DECISION TREES AND RANDOM ACCESS MACHINES [1])

by W. Paul and J. Simon

Abstract: Unit cost random access machines which add, substract and multiply require time $\Omega(n \log n)$ to sort n numbers or to perform certain sequences of n information retrievial operations. If the random access machines can also use integer division and boolean operations these bounds do not hold.

1. Introduction and results

Decision trees are an abstraction of random access machines (RAMS). In the decision tree model nontrivial lower bounds for several sorting and searching problems are known [AHU]. In this paper we exhibit a technique which allows in some cases to turn these lower bounds into bounds for the original RAM model.

Our RAM model is the model described in [AHU] with instruction set $+$, $\dot{-}$, $\times$ and uniform cost measure. Registers are addressed by natural numbers; for natural numbers i, $< i >$ denotes the content of register i; A denotes the content of the accumulator i.e. the content of the register with address 0. A program P is a finite sequence of lines $(l_1, \ldots, l_n)$ where each line has the format "label: instruction".

W.l.o.g. we may assume that for all j the label of line l_j is j. The instructions can come from the following list.

$$A \leftarrow i, \quad A \leftarrow < i >, \quad A \leftarrow << i >>$$
$$< i > \leftarrow A, \quad << i >> \leftarrow A$$
$$A \leftarrow A \, o < i > \quad \text{where} \quad o \in \{+, \dot{-}, \times\}, \quad i \in \mathbb{N}$$
$$\text{if } A > 0 \text{ then goto label } j_1 \text{ else goto label } j_2 \text{ start, stop.}$$

Execution of 1 line is counted as 1 *step*. *Inputs* for our RAMS are sequences $\underline{x} = (x_1, \ldots, x_n)$ of natural numbers. They are assumed to be initially in registers $1, \ldots, n$. The content of register i after execution of the

[1]) Research partially supported by NSF grant.

program is the i'th *output*. Let $n \in \mathbf{N}$. A program *sorts* n numbers if for all input sequences $\underline{x} = (x_1, \ldots, x_n)$ the first n outputs produced by the program are $x_{\pi(1)}, \ldots, x_{\pi(n)}$ where π is a permutation of $(1, \ldots, n)$ and $x_{\pi(1)} \leqslant \ldots \leqslant x_{\pi(n)}$. Clearly n numbers can be sorted in $O(n \log n)$ steps say by merge sort.

THEOREM 1. *For large n the worst case complexity of any program that sorts n numbers is at least* $(n/2)(\log n - 2)$.

This theorem has independently been obtained by Charles Rackoff. For machines that also use division but no indirect addressing the same result has been proven by Hong Jiawei.

A program solves the *nearest neighbour problem* of size n if for all inputs $(x_1, \ldots, x_n, y_1, \ldots, y_n)$ the first n outputs produced by the program are $(x_{i_1}, \ldots, x_{i_n})$ where for all l

$$i_l \in \{ i \mid |x_i - y_l| \leqslant |x_j - y_l| \quad \text{for all} \quad j \} .$$

This problem can easily be solved on-line by using balanced trees, even without using multiplication instructions, in time $O(n \log n)$.

THEOREM 2. *For large n the worst case complexity of any program that solves the nearest neighbour problem of size n is at least* $(n/2)(\log n - 2)$.

A program solves the *symbol table problem* of size n if for all inputs $(x_1, \ldots, x_n, y_1, \ldots, y_n)$ the first n outputs produced by the program are $z_1, \ldots, z_n$ where for all l

$$z_l = \begin{cases} 0 & \text{if} \quad y_l \notin \{x_1, \ldots, x_n\} \\ \text{some index } i \text{ such that } y_l = x_i & \text{otherwise.} \end{cases}$$

This problem can be solved in real time by making $< x_i > = i$ for all $i \in \{1, \ldots, n\}$, even without using any arithmetic operations. The space (largest address visited) used by this program is not bounded by any function of n. Using balanced trees the problem can be solved on-line in time $O(n \log n)$ and space $O(n)$, thus space has been traded against time. This cannot be avoided:

THEOREM 3. *For large n the worst case complexity of any program that solves the symbol table problem of size n using space bounded by any function $f(n)$ is at least* $(n/2)(\log n - 2)$.

If the instruction set of our RAMS is augmented by integer division and by *boolean operations* (componentwise negation and componentwise AND of register-contents, which are imagined to be binary representations of numbers) the bounds of theorems 1 and 3 collapse.

THEOREM 4. RAMS *with integer division and boolean operations can sort n numbers and solve the symbol table problem of size n in time and space $O(n)$.*

2. LOWER BOUNDS

Let $P = ((1 : \alpha_1), ..., (p : \alpha_p))$ be a program which halts for all n-tuples of inputs $x = (x_1, ..., x_n)$ after at most t steps.

We associate with P a decision tree T, i.e. a rooted binary tree where to each node v an instruction $\alpha(v)$ is associated and some edges are labelled with "yes" or "no". The tree is an abstract model of computation and the instructions $\alpha(v)$ do not necessarily come from the list of instructions of the RAM. The tree T is inductively defined by the following process:

(2.1) If v is the root then $\alpha(v)$ is the start instruction.

(2.2) If $\alpha(v)$ is the stop instruction v is a leaf.

(2.3) If $\alpha(v)$ equals "if $A > 0$ then goto j_1 else goto j_2" then v is replaced by figure 1.

(2.4) If $\alpha(v) = \alpha_j$ and α_j equals "$A \leftarrow A \dot{-} < i >$" then v is replaced by figure 2.

(2.5) In all other cases if $\alpha(v) = \alpha_j$ then v gets one son v' and $\alpha(v') = \alpha_{j+1}$.

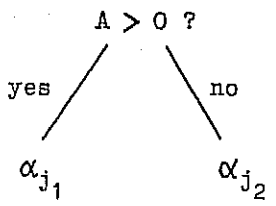

Fig. 1

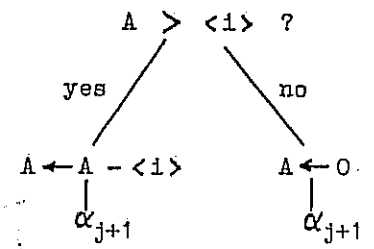

Fig. 2

In an obvious way the decision tree T "does the same" as the program. It may be infinite but because the program stops for all n-tuples of inputs after at most t steps no node in the tree of depth greater than $2t$ is reached. Therefore we may as well prune the tree at depth $2t$. Let $I = \{i_0, ..., i_m\}$ be the set of addresses occuring in the instructions of program P, w.l.o.g. $m \geqslant n$ and $i_0 = 0, ..., i_n = n$. With each node v of T and each $j \in \mathbf{N}$ we associate a polynomial $p_{v,j}$ in the indeterminates $X_1, ..., X_n$. The intended meaning of $p_{v,j}(\underline{x})$ is the content of register j if the input is $\underline{x}$ and control has just passed node v in the decision tree. The polynomials are inductively defined:

(2.6) If v is the root then

$$p_{v,j} = \begin{cases} X_j & \text{if} \quad j \in \{1, ..., n\} \\ 0 & \text{otherwise.} \end{cases}$$

If v is a son of u and $p_{w,j}$ is defined for all ancestors w of v (including u) and all $j \in \mathbf{N}$ then $p_{v,j}$ is defined in the following way:

(2.7) If $\alpha(v)$ equals $A \leftarrow A\, o < i >$, $o \in \{+, -, x\}$, then

$$p_{v,j} = \begin{cases} p_{u,0}\, o\, p_{u,i} & \text{if} \quad j = 0 \\ p_{u,j} & \text{otherwise.} \end{cases}$$

(2.8) If $\alpha(v)$ equals $< i > \leftarrow A$ or

$<< k >> \leftarrow A$ and $p_{u,k} \equiv i$ for some $i \in \mathbf{N}$ then

$$p_{v,j} = \begin{cases} p_{u,0} & \text{if} \quad j = i \\ p_{u,j} & \text{otherwise.} \end{cases}$$

(2.9) If $\alpha(v)$ is a test ($A > 0$? or $A > < i >$?) or the stop instruction or $\alpha(v)$ equals $<< k >> \leftarrow A$ and $p_{u,k} \not\equiv i$ for all $i \in \mathbf{N}$ then $p_{v,j} = p_{u,j}$ for all j.

(2.10) If $\alpha(v)$ equals $A \leftarrow i$ then

$$p_{v,j} = \begin{cases} i & \text{if} \quad j = 0 \\ p_{u,j} & \text{otherwise.} \end{cases}$$

(2.11) If $\alpha(v)$ equals $A \leftarrow < i >$ or $\alpha(v)$ equals $A \leftarrow << k >>$ and $p_{u,k} \equiv i$ for some $i \in \mathrm{N}$ then

$$p_{v,j} = \begin{cases} p_{u,i} & \text{if} \quad j = 0 \\ p_{u,j} & \text{otherwise.} \end{cases}$$

(2.12) If $\alpha(v)$ equals $A \leftarrow << k >>$ and $p_{u,k} \not\equiv i$ for all $i \in \mathrm{N}$ then trace the path in T from the root to v until the last node w such that $\alpha(w)$ equals $<< l >> \leftarrow A$ and $p_{w,l} \equiv p_{u,k}$. Then

$$p_{v,j} = \begin{cases} p_{w,0} & \text{if} \quad j = 0 \\ p_{w,j} & \text{otherwise.} \end{cases}$$

If no such node w can be found then

$$p_{v,j} = \begin{cases} 0 & \text{if} \quad j = 0 \\ p_{u,j} & \text{otherwise.} \end{cases}$$

So far this is just a formal definition. We have treated the polynomials associated with indirect address like values. Figure 3 illustrates a situation where the polynomials certainly do not have the intended meaning if $p_{u,k} \equiv p_{w,l} \not\equiv p_{z,q}$ but the input $\underline{x}$ is such that $p_{w,l}(\underline{x}) = p_{z,q}(\underline{x})$ and $p_{w,0}(\underline{x}) \neq p_{z,0}(x)$.

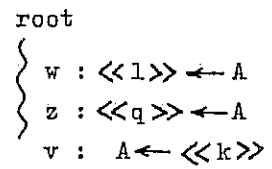

Fig. 3

Let $I' = I \cup \{ i \mid p_{u,k} \equiv i \text{ for some } u, k \}$.

Let Q be a set of polynomials in the variables $X_1, \ldots, X_n$. We call $\underline{x}$ *admissible* for Q if for all $p_1, p_2 \in Q$ with $p_1 \not\equiv p_2$ we have $p_1(\underline{x}) \neq p_2(\underline{x})$. By induction on the tree T one can show now, that for $i \in I'$ the values $p_{v,i}(x)$ have the intended meaning if $\underline{x}$ is admissible for $Q_T := I \cup \bigcup_{v,i} p_{u,i}$.

We explain only the inductive step for the case (2.12). By induction hypothesis $p_{u,k}(\underline{x})$ is the address of the cell whose content is transferred into the accumulator when control reaches node v. Suppose a node w can

be found on the path from the root of T to v such that $\alpha(w)$ equals $<< l >>$ $\leftarrow A$ and $p_{w,l} \equiv p_{u,k}$, and suppose w is the last such node. Again by induction hypothesis $p_{w,n}(\underline{x})$ is the value which was written into register $p_{w,l}(\underline{x})$ $= p_{u,k}(\underline{x})$ when control was at node w. We have to show that nothing was written into register $p_{u,k}(\underline{x})$ since control passed node w. Thus let w' be a node between w and w' and $\alpha(w')$ equals $< i > \leftarrow A$ or $<< m >> \leftarrow A$. In the first case $i \neq p_{u,k}(\underline{x})$ because $I \subset Q_T$, the polynomial $p_{u,k}$ is not constant and x is admissible for Q_T. In the second case by induction hypothesis $p_{w',m}(\underline{x})$ is the address of the cell into which the content of the accumulator is written when control is at node w'. By definition of w we have $p_{w',m} \neq p_{u,k}$, thus by the admissibility of x we can again conclude $p_{w',m}(\underline{x})$ $\neq p_{u,k}(\underline{x})$.

If no node w as above can be found then one proves in the same way that nothing was ever written into register $p_{u,k}(\underline{x})$ before control reached node v.

If Q is finite then the polynomial

$$R_Q := \prod_{\substack{p_i,\, p_j \in Q \\ p_i \neq p_j}} (p_i - p_j) \tag{2.14}$$

is non-zero and $\underline{x}$ is admissible for Q iff $R_Q(\underline{x}) \neq 0$.
If R is a polynomial,

$$R(\underline{X}) = \sum_{i=0}^{m} \sum_{\alpha_1 + \ldots + \alpha_n = i} a_\alpha X_1^{\alpha_1}, \ldots, X_n^{\alpha_n} \tag{2.15}$$

and $a_\alpha \neq 0$ for some α with $\alpha_1 + \ldots + \alpha_n = m$, then m is called the *degree* of R, deg $R = m$.

Let π be a permutation of $(1, \ldots, n)$. $\underline{x}$ has *order type* π if $x_{\pi(1)} < \ldots$ $< x_{\pi(n)}$. The existence of inputs of any order type which are admissible for Q_T follows from

LEMMA 1. *If R is a polynomial in $X_1, \ldots, X_n$, deg $R = m$ and π is a permutation of $(1, \ldots, n)$, then there is $\underline{x} \in \mathbb{N}^n$ of order type π such that $R(\underline{x}) \neq 0$ and $(x_1, \ldots, x_n) \in \{1, \ldots, m+n\}^n$.*

It suppices to *prove* the lemma for π the identical permutation. This is done by induction on n. For $n = 1$ the lemma holds because an m'th-degree polynomial in one variable has at most m zeros. For the induction step let

$$R(\underline{X}) = \sum_{i=0}^{s} R_i(X_1, \ldots, X_{n-1}) X_n^i. \tag{2.16}$$

Since $R \not\equiv 0$ we may assume $R_s \not\equiv 0$. Let $r = \deg R_s$. Then $r + s \leqslant m$. By induction hypothesis $R_s(x_1, \ldots, x_{n-1}) \neq 0$ for some $(x_1, \ldots, x_{n-1}) \in \{1, \ldots, r+n-1\}^{n-1}$ with $x_1 < \ldots < x_{n-1}$. Choose $x_n \in \{r+n, \ldots, r+s+n\}$ such that the s'th-degree polynomial

$$\sum_{i=0}^{s} R_i(x_1, \ldots, x_{n-1}) X_n^i$$

is non zero in x_n. □

Proof of theorem 1: Let P be a program that sorts n numbers, T the associated tree, π a permutation of $(1, \ldots, n)$, $\underline{x}$ an admissible input for Q_T of type π and v the leaf which is reached if the decision tree T is used with input $\underline{x}$. Then $p_{v,i}(\underline{x}) = x_{\pi(i)}$ for all $i \in \{1, \ldots, n\}$. Because $X_1, \ldots, X_n \in Q_T$ and $\underline{x}$ is admissible for Q_T this is only possible if $p_{v,i} \equiv X_{\pi(i)}$ for all i $i \in \{1, \ldots, n\}$. Therefore T has at least $n!$ leaves, hence its depth is not less than $n(\log n - 2)$. □

Proof of theorem 2: Let $(\underline{x}, \underline{y}) = (x_1, \ldots, x_n, y_1, \ldots, y_n) \in \mathbb{N}^{2n}$ and π a permutation of $(1, \ldots, n)$. We say $(\underline{x}, \underline{y})$ has *neighbour type* π if for all $i, j \in \{1, \ldots, n\}$ $|x_{\pi(i)} - y_i| \leqslant |x_j - y_i|$.

In a way analogous to the proof of lemma 1 one easily shows that if R is a polynomial in $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ and π is a permutation of $(1, \ldots, n)$ then there is $(\underline{x}, \underline{y}) \in \mathbb{N}^{2n}$ of neighbour type π such that $R(\underline{x}, \underline{y}) \neq 0$. Theorem 2 follows.

Proof of theorem 3: If $p(\underline{X}, \underline{Y})$ is a polynomial with variables $X_1, \ldots, X_n$, $Y_1, \ldots, Y_n$ and π a permutation of $(1, \ldots, n)$ then let

$$p^{\pi}(\underline{X}) = p(\underline{X}, X_{\pi(1)}, \ldots, X_{\pi(n)}).$$

Note that if $p^{\pi}(\underline{X}) \equiv f$ for some $f \in \mathbb{N}$, then $p(0, \ldots, 0) = f$, hence f is the constant term of p.

Now suppose the polynomials $p_{v,i}(\underline{X}, \underline{Y})$ have been defined for a tree T corresponding to a program P that solves the symbol table problem of size n in space $f(n)$, but (2.12) has been replaced by

(2.12)' If $\alpha(v)$ equals $A \leftarrow \langle\langle k \rangle\rangle$ and the constant term of $p_{u,k}$ is f then

$$p_{v,j} = \begin{cases} p_{u,f} & \text{if} \quad j = 0 \\ p_{u,j} & \text{otherwise.} \end{cases}$$

Let x be admissible for the set of polynomials

$$Q = \bigcup_{v,i,\pi} p^{\pi}_{v,i} \cup \{1, \ldots, f(n)\}.$$

We claim for all permutations π

(2.17) If during the computation with input $(x, x_{\pi(1)}, \ldots, x_{\pi(n)})$ node v in T is reached then for all $i \in \{1, \ldots, f(n)\}$ the values $p_{v,i}(\underline{x}, x_{\pi(1)}, \ldots, x_{\pi(n)})$ have the intended meaning.

This is proven by induction on the tree and it is obviously true for the root. Suppose it holds for the father u of v.

It only remains to consider the case where $\alpha(v)$ equals $A \leftarrow << k >>$. By the space bound and the induction hypothesis $p^{\pi}_{u,k}(x) \in \{1, \ldots, f(n)\}$. Because $\underline{x}$ is admissible for Q we conclude $p^{\pi}_{u,k}(\underline{X}) \equiv f$ for some $f \in \{1, \ldots, f(n)\}$. Moreover, f is the constant term of $p_{u,k}$. (2.17) now follows from the definitions of the polynomials $p_{v,j}$ and the induction hypothesis.

Now if π is a permutation and v the leaf of T which is reached with input $(\underline{x}, x_{\pi(1)}, \ldots, x_{\pi(n)})$ then $p^{\pi}_{v,i}(\underline{x}) = \pi(i)$. Because x is admissible for Q this implies $p^{\pi}_{v,i} \equiv \pi(i)$ which in turn implies that the constant term of $p_{v,i}$ is $\pi(i)$. The existence of $n!$ leaves in T follows. □

3. Upper bounds

Proof of theorem 4: a) For sorting: Given integers $\kappa_1, \ldots, \kappa_n$ in registers $x_1, \ldots, x_n$, find the maximum, say x_m of all κ_i. Let l be the length of the binary representation of x_m.

We shall compute the rank of every κ_i — i.e. compare each κ_i with all κ_j and count how many are greater than it. Note that if we have

$$A \ \ldots \ 1 \ \kappa_i \ 1 \ \ldots$$
$$B \ \ldots \ 0 \ \kappa_j \ 0 \ \ldots$$

in registers A and B, with κ_j, x_i in matching positions, then, regardless to the rest of register A and B, $C = A - B$ will contain a 1 in the bit positions immediately to the left of the most significant bit of κ_j (where A did have a 1) iff $\kappa_j \leq \kappa_i$.

So, to sort, we obtain, in a single register, n copies of $1\,\kappa_1\,1\,|\,\kappa_2\,1\,\ldots\,1\,\kappa_n\,1$ concatenated to each other, and $(0\kappa_1 0)_2\,(0\kappa_2 0)^n \ldots (0\kappa_n 0)^n$ in another with the length of all the x_i padded out to l with leading zeros. This can be done with $0(n)$ instructions. A single substraction yields all the comparisons. An AND with the bit string $(10^{l+1})^{n^2}$ retrieves them, and $O(\log n)$ shift and add moves compute the ranks at positions $i(l+2n)$. The κ_i can be ordered in $O(n)$ moves with this information, by retrieving the rank of each κ_i, and putting it in its proper position in a constant number of moves. □

b) For the symbol table: We show that the x_i's can be processed in linear time and that the y_i's can be processed in real time. We treat all the x_i's as l-bit strings for the same l. For $i \in \{1, \ldots, n\}$ bin (i) denotes the binary representation of i of length $\lfloor \log n \rfloor + 1$. Let $k = l + \lfloor \log n \rfloor + 2$. Suppose $x_1, \ldots, x_m$ have already been processed and $x_{i_1}, \ldots, x_{i_s}$ are the distinct values of $x_1, \ldots, x_m$. Suppose the following strings have been built up.

$$Z = Z_{i_s}, \ldots, Z_{i_1} \quad \text{where for each } j \qquad (3.1)$$
$$Z_{i_j} = 0\, x_{i_j}\, \text{bin}\,(i_j)\,.$$

$$M = (10^{k-1})^m\,. \qquad (3.2)$$

We show how to process x_{m+1}: Multiply the mask M by x_{m+1}, shift l bits right; a boolean "$\equiv$" with z gives 1^l in positions $k_{j-1}, \ldots, k_{j-l}$ iff $x_{m+1} = x_{i_j}$. Shifting M l bits right and adding it gives 1 in position k_j iff $x_{m+1} = x_{i_j}$. An AND with M shifted 1 to the left gives a string with at most one 1 which is in position k_j iff $x_{m+1} = x_{i_j}$. Test for 0 in order to decide if Z and M have to be up-dated. Updating can easily be done in constant time.

The problem of processing a y_i is almost identical with the problem of processing x_{m+1}. If the test for 0 above is negative one has to extract bin (i_j) from Z which is easy with the help of the string with the 1 in position k_j.

Acknowledgments: We thank Joel Seiferas for his help and comments.

REFERENCES

[AHU] AHO, HOPCROFT and ULLMAN. *The design and analysis of computer algorithms.* Addison Wesley 1974.

[J] „ JIAWEI. On lower bounds of time complexity of some algorithms. *Scientia sinica 23, 8/9,* pp. 890-900.

Wolfgang J. Paul

Fakultät für Mathematik
Universität Bielefeld
D-4800 Bielefeld 1

Janos Simon

Department of Computer Science
Penn State University
University Park, Pa 16802, USA