Theoretical Computer Science 43 (1986) 345–350
North-Holland

NOTE

A SIMPLE PROOF OF A TIME–SPACE TRADE-OFF FOR SORTING WITH LINEAR COMPARISONS *

Donald B. JOHNSON

Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755, U.S.A.

Communicated by R.M. Karp
Received December 1983
Revised December 1985

Abstract. It is shown how to extend the techniques originally used to prove a lower bound of $\Omega(n^2)$ for the product of the time and space consumed for sorting in branching programs with elementary comparisons, to the case of linear branching programs where linear functions on n input elements can be computed in unit time.

1. Introduction

The time complexity of sorting n keys $x=(x_1,\ldots,x_n)$ is well known to be $\Theta(n \log n)$ in the decision tree model of computation where comparisons on keys are restricted to linear combinations of the keys and each comparison is assumed to be done at unit cost. While decision trees capture faithfully the time expended by a variety of computations in which comparisons dominate other operations, they do not reflect the space consumed by such computations. To capture a notion of space consumption, an extension of the decision tree model called *branching programs* was introduced in [2]. Using this model, the authors of [2] showed that the time-space product for sorting in branching programs that allow only elementary comparisons satisfies $\Omega(n^2)$. Yao [3] has extended this result to branching programs where comparisons on linear combinations of the keys are allowed. Borodin and Cook [1] have extended these results to an even more general model, obtaining a bound of $TS=\Omega(n^2/\log n)$. In the present note we shall give a simplification of Yao's proof.

Yao's proof rests on two results. First, it is shown how to replace any linear branching program with a MIN branching program with the same time and capacity. A MIN branching program is one in which each non-leaf node v tests a subset X_v of the set of input values X, and branches according to the index of the smallest

* This research was partially supported by the National Science Foundation under Grant MCS-8002684.

value in X_v. The second is a result on slanted partial orders which is used to prove the following theorem.

Theorem ([3]). *The relation* $TS = \Omega(n^2)$ *holds for any* MIN *branching program of capacity* S *that sorts an input* x *with* n *elements in no more than* T *comparisons.*

We show how to extend the proof in [2] to obtain the theorem directly without using Yao's result on slanted partial orders.

2. Branching programs

An *elementary branching program* (*on n inputs*) $C^{(n)}$ is a directed graph with a distinguished *initial vertex* of indegree zero. A vertex with outdegree zero is a *leaf.* At each non-leaf vertex there is a comparison of the form $x_i : x_j$ where x is a vector $(x_1, \ldots, x_n)$ of input values. Each non-leaf vertex has three outgoing edges corresponding to the outcomes on $x_i : x_j$ of "<", "=", and ">". An elementary branching program takes any input x for which these outcomes are defined. Edges also specify zero or more outputs. The computations of an elementary branching program are analogous to those of computation trees. For any input, computation begins at the initial vertex and traverses some path to a leaf. The output of the program is the sequence, in path order, of outputs associated with the edges of the computation path traversed. We require that the program always reach a leaf in a finite number of steps. Therefore, we may define the time $T(C^{(n)})$ as the maximum over all inputs of the length of a computation path in $C^{(n)}$. Length is measured as the number of edges traversed. The capacity $S(C^{(n)})$ is defined as $\log_2(|C^{(n)}|)$, where $|C^{(n)}|$ is the number of vertices in $C^{(n)}$. Pippenger has observed (see [2]) that for any branching program there is an acyclic branching program which makes the same tests and performs the same computation in the same time bound T and has capacity no more than twice that of the given program. Therefore, for the purpose of lower bounds on the functional growth of the time-space product, we may assume that elementary branching programs are acyclic. This observation extends as well to MIN branching programs and linear branching programs, which we shall define now.

A *linear branching program* is defined similarly to an elementary branching program, except that it employs comparisons of the form $l(x) : 0$ where $l(x) = c + \sum_{i=1}^{n} \alpha_i x_i$ for real valued x, c, and $\{\alpha_i\}$. Each elementary branching program is isomorphic to a linear branching program for which, in any linear comparison $l(x) : 0$, $c = 0$ and $\alpha_i = 0$ for all i except for distinct i_1 and i_2 for which $\alpha_{i_1} = 1$ and $\alpha_{i_2} = -1$. A *MIN branching program* is also defined similarly to an elementary branching program, except that each non-leaf vertex v identifies $\text{MIN}(X_v)$ for some $X_v \subseteq X = \{x_1, x_2, \ldots, x_n\}$ and chooses its output edge according to index i of some $x_i = \text{MIN}(X_v)$. Elementary and MIN branching programs are *conservative* in the sense that the only operations allowed on elements of x are comparisons. A *tree program* is a branching program in which no vertex has indegree greater than one.

3. A lower bound on the depth of correct k-ranking programs

Let π_x denote a sorting permutation for x, that is, for all i and j, $1 \leq i, j \leq n$, $\pi_x(i) \leq \pi_x(j)$ implies $x_i \leq x_j$. Let M be a MIN branching program with output $(i_{j_1} : r_{j_1}, i_{j_2} : r_{j_2}, \ldots, i_{j_m} : r_{j_m})$ on an *allowed input* x (an input in which all elements are distinct). We call M a *k-big ranking program* if the following conditions are met:

(i) the output is correct ($\pi_x(i_{j_l}) = r_{j_l}$, for $l = 1, \ldots, m$), though it may not show the ranks of all n elements, and

(ii) at least k of the larger half of the input elements are ranked correctly (there exists a set $I \subseteq \{1, \ldots, n\}$ for which $|I| \geq k$, $\pi_x(i) \geq \lceil \frac{1}{2}n \rceil$ for $i \in I$, and for each $h \in I$, there is at least one index l, $1 \leq l \leq m$, for which $h = i_{j_l}$).

In [3] the quantity $\lceil \frac{1}{2}n \rceil$ is parameterized as n_0, but this generalization is unnecessary for the result. Any program that sorts must be a $\lfloor \frac{1}{2}n \rfloor$-big ranking program, that is, it has to get at least the $\lfloor \frac{1}{2}n \rfloor$ largest elements correct.

It follows from the input conditions that all outcomes implied by MIN(X_v) will be strict, that is, in $\{<, >\}$. Let the outcome $i > j$ define a directed edge from a point labeled i to a point labeled j. Then the MIN's performed on any path ϑ taken in M by some allowed input define a directed acyclic graph H_ϑ over the indices $\{i_1, \ldots, i_n\} = \{1, \ldots, n\}$ of the input x. We call the graph induced by these MIN operations a *Hasse diagram* over $\{i_1, \ldots, i_n\}$.

Vertex l *dominates* vertex m in H if there is a nontrivial directed path $(l, \ldots, m)$ in H. A permutation $\pi(i_1, \ldots, i_n)$ is *consistent* with a Hasse diagram H if $\pi(l) > \pi(m)$ whenever l dominates m.

Given a Hasse diagram H and a set of vertices $\{i_1, \ldots, i_k\}$, we denote the set of permutations π of x, consistent with H and in which $\pi(i_l) = r_l$ for $l = 1, \ldots, k$, as $P(H; i_1 : r_1, \ldots, i_k : r_k)$. We also define $D(H, i)$ to be the set of vertices that contains i and also all vertices that i dominates in H. Notice that $i_h \in D(H, i)$ implies $i \notin D(H, i_h)$ if $i \neq i_h$. As shown in [2], $|P(H; i : r)| \leq |P(H - i)|$, where $H - i$ is H from which i is removed and to which edge (h, l) is added whenever both edges (h, i) and (i, l) are present in H.

Lemma 3.1. *For any H, and for all r and i satifying $\lceil \frac{1}{2}n \rceil \leq r \leq n$ and $\pi(i) = r$ for some π consistent with H,*

$$\lceil \tfrac{1}{2}n \rceil |P(H; i : r)| \leq |D(H, i)||P(H)|.$$

Proof. The quantity $r|P(H - i)|$ is the number of permutations π, consistent with the class of allowed inputs and with $H - i$ and in which $\pi(i) \leq r$. To show that the expression $|D(H, i)||P(H)|$ is an upper bound for $n|P(H - i)|$, it will suffice to show how to assign each of these permutations π to a unique pair (i_m, π_l) where $i_m \in D(H, i)$ and $\pi_l \in P(H)$. As shown in [2], given π, let the path from i to i_m in H be $i = i_0, i_1, \ldots, i_m$. Thus, there is a unique sequence of elements $\Sigma = (\pi(i) = \pi(i_0) > \pi(i_1) > \cdots > \pi(i_m))$ and a π' defined by Σ for which the permutation $\pi'(\pi) = \pi_l$. The permutation π' is the identity on every element not in the sequence

Σ. Otherwise, $\pi'(\pi(i_0)) = \pi(i_m)$, and, for $j = 1, \ldots, m$, $\pi'(\pi(i_j)) = \pi(i_{j-1})$. The result follows since $\lceil \frac{1}{2}n \rceil \leq r$. □

Lemma 3.2. *For any set* $\{i_1, \ldots, i_k\}$ *of vertices and for* $\{r_1, \ldots, r_k\}$ *satisfying* $\lceil \frac{1}{2}n \rceil \leq \min\{r_1, \ldots, r_k\} \leq n$,

$$\frac{\lceil \frac{1}{2}n \rceil !}{(\lceil \frac{1}{2}n \rceil - k)!} |P(H; i_1 : r_1, \ldots, i_k : r_k)| \leq \left(\prod_{h=1}^{k} |D(H, i_h)| \right) |P(H)|.$$

Proof. When $k = 1$ the lemma reduces to Lemma 3.1. Therefore, we proceed by induction, essentially as in [2]. For $k > 1$,

$$\begin{aligned}
&\frac{\lceil \frac{1}{2}n \rceil !}{(\lceil \frac{1}{2}n \rceil - k)!} |P(H; i_1 : r_1, \ldots, i_k : r_k)| \\
&\quad = \lceil \tfrac{1}{2}n \rceil \frac{(\lceil \frac{1}{2}n/2 \rceil - 1)!}{((\lceil \frac{1}{2}n \rceil - 1) - (k-1))!} |P(H; i_1 : r_1) \cap P(H; i_2 : r_2, \ldots, i_k : r_k)| \\
&\quad \leq \lceil \tfrac{1}{2}n \rceil \frac{(\lceil \frac{1}{2}n \rceil - 1)!}{((\lceil \frac{1}{2}n \rceil - 1) - (k-1))!} |P(H - i_1; i_2 : r_2, \ldots, i_k : r_k)| \\
&\quad \leq \lceil \tfrac{1}{2}n \rceil \left(\prod_{h=2}^{k} |D(H - i_1, i_h)| \right) |P(H - i_1)| \\
&\quad \leq \lceil \tfrac{1}{2}n \rceil |P(H - i_1)| \left(\prod_{h=2}^{k} |D(H, i_h)| \right) \\
&\quad \leq \left(\prod_{h=1}^{k} |D(H, i_h)| \right) |P(H)|. \qquad \square
\end{aligned}$$

The proof of the theorem uses the above result by first showing that any MIN branching program needs time to output some small number, k, of ranks that are no smaller than $\lceil \frac{1}{2}n \rceil$ or equivalently, as is expressed in the following lemma, that there is an upper bound on the number of permutations that any MIN branching program can k-big rank correctly when there is some bound t on the time allowed for the computation. This result does not depend on capacity. Therefore, we state it in terms of a computation tree and use the fact that a computation tree partitions input permutations among its leaves.

Lemma 3.3. *Let* $P(\tau)$ *be the set of permutations of* x *for which a* MIN *tree program* τ *computes a k-big ranking of* x *correctly on allowed input* x. *Then for* $\lceil \frac{1}{2}n \rceil \geq k$,

$$P(\tau)| \leq \frac{(\lceil \frac{1}{2}n \rceil - k)!}{\lceil \frac{1}{2}n \rceil !} n!(t+1)^k,$$

where t *is the length of the longest path in* τ.

·oof. The computation of τ remains unchanged if all edges unreachable over any put are pruned. Assume such a pruning. Let $\phi \in \tau$ be a (reachable) leaf of τ, and $_\phi$ its Hasse diagram. Let a set of k-big elements ranked correctly by τ at leaf ϕ : indexed from the set $\{i_{\phi,1}, \ldots, i_{\phi,k}\}$. Thus,

$$|P(\tau)| \leq \sum_{\phi \in \tau} |P(H_\phi ; i_{\phi,1} : r_{\phi,1}, \ldots, i_{\phi,k} : r_{\phi,k})|,$$

ıd, applying Lemma 3.2, the observation that each execution of MIN can increase e size of a set $D(H_\phi, i_{\phi,h})$ by at most one, and the fact that the leaves of τ partition e $n!$ permutations of x, we have

$$\begin{aligned}
|P(\tau)| &\leq \sum_{\phi \in \tau} \left(\frac{(\lceil \tfrac{1}{2}n \rceil - k)!}{\lceil \tfrac{1}{2}n \rceil !} \left(\prod_{h=1}^{k} |D(H_\phi, i_{\phi,h})| \right) |P(H_\phi)| \right) \\
&\leq \sum_{\phi \in \tau} \left(\frac{(\lceil \tfrac{1}{2}n \rceil - k)!}{\lceil \tfrac{1}{2}n \rceil !} (t+1)^k |P(H_\phi)| \right) \\
&\leq \frac{(\lceil \tfrac{1}{2}n \rceil - k)!}{\lceil \tfrac{1}{2}n \rceil !} (t+1)^k \sum_{\phi \in \tau} |P(H_\phi)| \\
&= \frac{(\lceil \tfrac{1}{2}n \rceil - k)!}{\lceil \tfrac{1}{2}n \rceil !} n!(t+1)^k. \qquad \square
\end{aligned}$$

The lower bound for MIN branching programs

We now prove the theorem using essentially the strategy employed in [1, 2].

·oof of the Theorem. Let M be a MIN branching program that sorts x, given lowed input x, and let M have capacity $S = S(M)$ and time $T = T(M)$.

We first observe that $T \geq n-1$ by the fact that a MIN branching program can entify at any single vertex at most one new x_i for which the relation $x_i < x_j$ holds r some x_j. It follows that $S > \log_2 n$, for n sufficiently large. We may assume also at $S < \frac{1}{4}n$, for, otherwise, the result follows without further argument.

Since M is acyclic, we may decompose it into $L = \lceil 4T/(\lceil \frac{1}{2} \rceil - S - 1) \rceil$ levels, each depth t_i for $\frac{1}{5}(\lceil \frac{1}{2}n \rceil - S - 1) < t_i \leq \frac{1}{4}(\lceil \frac{1}{2}n \rceil - S - 1)$ for $i = 0, \ldots, L-1$. (Observe that $n \rceil - S - 1 > 0$.) Let $t = \max_i \{t_i\}$. A vertex v belongs to level i if the longest path it from the initial vertex is of length l_v for

$$\sum_{j=0}^{i-1} t_j < l_v \leq \sum_{j=0}^{i} t_j.$$

ɜt V_i be the set of vertices in level i, and let $V = \bigcup V_i$. We observe that $|V| \leq 2^S$. ɔr $v \in V_i$, let M_v be the subgraph of M rooted in v and restricted to edges incident ɔm vertices in V_i. For each $i = 0, \ldots, L-1$ and for each $v \in V_i$, let T_v be the ɛmentary tree program that is the union of the paths in M_v.

By Lemma 3.3, at most

$$|P_v| \leq n! \frac{(\lceil \frac{1}{2}n \rceil - S)!}{\lceil \frac{1}{2}n \rceil !} (t+1)^S$$

input permutations π_x^{-1} produce in T_v as many as S of the elements no smaller than $\lceil \frac{1}{2}n \rceil$ in π_x, provided $\lceil \frac{1}{2}n \rceil \geq S$. This same bound must hold for M_v since it does not depend on the capacity of T_v but only on t_i, n, and $\lceil \frac{1}{2}n \rceil$. Let P_i be the number of input permutations for which edges incident from vertices in level i output as many as S of the elements no smaller than $\lceil \frac{1}{2}n \rceil$ in π_x, and let there be $\beta_i 2^S$ vertices in level i. Thus, summing the preceding inequality over all v in V_i, we have

$$\begin{aligned}
|P_i| &\leq \beta_i 2^S n! \frac{(\lceil \frac{1}{2}n \rceil - S)!}{\lceil \frac{1}{2}n \rceil !} (t+1)^S \\
&= \frac{\beta_i n!}{\lceil \frac{1}{2}n \rceil \cdots (\lceil \frac{1}{2}n \rceil - S - 1)} (2(t+1))^S \\
&< \beta_i n! \left(\frac{2(t+1)}{(\lceil \frac{1}{2}n \rceil - S)} \right)^S \\
&\leq \beta_i n! \left(\frac{2(\lceil \frac{1}{2}n \rceil - S - 1)/4}{\lceil \frac{1}{2}n \rceil - S - 1} \right)^S \\
&< \beta_i n! (\tfrac{1}{2})^S.
\end{aligned}$$

It follows that

$$\sum_{i=0}^{L-1} |P_i| < n!,$$

indicating that at least one input permutation π_x^{-1} produces fewer than S new elements of π_x no smaller than $\lceil \frac{1}{2}n \rceil$ in each level. Since M must produce $n - \lceil \frac{1}{2}n \rceil = \lfloor \frac{1}{2}n \rfloor$ elements of π_x no smaller than $\lceil \frac{1}{2}n \rceil$ over all levels, $L > \lfloor \frac{1}{2}n \rfloor / S$, and therefore,

$$\begin{aligned}
&\left\lceil \frac{4T}{(\lceil \frac{1}{2}n \rceil - S - 1} \right\rceil > \frac{\lfloor \frac{1}{2}n \rfloor}{S}, \\
&4TS > (\lceil \tfrac{1}{2}n \rceil - S - 1) \lfloor \tfrac{1}{2}n \rfloor, \\
&T(M)S(M) = \Omega(n^2).
\end{aligned}$$

This result holds for all n, since when n is not large enough for our arguments above, $\Omega(n^2)$ follows trivially. □

References

[1] A. Borodin and S. Cook, A time-space tradeoff for sorting on a general sequential model of computation, *Proc. 12th Ann. ACM Symp. on Theory of Computing*, Los Angeles, 1980, 294–301.

[2] A. Borodin, M.J. Fischer, D.G. Kirkpatrick, N.A. Lynch and M. Tompa, A time-space tradeoff for sorting on non-oblivious machines, *Proc. 20th Ann. IEEE Symp. on the Foundations of Computer Science*, San Juan, Puerto Rico, 1979, 319–327.

[3] A.C.-C. Yao, On the time-space tradeoff for sorting with linear queries, *Theoret. Comput. Sci.* **19**(2) (1982) 203–218.