THE JOURNAL OF SYMBOLIC LOGIC
Volume 53, Number 4, Dec. 1988

ON THE USE OF INACCESSIBLE NUMBERS AND ORDER INDISCERNIBLES IN LOWER BOUND ARGUMENTS FOR RANDOM ACCESS MACHINES

WOLFGANG MAASS[1]

Abstract. We prove optimal lower bounds on the computation time for several well-known test problems on a quite realistic computational model: the random access machine. These lower bound arguments may be of special interest for logicians because they rely on finitary analogues of two important concepts from mathematical logic: inaccessible numbers and order indiscernibles.

§1. Introduction and definitions. We develop in this paper a new lower bound technique in order to prove optimal lower bounds on the computation time for several well-known test problems on a quite realistic computational model: the random access machine (RAM) with polynomially many registers. These lower bound arguments may be of special interest for logicians because they rely on finitary analogues of two concepts from mathematical logic: *inaccessible numbers* and *order indiscernibles*.

In particular we prove in §2 an optimal lower bound of $\Omega(n \log n)$ for the problems

$$\text{ELEMENT DISTINCTNESS} := \{\langle x_1, \ldots, x_n \rangle \in \mathbf{N}^n \mid x_i \neq x_j \text{ for } i \neq j\}$$

and

$$\text{DISJOINT SETS} := \{\langle y_1, \ldots, y_{n/2}, z_1, \ldots, z_{n/2} \rangle \in \mathbf{N}^n \mid n \text{ is even and } \{y_1, \ldots, y_{n/2}\} \cap \{z_1, \ldots, z_{n/2}\} = \varnothing\}$$

on random access machines where the number of registers is bounded by an arbitrary function of n. This yields the first nontrivial (i.e., superlinear) optimal lower bound on the computation time for a decision problem on a random access machine. We will also show that this lower bound of $\Omega(n \log n)$ is relatively stable insofar as it remains valid if we attach an arbitrary oracle $Q \subseteq \mathbf{N}^q$ to the RAM, which may be queried about arbitrary q-tuples of input numbers (q is an arbitrary constant).

Applications of our lower bound arguments to other problems will be discussed in Remark 2.5.

Received June 10, 1985; revised June 15, 1987.
[1]Supported in part by NSF grant DCR-8504247.

0022-4812/88/5304-0007/$02.20

The computational model that we consider in this paper, the random access machine (RAM), has become a standard model for the machine-independent analysis of the time and space complexity of concrete algorithms—see Cook and Reckhow [5], Aho, Hopcroft and Ullman [1], Hartmanis and Simon [10], Paul and Simon [20], and Klein and Meyer auf der Heide [12]. We apply in our lower bound arguments the usual uniform cost criterion, where one charges one unit for each execution of an instruction, independent of the size and address of the operands. The length of a RAM computation is defined as the number of instructions that are executed.

One assumes that the memory of a RAM consists of a sequence $r_0, r_1, r_2, \ldots$ of registers. Each register is capable of holding an integer (of arbitrary size). We write $\langle i \rangle$ for the current content of register r_i. All arithmetical operations take place in the first register r_0 (the "accumulator"). The n input numbers are located at the beginning of the computation in the registers $r_1, \ldots, r_n$ (one number per register).

A program for a RAM is a finite sequence of instructions $I_1, \ldots, I_p$ from the following standard instruction set:

$$\text{c-LOAD}(k)\colon \langle 0 \rangle \leftarrow k$$

(the constant k is placed into the accumulator r_0);

$$\text{LOAD}(k)\colon \langle 0 \rangle \leftarrow \langle k \rangle$$

(the content of register r_k is placed into the accumulator);

$$\text{i-LOAD}(k)\colon \langle 0 \rangle \leftarrow \langle\langle k \rangle\rangle$$

(the content of r_j is placed into the accumulator, where j is the absolute value of $\langle k \rangle$; this instruction uses "indirect addressing");

$$\text{STORE}(k)\colon \langle k \rangle \leftarrow \langle 0 \rangle$$

(the content of r_0 is placed into r_k);

$$\text{i-STORE}(k)\colon \langle\langle k \rangle\rangle \leftarrow \langle 0 \rangle$$

(the content of r_0 is placed into r_j, where j is the absolute value of the content of r_k; this instruction uses "indirect addressing");

$$\text{ADD}(k)\colon \langle 0 \rangle \leftarrow \langle 0 \rangle + \langle k \rangle$$

(the new content of r_0 is the sum of the preceding content of r_0 and the content of r_k);

$$\text{SUB}(k)\colon \langle 0 \rangle \leftarrow \langle 0 \rangle - \langle k \rangle$$

(subtraction);

$$\text{COMPARE}(j)\colon \text{if } \langle 0 \rangle > 0 \text{ then go to } I_j$$

(conditional jump: if $\langle 0 \rangle \leq 0$ then one proceeds as usual to the next instruction in the program, otherwise one jumps to instruction I_j);

ACCEPT: the computation halts and accepts the input;

REJECT: the computation halts and rejects the input.

The obvious advantage of the RAM-model is the fact that one can implement on it the standard algorithms in a rather straightforward manner. Furthermore, computation time (= number of instructions that are executed) and computation space (= number of registers that are used) on a RAM correspond quite well to time and space on a real computer. Both advantages distinguish the RAM from the Turing machine.

A less satisfactory aspect of RAM's is the fact that the definition of the instruction set is not canonical, although the preceding standard instruction set is in fact used almost everywhere in the literature (often with minor variations that change the time and space of the computation by at most a constant factor). Thus it is of some interest to know which changes in the instruction set of a RAM have a significant impact on the use of computational resources. This research area is still relatively unexplored because there are so few techniques for proving lower bounds on RAM's.

One natural candidate for the inclusion in the instruction set is the following instruction that allows us to multiply two register contents in one step:

$$\text{MULT}(k)\colon \langle 0\rangle \leftarrow \langle 0\rangle \cdot \langle k\rangle$$

(multiplication of the contents of r_0 and r_k). We will point out in Remark 2.2 that the inclusion of this instruction has a significant impact on the structure of RAM's that decide ELEMENT DISTINCTNESS: it reduces drastically the required number of conditional jumps in the computation.

In Theorem 2.3 we address another question with regard to other possible instruction sets. The conditional jump COMPARE in the previously defined standard instruction set tests whether the current content of the accumulator lies in the set $\{x|x > 0\}$. Although this "test predicate" $\{x|x > 0\}$ is quite natural, one can think of many other predicates of one or more variables that might potentially be useful as test predicates in a perhaps more powerful new type of conditional jump. We will show in Theorem 2.3 that the lower bound result of Theorem 2.1 is quite stable insofar as any new type of conditional jump with an *arbitrary* test predicate $Q \subseteq \mathbf{N}^q$ (for any constant q) that may be applied to arbitrary q-tuples of input numbers reduces the required number of conditional jumps for deciding ELEMENT DISTINCTNESS by at most a constant factor.

In analogy to corresponding concepts for Turing machines we will refer to such an arbitrary new test predicate Q as an "oracle". We will write R^Q for a RAM R that has in addition two new types of conditional jumps that allow the RAM to query the oracle $Q \subseteq \mathbf{N}^q$ about q-tuples $\langle x_{k_1}, \ldots, x_{k_q}\rangle$ of input numbers, which may be addressed directly or indirectly:

$\text{ORACLEQUERY}(j; k_1, \ldots, k_q)$: if $\langle x_{k_1}, \ldots, x_{k_q}\rangle \in Q$ then jump to instruction I_j;
otherwise go to the next instruction;

and

$\text{i-ORACLEQUERY}(j; i_1, \ldots, i_q)$: if $\langle x_{k_1}, \ldots, x_{k_q}\rangle \in Q$ where
$(k_s - 1) \equiv \langle i_s\rangle \pmod n$ for $s = 1, \ldots, q$,
then jump to instruction I_j;
otherwise go to the next instruction.

The result of Theorem 2.3 is only a partial stability result insofar as the conditional jumps of R^Q may only be applied to q-tuples of *input numbers*. This leaves open the question whether the same lower bound holds if the test predicate Q can be applied to arbitrary q-tuples of *register contents*. Nevertheless this partial stability result is of some technical interest because it exhibits a new feature of our lower bound argument which apparently distinguishes it from all preceding lower bound techniques for RAM's (see the references below). Previous lower bound arguments for RAM's relied on an analysis of the geometrical structure of the set of all inputs that follow a fixed computation path. Such geometrical arguments do not remain valid if one adds an arbitrary oracle Q to the RAM, because the geometrical structure of Q may be arbitrarily complicated.

With regard to the complexity measures that are considered in this paper we would like to point out that we measure here the complexity of a computation on an input $\langle x_1, \ldots, x_n \rangle$ in terms of the dimension n (or: number of "input words") of the input, not in terms of the number of input bits. This complexity measure is customarily used for the analysis of computation on RAM's and computation trees, where one charges correspondingly only one unit of time for each operation (independently of the size of the operands).

The main technical tool that we introduce in this paper is the construction of "hard" instances of the considered computational problems with the help of numbers x_i that are chosen to be "mutually inaccessible" from the point of view of the considered RAM. This means that the numbers x_i are spaced so far apart that the RAM cannot construct within the considered computation time any number $\geq x_{i_0}$ by using only numbers x_i with $x_i < x_{i_0}$. This technique may be viewed as an extension of a previously known method that focused on inputs $\langle x_1, \ldots, x_n \rangle$ that were not solutions of a number of linear or algebraic equalities (see Yao [24], Hong [11], and Paul and Simon [20]). Inputs $\langle x_1, \ldots, x_n \rangle$ that are constructed with the help of inaccessible numbers have an important further advantage: we know that certain minor perturbations of such inputs are also not solutions of the considered algebraic equalities. This novel stability property will in fact be the key for the "fooling argument" in our lower bound proofs. In the proof of Theorem 2.1 we will consider perturbations $\tilde{I}$ of the original inputs I and I' (I and I' are constructed from "inaccessible" numbers and are processed by the RAM in the same way) and we will be able to argue that for each COMPARE-instruction that is executed by the RAM the original inputs I, I' and the changed input $\tilde{I}$ lie on the same side of the hyperplane that is defined by this comparison.

We refer to Dietzfelbinger and Maass [6], [7] for further applications of "inaccessible" numbers in lower bound arguments.

The lower bound argument for oracle-RAM's in Theorem 2.3 uses as an additional tool, Ramsey's theorem from combinatorics (see [22] and [9]), in order to make the input numbers x_i "order indiscernible" with regard to an arbitrary fixed oracle $Q \subseteq \mathbf{N}^q$ (i.e., this oracle can *only discern the order* of any q-tuple $\langle x_{k_1}, \ldots, x_{k_q} \rangle$ of numbers from the input $\langle x_1, \ldots, x_n \rangle$). A useful property of the construction of "order indiscernibles" is the fact that it is compatible with the requirement of making the numbers x_i "mutually inaccessible".

The technique of selecting "order indiscernible" numbers from an infinite set via Ramsey's theorem is a standard tool in model theory (see [23]). The relevance of this

technique for lower bound arguments in complexity theory (where one applies it in a finite setting) had been realized independently by Moran, Snir and Manber [18], [19] (for decision trees) and the author of this paper [13] (for RAM's). Various other applications of Ramsey theory in complexity theory can be found for example in Yao [25], Pudlák [21], Chandra, Furst and Lipton [4], Meyer auf der Heide and Wigderson [17], and Alon and Maass [2].

The following other preceding results are related to the results of this paper. Paul and Simon [20] and Rackoff have proved optimal $\Omega(n \log n)$ lower bounds for the computation of several concrete functions on RAM's (in particular for sorting). Their lower bound arguments cannot be applied to decision problems.

Klein and Meyer auf der Heide [12] have extended methods of Dobkin and Lipton [8] and Paul and Simon [20] to prove an $\Omega(n^2)$ lower bound for the decision problem KNAPSACK on RAM's (this lower bound is probably not optimal since KNAPSACK is NP-complete). Meyer auf der Heide and Reischuk [15], [16] have used this method to prove an $\Omega(n^2 \log k)$ lower bound for a more complex problem $L_{n,k}$. These lower bound techniques can only yield lower bounds that lie *above* the $n \log n$ range because of the large number of "forbidden hyperplanes" that arise in the simulation of a RAM by a linear decision tree (therefore they cannot be applied to problems that have an *upper* bound of $O(n \log n)$).

There are, in addition, a number of related lower bound results for decision trees with nodes of constant degree. Such a decision tree is a somewhat weaker model than a RAM insofar as it does not allow indirect addressing (note that one *can* simulate a RAM with bounded memory by a decision tree with nodes of *large* degree, but no lower bound arguments are known for such tree). Ben-Or [3] has shown optimal $\Omega(n \log n)$ lower bounds for ELEMENT DISTINCTNESS, DISJOINT SETS and several other problems on a quite general type of decision tree with nodes of constant degree where several concrete algebraic operations are permitted. However, Ben-Or's argument does not remain valid if one allows to query an arbitrary oracle Q in the decision tree. On the other hand, Moran, Snir and Manber [18], [19] have shown a lower bound of $\Omega(n \log n)$ for ELEMENT DISTINCTNESS on a different type of decision tree where *only* queries to an arbitrary oracle $Q \subseteq \mathbf{N}^q$ are permitted (for $q \leq n^{1/2-\varepsilon}$), however no linear tests (note that a linear test, as it occurs for example in the computation of a RAM, may involve more than $n^{1/2}$ many input variables x_i and therefore cannot be viewed as a special case of such an oracle query). As mentioned above, this result uses also Ramsey's theorem. To our knowledge no lower bound argument was previously known that can be applied to a decision tree where both linear tests and queries to an arbitrary oracle are permitted.

§2. Lower bound arguments. We analyze the length of RAM-computations for the following two well-known decision problems:

$$\text{ELEMENT DISTINCTNESS} = \{\langle x_1, \ldots, x_n \rangle \in \mathbf{N}^n \mid x_i \neq x_j \text{ for } i \neq j\}$$

and

$$\text{DISJOINT SETS} = \{\langle y_1, \ldots, y_{n/2}, z_1, \ldots, z_{n/2} \rangle \in \mathbf{N}^n \mid n \text{ is even and } \{y_1, \ldots, y_{n/2}\} \cap \{z_1, \ldots, z_{n/2}\} = \emptyset\}.$$

We now consider the "test set" $T \subseteq$ ELEMENT DISTINCTNESS that consists of all $n!$ permutations of the n numbers in the set **IN**. We will show that R uses $\log(n!) = \Omega(n \log n)$ conditional jumps for some input from this "test set" T.

The strategy of the proof is as follows. We assign to every input $I \in T$ a binary sequence $P(I)$ which records the outcomes of all conditional jumps (=COMPARE-instructions) in the computation of R on input I. It is sufficient to show that $P(I) \neq P(I')$ for any two different inputs I, I' from the test set T (since $\log|T| = \log(n!) = \Omega(n \log n)$).

Assume for a contradiction that there are in T two different inputs $I = \langle x_1, \ldots, x_n \rangle$ and $I' = \langle x'_1, \ldots, x'_n \rangle$ with $P(I) = P(I')$. Let π be the permutation of $\{1, \ldots, n\}$ so that $x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}$. Choose l minimal so that $x'_{\pi(l+1)} < x'_{\pi(l)}$. Consider the input $\tilde{I} = \langle \tilde{x}_1, \ldots, \tilde{x}_n \rangle$ with

$$\tilde{x}_i = \begin{cases} x_i, & \text{if } i \neq \pi(l+1), \\ x_{\pi(l)}, & \text{if } i = \pi(l+1) \end{cases}$$

(in other words: one replaces $x_{\pi(l+1)}$ in I by another copy of $x_{\pi(l)}$). Obviously we have $\tilde{x}_{\pi(l)} = \tilde{x}_{\pi(l+1)}$ and thus $\tilde{I} \notin$ ELEMENT DISTINCTNESS. We will show that nevertheless all three inputs I, I' and $\tilde{I}$ are processed in the same way by R (the idea is that because R did not "notice" that the $\pi(l)$th and the $\pi(l+1)$th component have a different relative order in the inputs I and I', R will also not "notice" that the $\pi(l)$th and the $\pi(l+1)$th component of input $\tilde{I}$ are in fact equal). This implies the desired contradiction since in particular $\tilde{I}$ is accepted by R in the same way as I and I'. More precisely, the following claim implies that the instruction ACCEPT is the last instruction that is executed in all three computations.

Claim. At every step t, R executes for all three inputs I, I' and $\tilde{I}$ the same instruction, and if a conditional jump is executed at step t it has the same outcome for all three inputs. Furthermore if a register r_a holds for input I at the end of step t the number $s_0 + \sum_{i=1}^n s_i \cdot x_i$ with $s_i \in \mathbf{Z}$, $|s_i| \leq 2^{g(n)}$, then for input $I'(\tilde{I})$ the same register r_a holds at the end of step t the number $s_0 + \sum_{i=1}^n s_i \cdot x'_i$ $(s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}_i)$ with the same coefficients $s_0, \ldots, s_n$.

We will prove this claim by induction on t (the purpose of the second part of the claim is to keep this induction going). Before we proceed with the induction we first note that every register content that occurs in the computation of R on input $I = \langle x_1, \ldots, x_n \rangle$ can be written *uniquely* in the *normal form*

$$(*) \qquad s_0 + \sum_{i=1}^{n} s_i \cdot x_i, \quad \text{where } s_0, \ldots, s_n \text{ are integers of absolute value } \leq 2^{g(n)}$$

(the analogous fact holds for input $I' = \langle x'_1, \ldots, x'_n \rangle$). In order to prove this fact we first note that any number which is computed by R from the input numbers $x_1, \ldots, x_n$ with $\leq d$ arithmetical operations can be written in the form $s_0 + \sum_{i=1}^n s_i \cdot x_i$ with $s_i \in \mathbf{Z}$ and $|s_i| \leq 2^d$ (this is easily verified by induction on t). This implies for $d = g(n)$ that any register content of R in the considered computation on input $\langle x_1, \ldots, x_n \rangle$ can be written in the normal form $(*)$. Assume for a contradiction that this normal form is not unique and $s_0 + \sum_{i=1}^n s_i \cdot x_i = \tilde{s}_0 + \sum_{i=1}^n \tilde{s}_i \cdot x_i$, where $s_i, \tilde{s}_i \in \mathbf{Z}$, $|s_i|, |\tilde{s}_i| \leq 2^{g(n)}$ for all $i \in \{0, \ldots, n\}$, and $s_i \neq \tilde{s}_i$ for some $i \in \{0, \ldots, n\}$.

Obviously a RAM can decide both of these problems in $O(n)$ steps (write each input number x into the register with address x, but check first whether it is already occupied). However these algorithms require a very large number of registers and the question arises whether these problems can also be solved in linear time on a RAM with a "reasonable" memory size (e.g. polynomially in n many registers). There exist on the other hand trivial algorithms (via sorting) that solve both of these problems in $O(n \log n)$ steps with only $O(n)$ many registers. We show in Theorems 2.1 and 2.4 that this time bound is in fact optimal if one allows polynomially in n many registers, or any number of registers that is bounded by a function of n. Furthermore we will show in Theorem 2.3 that this lower bound remains valid even if one allows the RAM to query in addition an arbitrary oracle $Q \subseteq \mathbf{N}^q$ about arbitrary q-tuples of input numbers (q is a constant).

Lower bounds for further problems will be discussed in Remark 2.5. We will consider throughout this section RAM's that use in computations on n input numbers only registers whose address has at most $f(n)$ bits (i.e., only the registers $r_0, \ldots, r_{2^{f(n)}-1}$ are used), where $f: \mathbf{N} \to \mathbf{N}$ is an arbitrary function. We assume that in a situation where one of these registers r_k is used for indirect addressing and the absolute value of its current content $\langle k \rangle$ is larger than $2^{f(n)} - 1$, the low order $f(n)$ bits of $|\langle k \rangle|$ are interpreted as the intended address.

THEOREM 2.1. *Let R be a random access machine (RAM) that recognizes* ELEMENT DISTINCTNESS. *Assume that R uses for inputs from $\mathbf{N}^n$ only register addresses with at most $f(n)$ bits, where $f: \mathbf{N} \to \mathbf{N}$ is an arbitrary function. Then R needs $\Omega(n \log n)$ computation steps. This lower bound is optimal.*

REMARK 2.2. The proof of Theorem 2.1 provides some further information: there is *no tradeoff* where R can use *more* than $\Omega(n \log n)$ *arithmetical operations* in order to get away with $o(n \log n)$ *conditional jumps*. It turns out that whenever R executes $\leq g(n)$ arithmetical operations, where $g: \mathbf{N} \to \mathbf{N}$ is an *arbitrary* function, then R executes $\Omega(n \log n)$ conditional jumps.

Note that this refined lower bound *does not hold for RAM's with multiplication.* Such a RAM can compute the product $\prod_{i \neq j}(x_i - x_j)$ with $O(n^2)$ arithmetical operations, and then check with *one* conditional jump whether it has value 0.

PROOF OF THEOREM 2.1. Let R be a RAM as in the claim. Our goal is to show that R uses $\Omega(n \log n)$ computation steps for some input $I = \langle x_1, \ldots, x_n \rangle$. Thus we can assume that R executes less than $n \log n$ arithmetical operations on any such input (otherwise we are already done). However in order to prove the slightly stronger claim of Remark 2.2 we assume only that there is *some* function $g: \mathbf{N} \to \mathbf{N}$ with $g(n) \geq n$ so that R executes $\leq g(n)$ arithmetical operation for any input $I = \langle x_1, \ldots, x_n \rangle$. We will show under this weaker assumption that R executes $\Omega(n \log n)$ conditional jumps.

For the rest of the proof we fix some natural number n. Obviously R can generate with $\leq g(n)$ arithmetical operations, starting from numbers $\leq b$, only numbers of size $\leq 2^{g(n)} \cdot b$. Therefore the numbers in the set

$$\mathbf{IN} := \{2^{f(n)+2i \cdot g(n)} \mid 1 \leq i \leq n\}$$

are "mutually inaccessible" with regard to these computations of R on inputs that consist of numbers from **IN**.

Obviously this implies that $s_i \neq \tilde{s}_i$ for some $i \geq 1$. Choose $i_0 \geq 1$ such that $s_{i_0} \neq \tilde{s}_{i_0}$ and $s_i = \tilde{s}_i$ for all $i \geq 1$ with $x_i > x_{i_0}$. We then have (with $\max\{x_i \mid x_i < x_{i_0}\} := 1$ if $x_i > x_{i_0}$ for all $i \neq i_0$):

$$x_{i_0} \leq |s_{i_0} - \tilde{s}_{i_0}| \cdot x_{i_0} \leq |s_0| + |\tilde{s}_0| + \sum_{x_i < x_{i_0}} (|s_i| + |\tilde{s}_i|) \cdot x_i$$

$$\leq 2n \cdot 2^{g(n)} \cdot \max\{x_i \mid x_i < x_{i_0}\} < 2^{2g(n)} \cdot \max\{x_i \mid x_i < x_{i_0}\} \leq x_{i_0},$$

a contradiction. Thus the normal form (*) is unique.

With the help of the unique normal form (*) we can now analyze register contents of R on input $I = \langle x_1, \ldots, x_n \rangle$ as if they were elements of an $(n+1)$-dimensional vector space with basis $\{1, x_1, \ldots, x_n\}$ (over a suitable finite field). In particular we will consider for the two inputs $I = \langle x_1, \ldots, x_n \rangle$ and $I' = \langle x'_1, \ldots, x'_n \rangle$ a map between corresponding register contents for both inputs. This map can be viewed as a vector space isomorphism from the vector space with basis $\{1, x_1, \ldots, x_n\}$ onto the vector space with basis $\{1, x'_1, \ldots, x'_n\}$, where $1 \mapsto 1$ and $x_i \mapsto x'_i$.

We will now prove the preceding claim by induction on t. The case $t = 1$ is trivial. Assume that $t > 1$ and that the claim holds for all $\tilde{t} < t$. This implies in particular that the same instruction was executed at step $t-1$ for all three inputs I, I', $\tilde{I}$, and that in the case of a conditional jump at step $t-1$ the outcome was the same for all three inputs. Thus the same instruction is executed at step t for I, I' and $\tilde{I}$. It remains to show that in the case where an instruction COMPARE(j) is executed at step t the outcome is the same for I, I' and $\tilde{I}$ (this is the only nontrivial point of the whole inductive proof.) Let $s_0 + \sum_{i=1}^n s_i x_i$ be the content of r_0 at the end of step $t-1$ for input I (in normal form). By the induction hypothesis we know that $s_0 + \sum_{i=1}^n s_i \cdot x'_i$ ($s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}$) are the corresponding contents of r_0 at the end of step $t-1$ for input I' ($\tilde{I}$). Consider the case where $s_0 + \sum_{i=1}^n s_i \cdot x_i > 0$. By assumption we have $P(I) = P(I')$ and therefore $s_0 + \sum_{i=1}^n s_i \cdot x'_i > 0$. We have to show that $s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}_i > 0$. Let j be maximal so that $s_{\pi(j)} \neq 0$ (in the considered case this implies that $s_{\pi(j)} > 0$). If $j > l+1$ this immediately implies that $s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}_i > 0$. If $j < l+1$ then $s_0 + \sum_{i=1}^n s_i \cdot x_i = s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}_i$. Thus the only nontrivial case occurs when $j = l+1$. We have (by the choice of l) $x'_{\pi(1)} < \cdots < x'_{\pi(l)}$ and $x'_{\pi(l+1)} < x'_{\pi(l)}$; further, $s_{\pi(i)} = 0$ for $i > l+1$ (by the choice of j). Thus all $x'_i > x'_{\pi(l)}$ have the coefficient 0 in the term $s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}'_i$. On the other hand this term has a positive value (as noted above), which implies that $s_{\pi(l)} \geq 0$ (by the "inaccessibility" property of numbers in IN). Since by assumption $s_{\pi(l+1)} > 0$, and since the definition of input $\tilde{I} = \langle \tilde{x}_1, \ldots, \tilde{x}_n \rangle$ implies that $\tilde{x}_{\pi(1)} < \cdots < \tilde{x}_{\pi(l)} = \tilde{x}_{\pi(l+1)}$, these facts together imply that $s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}_i > 0$.

The case where $s_0 + \sum_{i=1}^n s_i \cdot x_i < 0$ is handled analogously, and $s_0 + \sum_{i=1}^n s_i \cdot x_i = 0$ implies that $s_0 = s_1 = \cdots = s_n = 0$ (by the uniqueness of the normal form).

In order to prove the second part of the claim for step t we first note that indirect addressing causes no problem. In the case where the instruction i-LOAD(k) or i-STORE(k) is executed at step t, when $s_0 + \sum_{i=1}^n s_i \cdot x_i$ is the content $\langle k \rangle$ of register r_k at the end of step $t-1$ for input I, we known by the induction hypothesis that $s_0 + \sum_{i=1}^n s_i \cdot x'_i$ ($s_0 + \sum_{i=1}^n s_i \cdot \tilde{x}_i$) is the corresponding register content for input I' ($\tilde{I}$). Furthermore, all numbers in IN are multiples of $2^{f(n)}$ and by our convention only the low order $f(n)$ bits of the absolute values of these register contents are

interpreted as address. Thus the indirect address depends only on s_0; in particular, it is the same address for all three inputs.

Finally we consider the case where ADD(k) is executed at step t (the case of SUB(k) is analogous) and $s_0 + \sum_{i=1}^{n} s_i \cdot x_i$ is the content (in normal form) of register r_0 at the end of step t for input I. If $s_0' + \sum_{i=1}^{n} s_i' \cdot x_i$ and $s_0'' + \sum_{i=1}^{n} s_i'' \cdot x_i$ are the contents (in normal form) of the registers r_0 and r_k at the end of step $t-1$ for input I, the uniqueness of the normal form implies that $s_i = s_i' + s_i''$ for $i = 0, \ldots, n$. Therefore the induction hypothesis implies that $s_0 + \sum_{i=1}^{n} s_i \cdot x_i' (s_0 + \sum_{i=1}^{n} s_i \cdot \tilde{x}_i)$ is the content of register r_0 at the end of step t for input I' ($\tilde{I}$). This finishes the proof of the claim. Thus the proof of Theorem 2.1 is complete.

Note. In order to show that a RAM with $2^{f(n)}$ registers needs $\Omega(n \log n)$ steps to decide ELEMENT DISTINCTNESS the proof of Theorem 2.1 requires the consideration of input numbers with $O(f(n) + n^2 \log n)$ bits. Of course the same lower bound holds for RAM's that are used to decide ELEMENT DISTINCTNESS for smaller input numbers if its algorithm (and its analysis) does not actually *exploit* that the input numbers are small.

THEOREM 2.3. *The lower bound of Theorem* 2.1 *remains valid for RAM's* R^Q *with an arbitrary oracle* $Q \subseteq N^q$ (*for an arbitrary constant* $q \in \mathbf{N}$).

PROOF. We only describe the changes that have to be made in the proof of Theorem 2.1. The key point is the fact that one can choose a set **IN** of n numbers that are "mutually inaccessible" *and* order indiscernible (see §1 for the definition).

Fix a RAM R^Q with oracle $Q \subseteq \mathbf{N}^q$ as in the theorem. We start with the *infinite* set

$$\widehat{\mathbf{IN}} := \{n \cdot 2^{f(n) + 2i \cdot g(n)} \mid i \in \mathbf{N}\}$$

of "mutually inaccessible" numbers. According to Ramsey's theorem ([22]; see also [9]) one can select from $\widehat{\mathbf{IN}}$ a set **IN** of n numbers that are *order indiscernible* with regard to Q, i.e., for any two q-tuples $\langle y_1, \ldots, y_q \rangle$ and $\langle z_1, \ldots, z_q \rangle$ of the same order type with $y_1, \ldots, y_q, z_1, \ldots z_q \in \mathbf{IN}$ one has

$$\langle y_1, \ldots, y_q \rangle \in Q \Leftrightarrow \langle z_1, \ldots z_q \rangle \in Q.$$

As before we consider the test set T of all $n!$ permutations of this new set **IN**. To each $I \in T$ one assigns a sequence $P(I) \in \{1, \ldots, q^q\}^*$ (we assume for simplicity that $q \geq 2$) which records for the computation of the RAM R^Q on input I the outcome of each conditional jump COMPARE and for each execution of an instruction ORACLEQUERY or i-ORACLEQUERY the order type of the q-tuple of numbers from **IN** about which the oracle is queried. In order to show that at least one of these sequences $P(I)$ has length $\Omega(n \log n)$ (the constant factor will depend on q) it is sufficient to show that the sequences $P(I)$, $I \in T$, are pairwise different.

Assume for a contradiction that there are in T two different inputs $I = \langle x_1, \ldots, x_n \rangle$ and $I' = \langle x_1', \ldots, x_n' \rangle$ with $P(I) = P(I')$. A third input $\tilde{I} = \langle \tilde{x}_1, \ldots, \tilde{x}_n \rangle \notin$ ELEMENT DISTINCTNESS is defined as in the proof of Theorem 2.1. One proves exactly the same claim for the computations on inputs I, I' and $\tilde{I}$, which yields the desired contradiction.

The inductive proof of this claim now requires a little bit more work because one has to take into account the two new conditional jumps ORACLEQUERY and

i-ORACLEQUERY in the first part of the claim. The indirect addressing in i-ORACLEQUERY causes no problem because the numbers in **IN** have been chosen to be multiples of n. Thus it is enough to show that in the case where an instruction ORACLEQUERY $(j; k_1, \ldots, k_q)$ is executed at step t for the inputs I, I' and $\tilde{I}$ the outcome is for all three inputs the same, i.e.,

$$\langle x_{k_1}, \ldots, x_{k_q} \rangle \in Q \Leftrightarrow \langle x'_{k_1}, \ldots, x'_{k_q} \rangle \in Q \Leftrightarrow \langle \tilde{x}_{k_1}, \ldots, \tilde{x}_{k_q} \rangle \in Q.$$

Since $P(I) = P(I')$, we know already that the q-tuples $\langle x_{k_1}, \ldots, x_{k_q} \rangle$ and $\langle x'_{k_1}, \ldots, x'_{k_q} \rangle$ have the same order type (which implies that $\langle x_{k_1}, \ldots, x_{k_q} \rangle \in Q \Leftrightarrow \langle x'_{k_1}, \ldots, x'_{k_q} \rangle \in Q$). Therefore not both of the indices $\pi(l)$ and $\pi(l+1)$ (notation as in the proof of Theorem 2.1) can occur among the indices $k_1, \ldots, k_q$ (since $x_{\pi(l)} < x_{\pi(l+1)}$ and $x'_{\pi(l)} > x'_{\pi(l+1)}$). This implies that $\langle \tilde{x}_{k_1}, \ldots, \tilde{x}_{k_q} \rangle$ has the same order type as $\langle x_{k_1}, \ldots, x_{k_q} \rangle$, and therefore $\langle \tilde{x}_{k_1}, \ldots, \tilde{x}_{k_q} \rangle \in Q \Leftrightarrow \langle x_{k_1}, \ldots, x_{k_q} \rangle \in Q$. This finishes the proof of Theorem 2.3.

We would like to point out that in the situation of Moran, Snir and Manber [19] (where one has no comparisons of arithmetical terms) the lower bound on the depth of the tree also holds if the arity q of the "oracle" grows with n (provided it does not grow too fast). However their method cannot be readily adapted to the situation considered here (because a comparison may depend on $\Omega(n)$ input numbers).

THEOREM 2.4. *The optimal lower bounds of Theorems* 2.1 *and* 2.3 *also hold for the problem* DISJOINT SETS.

PROOF. The proofs are the same as for ELEMENT DISTINCTNESS, except that here we have to make sure that the "test set" T is contained in DISJOINT SETS and that the "fooling input" $\tilde{I}$, that is constructed from inputs $I, I' \in T$ as in the proof of Theorem 2.1, lies outside of DISJOINT SETS.

The set $\mathbf{IN} = \{a_1, \ldots, a_n\}$ of n "mutually inaccessible" (and simultaneously order indiscernible) numbers is constructed exactly as in the preceding proofs. Assume that $a_1 < a_2 < \cdots < a_n$ and that n is even. The test set T consists of those $(n/2)!$ inputs $\langle a_1, a_3, a_5, \ldots, a_{n-1}, z_1, \ldots, z_{n/2} \rangle$, where $\langle z_1, \ldots, z_{n/2} \rangle$ is an arbitrary permutation of $\langle a_2, a_4, \ldots, a_n \rangle$. For inputs $I = \langle x_1, \ldots, x_n \rangle$, $I' = \langle x'_1, \ldots, x'_n \rangle$ and $\tilde{I} = \langle \tilde{x}_1, \ldots, \tilde{x}_n \rangle$ as in the proof of Theorem 2.1 one sees immediately that *exactly one* of the two indices $\pi(l)$, $\pi(l+1)$ is larger than $n/2$ (in any input from T consecutive elements of **IN** are located by construction in different "halves"). Therefore if one replaces $x_{\pi(l+1)}$ by $x_{\pi(l)}$ the resulting input $\tilde{I}$ is not in DISJOINT SETS. The desired lower bound follows since $\log(|T|) = \log((n/2)!) = \Omega(n \log n)$.

REMARK 2.5. Another problem for which the same lower bounds as in Theorem 2.1 and Theorem 2.3 can be shown is:

$$\text{EVEN PERMUTATIONS} := \{\langle x_1, \ldots, x_n \rangle \in \mathbf{N}^n \mid \text{the sign of permutation } \pi \text{ with } x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)} \text{ is even}\}.$$

In this case T consists of all $n!/2$ even permutations of **IN**. The "fooling input" $\tilde{I}$ is defined differently than in the previous cases: $\tilde{I}$ results from I by *exchanging* $x_{\pi(l)}$ and $x_{\pi(l+1)}$. It is obvious for $I \in$ EVEN PERMUTATIONS that $\tilde{I} \notin$ EVEN PERMUTATIONS.

This problem and the preceding two belong to the list of those problems for which Ben-Or [3] had already previously shown an $\Omega(n \log n)$ lower bound on algebraic

computation trees. However it is not true that one can automatically apply to every combinatorial problem from Ben-Or's list the lower bound arguments of this paper (SET EQUALITY is an example where one cannot construct a "test set" T as for the other problems above).

In [14] we have shown that one can apply the methods of this paper to prove an optimal quadratic lower bound on RAM's R^Q with arbitrary oracles Q for the following pattern matching problem:

$$\mathrm{PM} := \{\langle x_1, \ldots, x_n, d\rangle \mid x_1, \ldots, x_n \text{ are binary strings where some "pattern" of length } d \text{ occurs in at least two of these strings}\}.$$

Acknowledgements. The author would like to thank Alan Borodin, Leo Harrington, Richard Karp and Avi Widgerson for helpful conversations.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT and J. D. ULLMAN, ***The design and analysis of computer algorithms***, Addison-Wesley, Reading, Massachusetts, 1974.

[2] N. ALON and W. MAASS, *Ramsey theory and lower bounds for branching programs*, ***Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*** (1986), pp. 410–417.

[3] M. BEN-OR, *Lower bounds for algebraic computation trees*, ***Proceedings of the 15th ACM Symposium on Theory of Computing*** (1983), pp. 80–86.

[4] A. K. CHANDRA, M. L. FURST and R. J. LIPTON, *Multi-party protocols*, ***Proceedings of the 15th ACM Symposium on Theory of Computing*** (1983), pp. 94–99.

[5] S. A. COOK and R. A. RECKHOW, *Time bounded random access machines*, ***Journal of Computer and System Sciences***, vol. 7 (1973), pp. 354–375.

[6] M. DIETZFELBINGER and W. MAASS, *Two lower bound arguments with "inaccessible" numbers*, ***Proceedings of the structure in complexity theory conference (1986)***, Lecture Notes in Computer Science, vol. 223, Springer-Verlag, Berlin, 1986, pp. 163–183.

[7] ———, *Three lower bound arguments with "inaccessible" numbers*, to appear in a special issue of the ***Journal of Computer and System Sciences*** for the 1986 Structure in Complexity Theory conference.

[8] D. DOBKIN and R. LIPTON, *A lower bound of $n^2/2$ on linear search programs for the knapsack problem*, ***Journal of Computer and System Sciences***, vol. 16 (1975), pp. 417–442.

[9] R. L. GRAHAM, B. L. ROTHSCHILD and J. H. SPENCER, ***Ramsey theory***, Wiley, New York, 1980.

[10] J. HARTMANIS and J. SIMON, *On the power of multiplication in random access machines*, ***Conference Record of the 15th IEEE Symposium on Switching and Automata Theory*** (1974), pp. 13–23.

[11] J. HONG, *On lower bounds of time complexity of some algorithms*, ***Scientia Sinica***, vol. 22 (1979), pp. 890–900.

[12] P. KLEIN and F. MEYER AUF DER HEIDE, *A lower time bound for the knapsack problem on random access machines*, ***Acta Informatica***, vol. 19 (1983), pp. 385–395.

[13] W. MAASS, ***An optimal quadratic lower bound for random access machines and other applications of Ramsey's theorem***, Preliminary report, University of California, Berkeley, California, 1984.

[14] ———, ***On the use of inaccessible numbers and order indiscernibles in lower bound arguments for random access machines***, Research Report in Computer Science, no. 4, University of Illinois at Chicago, Chicago, Illinois, 1985.

[15] F. MEYER AUF DER HEIDE and R. REISCHUK, *On the limits to speed up parallel machines by large hardware and unbounded communication*, ***Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*** (1984), pp. 56–84.

[16] F. MEYER AUF DER HEIDE, *Lower bounds for solving linear Diophantine equations on random access machines*, ***Journal of the Association for Computing Machinery***, vol. 32 (1985), pp. 929–937.

[17] F. MEYER AUF DER HEIDE and A. WIGDERSON, *The complexity of parallel sorting*, ***Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*** (1985), pp. 532–540.

[18] S. MORAN, M. SNIR and U. MANBER, *Applications of Ramsey's theorem to decision tree complexity*, ***Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*** (1984), pp. 332–337.

[19] ———, *Applications of Ramsey's theorem to decision tree complexity*, ***Journal of the Association for Computing Machinery***, vol. 32 (1985), pp. 938–949.

[20] W. J. PAUL and J. SIMON, *Decision trees and random access machines*, ***Logic and algorithmic (symposium in honor of Ernest Specker, Zürich, 1980)***, Monographies de L'Enseignement Mathématique, vol. 30, Université de Genève, Geneva, 1982, pp. 331–340.

[21] P. PUDLÁK, *A lower bound on the complexity of branching programs*, ***Mathematical foundations of computer science (proceedings, Prague, 1984)***, Lecture Notes in Computer Science, vol. 176, Springer-Verlag, Berlin, 1984, pp. 480–489.

[22] F. P. RAMSEY, *On a problem of formal logic*, ***Proceedings of the London Mathematical Society***, ser. 2, vol. 30 (1930), pp. 264–286.

[23] G. E. SACKS, ***Saturated model theory***, Benjamin, Reading, Massachusetts, 1972.

[24] A. C. YAO, *On the complexity of comparison problems using linear functions*, ***Proceedings of the 16th IEEE Symposium on Foundations of Computer Science*** (1975), pp. 85–89.

[25] ———, *Should tables be sorted?* ***Journal of the Association for Computing Machinery***, vol. 28 (1981), pp. 615–628.

DEPARTMENT OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT CHICAGO
CHICAGO, ILLINOIS 60680