SIAM J. COMPUT.
Vol. 20, No. 2, pp. 270-277, April 1991

A GENERAL SEQUENTIAL TIME-SPACE TRADEOFF FOR FINDING UNIQUE ELEMENTS*

PAUL BEAME†

Abstract. An optimal $\Omega(n^2)$ lower bound is shown for the time-space product of any R-way branching program that determines those values which occur exactly once in a list of n integers in the range $[1, R]$ where $R \geqq n$. This $\Omega(n^2)$ tradeoff also applies to the sorting problem and thus improves the previous time-space tradeoffs for sorting. Because the R-way branching program is such a powerful model, these time-space product tradeoffs also apply to all models of sequential computation that have a fair measure of space such as off-line multitape Turing machines and off-line log-cost random access machines (RAMs).

Key words. lower bounds, time-space tradeoff, computational complexity, sorting, branching programs

AMS(MOS) subject classifications. 68P10, 68Q10, 68Q25

1. Introduction.

The goal of producing nontrivial lower bounds on the time or space complexity for specific computational problems in $\mathcal{NP}$ has largely been elusive. Also, concentration on a single resource does not always accurately represent all of the issues involved in solving a problem. For some computational problems it is possible to obtain a whole spectrum of algorithms within which one can trade time performance for storage or vice versa. Thus the question of obtaining lower bounds that say something about time and space simultaneously has received considerable study as well.

The most interesting model for studying time-space tradeoff lower bounds that has been developed is the R-way branching program model. The R-way branching program is an unstructured model of computation that has unrestricted random access to its inputs and which makes no assumption about the way its internal storage is managed. The model is powerful enough that lower bounds proven in it apply to a wide variety of sequential computing models including off-line multitape Turing machines with random-access input heads. A particularly convenient model for which the lower bounds for R-way branching programs apply is that of a random access machine (RAM) with its input stored in a read-only memory, with a unit-cost measure of time and with its read-write storage charged on a log-cost basis.

The R-way branching program model was introduced by Borodin and Cook [BC82], who used it in showing the first nontrivial general sequential time-space tradeoff lower bound for any problem. They showed that any R-way branching program requires a time-space product of $\Omega(n^2/\log n)$ to sort n integers in the range $[1, n^2]$.

Since [BC82], time-space tradeoff lower bounds on R-way branching programs have been shown for a number of algebraic problems such as discrete Fourier transforms, matrix-vector products, and integer and matrix multiplication [Yes84], [Abr86]. In addition to these results, Reisch, in [RS82], has claimed an improvement of the sorting lower bound to $\Omega(n^2 \log\log n/\log n)$ using the same approach as in [BC82]. [RS82] presents an improvement of only one of the two key lemmas in [BC82]; this change appears to necessitate an overhaul of the second, more complex lemma as well in order to obtain the claimed bound. However, even this bound leaves a gap between the upper and lower bounds for sorting.

* Received by the editors March 16, 1989; accepted for publication (in revised form) June 28, 1990. This research was supported by National Science Foundation grant CCR-8858799.

† Computer Science Department, FR-35, University of Washington, Seattle, Washington 98195.

The first problem considered here, the UNIQUE ELEMENTS problem, is to produce, given an input list of n integers, a list of those integers that occur only once in the input. No particular order is required for the output of UNIQUE ELEMENTS. It is related to sorting, but its output provides much less information about the input than a sorted list provides. The main result of this paper is that any R-way branching program for UNIQUE ELEMENTS requires time T and space S such that $S \cdot T = \Omega(n^2)$ and that this bound is achievable using a simple RAM algorithm.

Borodin and Cook showed their time-space tradeoff for a somewhat unusual form of the sorting problem and derived bounds for the usual form of sorting by a straightforward reduction. In this paper their problem will be termed the RANKING problem and SORTING will be reserved for the usual form of output in which the elements are presented in sorted order. An easy reduction of UNIQUE ELEMENTS to the SORTING problem is shown which also yields an $\Omega(n^2)$ time-space product lower bound for SORTING. This improves the previous bounds in the amount of the tradeoff and also improves the range of inputs for which the bound holds. In this expanded range it is also shown that the tradeoff gap for sorting is closed, at least for R-way branching programs, since the $\Omega(n^2)$ time-space product is optimal.

A particularly remarkable feature of these results is the relative simplicity of the arguments required when compared with the involved arguments used in [BC82].

2. Definitions. An *$R(n)$-way branching program* consists of a directed acyclic rooted graph of out-degree $R = R(n)$ with each nonsink node labelled by an index from $\{1, \cdots, n\}$ and with the R out-edges of each node labelled $1, \cdots, R$. Edges of the branching program may also be labelled by a sequence of values from some output domain. The *size* of a branching program is the number of nodes it has. An R-way branching program is *levelled* if the nodes of the underlying graph are assigned levels so that the root has level 0 and the out-edges of a node at level l only go to nodes at level $l+1$.

Let $x = (x_1, \cdots, x_n)$ be an n-tuple of integers chosen from the range $[1, R]$. An R-way branching program computes a function of input x as follows. The computation starts at the root of the branching program. At each nonsink node v encountered, the computation follows the out-edge labelled with the value of x_i where i is the index that labels node v. (Variable x_i is *queried* at v.) The computation terminates when it reaches a sink node. The sequence of nodes and edges encountered is the *computation path* followed by x. The concatenation of the sequences of output values encountered along the path that x follows is the output of the branching program on input x.

The *time* used by a branching program is the length of the longest computation path followed by any input. The *space* used by a branching program is the logarithm base 2 of its size.

Any branching program can be levelled without changing its time and with at most squaring its size (see [BFK$^+$81]). Because this leaves the time used unchanged and changes the space used by no more than a factor of 2, it will usually be assumed without loss of generality that R-way branching programs are levelled.

Let $x = (x_1, \cdots, x_n)$ be an n-tuple of integers. An input value x_i is *unique* in x if there is no $j \neq i$ such that $x_i = x_j$. The UNIQUE ELEMENTS problem is, given an n-tuple of integers x as input, to produce as output a list (in arbitrary order) of exactly those values x_i that are unique in x.

In addition to the UNIQUE ELEMENTS problem the following two problems will also be of interest. The SORTING problem is, given an n-tuple of integers x as input, to produce as output the values of the x_i's in sorted (e.g., nondecreasing) order. The

RANKING problem is, given an n-tuple of integers x as input, to produce a list (in arbitrary order) of the ranks of all the inputs x_i in the sorted order of x where x_i's rank is output as a pair $(i, \text{rank}(x_i))$.

3. Unique elements.

THEOREM 1. *Any R-way branching program computing* UNIQUE ELEMENTS *for input integers in the range $[1, R]$, where $R \geqq n$, requires time T and space S such that $S \cdot T = \Omega(n^2)$.*

The general outline for the proof of this theorem is essentially the same as was used in the previous proofs of time-space tradeoff lower bounds for R-way branching programs [BC82], [Yes84], [Abr86], [Abr87] and which was originated in the context of comparison branching programs in [BFK$^+$81]. The R-way branching program is broken up into layers and each layer is considered as a collection of shallow branching programs, one rooted at each node on the interlayer boundary. It is shown that any shallow branching program produces many output values for only a tiny fraction of the input n-tuples. Because the problem requires a large number of outputs to be made, if the time is not large then a large number of outputs must be made during some layer and therefore during some shallow branching program. The bound then follows since the total number of shallow branching programs must be sufficient to compensate for the small fraction of inputs for which each produces enough outputs.

Most problems for which time-space tradeoff lower bounds have been shown require a fixed large number of outputs, e.g., n outputs are required for sorting. In contrast, certain input vectors for UNIQUE ELEMENTS require few outputs or possibly none at all. However, a large number of outputs is required for a sufficiently large fraction of the inputs that the technique still applies.

It will be convenient to express the argument in a probabilistic format. Denote the uniform distribution on $[1, R]^n$ by U_R^n.

LEMMA 2. *If x is chosen at random from U_n^n then*

$$\Pr[x \textit{ contains } \geqq n/(2e) \textit{ unique elements}] > 1/(2e-1).$$

Proof. Let $u(x)$ denote the number of unique elements in x and let

$$Y_i = \begin{cases} 1 & \text{if } x_i \text{ is unique in } x, \\ 0 & \text{if not.} \end{cases}$$

Then $E[Y_i] = \Pr[Y_i = 1] = [(n-1)/n]^{n-1} = (1-1/n)^n/(1-1/n) > e^{-1}$. Thus

$$E[u(x)] = E\left[\sum_{i=1}^{n} Y_i\right] = \sum_{i=1}^{n} E[Y_i] > \frac{n}{e}.$$

Since there are never more than n unique elements in x, an application of Markov's inequality shows that $\Pr[u(x) \geqq n/(2e)] > 1/(2e-1)$. (For let $\alpha = \Pr[u(x) \geqq n/(2e)]$. Then $\alpha \cdot n + (1-\alpha) \cdot n/(2e) > E[u(x)] > n/e$. Solving for α yields the desired result.) $\square$

For the UNIQUE ELEMENTS problem, say that an output value is *correct* for input x if it is the value of a unique element in x. Say that a branching program $\mathcal{P}$ *correctly outputs at least m values* on input x if all values output along the computation path in $\mathcal{P}$ that x follows are correct for x and at least m values are output along that computation path.

LEMMA 3. *Let $\mathcal{P}$ be an R-way branching program of height $\leqq n/4$ where $R \geqq n$. Let x be chosen at random from U_n^n. For $m \leqq n/4$,*

$$\Pr[\mathcal{P} \text{ correctly outputs at least } m \text{ values on input } x] \leqq e^{-m/2}.$$

Proof. Consider a computation path π in $\mathcal{P}$. Let Q_π be the set of indices of variables that are queried along π and V_π be the set of the first m values that are output along π. Some of the values in V_π can be values of variables queried along π but it is possible that some values in V_π are not. Call the values in V_π that are not values of any variable queried along π *extra* values and suppose that there are exactly k extra values in V_π. Let $s = n - |Q_\pi| - k$. Since $|Q_\pi| \leqq n/4$ and $k \leqq |V_\pi| = m \leqq n/4$, it follows that $s \geqq n/2$.

Assume that x has nonzero probability in U_n^n. The fact that an input x follows the path π in $\mathcal{P}$ only determines the values of the variables whose indices are in Q_π. The remaining $s+k$ variables are completely unconstrained so there are exactly n^{s+k} possible inputs in $[1, n]^n$ that can follow π in $\mathcal{P}$. For how many of these inputs are the values in V_π correct? In order for all the values in V_π to be correct it must be the case that, whatever the location of the k extra values in V_π among the $s+k$ unconstrained input variables, each of the remaining s variables must avoid all m values in V_π. Thus there are at most $(n-m)^s$ choices of the remaining s variables that would permit the values in V_π to be correct. Since there are exactly $(s+k)!/s!$ ways that the k extra values can occur in the input,

$$\begin{aligned}
\Pr[V_\pi \text{ is correct for } x \mid x \text{ follows } \pi] &\leqq \frac{(s+k)!}{s!} \cdot \frac{(n-m)^s}{n^{s+k}} \\
&< \left[\frac{s+k}{n}\right]^k \cdot \left[\frac{n-m}{n}\right]^s \\
&< \left[1 - \frac{m}{n}\right]^s \\
&\leqq \left[1 - \frac{m}{n}\right]^{n/2} \\
&< e^{-m/2}.
\end{aligned}$$

Since each input follows exactly one path π in $\mathcal{P}$, the statement of the lemma follows. $\square$

Proof of Theorem 1. Consider an R-way branching program $\mathcal{B}$ for UNIQUE ELEMENTS. Assume without loss of generality that $\mathcal{B}$ is levelled. Suppose that $\mathcal{B}$ uses time T and space S, i.e., $\mathcal{B}$ has height T and has 2^S nodes. For convenience we can also assume without loss of generality that n is a multiple of 4 and that T is a multiple of $n/4$. (Since $\mathcal{B}$ must at least query all inputs to produce an output, T is at least n anyway.)

Divide the levels of $\mathcal{B}$ into layers of height $n/4$ where layer i consists of the portion of the branching program from level $(i-1)n/4$ to level $in/4$. View each node v at a level that is a multiple of $n/4$ as the root of an R-way subbranching program of height $n/4$ consisting of all nodes reachable from v in the layer whose levels start at v's level.

There are $T/(n/4) = 4T/n$ layers in $\mathcal{B}$. By Lemma 2, a large fraction of x in $[1, n]^n$ require at least $n/(2e)$ output values. For each such input x, at least $(n/(2e))/(4T/n) = n^2/(8eT)$ outputs must be made during some single layer. An input reaches at most one node at each level and so reaches the root of only one subbranching program per layer. Now, for an input x chosen at random from U_n^n, by Lemma 3 each subbranching program can correctly output $\geqq n^2/(8eT)$ values on x only with probability $<e^{-(n^2/16eT)}$. (Note that $n^2/(8eT)<n/4$ since $T \geqq n$.)

Consider the probability, for x chosen at random from U_n^n, that there exists a subbranching program in $\mathscr{B}$ that correctly outputs at least $n^2/(8eT)$ values on input x. Since there are only 2^S nodes in $\mathscr{B}$, the number of subbranching programs that need to be considered is no more than 2^S and thus this probability is less than $2^S e^{-(n^2/16eT)}$. But, by Lemma 2, for x chosen at random from U_n^n the probability is $>1/(2e-1)$ that the Unique Elements problem requires at least $n/(2e)$ output values. Therefore $\mathscr{B}$ must have

$$2^S e^{-(n^2/16eT)} > \frac{1}{2e-1}$$

so that $S=\Omega(n^2/T)$, i.e., $ST=\Omega(n^2)$. □

Because the proof technique for Theorem 1 is probabilistic, it can be applied to show that the tradeoff for Unique Elements holds for *average* time and space as well (see [Abr86]) in the case that the input integers are chosen uniformly from $[1, n]$.

Any problem for which n input variables must be read before some output is produced requires branching program time $T \geqq n$ and therefore space $S \geqq \log n$. Thus, for inputs in the range $[1, n]$, the following theorem demonstrates that the tradeoff in Theorem 1 is optimal.

Theorem 4. *For any S with $n \geqq S \geqq \log n$ there is an n-way branching program that solves the* Unique Elements *problem for inputs in the range $[1, n]$ using $O(S)$ space and $O(n^2/S)$ time.*

Proof. The n-way branching program is a straightforward implementation of the following RAM program:

```
Algorithm Unique Elements.
$b \leftarrow 0$
**for** $j = 1$ **to** $\lceil n/S \rceil$ **do**
  **for** $i = 1$ **to** $S$ **do**
      $A[i] \leftarrow 0$
  **end for**
  **for** $i = 1$ **to** $n$ **do**
      **if** $b < x_i \leqq b + S$ **then do**
          $k \leftarrow x_i - b$
          **if** $A[k] < 2$ **then** $A[k] \leftarrow A[k] + 1$
      **end if**
  **end for**
  **for** $i = 1$ **to** $S$ **do**
      **if** $A[i] = 1$ **then** Output $b + i$
  **end for**
  $b \leftarrow b + S$
**end for**
```

Each of the S entries in the array A only contains either 0, 1, or 2, and the other variables only store values that require only $O(\log n)$ bits of storage. Each of the $O(n/S)$ passes through the outer loop uses only $O(n)$ time. Thus the program uses $O(S)$ space and $O(n^2/S)$ time. □

The technique of Theorem 4 does not apply if the range of inputs is significantly larger, say $[1, n^c]$ for $c > 1$. For inputs in this range the best upper bound known is an $O(n^2 \log n)$ time-space product used by a number of straightforward algorithms. If this really is the best possible then it leaves a $\log n$ gap which seems to be difficult to close using the approach of Theorem 1 since larger ranges of inputs only increase the likelihood that inputs are unique. It would be interesting to close this gap.

4. Sorting. Borodin and Cook's time-space tradeoff [BC82] of $S \cdot T = \Omega(n^2/\log n)$ for sorting n distinct integers actually holds for the RANKING problem on inputs in the range $[1, n^2]$. In order to get the bound for SORTING they use an easy reduction which uses small amounts of additional time and space from RANKING for inputs in the range $[1, n^2]$ to SORTING for inputs in the range $[1, n^3]$. In [RS82], Reisch's better bound of $S \cdot T = \Omega(n^2 \log\log n/\log n)$ is for RANKING distinct integers in the range $[1, n \log n/\log\log n]$ and gives a similar reduction in the range for SORTING. The above bounds for UNIQUE ELEMENTS will yield an improvement in both of these results.

With the output of the SORTING problem on inputs in the range $[1, n]$ it is possible to solve the UNIQUE ELEMENTS problem for inputs in the range $[1, n]$ using only small amounts of additional storage and time. Intuitively think of the index and value of the most recently generated output of the sorting program being stored along with a flag bit that is 1 if the stored value is the only one of its kind seen so far. When a new output for the sorting problem is produced it is compared with the stored value. If the flag bit is 1 and the compared values are different, then the stored value is output as a unique element. The flag and the stored value are then reset appropriately.

In the context of n-way branching programs the reduction is implemented by creating $2n$ copies of the program for SORTING to handle the different values of the stored input as well as the flag bit. The test of the flag bit and of the stored output of the sorting problem against the new one is handled implicitly since the state information of the modified branching program will be sufficient. The edges on which outputs occurred in the SORTING program have to be routed to the appropriate copy of the original program and the outputs for SORTING have to be replaced by the unique elements also where appropriate. This reduction uses no additional time and only $O(\log n)$ additional space. Thus the following corollary of Theorem 1 is obtained.

COROLLARY 5. *Any R-way branching program for* SORTING *for input integers in the range $[1, R]$, where $R \geqq n$, requires time T and space S such that $S \cdot T = \Omega(n^2)$.*

It is worth remarking that the bounds in [BC82] and [RS82] hold for *distinct* inputs to the SORTING problem. However, if the range of inputs is restricted to $[1, n]$, as could be the case for the bound proven here, then the problems for distinct inputs are trivially solvable. The statement to Corollary 5 does hold for distinct inputs except that R must be at least n^2 in this case. The reduction from UNIQUE ELEMENTS to sorting distinct values is achieved by appending an input's index to its value as the low order $\log n$ bits. The sorting algorithm is used as above except that outputs are not checked for simple uniqueness but rather for uniqueness in an interval of n consecutive values in $[1, R]$. In fact, in general, SORTING can be used in this same way to solve the following UNIQUE INTERVALS problem: Given an n-tuple of integers x in the range $[1, R]$, produce a list (in arbitrary order) of those $i \in [1, n]$ such that the interval $[(i-1)R'+1, iR']$ for $R' = \lceil R/n \rceil$ contains a unique value from x.

The average case argument for UNIQUE ELEMENTS with input integers chosen uniformly from $[1/n]$ that was alluded to in the last section can be used to show that the tradeoff for SORTING in Corollary 5 also holds for average time and space for input integers chosen uniformly from $[1, R]$ for $R \geqq n$: The key observation is that the probabilistic analysis of a branching program for the UNIQUE INTERVALS problem for a random input n-tuple in $[1, R]$ is essentially the same as for UNIQUE ELEMENTS in the range $[1, n]$. Then the reduction from the UNIQUE INTERVALS problem to SORTING gives the desired average case lower bound tradeoff.

The following theorem shows that Corollary 5 for SORTING is optimal for input integers in the range $[1, n]$ although the branching program that shows this is not obviously expressible as a RAM program, as was the case with the program for UNIQUE ELEMENTS in the proof of Theorem 4. As was the case for UNIQUE ELEMENTS, there is a $\log n$ gap for inputs in ranges such as $[1, n^2]$.

THEOREM 6. *For any S with $n \geqq S \geqq \log n$ there is an n-way branching program that solves the* SORTING *problem for inputs in the range $[1, n]$ using $O(S)$ space and $O(n^2/S)$ time.*

Proof. First consider the situation when $S = n$. The output of the SORTING problem only depends on the number of inputs of each value, not on their order in the input. There are $\binom{n+k-1}{n-1}$ ways of selecting k numbers in the range $[1, n]$. The branching program has a root node and a node for each of the ways of selecting k inputs from $[1, n]$ where $1 \leqq k \leqq n$. The program makes one pass through the inputs, traversing the nodes in the obvious way. On the edges leading to each node that describes the entire selection of n numbers, the entire sorted sequence is output. Since $\binom{n+k-1}{n-1} < 2^{n+k-1}$ the program uses $O(n)$ space and $O(n)$ time.

The modification of the branching program for smaller values of S gets a little trickier. It would be nice simply to make a pass through the inputs and record how many inputs there are of each value in the range 1 through S, then output that prefix of the sorted list and proceed on through the ranges $S+1$ to $2S$, $2S+1$ to $3S$, etc. The problem is that there might be too many inputs in some of these ranges. In order to handle this, the program never stores more than S values below the largest value in the range for which it is currently recording. When the $(S+1)$st value is found, the upper end of the range being recorded is lowered until the count of inputs below the largest value in the range is at most S. At the end of the pass through the input the portion of the sorted list corresponding to the range is output. Each range is initially set at length S and starts where the previous range left off.

To store the necessary markers and to record the number of inputs which take on the largest value of the range requires only $O(\log n)$ space and the record for the remainder of the range requires at most $O(S)$ space. Each pass through the input either reduces the number of inputs by S or outputs the sorted list for all values in a range of length S. Thus there are a total of $O(n/S)$ passes through the input for a total time of $O(n^2/S)$. □

5. Conclusions.

This paper shows the first optimal time-space tradeoff lower bounds for UNIQUE ELEMENTS and for SORTING that apply to any general sequential model that has a fair measure of space, without restrictions on its mode of accessing the inputs or on the structure of its computation. In comparison with previous work for SORTING, the bounds are better and the argument is considerably simpler. In addition, the UNIQUE ELEMENTS problem deals with questions of distinctness in a direct way and arguments about it may provide intuition that will be helpful for studying the element distinctness problem which seems to be the next natural problem to be attacked in the area of time-space tradeoffs for R-way branching programs.

There is a sense in which the lower bounds proven here for SORTING and those in [BC82] and [RS82] are orthogonal. The bounds in [BC82] and [RS82] hold for RANKING as well as for the SORTING problem. There do not seem to be any reductions using small time and space between the RANKING and UNIQUE ELEMENTS problems, so the previous bounds for RANKING are not improved by these results.

Clearly, the most interesting open problem in the area of time-space tradeoff is that posed in [BC82]—namely, to prove a nontrivial tradeoff for some decision problem in an unrestricted unstructured model like the R-way branching program. As mentioned above, the element distinctness problem seems to be a natural candidate. Good time-space tradeoff lower bounds have been shown for models with unstructured computation but restricted access to the inputs [Kar86], as well as for structured models with unrestricted access to the inputs [BFM$^+$87], [Kar86], [Yao88]. However, there appears to be a big stumbling block in the way of achieving similar results for R-way branching programs since, so far, the limit of time-space product lower bounds for R-way branching programs has been $O(nm)$ where n is the number of inputs and m is the number of outputs.

In one aspect, the inability to prove tradeoffs for decision problems may be due to a lack of intuition about measures of "progress" for solving them; in structured models a good measure of progress for the problem of element distinctness has led to interesting and nearly optimal time-space tradeoff lower bounds [BFM$^+$87], [Yao88]. However, it also seems likely that in addition to better measures of progress a more sophisticated handling of how branching programs make their progress is also needed. In the general framework for the time-space tradeoff shown here and in virtually all similar tradeoffs for branching programs it is granted that every input that could make good use of a subbranching program has been routed to the root of that subbranching program. The recent result of Yao [Yao88] for comparison branching programs uses a more careful accounting but it remains to be seen if even this accounting will be effective for R-way branching programs.

Acknowledgments. I would like to thank Richard Anderson, Larry Ruzzo, Martin Tompa, All Borodin, and Steve Cook for several helpful discussions and suggestions concerning these results.

REFERENCES

[Abr86] K. ABRAHAMSON, *Time-space tradeoffs for branching programs contrasted with those for straight-line programs*, in Proc. 27th IEEE Symposium on the Foundations of Computer Science, 1986, Toronto, Ontario, Canada, IEEE Computer Society, Washington, DC, pp. 402-409.

[Abr87] ———, *Generalized string matching*, SIAM J. Comput., 16 (1987), pp. 1039-1051.

[BC82] A. BORODIN AND S. COOK, *A time-space tradeoff for sorting on a general sequential model of computation*, SIAM J. Comput., 11 (1982), pp. 287-297.

[BFK$^+$81] A. BORODIN, M. FISCHER, D. KIRKPATRICK, N. LYNCH, AND M. TOMPA, *A time-space tradeoff for sorting on non-oblivious machines*, J. Comput. System Sci., 22 (1981), pp. 351-364.

[BFM$^+$87] A. BORODIN, F. FICH, F. MEYER AUF DER HEIDE, E. UPFAL, AND A. WIGDERSON, *A time-space tradeoff for element distinctness*, SIAM J. Comput., 16 (1987), pp. 97-99.

[Kar86] M. KARCHMER, *Two time-space tradeoffs for element distinctness*, Theoret. Comput. Sci., 47 (1986), pp. 237-246.

[RS82] S. REISCH AND G. SCHNITGER, *Three applications of Kolmogorov complexity*, in Proc. 23rd IEEE Symposium on the Foundations of Computer Science, 1982, Chicago, IL, IEEE Computer Society, Washington, DC, pp. 45-52.

[Yao88] A. YAO, *Near optimal time-space tradeoff for element distinctness*, in Proc. 29th IEEE Symposium on the Foundations of Computer Science, 1988, White Plains, NY, IEEE Computer Society, Washington, DC, pp. 91-97.

[Yes84] Y. YESHA, *Time-space tradeoffs for matrix multiplication and the discrete Fourier transform on any general sequential random-access computer*, J. Comput. System Sci., 29 (1984), pp. 183-197.