additional revealed bits in I by $\mathcal{P}_I$ in every input instance. The number of nonredundant comparisons made by $\mathcal{P}$ is $O(S\log S)$, since $|L| = O(S)$, if (‡). By Lemma 3.3, w.h.p. the number of additional revealed bits in I by $\mathcal{P}_I$ in every input instance is $O(S\log S)$ or (‡) is false. Our lemma follows since there are polynomially number of different I's. □

4.3 The overall proof

We will finally present the proof of the main result, after first isolating some technical probabilistic facts. Unlike in Section 3.3, we cannot alter the free elements to force the answer to lie in a particular subinterval, but has to let the answer fall where it may "naturally". Fortunately, there is enough variability in nature, as quantified in the following statements:

Proposition 4.3 *Fix k. In the range $I = [0, n)$, fix $n/2$ elements and add $n/2$ independent and uniformly distributed, random elements. Let x^* be the overall k-th smallest element. Then*

(i) *we can find a range J of length $O(\sqrt{bn})$ such that $\Pr\{x^* \in J\} \geq 1 - 2^{-\Omega(b)}$;*

(ii) *for each fixed range $J' \subseteq [n/a, (1-1/a)n]$ and fixed k',*

$$\Pr\{x^* \textit{ is the } k'\textit{-th smallest element in } J'\} = O(\sqrt{a/n}).$$

Proof:

(i) Let $\mu(x)$ denote the expected number of elements in $[0, x)$, i.e., let $\mu(x)$ be equal to $xn/2$ plus the number of fixed elements in $[0, x)$. Pick x_0 so that $\mu(x_0) = k$. Set $J = [x_0 - \sqrt{bn}, x_0 + \sqrt{bn}]$. Observe that $\mu(x_0) - \mu(x_0 - \sqrt{bn})$ is at least $n/2 \cdot \sqrt{bn}/n = \sqrt{bn}/2$. The number of elements in $[0, x_0 - \sqrt{bn})$ is a sum of n independent 0-1 random variables (the fixed ones would have 0 variance), with overall mean $\mu(x_0 - \sqrt{bn}) \leq k - \sqrt{bn}/2$. By the standard Chernoff bound, this number is greater than k with probability at most $2^{-\Omega(b)}$. Thus, the probability that x^* is to the left of J is at most $2^{-\Omega(b)}$. The case where x^* is to the right of J is symmetric.

(ii) Let p be the length of the left portion of $I - J'$ divided by n. Note that $1/a \leq p \leq 1 - 1/a$. Let k'' be the number of fixed elements to the left of J'. Then x^* is the k'-th smallest in J' iff the number of random elements to the left of J' is precisely $j := k - k' - k''$. The probability of this event is $\binom{n/2}{j}p^j(1-p)^{n/2-j}$, which is maximized when $j \approx pn/2$. The maximum is at most $O(1/\sqrt{\min\{pn, (1-p)n\}})$, for example, by Stirling's formula. □

Theorem 4.4 *Any randomized median-finding algorithm in the comparison RAM model that uses S bits of space must require $\Omega(N \log\log_S N)$ expected time.*

Proof: Let a and b be parameters to be set later, which are nonconstant but small, e.g., $o(\log N)$. Consider a random input. Let $n_0 = N$, $n_{i+1} = n_i^{1/4}$, and $d_i = \log(N/n_i)$, for $i = 1, \ldots, t$, where $n_t > \max\{S/\delta, \log^3 N\}$. We simulate $\mathcal{A}$ for t phases each of δN steps, revealing bits along the way according to the scheme in Section 3.2. Let I_i be the depth-d_i interval that contains the answer. Suppose the answer, denoted x^*, is the k_i-th smallest in I_i.

Call an interval I *good* if there are at least $0.6n_I$ free elements in I, or *bad* otherwise. For each i, let $(\dagger)_i$ be the following conjunction of events:

(I)$_i$ at the beginning of the i-th phase, I_i is good, and

(II)$_i$ $k_i \in [n_i/a, (1-1/a)n_i]$.

The presence of free elements means that if $(\dagger)_i$ is true, $\mathcal{A}$ cannot finish before phase i. We will bound the probability that $(\dagger)_{i+1}$ is false when $(\dagger)_i$ is true.

First let (*) be the following conjunction of events: for every interval I with $n_I > \max\{S/\delta, \log^3 N\}$,

(III) the number of elements in I is $n_I \pm o(n_I/a)$,

(IV) the simulation of the δN steps in each phase encounters at most $0.1n_I$ elements in I, and

(V) the entire simulation, of at most $\delta t N$ steps, reveals at most $O(tn_I \log S)$ bits in I.

By Lemmas 3.1, 4.1, and 4.2, (*) holds w.h.p. if δ is a sufficiently small constant.

We proceed as follows:

(1) Fix all bits up to depth d_i of all elements and fix all bits revealed up to the end of phase i. Then we know I_i, k_i, and whether $(\dagger)_i$ holds. Assume $(\dagger)_i$. By (I)$_i$, at least $0.6n_i$ elements are free in I_i at the beginning of phase i. By (IV), after the phase, at least $n_i/2$ elements remain free in I_i if (*).

(2) Further fix the constrained elements in I_i but leave $n_i/2$ free elements in I_i random.

By Proposition 4.3(i), there exists a range J of length $\Theta(\sqrt{bn_i})$ such that $\Pr\{x^* \in J \text{ or not } (*)\} \geq 1 - 2^{-\Omega(b)}$. By (II)$_i$ and (III), we may assume J satisfies the property that the length of either portion of $I - J$ is $\Omega(n_i/a)$. We may further assume that J is a union of two intervals $\{J_j\}$ of length $\Theta(\sqrt{bn_i})$.

(3) By (V), the total number of bits revealed in J_j is at most $O(t\sqrt{bn_i}\log S)$ if (*). For each constrained element in a depth-d_{i+1} subinterval of J_j, the number of its revealed bits in J_j must be at least $d_{i+1} - d_{J_j} = \Omega(\log(\sqrt{bn_i}/n_{i+1})) = \Omega(\log n_i)$, since $n_{i+1} = n_i^{1/4}$. Thus, the number of bad depth-d_{i+1} subintervals in J_j must be at most $O\left(\frac{t\sqrt{bn_i}\log S}{n_{i+1}\log n_i}\right)$ if (*).

(4) For each fixed depth-d_{i+1} subinterval J' of J,

$$\Pr\{x^* \in J' \text{ and } (*)\} = O(n_{i+1}\sqrt{a/n_i})$$

by Proposition 4.3(ii) where we sum over all $k' = 1, \ldots, n_{i+1} + o(n_{i+1}/a)$. Thus,

$$\begin{aligned}
&\Pr\{(\text{not (I)}_{i+1}) \text{ and } (*)\} \\
&\quad\leq \sum_{\text{bad depth-}d_{i+1}\ J' \subseteq J} \Pr\{x^* \in J' \text{ and } (*)\} \;+\; \Pr\{x^* \notin J \text{ and } (*)\} \\
&\quad= O\left(\frac{t\sqrt{bn_i}\log S}{n_{i+1}\log n_i} \cdot n_{i+1}\sqrt{a/n_i} \;+\; 2^{-\Omega(b)}\right) \;=\; O\left(\sqrt{ab}\,t/\log_S n_i + 2^{-\Omega(b)}\right).
\end{aligned}$$

(5) On the other hand, for each fixed depth-d_{i+1} subinterval J' of J,

$$\Pr\{x^* \in J' \text{ and } k_{i+1} \notin [n_{i+1}/a, (1-1/a)n_{i+1}] \text{ and } (*)\} \;=\; O((n_{i+1}/a)\sqrt{a/n_i})$$

by Proposition 4.3(ii) where we sum over all $k' = 1, \ldots, n_{i+1}/a$ and $k' = (1 - 1/a)n_{i+1}, \ldots, n_{i+1} + o(n_{i+1}/a)$. Thus,

$$\begin{aligned}
&\Pr\{(\text{not (II)}_{i+1}) \text{ and } (*)\} \\
&\quad\leq \sum_{\text{depth-}d_{i+1}\ J' \subseteq J} \Pr\{x^* \in J' \text{ and } k_{i+1} \notin [n_{i+1}/a, (1-1/a)n_{i+1}] \text{ and } (*)\} \\
&\qquad\quad + \Pr\{x^* \notin J \text{ and } (*)\} \\
&\quad= O\left(\frac{\sqrt{bn_i}}{n_{i+1}} \cdot (n_{i+1}/a)\sqrt{a/n_i} \;+\; 2^{-\Omega(b)}\right) \;=\; O\left(\sqrt{b/a} + 2^{-\Omega(b)}\right).
\end{aligned}$$

Set $b = t$ and $a = 2^t$, for example. We conclude that

$$\Pr\{(\text{not } (\dagger)_{i+1}) \text{ and } (\dagger)_i \text{ and } (*)\} \;\leq\; 2^{O(t)}/\log_S n_i + 2^{-\Omega(t)}.$$

All the above probability statements are originally conditioned to the bits that have been fixed, but we may now infer that the conclusion holds unconditionally.

Summing over $i = 1, \ldots, t$ and recalling that $n_i = N^{1/4^i}$, we can bound the probability that some $(\dagger)_i$ is false by $O(2^{O(t)}/\log_S N + 2^{-\Omega(t)} + N^{-c_0})$. For $t = \varepsilon \log\log_S N$ with a sufficiently small ε, this probability is $O(\log_S N)^{-\Omega(1)}$. Therefore, $\mathcal{A}$ takes $\Omega(N \log\log_S N)$ time with probability at least $1 - O(\log_S N)^{-\Omega(1)}$, which is $\Omega(1)$ (we may assume $\log_S N = \omega(1)$, for otherwise we have nothing to prove). In particular, the expected running time must be $\Omega(N \log\log_S N)$. □

5 Final Remarks

A fascinating open problem is to prove a deterministic $\Omega(N \log^* N)$-time lower bound for selection on the comparison RAM for, say, $S = N^\varepsilon$. Most of the relevant comparison-RAM lower bound techniques simply consider a randomly generated input, but these cannot work here because for $S = N^\varepsilon$, the randomized time complexity is $\Theta(N)$. One likely has to combine random input with adversary arguments as in Section 3, but the proof of the starting lemma, Lemma 4.1, already fails for subtle reasons: The way that some elements are altered might accidentally leak information that could allow the algorithm to encounter the elements in a given interval more quickly. Although intuitively this seems unlikely since the adversary is doing the alteration, ruling out this possibility rigorously seems quite difficult.

Another reason that the above open problem is interesting is that it is inherently about comparison-based algorithms (and so communication complexity techniques like [12] do not seem to apply). For $O(\log N)$-bit integers, it is not difficult to modify Munro and Paterson's multi-pass algorithm [25] to get a deterministic $O(N)$-time algorithm with $O(N^{\varepsilon})$ space, by using radix-sort at the base levels.

A related question is to prove a deterministic $\Omega(N \log_S N)$-time lower bound on the comparison RAM for, say, S polylogarithmic. This would, in a sense, extend the $\Omega(\log_S N)$-pass, deterministic lower bound of Munro and Paterson [25] (and a similar, non-comparison-based, deterministic lower bound of Guha and McGregor [21]) for multi-pass algorithms.

By a technique of Moran *et al.* [23], we can extend the types of comparisons allowed for our lower bounds, by allowing arbitrary (algebraic or nonalgebraic) predicates over a constant number of input elements. Although Moran *et al.*'s technique was originally for decision trees, it is easy to see that it applies to branching programs as well. (Basically, given N, the technique applies Ramsey's theorem to construct a subdomain $D \subset \mathbb{R}$ so that for any predicate encountered along any computation path, the predicate reduces to direct comparisons when the arguments are restricted to lie in D.)

It is interesting to note that while selection and low-dimensional linear programming appear to have similar complexities in regards to streaming algorithms, they are different on the RAM: Chan and Chen [14] have obtained a randomized RAM algorithm for linear programming in fixed dimensions that runs in $O(n)$ expected time with logarithmic space, but our lower bound says this is not possible for selection.

Finally, with current communication complexity techniques, could one hope for a proof of the randomized $\Omega(N \log\log_S N)$ lower bound for selection on the general R-way branching program model?

References

[1] M. Ajtai. A non-linear time lower bound for Boolean branching programs. In *Proc. 40th IEEE Sympos. Found. Comput. Sci.*, pages 60–70, 1999.

[2] M. Ajtai. Determinism versus nondeterminism for linear time RAMs with memory restrictions. *J. Comput. Sys. Sci.*, 65:2–37, 2002.

[3] L. Arge and J. Pagter. I/O-space trade-offs. In *Proc. 7th Scand. Workshop Algorithm Theory*, Lect. Notes Comput. Sci., vol. 1851, Springer-Verlag, pages 448–461, 2000.

[4] P. Beame. A general sequential time–space tradeoff for finding unique elements. *SIAM J. Comput.*, 20:270–277, 1991.

[5] P. Beame, T. S. Jayram, and M. Saks. Time–space tradeoffs for branching programs. *J. Comput. Sys. Sci.*, 63:542–572, 2001.

[6] P. Beame, M. Saks, X. Sun, and E. Vee. Time–space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50:154–195, 2003.

[7] S. W. Bent and J. W. John. Finding the median requires $2n$ comparisons. In *Proc. 17th ACM Sympos. Theory Comput.*, pages 213–216, 1985.

[8] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *J. Comput. Sys. Sci.*, 7:448–461, 1973.

[9] A. Borodin and S. A. Cook. A time–space tradeoff for sorting on a general sequential model of computation. *SIAM J. Comput.*, 11:287–297, 1982.

[10] A. Borodin, F. E. Fich, F. Meyer auf der Heide, E. Upfal, and A. Wigderson. A time–space tradeoff for element distinctness. *SIAM J. Comput.*, 16:97–99, 1987.

[11] A. Borodin, M. J. Fischer, D. G. Kirkpatrick, N. A. Lynch, and M. Tompa. A time-space tradeoff for sorting on non-oblivious machines. *J. Comput. Sys. Sci.*, 22:351–364, 1981.

[12] A. Chakrabarti, T. S. Jayram, and M. Pătraşcu. Tight lower bounds for selection in randomly ordered streams. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pages 720–729, 2008.

[13] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *Proc. 40th ACM Sympos. Theory Comput.*, pages 641–650, 2008.

[14] T. M. Chan and E. Y. Chen. Multi-pass geometric algorithms. *Discrete Comput. Geom.*, 37:79–102, 2007.

[15] A. Cobham. The recognition problem for the set of perfect squares. In *Proc. 7th IEEE Sympos. Found. Comput. Sci.*, pages 78–87, 1966.

[16] W. Cunto and J. I. Munro. Average case selection. *J. ACM*, 36:270–279, 1989.

[17] D. Dor and U. Zwick. Selecting the median. *SIAM J. Comput.*, 28:1722–1758, 1999.

[18] D. Dor and U. Zwick. Median selection requires $(2+\epsilon)n$ comparisons. *SIAM J. Discrete Math.*, 14:312–325, 2001.

[19] R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *Commun. ACM*, 18:165–173, 1975.

[20] G. N. Frederickson. Upper bounds for time–space trade-offs in sorting and selection. *J. Comput. Sys. Sci.*, 34:19–26, 1987.

[21] S. Guha and A. McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *Proc. 34th Int. Colloq. Automata, Languages, and Programming*, Lect. Notes Comput. Sci., vol. 4596, Springer-Verlag, pages 704–715, 2007.

[22] T. W. Lai and D. Wood. Implicit selection. In *Proc. 1st Scand. Workshop Algorithm Theory*, Lect. Notes Comput. Sci., vol. 318, Springer-Verlag, pages 14–23, 1988.

[23] S. Moran, M. Snir, and U. Manber. Applications of Ramsey's Theorem to decision tree complexity. *J. ACM*, 32:938–949, 1985.

[24] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[25] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. *Theoret. Comput. Sci.*, 12:315–323, 1980. Preliminary version in FOCS 1978.

[26] J. I. Munro and V. Raman. Selection from read-only memory and sorting with minimum data movement. *Theoret. Comput. Sci.*, 165:311–323, 1996. Preliminary version in FCTTCS 1992.

[27] J. Pagter and T. Rauhe. Optimal time-space trade-offs for sorting. In *Proc. 39th IEEE Sympos. Found. Comput. Sci.*, pages 264–268, 1998.

[28] V. Raman and S. Ramnath. Improved upper bounds for time–space trade-offs for selection. *Nord. J. Comput.*, 6:162–180, 1999.

[29] A. Schönhage, M. Paterson, and N. Pippenger. Finding the median. *J. Comput. Sys. Sci.*, 13:184–199, 1976.

[30] A. C.-C. Yao. Near-optimal time–space tradeoff for element distinctness. *SIAM J. Comput.*, 23:966–975, 1994.

Appendix: A Randomized $O(N \log\log_s N)$ Algorithm

Munro and Raman [26] showed that the selection problem can be solved on the comparison RAM with $O(s)$ storage cells in $O(N \log\log_s N)$ expected time, provided that each permutation of the input is equally likely. We observe that this assumption can be removed by randomizing the algorithm. The observation is simple, but appears new.

As in Munro and Raman's paper, the algorithm is just a variant of Floyd and Rivest's standard random sampling algorithm [19]. We will follow the version from Motwani and Raghavan's textbook [24, Section 3.3]. An apparent difficulty is that we do not have space to store a random sample. (This wasn't a problem for randomly ordered input, since a prefix would make a good sample.) The key is to realize that we do not need a sample of totally independent elements, but only pairwise independent elements, if we use an analysis based on second moments (Chebyshev's inequality), as was done in [24], rather than Chernoff's inequality. The resulting sample can then be specified in $O(1)$ space by standard constructions ("2-point sampling"—see [24, Section 3.4]). Of course, the utility of such constructions is well known, e.g., in reducing the amount of random bits or derandomizing algorithms, but still the application in this context is interesting.

For completeness, we spell out the entire pseudocode. We first assume N is prime, by increasing N by at most a constant factor and implicitly padding the array $A[0,\ldots,N-1]$ with extra infinities. Initialize $I = (-\infty, \infty)$ and repeat the following:

1. Let n be the number of elements in I. If $n \le s$, return the k-th smallest in I directly.
2. Choose random $a, b \in \{0,\ldots,N-1\}$. Let $R = \{A[(aj+b) \bmod N] : j = 1,\ldots,N/n^{1/4}\}$.
3. Let $k^- = \max\{k/n^{1/4} - \sqrt{n},\, 1\}$ and $k^+ = \min\{k/n^{1/4} + \sqrt{n},\, n\}$. Compute the k^--th smallest x^- and k^+-th smallest x^+ of the elements in $R \cap I$.
4. Let ℓ^- be the number of elements in $I \cap (-\infty, x^-]$ and ℓ^+ be the number of elements in $I \cap (-\infty, x^+]$. Check that $\ell^- > k$, $\ell^+ \le k$, and $\ell^+ - \ell^- \le 4n^{3/4} + 2$. If false, go back to line 2.
5. Set $I = I \cap (x^-, x^+]$ and $k = k - \ell^-$.

For line 3, we can use a variant of the standard randomized quickselect algorithm, which requires an expected $O(\log |R \cap I|)$ passes over the elements in R. The implementation takes expected time $O(|R| \log |R \cap I|) = O((N/n^{1/4}) \log n) = O(N)$ and space $O(1)$. Note that R is not stored explicitly. Line 4 takes $O(N)$ time.

The conditions in line 4 hold with probability $1 - O(n^{-1/4})$ by virtually the same analysis as in [24], because the variables $\{(aj+b) \bmod N : j = 1,\ldots,N/n^{1/4}\}$ are pairwise independent.

The total expected time satisfies the recurrence $T(n) = T(O(n^{3/4})) + O(N)$ for all $n > s$, and solves to $T(N) = O(N \log\log_s N)$.