Lower Bounds for External Algebraic Decision Trees [*]

Jeff Erickson[†]

Abstract

We propose a natural extension of algebraic decision trees to the external-memory setting, where the cost of disk operations overwhelms CPU time, and prove a tight lower bound of $\Omega(n \log_m n)$ on the complexity of both sorting and element uniqueness in this model of computation. We also prove a $\Omega(\min\{n \log_m n, N\})$ lower bound for both problems in a less restrictive model, which requires only that the worst-case internal-memory computation time is finite. Standard reductions immediately generalize these lower bounds to a large number of fundamental computational geometry problems.

1 Introduction

Lower bounds for computational geometry problems are frequently stated in the *algebraic decision tree* model defined by Steele and Yao [15]. Examples include convex hulls, Voronoi diagrams, Euclidean minimum spanning trees, closest and farthest pairs, line segment intersections, and batched planar point location. Tight $\Omega(N \lg N)$ lower bounds in the algebraic decision tree model have been known for these problems for two decades; these lower bounds were all proved by reductions from sorting or element uniqueness [7, 14, 15]. On the other hand, geometric algorithms are usually designed around a core of low-degree polynomial primitives, such as triangle orientation tests and in-circle tests, which are treated as black boxes. Algebraic decision trees describe these algorithms perfectly.

More recently, researchers have developed efficient *external memory* or *out-of-core* algorithms for these same geometric problems [1, 5, 6, 10]. Algorithms in this model assume that the input data is too large to fit into main memory, and their analysis assumes that the cost of swapping blocks of data between memory and disk overwhelms the CPU time. The running time of an external-memory algorithm is the number of I/O operations it performs, expressed as a function of the input size N, the block size B, and the size of internal memory M. For notation convenience, we define $n = \lceil N/B \rceil$ and $m = \lceil M/B \rceil$. All the geometric problems listed above can be solved in $O(n \log_m n)$ I/O's, which is the same as the I/O bound for sorting. Tight I/O lower bounds for sorting and testing element uniqueness are known, but only for comparison-based external-memory algorithms [1] and for algorithms that accept only integer input [4]. Most interesting geometry problems cannot be solved with comparisons alone, and their most natural formulations call for arbitrary real-valued inputs. Arguably, therefore, no nontrivial lower bounds are known for the I/O complexity of these geometric problems.

In this paper, we define the first workable extension of the algebraic decision tree model to the external-memory setting. Using a generalization of Aggarwal and Vitter's argument for external comparison trees [1], we prove a tight lower bound of $\Omega(n \log_m n)$ on the complexity of sorting in this model. Similar arguments imply a tight lower bound of $\Omega(n \log_m n)$ for element uniqueness and other decision problems in the external *linear* decision tree model, but yield only trivial lower bounds for decision trees of higher degree. We strengthen this bound using a two-stage argument. First we describe a randomized adversarial reduction to element uniqueness from a new problem called *proximate visitors*, related to the proximate neighbor problem considered by previous authors [2, 8, 12]. We then prove that the proximate visitor problem requires $\Omega(n \log_m n)$ I/O's to solve in our model.

Finally, we briefly consider a less restrictive version of our model, which assumes only that the worst-case internal-memory computation time is *finite*; algorithms in this model may perform any finite number of arbitrary polynomial queries, of arbitrary degree, before each I/O operation. Even in this very permissive model, $\Omega(\min\{n \log_m n, N\})$ I/Os are required to solve the element uniqueness problem, and therefore to sort. This bound matches the number of I/O's required to permute N items into some desired order.

Standard linear-time reductions [14], originally developed for the internal-memory setting, immediately generalize all of our lower bounds to a huge number of computational geometry problems, including all the examples listed above.

[*]See http://www.cs.uiuc.edu/~jeffe/pubs/extlower.html for the most recent version of this paper.

[†]Department of Computer Science, University of Illinois at Urbana-Champaign, and LORIA-CNRS-INRIA Lorraine; jeffe@cs.uiuc.edu; http://www.cs.uiuc.edu/~jeffe/. Partially supported by NSF CAREER award CCR-0093348 and NSF ITR grants DMR-0121695 and CCR-0219594.

Previous and related work. Aggarwal and Vitter [1] introduced the standard external-memory model of computation and derived tight upper and lower bounds for several fundamental problems, including sorting and permuting. They proved a tight lower bound of $\Omega(n \log_m n)$ for in the *comparison I/O model*, in which internal-memory computation is restricted to comparisons between input records. The bound is derived by (over-)counting the number of permutations that can be sorted using a certain number of I/Os. This result was later generalized by Arge, Knudsen, and Larsen [3], who proved that an $\Omega(N \lg N)$ lower bound for *any* problem in the standard (internal-memory) comparison tree model implies an $\Omega(n \log_m n)$ bound for the same problem in the comparison I/O model. This reduction applies even to decision problems such as element uniqueness, where Aggarwal and Vitter's counting arguments cannot be used.

Aggarwal and Vitter's $\Omega(\min\{n \log_m n, N\})$ lower bound for permuting, on the other hand, does not restrict internal-memory computations at all, but does require the following *indivisibility* assumption: Only complete input records can be written to disk. This assumption is automatically satisfied in the comparison I/O model, as well as in the models we consider in this paper. Aggarwal and Vitter's lower bound was later strengthened and generalized by Arge [2], Chiang *et al.* [8], and Kameshwar and Ranade [12].

The strongest known lower bounds for several computational geometry problems were developed by Arge and Miltersen [4] in their *external-memory Turing machine* model of computation. *Any* external-memory algorithm that obeys the indivisibility assumption can be modeled as an external-memory Turing machine, provided the input consists of w-bit integers for some word length w. (Arge and Miltersen's lower bounds do not depend on w; algorithms in their model are assumed to be correct for any word length.) This model describes actual computation much more accurately than algebraic decision trees—after all, real computers can't manipulate arbitrary real numbers—but it forbids the natural abstract formulation of geometric problems with real-valued inputs.

Arge and Miltersen also consider an external-memory version of the algebraic computation tree model of Ben-Or [7]. Arge and Miltersen showed that *any* continuous function can be approximated arbitrarily closely by an external-memory algebraic computation tree that uses just $2n$ I/Os. Their universal approximation algorithm relies on two crucial properties of the computation model: (1) intermediate algebraic results can be kept in internal memory across I/O operations, and (2) the model does not require finite worst-case internal-memory computation time. The models considered in this paper have neither of these properties.

2 External-Memory Algebraic Decision Trees

We define the *external-memory algebraic decision tree model* as follows. Let Q be a fixed finite set of multivariate *query polynomials*. Let t denote the maximum number of terms in any polynomial in Q, and let d denote the the maximum degree of any term. The parameters t and d and the cardinality of Q are considered fixed constants, independent of the block size B, the memory size M, or the input size N. (Our lower bounds will ultimately depend on these constants.) We also assume that B and M are fixed; we are primarily interested in the asymptotic running time of our algorithms as N tends to infinity.

We model an algorithm by a family of decision trees, one for each possible input size N. The input to the Nth tree is a real vector in $\mathbb{R}^N$, which we imagine to be stored in the first n blocks on disk. In each tree, each internal node represents the following sequence of actions:

1. Write B arbitrary real numbers from internal memory to an unused disk block. Without loss of generality, if this is the Tth output operation, the numbers are written to the $(n+T)$th block on disk. Enforcing this assumption increases the number of I/Os by at most a constant factor. Also without loss of generality, we assume that the numbers within the block are written in sorted order.

2. Read a block of B real numbers from disk, replacing B arbitrary numbers in main memory. The block to be read must be either an input block or a previously written block.

3. Compute the sign of *every* polynomial in Q at *every* possible tuple of real numbers in main memory.

4. Branch to the child of v corresponding to the vector of signs computed in the previous step.

Each leaf is labeled with a possible output value. The running time of the algorithm is the maximum depth of the Nth tree, expressed as a function of the input size N. This cost counts only the number of I/O operations; the actual cost of computing the signs of query polynomials is completely ignored.

Restricting the entire family of decision trees to the same finite set Q of query polynomials imposes some weak but natural uniformity on the model. One can think of Q as the set of algebraic primitives employed by some uniform geometric algorithm. However, our lower bounds do not require true uniformity; trees for different input sizes may have radically different structures, or even different query polynomials as long as $|Q|$, d, and t remain constant. An external algebraic decision tree whose query polynomials all have degree 1 is called an external *linear* decision tree. The external *comparison* trees of Aggarwal and Vitter [1] are just external linear decision trees where only the only query polynomial is $x - y$.

3 Lower Bounds for Sorting

In this section, we generalize Aggarwal and Vitter's $\Omega(n \log_m n)$ sorting lower bound to the external algebraic decision-tree model. To help keep our argument self-contained, we include a slight simplification of their proof here.

Theorem 3.1 ([1]). *Any external-memory comparison tree that sorts an array of N elements has depth $\Omega(n \log_m n)$.*

Proof: Without loss of generality, we assume that N is a multiple of B and that each block in the input array is initially sorted; we can enforce the second precondition by making a single scan over the data in $O(n)$ I/Os. These simplifying assumptions imply that the number of possible output permutations is exactly $N!/(B!)^n$.

We can also safely assume that after the algorithm dumps any block to disk, it sorts the remaining $M - B$ elements in internal memory, because such an internal sort requires no additional I/O operations. Thus, when the algorithm loads a block of B numbers from disk, the only useful comparisons are between one of the B new elements and one of the $M - B$ old elements already in memory. It follows that the degree of any node in any external comparison tree is at most $\binom{M}{B}$. (The degree could be smaller than $\binom{M}{B}$ if some old number and some new number were compared in some earlier node, but this can only help us.)

We conclude that any external comparison tree for sorting has at least $N!/(B!)^n > n^N$ leaves, and each internal node has degree at most $\binom{M}{B} < M^B/B!$, so its depth is at least

$$\begin{aligned}\frac{\log(n^N)}{\log(M^B/B!)} &= \Omega\left(\frac{N \log n}{B \log M - B \log B}\right) \\ &= \Omega\left(\frac{N \log n}{B \log m}\right) \\ &= \Omega(n \log_m n). \qquad \square\end{aligned}$$

First we generalize this proof to linear decision trees.

Theorem 3.2. *Any external-memory linear decision tree that sorts an array of N elements has depth $\Omega(n \log_m n)$.*

Proof: Fix a finite set Q of linear forms over the formal variables $x_1, x_2, \ldots, x_t$. As in the previous proof, we assume that the input blocks are presorted, so the number of possible output permutations is $N!/B!^n > n^N$. (This assumption is only reasonable if the set Q contains some linear form $x_i - x_j$; on the other hand, if Q does not contain such a linear form, the algorithm cannot sort at all!)

Consider an arbitrary linear form $\alpha_0 + \sum_{i=1}^t \alpha_i x_i$ in Q. During the execution of a single I/O node, we evaluate this form on all possible t-tuples of numbers in internal memory. In order for a t-tuple to yield any new information, at least one of its t components must come from the new block. We can safely assume that the variable x_1 is always assigned a new value (and that $\alpha_1 \neq 0$), by increasing the number of forms in Q by at most a factor of t if necessary. We can express a test of the sign of this linear form as the query

$$x_1 \lesseqgtr -\frac{\alpha_0}{\alpha_1} - \sum_{i=2}^{t} \frac{\alpha_i}{\alpha_1} x_i ?$$

The results of all such queries are determined by the relative order of the new block and the set

$$\widehat{\mathcal{M}} = \left\{ -\frac{\alpha_0}{\alpha_1} - \sum_{i=2}^{t} \frac{\alpha_i}{\alpha_1} x_i \;\middle|\; x_2, \ldots, x_t \in \mathcal{M} \right\},$$

where $\mathcal{M}$ is the set of M numbers in internal memory. $\widehat{\mathcal{M}}$ clearly has at most M^{t-1} elements, so the total number of outcomes of all evaluations of our linear form is at most $\binom{M^{t-1}}{B} < M^{(t-1)B}/B!$.

It follows that any internal node in an external-memory linear decision tree has degree at most $(M^{(t-1)B}/B!)^{|Q|}$. (This upper bound is a gross overestimate; outcomes of tests with different linear forms are generally not independent.) We conclude that the depth of the tree is at least

$$\frac{N \log n}{|Q| \, \log\left(M^{(t-1)B}/B!\right)} = \Omega\left(\frac{n \log_m n}{(t-1)\,|Q|}\right).$$

Since t and $|Q|$ are fixed constants, the proof is complete. $\square$

Further generalizing our proof to algebraic decision trees is now straightforward.

Theorem 3.3. *Any external-memory algebraic decision tree that sorts an array of N elements has depth $\Omega(n \log_m n)$.*

Proof: Fix a finite set Q of query polynomials. We assume without loss of generality that each polynomial in Q has exactly t terms, each the product of exactly d variables, by keeping the constants 0 and 1 in internal memory at all times if necessary. As before, we assume that the input blocks are presorted, so the number of possible output permutations is $N!/B!^n > n^N$.

Consider an arbitrary query polynomial

$$\sum_{i=1}^{t} \alpha_i \prod_{j=1}^{d} x_{ij}$$

in Q. In order for an evaluation of this polynomial to yield any new information, at least one variable in some non-zero term must be assigned a value from the new block. We can safely assume that the the variable x_{11} is always assigned a new value and that no variable x_{1j} is assigned the value 0, by increasing the number of polynomials in Q by at most a factor of t if necessary. We can express testing the sign of this polynomial as the query

$$x_1 \lessgtr \frac{-\sum_{i=2}^{t} \alpha_i \prod_{j=1}^{d} x_{ij}}{\alpha_{11} \prod_{j=2}^{d} x_{1j}}?$$

The results of all such queries are determined by the relative order of the new block and the set

$$\widehat{\mathcal{M}} = \left\{ \frac{-\sum_{i=2}^{t} \alpha_i \prod_{j=1}^{d} x_{ij}}{\alpha_{11} \prod_{j=2}^{d} x_{1j}} \;\middle|\; x_{ij} \in \mathcal{M},\, x_{1j} \neq 0 \text{ for all } i,j \right\},$$

where $\mathcal{M}$ is the set of M numbers in internal memory. The set $\widehat{\mathcal{M}}$ clearly has at most M^{dt-1} elements, so the total number of outcomes from evaluating this polynomial is at most $\binom{M^{dt-1}}{B}$.

Following the analysis in the previous proof, we conclude that any external algebraic decision tree that sorts N elements must have depth at least

$$\Omega\left(\frac{n \log_m n}{(dt-1)|Q|}\right) = \Omega(n \log_m n). \qquad \square$$

4 Lower Bounds for Decision Problems

The proofs in the previous section relied on the fact that the sorting problem has a large number of possible outputs. In the standard algebraic decision tree model, lower bounds for *decision* problems, where there are only two possible outputs, can be proved by exploiting topological properties of the set of inputs that lead to each answer.

We consider the following decision problems, each of which can be solved in $O(n \log_m n)$ I/Os by sorting and scanning the data once. In internal memory, Dobkin and Lipton [9] proved $\Omega(N \log N)$ lower bounds for all these problems in the linear decision tree model. Their technique was generalized to algebraic decision and computation trees by Steele and Yao [15] and Ben-Or [7], again yielding $\Omega(N \log N)$ lower bounds.

- Element Uniqueness: Given an unsorted array $A[1..N]$ of real numbers, is $A[i] = A[j]$ for any pair of indices $i \neq j$?
- Set Intersection: Given two arrays $A[1..N]$ and $B[1..N]$ of distinct real numbers, at most one of which is sorted, is $A[i] = B[j]$ for any pair of indices i, j?
- Set Equality: Given two arrays $A[1..N]$ and $B[1..N]$ of distinct real numbers, at most one of which is sorted, is there a permutation π such that $A[i] = B[\pi(i)]$ for all i?

4.1 Counting components. In the external *linear* decision tree model, tight lower bounds for these problems follow essentially from Dobkin and Lipton's component-counting argument [9].

Theorem 4.1. *Any external linear decision tree that solves either* Element Uniqueness, Set Intersection, *or* Set Equality *for inputs of size N has depth $\Omega(n \log_m n)$.*

Proof: As in the previous proofs, we assume without loss of generality that each lock of the input data is sorted. This assumption restricts the input vector to a convex region of $\mathbb{R}^N$. For each problem, the set W of negative instances has $N!/(B!)^n$ connected components. The inputs that traverse any root-to-leaf path in an external linear decision tree satisfy a finite sequence of linear inequalities, so they comprise a convex polytope, which is necessarily connected. Thus, the set of inputs that reach each leaf can intersect at most one connected component of W. We conclude any external linear decision tree for Element Uniqueness has at least $N!/(B!)^n$ leaves. The lower bound now follows from the same argument as Theorem 3.2. $\square$

Unfortunately, this argument does not generalize to the external algebraic decision tree model, because the set of inputs that traverse a root-to-leaf path can have a huge number of components. Following the proofs of Steele and Yao [15] and Ben-Or [7], we can bound the number of components using the following classical result of Petrovskiĭ and Oleĭnik [13], Thom [16], and Milnor [11].

Lemma 4.2. *A semi-algebraic set in $\mathbb{R}^N$ defined by h polynomial inequalities, each with degree d, has at most at most $d(2d-1)^{N+h-1}$ connected components.*

Each node in an external algebraic decision tree performs at most $M^{dt|Q|} = M^{O(1)}$ queries of degree at most d. Thus, the Petrovskiĭ-Oleĭnik-Thom-Milnor bound implies that the set of points traversing any path of length T has at most $d(2d-1)^{N+TM^{O(1)}}$ components. Our earlier arguments imply that if the tree has depth T, then it has at most $\binom{M^{O(1)}}{B}^{O(T)}$ leaves. If the tree is correct, the total number of connected components in all leaves is at least the number of components of W:

$$d(2d-1)^{N+TM^{O(1)}}\binom{M^{O(1)}}{B}^{O(T)} \geq n^N.$$

This inequality implies that the depth of the tree must be at least

$$\Omega\left(\min\left\{n\log_m n, \frac{N\log n}{M^{O(1)}}\right\}\right).$$

Unfortunately, this lower bound is trivial unless N is exponential in M.

4.2 An adversarial reduction. We improve this lower bound using a different proof technique, which has two stages. The first stage is an adversarial reduction to ELEMENT UNIQUENESS from the following problem:

> PROXIMATE VISITORS: Given an unsorted array $A[1\,..\,N]$ on disk, bring each adjacent pair of the form $(A[\pi(2i-1)], A[\pi(2i)])$ together into internal memory, where π is any permutation such that $A[\pi(j)] \leq A[\pi(j+1)]$ for all j.

As the name suggests, this problem is similar to the *proximate neighbor* problem considered by Chiang *et al.* [8], Arge [2], and Kameshwar and Ranade [12]: Given an unsorted array $A[1\,..\,N]$ with $N/2$ distinct values, where each value appears exactly twice, permute the array so that equal values are adjacent on disk. Indeed, any algorithm for the proximate neighbor problem must route each proximate pair through internal memory, thereby solving the proximate visitor problem. However, our problem is easier, since it does not require any actual output; once any proximate pair $(A[\pi(2i-1)], A[\pi(2i)])$ 'visit each other' in internal memory, it can be discarded forever.

Lemma 4.3. *Any external algebraic decision tree that solves* ELEMENT DISTINCTNESS, SET INTERSECTION, *or* SET EQUALITY *can be forced to solve* PROXIMATE VISITORS.

Proof: We prove the lemma using a randomized adversary argument. We explicitly consider only ELEMENT DISTINCTNESS; the arguments for the other two problems are almost identical. Our adversary chooses $N/2$ real numbers $r_1, r_2, \ldots, r_{N/2}$ independently and uniformly at random from the unit interval $[0,1]$ and an arbitrary (*not* random) permutation $\pi \in S_N$. The adversary then presents the input array $A[1\,..\,N]$, where $A[\pi(2i-1)] = r_i$ and $A[\pi(2i)] = r_i + \varepsilon$ for each $1 \leq i \leq N/2$ and ε is some small positive value to be determined later. We choose $\varepsilon < 1/N^3$, so that with very high probability, every proximate visitor pair has the form $(A[\pi(2i-1)], A[\pi(2i)]) = (r_i, r_i + \varepsilon)$. We emphasize that our argument relies only on choosing a difficult *set* of input values; the permutation π plays no role in the proof.

We claim that with positive probability, any external-memory algebraic decision tree that solves ELEMENT DISTINCTNESS must bring each proximate pair $(A[\pi(2i-1)], A[\pi(2i)])$ into memory at some point during the computation. If some pair $(A[\pi(2i-1)], A[\pi(2i)])$ is never in memory together, the adversary can 'collapse' the pair, decreasing $A[\pi(2i)]$ by ε so that it equals $A[\pi(2i-1)]$, without modifying the sign of any query polynomial evaluated by the algorithm. Since the algorithm cannot distinguish between the original input and the collapsed input, even though one has a duplicate and the other does not, the algorithm must not be correct.

Let $T(A)$ be the number of polynomial queries evaluated by the algorithm given input A, and assume that each such query has total degree at most d. For notational convenience, let us write $x = A[\pi(2i-1)]$ and $y = A[\pi(2i)]$. Suppose the algorithm never has both x and y in memory simultaneously. By fixing the other $N-2$ input values, we can express any query polynomial on the computation path as a univariate polynomial in x, a univariate polynomial in y, or a constant. (A query polynomial involving both x and y can only be evaluated if x and y are together in memory.) We require that no univariate query polynomial has a root in the interval $[x, y]$. In other words, if ρ is a real root of any query polynomial, $x = r_i$ must not lie in the *forbidden interval* $[\rho-\varepsilon, \rho]$. Since each query polynomial has at most d real roots, the total length of all forbidden intervals is at most $d\varepsilon T(A)$.

Let $T(N) = \max_{|A|=N} T(A)$; this is the just the depth of the corresponding internal-memory algebraic decision tree. If $\varepsilon < 1/2dT(N)$, the proximate pair $(A[\pi(2i-1)], A[\pi(2i)])$ is forced together in internal memory with probability at least $1/2$, no matter what other values are stored in in input array. Since each random value r_j is generated independently, the probability that all $N/2$ proximate pairs must be forced together is at least $2^{-N/2} > 0$. □

4.3 Counting permutations. The second stage of our proof replaces the Petrovskiĭ/Oleĭnik/Thom/Milnor lemma with a more straightforward counting argument that bounds. Specifically, we count the number of input permutations that can reach a single leaf in any algebraic decision tree that solves PROXIMATE VISITORS. Our proof is similar to earlier lower-bound arguments for the proximate neighbor problem [2, 8, 12].

Lemma 4.4. *Any external-memory algebraic decision tree that solves* Proximate Visitors *for inputs of size N has depth $\Omega(n \log_m n)$.*

Proof: To simplify the proof, we assume without loss of generality that both N and M are integer multiples of $2B$, and that $N > M^5$. (If $N \leq M^5$, then $n \log_m n = \Theta(n)$, and the lower bound becomes trivial.) We also assume, as in our earlier proofs, that each block in the input array is initially sorted.

For purposes of analysis, we establish an ongoing record of the contents of internal memory, called the *trace*. Whenever a block of data is read from the disk, the trace records which B items in internal memory are discarded and which B new items are read in. In addition, after every I/O operation, the trace records the B items in internal memory that were least recently written to the trace. Thus, an algorithm that performs T I/O operations leaves a trace of length at most $3T$. The trace does not store the actual *values* of elements in memory, but rather their indices in the original input array. Under these assumptions, any algorithm that solves Proximate Visitors leaves a trace with the following property:

> Every pair $(\pi(2i-1), \pi(2i))$ appears in the trace at most $3M$ addresses apart, where π is a permutation that sorts the input.

We emphasize that this requirement is independent of the actual *values* in the input array; only their *order* matters.

Suppose some algorithm performs T different I/O operations on a given input. We define a graph G with N vertices, with an edge between vertices i and j if and only if the indices i and j appear in the trace at most $3M$ addresses apart. This graph has at most $9MT$ edges. The algorithm has solved the given instance of Proximate Visitors only if G contains an edge between every pair of the form $(\pi(2i-1), \pi(2i))$; these edges, if they exist, form a perfect matching in G. There are (crudely) at most $\binom{9MT}{N/2}$ perfect matchings in G, and exactly $(N/2)!2^{N/2}$ different input permutations can give rise to each perfect matching. It follows that this trace 'resolves' at most $\binom{9MT}{N/2}(N/2)!2^{N/2} \leq (18MT)^{N/2}$ different input permutations.

As we observed earlier, an external-memory algebraic decision tree with depth T has at most $\binom{M^{O(1)}}{B}^T < (M^{O(B)}/B!)^T$ leaves. We easily observe that the trace depends only on which root-to-leaf path is taken through the decision tree. Thus, an external-memory algebraic decision tree of depth T correctly solves Proximate Visitors for all possible inputs of length N only if each of the $N!/(B!)^n > n^N$ possible permutations is 'resolved' by at least one root-to-leaf trace:

$$(18MT)^{N/2}(M^{O(B)}/B!)^T \geq n^N.$$

(Different inputs with the same permutation may follow different paths through the decision tree, but this can only help us.) In particular, one of two inequalities must hold. If $(M^{O(B)}/B!)^T \geq n^{N/8}$, then our earlier analysis implies that $T = \Omega(n \log_m n)$. Otherwise, $(18MT)^{N/2} \geq n^{7N/8}$; in this case, since $n > N/M > N^{4/5}$, we have $T \geq n^{7/4}/18M \geq N^{7/5}/18N^{1/5} = \Omega(N^{6/5}) = \Omega(n \log_m n)$. □

Lemmas 4.3 and 5.1 immediately imply a tight lower bound for all three canonical decision problems.

Theorem 4.5. *Any external algebraic decision tree that solves* Element Uniqueness, Set Intersection, *or* Set Equality *for inputs of size N has depth $\Omega(n \log_m n)$.*

5 Lower Bounds for Finite-CPU Algorithms

Finally, we consider a much less restrictive version of our model of computation, which we call *naïve* external-memory algebraic decision trees. As in our standard model, algorithms are described by a family of decision trees, one for each possible input size N. Each internal node represents writing a block of data from memory to disk, reading a block of data from disk to memory, computing the signs of several polynomials over the data in memory, and branching on the resulting sign vector. However, the *only* further restriction we impose is that the number of polynomial queries performed in each node is *finite*; equivalently, we require only that the underlying internal-memory algorithm have finite worst-case running time. We do not require the query polynomials to come from a fixed set, to have a bounded number of terms, or even to have bounded degree. In this last respect, our model is even less restrictive than the classical internal algebraic decision tree model.[1]

To prove nontrivial lower bounds in our naïve model, we follow the same two-step approach as for our more restrictive model. In fact, the first step of our argument is unchanged; our proof of Lemma 4.3 only requires that the algorithm evaluates a finite number of polynomials during its execution. The second step, however, requires a slight modification, since the degree of a node is no longer bounded.

Lemma 5.1. *Any naïve external-memory algebraic decision tree that solves* Proximate Visitors *for inputs of size N requires $\Omega(\min\{n \log_m n, N\})$ I/Os in the worst case.*

[1]Lower bounds in the classical algebraic decision tree necessarily depend on the maximum degree of a query polynomial; recall that the element uniqueness problem can be solved by evaluating the sign of the single polynomial $\prod_{i<j}(x_i - x_j)$.

Proof: We adopt the same assumptions as in the proof of Lemma 5.1; in particular, we assume that $N > M^5$. To prove the lemma, it suffices to prove a lower bound on the worst-case number of I/Os required to produce a trace where every pair $(\pi(2i-1), \pi(2i))$ appears in the trace at most $3M$ addresses apart.

Each time the algorithm reads a block from disk, it chooses B items in memory to forget (after writing them to the trace) and a block of B items to read into memory (before writing them to the trace). After T I/O operations, there are at most $T+n$ different blocks that can be read by the next I/O operation. Thus, very crudely, at most $(\binom{M}{B}(T+n))^T < (2TM^B/B!)^T$ different traces can be produced in T I/Os.

Recall from the proof of Lemma 5.1 that a trace produced in T I/O operations resolves at most $(18MT)^{N/2}$ different input permutations. Thus, in order for every possible permutation to be resolved in T I/Os, we must have

$$(2TM^B/B!)^T(6MTN)^{N/2} \geq n^N.$$

In particular, one of three inequalities must hold. If $T^T \geq n^{N/8}$, then $T = \Omega(N)$, since $n \geq N/M > N^{4/5}$. If $(2M^B/B!)^T \geq n^{N/8}$, then $T = \Omega(n \log_m n)$ by our earlier analysis. Finally, if $(6MT)^{N/2} \geq n^{3N/4}$, then $T \geq n^{3/2}/6M = N^{6/5}/6M = \Omega(N)$. □

Theorem 5.2. *Any naïve external-memory algebraic decision tree that solves* Element Uniqueness, Set Intersection, *or* Set Equality *for inputs of size N has depth $\Omega(\min\{n \log_m n, N\})$.*

Since all three of these decision problems can be solved in $O(n)$ I/Os when the input is sorted, the following corollary is immediate.

Corollary 5.3. *Any naïve external-memory algebraic decision tree that sorts an array of N elements has depth $\Omega(\min\{n \log_m n, N\})$.*

Acknowledgments. Portions of this work were done at the Eindhoven-Carleton Workshop on Computational Geometry in October 2003. Thanks to Mark de Berg, Jit Bose, and Otfried Cheong for organizing the workshop, and to Erik Demaine for helpful discussions.

References

[1] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM* 31:1116–1127, 1988.

[2] L. Arge. The I/O-complexity of ordered binary-decision diagram manipulation. *Proc. 6th Annu. Internat. Sympos. Algorithms Comput.*, 82–91, 1995. Lecture Notes Comput. Sci. 1004, Springer-Verlag.

[3] L. Arge, M. Knudsen, and K. Larsen. A general lower bound on the I/O-complexity of comparison-based algorithms. *Proc, 3rd Workshop Algorithms Data Struct.*, 83–94, 1993. Lecture Notes Comput. Sci. 709, Springer-Verlag.

[4] L. Arge and P. B. Miltersen. On showing lower bounds for external-memory computational geometry problems. *External Memory Algorithms and Visualization*, 139–160, 1999. DIMACS series in Discrete Mathematica and Theoretical Computer Science 50, AMS.

[5] L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. S. Vitter. Theory and practice of I/O-efficient algorithms for multidimensional batched searching problems. *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, 685–694, 1998.

[6] L. Arge, D. E. Vengroff, and J. S. Vitter. External-memory algorithms for processing line segments in georgraphic information systems. *Proc. 3rd European Sympos. Algorithms*, 295–310, 1995. Lecture Notes Comput. Sci. 979, Springer-Verlag. Full version to appear in *Algorithmica*.

[7] M. Ben-Or. Lower bounds for algebraic computation trees. *Proc. 15th Annu. ACM Sympos. Theory Comput.*, 80–86, 1983.

[8] Y.-J. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter. External-memory graph algorithms. *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, 139–149, 1995. ⟨http://www.cs.brown.edu/cgc/papers/cggtvv-emga-95.ps.gz⟩.

[9] D. P. Dobkin and R. J. Lipton. On the complexity of computations under varying sets of primitives. *J. Comput. Syst. Sci.* 18:86–91, 1979.

[10] M. T. Goodrich, J.-J. Tsay, D. E. Vengroff, and J. S. Vitter. External-memory computational geometry. *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci.*, 714–723, 1993.

[11] J. W. Milnor. On the Betti numbers of real algebraic varieties. *Proc. Amer. Math. Soc.* 15:275–280, 1964.

[12] K. Munagala and A. Ranade. I/O-complexity of graph algorithms. *Proc. 10th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 687–694, 1999.

[13] I. G. Petrovskiĭ and O. A. Oleĭnik. On the topology of real algebraic surfaces. *Isvestia Akad. Nauk SSSR. Ser. Mat.* 13:389–402, 1949. In Russian.

[14] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[15] J. M. Steele and A. C. Yao. Lower bounds for algebraic decision trees. *J. Algorithms* 3:1–8, 1982.

[16] R. Thom. Sur l'homologie des variétés algébriques reélles. *Differential and Combinatorial Topology*, 255–265, 1965. Princeton University Press.