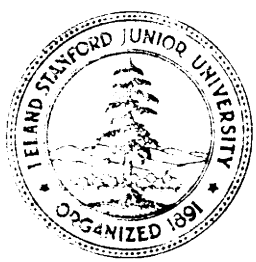
ON THE TIME-SPACE TRADEOFF FOR SORTING WITH LINEAR QUERIES

by

Andrew Chi-Chih Yao

STAN-CS-79-764
August 1979

DEPARTMENT OF COMPUTER SCIENCE
School of Humanities and Sciences
STANFORD UNIVERSITY



On the Time-Space Tradeoff for Sorting with Linear Queries */

Andrew Chi-Chih Yao

Computer Science Department
Stanford University
Stanford, California 94305

Abstract.

Extending a result of Borodin, et. al. [1], we show that any branching program using linear queries " $\sum_i \lambda_i x_i : c$ " to sort n numbers $x_1, x_2, \ldots, x_n$ must satisfy the time-space tradeoff relation $TS = \Omega(n^2)$. The same relation is also shown to be true for branching programs that uses queries " min R = ? " where R is any subset of $\{x_1, x_2, \ldots, x_n\}$.

Keywords: Branching program, linear query, partial order, sorting, time-space tradeoff, tree program.

*/ This research was supported in part by National Science Foundation under grant MCS-77-05313.

1. Introduction.

A fundamental problem in low-order computational complexity is the problem of sorting n numbers $x_1, x_2, \ldots, x_n$. In the standard decision tree model (see Knuth [5]), it is well known that $\approx$ n log n comparisons $x_i : x_j$ are necessary and sufficient in the worst case. This model assumes that all the test information can be retained, but does not address the question of space needed to store the information. Recently, Borodin, et. al. [1] studied "branching programs" for sorting which incorporate the concept of storage requirements. It was shown [1] that any branching program using " $x_i : x_j$ " to sort n elements must satisfy the time-space tradeoff relation $TS = \Omega(n^2)$, and that this bound can nearly be achieved. One open problem raised there is whether the same tradeoff relation also holds for programs with queries other than " $x_i : x_j$ ". The case of linear queries " $\sum_i \lambda_i x_i : c$ " is of special interest [2][7], both because it deals with the question whether arithmetic helps in a purely discrete problem and because linear queries are natural in problems such as network flows, bin packing, and finding shortest paths. The main purpose of the present paper is to prove a tradeoff $TS = \Omega(n^2)$ for branching programs with linear queries (Theorem 2). An intermediate step is to establish this tradeoff for programs employing only queries of the form "which element is the smallest in R?", which may be of interest by itself (Theorem 1).

There is an extensive literature on time-space tradeoffs for general computations. We refer the readers to [1] where further references can be found; however, their understanding is not necessary for this paper.

2. Model and Results.

In this section we review the essence of the branching-program model and state the results to be proved in this paper. The readers are referred to [1][6] for more motivations and discussions of this model.

Let n be a positive integer. We consider programs that compute an output vector for any input vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ in some input domain D . A tree program (or, decision tree) τ is a rooted tree with each internal node v labelled by a query*/ of $\vec{x}$ and each leaf ψ labelled by an output vector $\vec{e}_\psi$. Every edge out of an internal node v is labelled by a possible response to the query at v . For any input $\vec{x}$, the computation starts at the root, branches and traverses down the tree according to the responses of the queries until a leaf ψ is reached. The vector $\vec{e}_\psi$ is then the output. The time required by τ is the maximum number of queries encountered for any $\vec{x} \in D$. We remark that $\vec{e}_\psi$ may have different dimensions for different ψ .

Branching programs extend the concept of tree programs. A branching program τ is a directed multigraph with a distinguished vertex of indegree 0 called the source. Any vertex of outdegree 0 is a leaf, otherwise an internal node. Each internal node v is labelled by a query of $\vec{x}$, and each outgoing edge of v is labelled by a possible response to the query and by an output $[r_1, i_1; r_2, i_2; \ldots; r_\ell, i_\ell]$ (possibly empty). The last expression is to be interpreted as part of an output vector $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}))$, in the sense that

---

*/ A query of $\vec{x}$ is any function q(x) that can only take a finite number of distinct values (or, responses).

$f_{r_1}(f) = i_1$, $f_{r_2}(Z) = i_2$, ... $f_{r_\ell}(\vec{x}) = i_\ell$. As in tree programs, the computation for any input $\vec{x}$ starts at the source, traverses the graph in a natural way until a leaf is reached. The collection of outputs in the process gives the output vector for $\vec{x}$. The number of components in the output vector $\vec{f}$ may depend on $\vec{x}$, and, in general, some components $f_j(\vec{x})$ may be unspecified in the output. We only require that the computation halts in a finite number of steps, and that the outputs are consistent */ for any $\vec{x}$ in the desired input domain. The time required by τ is the maximum number of queries encountered for any $\vec{x} \in D$. The capacity required by τ is defined to be $\lceil \log_2 |V| \rceil$, where V is the set of vertices of τ that can be reached by some $\vec{x} \in D$; we shall regard the capacity as the storage requirement for τ .

We now consider the problem of sorting distinct numbers $x_1, x_2, \ldots, x_n$ with branching programs and tree programs, In this case, the output vector $\vec{f}(\vec{x})$ is required to be the permutation $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ such that $x_{\sigma_1} < x_{\sigma_2} < \ldots < x_{\sigma_n}$ Let K be any set of queries. A K-branching program is a branching program that uses queries in K only; a K-tree program is defined similarly. Let K_0 denote the set of queries $\{x_i : x_j \ (i \neq j)\}$. Borodin, et. al. [1] showed the following interesting result.

Theorem BFKLT [1]. Any K_0-branching program for sorting n distinct numbers in time T and capacity S requires $TS = \Omega(n^2)$.

---

*/ In the sense that, if a $f_j(\vec{x})$ has been specified in the outputs more than once, the values must be the same.

In this paper, we extend the above theorem to other query sets. Let MIN denote the set of queries " Min R = ? ", where $R \subseteq \{1,2,\ldots,n\}$ is any subset and Min R = i such that $i \in R$ and $x_i \leq x_j$ for all $j \in R$, Note that Min R can have $|R|$ responses, and that $x_i : x_j$ is a special case by taking $R = \{i,j\}$. Let LINEAR denote the set of queries " I(7) : 0 " with possible responses $<$, $=$, $>$, where $\ell(\vec{x}) = \sum_i \lambda_i x_i - c$ is any linear function. Our main results are the following theorems.*/

Theorem 1. Any MIN -branching program for sorting n distinct numbers in time T and capacity S requires $TS = \Omega(n^2)$.

Theorem 2. Any LINEAR -branching program for sorting n distinct numbers in time T and capacity S requires $TS = \Omega(n^2)$.

Before turning to the proofs in the next three sections, we list below some useful general properties for branching programs. The proofs can be found in [1]. Let τ be any branching program with required time T and capacity S .

Proposition 1. $S \geq \lceil \log_2 T \rceil$.

Proposition 2 (Pippenger). There exists a branching program τ' which uses the same set of queries, computes the same function as τ in time T and capacity $\leq 2S$, and has the property that its vertices can be partitioned

---

*/ For convenience, we have made the assumption that all x_i are distinct. For Theorem 2, this assumption clearly only makes the result stronger. To remove this assumption in Theorem 1, we have to define Min R when R contains some equal elements. As long as the extension preserves the original meaning when all elements in R are distinct, Theorem 1 of course remains true.

into $T+1$ sets $V_0, V_1, \ldots, V_T$ such that any edge emanating from a vertex in V_i terminates at a vertex in V_{i+1}.

Proposition 3. There is a tree program which, for each input $\vec{x}$, uses the same number of steps and has the same output as τ.

We shall call the τ' in Proposition 2 a normal form for τ. Clearly we need only consider branching programs in their normal forms, for the proofs of Theorems 1 and 2.

3. Guessing Ranks in a Partial Order.

We shall develop some lemmas concerning the accuracy with which one can guess the ranks of elements in a partial order.

We start with some conventions. A partial order P on a set $X = \{x_1, x_2, \ldots, x_n\}$ is a subset of $X \times X$ such that (1) $(x_i, x_i) \notin P$ for all i, and (2) $(x_i, x_j) \in P$ and $(x_j, x_k) \in P$ implies $(x_i, x_k) \in P$, i.e., it is "transitive". We write $x_i <_P x_j$ for $(x_i, x_j) \in P$, or simply $x_i < x_j$ when P is clear from the context. Any set $I \subseteq X \times X$ of consistent inequalities $\{x_{i_1} < x_{j_1}, x_{i_2} < x_{j_2}, \ldots\}$ generates a partial order P by taking the closure of I (i.e., adding to I all the inequalities implied by transitivity); we often write $P = \{x_{i_1} < x_{j_1}, x_{i_2} < x_{j_2}, \ldots\}$ if P can be generated by that set of inequalities. For any partial order P on X, let N(P) denote the number of linear orders on X that are consistent with P. We shall draw partial order P sideways as in Figure 1; an arrow from b to a means $a < b$ in P, and we only draw a subset of arrows whose corresponding inequalities generate P,

Let us consider the set $\mathcal{L}(X)$ of all n! linear orders on X as a probability space with each linear order assigned equal probability. Let $\mathrm{rank}(x_i)$ be the random variable whose value, for each linear order, is equal to the number of x_j less than or equal to x_i. Any set of inequalities I (or a partial order P) induces an event on $\mathcal{L}(X)$, and we shall use the same symbol I (or P) to denote the corresponding event. For example, $\Pr\{x_i < x_j \mid P\}$ will stand for $\Pr\{\text{event } x_i < x_j \mid \text{event P}\}$; clearly, $\Pr\{x_i < x_j \mid P) = N(P \cup \{x_i < x_j\})/N(P)$,

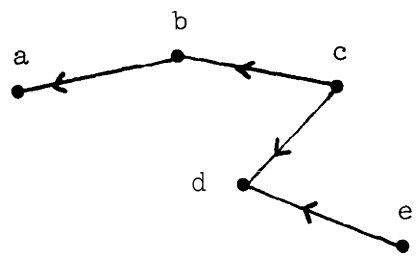


Figure 1. A partial order $P = \{a<b, b<c, d<c, d<e, a<c\}$; note that the arrow from c to a is not shown.

the probability that $x_i < x_j$ assuming all linear orders consistent with P equally likely. Note that for any two sets of inequalities I_1, I_2, the event corresponding to $I_1 \cup I_2$ is the event $I_1 \wedge I_2$.

Let P be a partial order on $A \cup B$ where $A = \{a_1, a_2, \ldots, a_t\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ are disjoint non-empty sets. We say P is slanted on (A,B) if no relation $b_i < a_j$ is contained in P, 2-covered (A, B) if $a_1 < a_2 < \ldots < a_t$ and $b_1 < b_2 < \ldots < b_m$ under P, and 2CS on (A,B) if P is both slanted and 2-covered on (A, B). (See Figure 2.)

Let Z and W be two partial orders on $A \cup B$, where A, B are disjoint. Suppose $Z \cap (A \times A) = W \cap (A \times A)$ and $Z \cap (B \times B) = W \cap (B \times B)$, i.e., Z and W are identical when restricted to either A or B. We say that Z is more A-selective than W if $Z \cap (A \times B) \supseteq W \cap (A \times B)$ and $Z \cap (B \times A) \subseteq W \cap (B \times A)$ (see Figure 3). Intuitively, the elements of A will be "smaller" under Z relative to B than under W. Note that if Z is more A-selective than W, then W is more B-selective than Z.

We need the following result from Graham, Yao, and Yao [4].

Lemma 1 [4, Corollary 2 to Theorem 1]. Let Z and W be 2-covered partial orders on (A,B), and Z is more A-selective than B. Then $\Pr\{I \mid Z\} \geq \Pr\{I \mid W\}$ for any $I \subseteq A \times B$.

The main results in this section are the next two lemmas. Let $t > 0$, $m > 0$, $n = t+m$, $1 \leq k \leq t$ be integers.

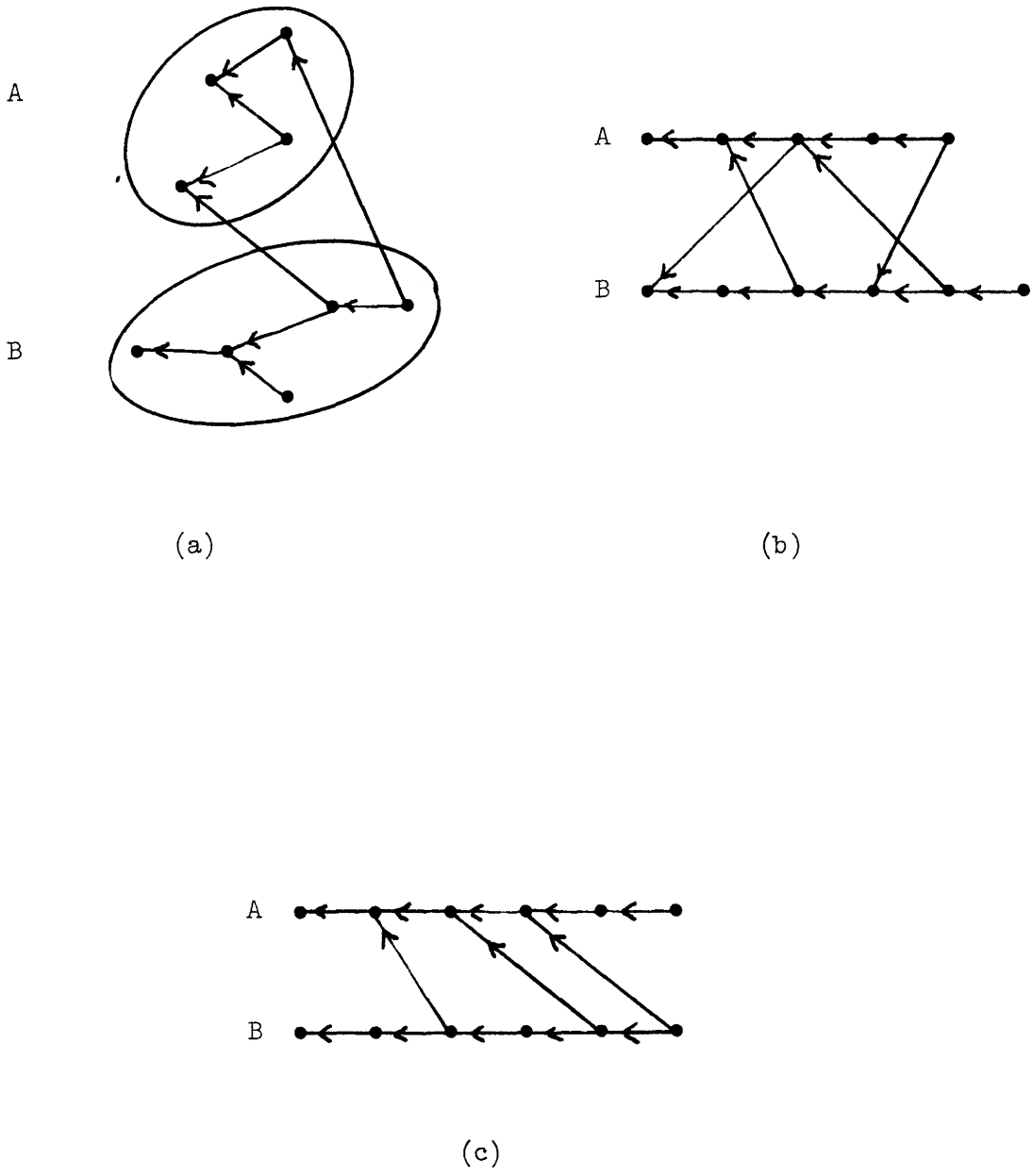


Figure 2. (a) A slanted partial order on (A,B) ; note that no arrow goes from A to B .
(b) A 2-covered partial order on (A,B) .
(c) A 2CS-partial order on (A,B) .

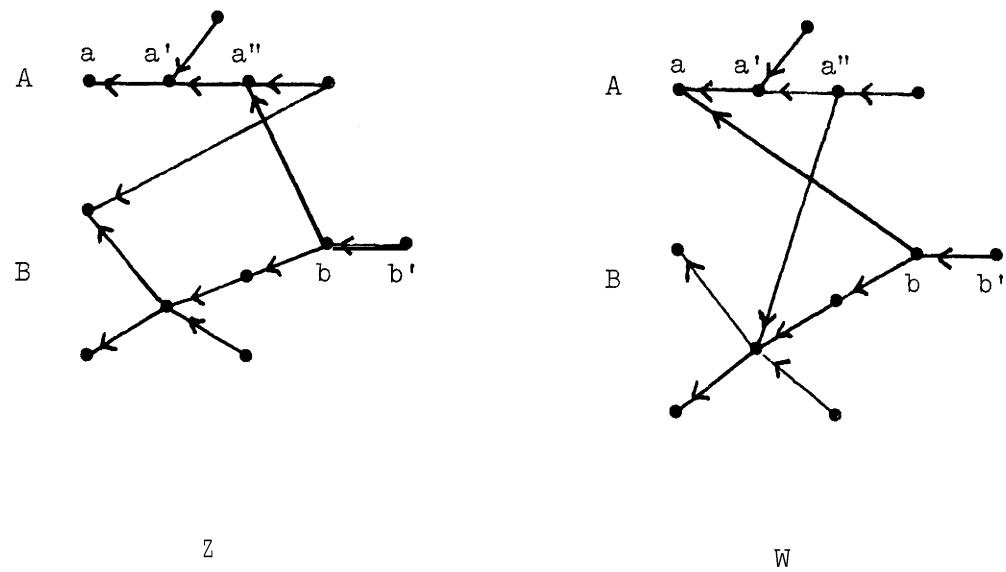


Figure 3. Z is more A-selective than W ; note that

$Z \cap (A \times B) = \{a<b, a'<b, a"<b, a<b', a'<b', a"<b'\}$

while $W \cap (A \times B) = \{a<b, a<b'\}$.

Lemma 2. Let P be a 2CS-partial order on (A,B), where $A = \{a_1, a_2, \ldots, a_t\}$ and $B = \{b_1, b_2, \ldots, b_m\}$. Then, for each k distinct $1 \le i_1, i_2, \ldots, i_k \le t$ and each $1 \le r_1, r_2, \ldots, r_k < n$,

$$\Pr\left\{ \bigwedge_{1 \le \ell \le k} (\mathrm{rank}(a_{i_\ell}) = r_\ell) \mid P_I \right\} < \prod_{1 \le \ell \le k} \frac{1}{r_\ell} .$$

Corollary. $$\Pr\left\{ \bigwedge_{1 \le \ell \le k} (\mathrm{rank}(a_{i_\ell}) = r_\ell) \mid P \right\} \le \left(t / \min_\ell r_\ell \right)^k .$$

Lemma 3. Let P be a slanted partial order on (A,B) where $|A| = t$ and $|B| = m$. Then, for any k distinct elements $a_{i_1}, a_{i_2}, \ldots, a_{i_k} \in A$ and any k integers $1 \le r_1, r_2, \ldots, r_k \le n$,

$$\Pr\left\{ \bigwedge_{1 \le \ell \le k} (\mathrm{rank}(a_{i_I}) = r_\ell) \mid P \right\} \le \left(t / \min_\ell r_\ell \right)^k .$$

Proof of Lemma 2. Before proceeding with the proof we introduce some notations involving $\pm\infty$. We regard the expression $x_i < +\infty$ (or $-\infty < x_i$, or $-\infty < x_i < +\infty$) as an event which is certain on $\mathcal{L}(X)$, i.e., an event that always occurs. We will also regard $x_i < +\infty$ (or $-\infty < x_i$, $-\infty < x_i < +\infty$) as the "null" inequality when it appears in a set of inequalities. For example, the set of inequalities (or partial order) $\{x_7 < x_3, x_5 < +\infty, -\infty < x_6, -\infty < x_7 < +\infty, -\infty < x_4 < x_8\}$ means exactly the set of inequalities (or partial order) $\{x_7 < x_3, x_4 < x_8\}$. Thus, for $A = \{a_1, a_2, \ldots, a_t\}$ and $B = \{b_1, b_2, \ldots, b_m\}$, we can write $I = \{a_1 < b_2, a_3 < b_4, -\infty < a_4, a_5 < +\infty\} \subseteq A \times B$ even though the displayed I is not exactly formally a subset of $A \times B$.

By definition $a_1 < a_2 < \ldots < a_t$ and $b_1 < b_2 < \ldots < b_m$ under P. Without loss of generality, we assume that $1 \le i_1 < i_2 < \ldots < i_k \le t$, $1 \le r_1 < r_2 < \ldots < r_k \le n$, and $r_\ell \ge i_\ell$ for all ℓ. Define $j_\ell = r_{RI} - i_\ell + 1$ for $1 \le \ell \le k$, then $1 \le j_1, j_2, \ldots, j_\ell \le m+1$. The condition $\text{rank}(a_{i_\ell}) = r$ is clearly equivalent to the condition $b_{j_\ell - 1} < a_{i_I} < b_{j_\ell}$, where we have adopted the convention $b_0 = -\infty$ and $b_{m+1} = +\infty$, to be used throughout the proof of Lemma 2 unless specified otherwise. We can thus further assume that $j_1 \le j_2 \le \cdots \le j_k$.

We now show that P can be restricted to a standard form. For convenience, let us use the notation $\alpha(i_1, \ldots, i_k; r_1, \ldots, r_k; P)$ for $\Pr\left\{ \bigwedge_{1 \le \ell \le k} (\text{rank}(a_{i_\ell}) = r_\ell) \mid P \right\}$.

Reduction 1. We can assume that P includes $a_{i_1} < b_{j_1}, \ldots, a_{i_k} < b_{j_k}$.

Proof. Otherwise, let $P' = P \cup \{a_{i_1} \le b_{j_1}, \ldots, a_{i_k} < b_{j_k}\}$. Clearly, $0 < N(P') \le N(P)$. Thus,

$$\alpha(i_1, \ldots, i_k; r_1, \ldots, r_k; P') = N(P' \cup \{b_{j_1-1} < a_{i_1} < b_{j_1}, \ldots, b_{j_k-1} < a_{i_k} < b_{j_k}\})/N(P')$$

$$\ge N(P \cup \{b_{j_1-1} < a_{i_1} < b_{j_1}, \ldots, b_{j_k-1} < a_{i_k} < b_{j_k}\})/N(P)$$

$$= \alpha(i_1, \ldots, i_k; r_1, \ldots, r_k; P).$$

The validity of the lemma for P' will imply that for P. □

Reduction 2. We can assume that $P = \{a_1 < a_2 < \ldots < a_t, b_1 < b_2 < \ldots < b_m, a_{i_1} < b_{j_1}, a_{i_2} < b_{j_2}, \ldots a_{i_k} < b_{j_k}\}$.

Proof. By Reduction 1, we can assume that P includes $a_{i_1} < b_{i_1}, \ldots a_{i_k} < b_{j_k}$. Let $P' = \{a_1 < \ldots < a_t, b_1 < \ldots < b_m, a_{i_1} < b_{j_1}, \ldots a_{i_k} < b_{j_k}\}$, then P' is more B-selective than P. Let $E \subset B \times A$ denote the conditions $(b_{j_1-1} < a_{i_1}, b_{j_2-1} < a_{i_2}, \ldots, b_{j_k-1} < a_{i_k})$. Then, by Lemma 1,

$$\alpha(i_1, \ldots, i_k; r_1, \ldots, r_k; P') = \Pr\{E \mid P'\}$$
$$\geq \Pr\{E \mid P\}$$
$$= \alpha(i_1, \ldots, i_k; r_1, \ldots, r_k; P) .$$

Again, it is sufficient to prove the lemma for P', □

Henceforth we assume that P is as given in Reduction 2. Let us denote the event $b_{j_\ell - 1} < a_{i_\ell}$ by E_ℓ for $1 \leq \ell \leq k$. Then

$$\alpha(i_1, \ldots, i_k; j_1, \ldots, j_k; P) = \Pr\{E_1 \wedge \ldots \wedge E_k \mid P\}$$
$$= \Pr\{E_k \mid P\}\Pr\{E_{k-1} \mid P \wedge E_k\} \ldots \Pr\{E_\ell \mid P \wedge E_k \wedge \ldots \wedge E_{\ell+1}\}$$
$$\ldots \Pr\{E_1 \mid P \wedge E_k \wedge \ldots \wedge E_2\} . \qquad (1)$$

Let us denote $\Pr\{b_{j_k-1} < a_{i_k} \mid a_{i_1} < b_{j_1}, a_{i_2} < b_{j_2}, \ldots, a_{i_k} < b_{j_k}\}$ as $h(i_1,\ldots,i_k; j_1,\ldots,j_k; t,m)$, where the dependency on t and m is explicitly exhibited. Keep in mind that $b_{j_s} = +\infty$ for $j_s = m+1$. By definition,

$$\Pr\{E_k \mid P\} = h(i_1,\ldots,i_k; j_1,\ldots,j_k; t,m) \quad . \tag{2}$$

For $1 \leq \ell < k$, one can show that

$$\Pr\{E_\ell \mid P \wedge E_k \wedge \ldots \wedge E_{\ell+1}\} = \begin{cases} 1 & \text{if } j_{\ell+1} = 1 \\ h(i_1,\ldots,i_\ell; j_1,\ldots,j_\ell; i_{\ell+1}-1, j_{\ell+1}-1) & \text{otherwise } */ \end{cases} \tag{3}$$

by the following argument. When $j_{\ell+1} = 1$, we must have $j_\ell = 1$ and the event E_ℓ is thus $-\infty < a_{i_\ell}$, a certainty. In the other case, under $P \wedge E_k \wedge \ldots \wedge E_{\ell+1}$, the elements in $\{a_s \mid s \geq i_{\ell+1}\} \cup \{b_s \mid s \geq j_{\ell+1}\}$ have ranks $r_{\ell+1}, r_{\ell+1}+1, \ldots, n$, and for any relative order among these elements, the probability distribution of the linear order on $\{a_1, a_2, \ldots, a_{i_{\ell+1}-1}, b_1, b_2, \ldots, b_{j_{\ell+1}-1}\}$ is identical to that under the partial order $\{a_1 < a_2 < \ldots < a_{i_{\ell+1}-1}, b_1 < b_2 < \ldots < b_{j_{\ell+1}-1}, a_{i_1} < b_{j_1} < \ldots < a_{i_\ell} < b_{j_\ell}\}$. (See Figure 4.)

We now digress to derive certain properties of the function h. By Lemma 1, we have, for $j_k \neq 1$,

$$\Pr\{a_{i_k} < b_{j_k-1} \mid a_{i_1} < b_{j_1} < \ldots < a_{i_k} < b_{j_k}\} \geq \Pr\{a_{i_k} < b_{j_k-1} \mid a_{i_k} < b_{j_k}\},$$

---

*/ We emphasize that $h(i_1,\ldots,i_\ell; j_1,\ldots,j_\ell; i_{\ell+1}-1, j_{\ell+1}-1)$ is $\Pr\{b_{j_\ell} < a_{i_\ell} \mid a_{i_1} < b_{j_1}, a_{i_2} < b_{j_2}, \ldots, a_{i_\ell} < b_{j_\ell}\}$ in $A' \cup B'$, where $A' = \{a_1, a_2, \ldots, a_{t'}\}$, $B' = \{b_1, b_2, \ldots, b_{m'}\}$ with $t' = i_{\ell+1}-1$, $m' = j_{\ell+1}-1$, and where the value b_s is $+\infty$ if $s = m'+1$ and $-\infty$ if $s = 0$.

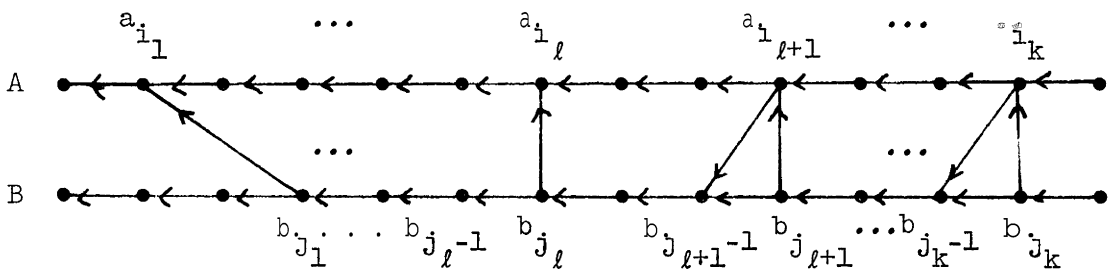


Figure 4. The element $a_{i_{\ell+1}}$ divides $A \cup B$ into the "left" part $\{a_1, \ldots, a_{i_{\ell+1}-1}, b_1, \ldots, b_{j_{\ell+1}-1}\}$ and the "right" part $\{a_{i_{\ell+1}+1}, \ldots, a_t, b_{j_{\ell+1}}, \ldots, b_m\}$; the right part occupies ranks $r_{\ell+1}+1, r_{\ell+1}+2, \ldots, n$, and the actual rankings within it does not affect the probability of the event E_ℓ .

which implies

$$\Pr\{b_{j_k-1} < a_{i_k} < a_{i_l} < b_{j_l} < \cdots < a_{i_1} < b_{j_k}\} \geq \Pr\{b_{j_k-1} < a_{i_k} \mid a_{i_1} < b_{j_k}\},$$

where the probabilities are taken with $|A| = t$ and $|B| = m$. The last inequality is clearly also true for $j_k = 1$. Therefore,

$$h(i_1,\ldots,i_k; j_1,\ldots,j_k; t,m) \leq h(i_k,j_k; t,m) . \tag{4}$$

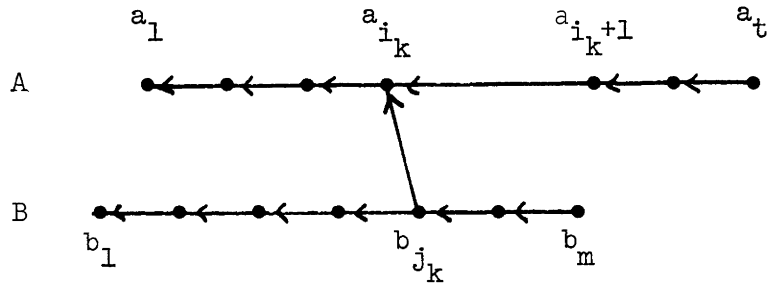
By definition, $1 \leq i_k < t$ and $1 \leq j_k < m+1$. For the moment assume that $i_k < t$ and $j_k < m+1$. Let $Q_1 = \{a_1 < a_2 < \ldots < a_t, b_1 < b_2 < \ldots < b_m, a_{i_k} < b_{j_k}\}$, and $Q_2 = Q_1 \cup \{b_m < a_{i_k+1}\}$. Then Q_2 is more B-selective than Q_1. Using Lemma 1 and the fact that the ranks of all a_ℓ $(\ell > i_k)$ and b_s $(s > j_k)$ are fixed under Q_2 (see Figure 5), we obtain

$$\begin{aligned} h(i_k,j_k; i_k,j_k) &= \Pr\{b_{j_k-1} < a_{i_k} \mid Q_2\} \\ &\geq \Pr\{b_{j_k-1} < a_{i_k} \mid Q_1\} \\ &= h(i_k,j_k; t,m) . \end{aligned}$$

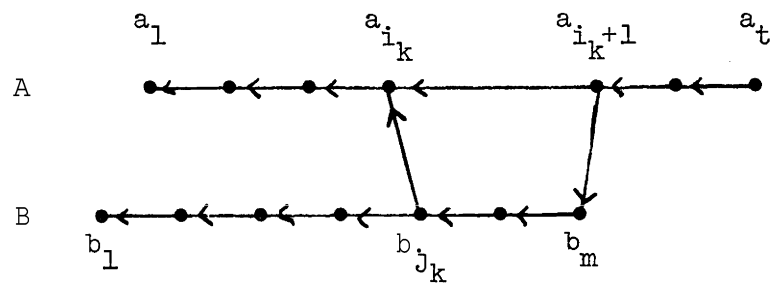
We now claim the inequality,

$$h(i_k,j_k; i_k,j_k) \geq h(i_k,j_k; t,m) , \tag{5}$$

is true for all $1 \leq i_k < t$ and $1 \leq j_k < m+1$. There are three remaining cases :



Q_1



Q_2

Figure 5. Partial orders Q_1 and Q_2 for $i_k < t$ and $j_k < m+1$.

Case 1. $i_k < t$ and $j_k = m+1$. Define Q_1 and Q_2 formally as before (see Figure 6). Utilizing Lemma 1, we obtain

$$h(i_k, j_k ; i_k, m) = \Pr\{b_{j_k-1} < a_{i_k} \mid Q_2\}$$

$$\geq \Pr\{b_{j_k-1} < a_{i_k} \mid Q_1\}$$

$$= h(i_k, j_k ; t, m) .$$

Formula (5) follows by observing that $h(i_k, j_k ; i_k, m)$ $= h(i_k, j_k ; i_k, j_k)$ when $j_k = m+1$ (see Figure 7).

Case 2. $i_k = t$ and $j_k < m+1$. Define Q_1 as before. Then, as the rank of b_s is fixed at $t+s$ for each $s > j_k$ (see Figure 8), we have

$$h(i_k, j_k ; i_k, j_k) = \Pr\{b_{j_k-1} < a_{i_k} \mid Q_1\}$$

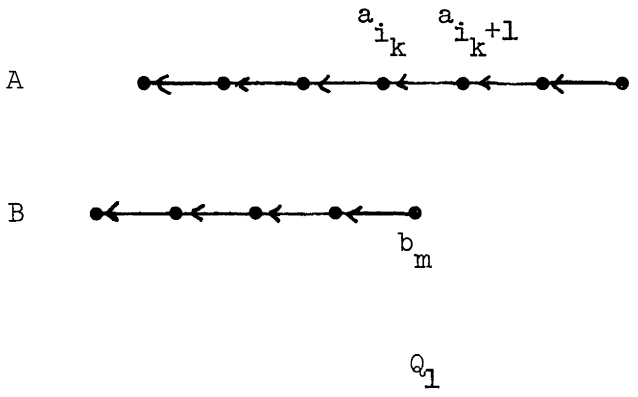
$$= h(i_k, j_k ; t, m) .$$

Case 3. $i_k = t$ and $j_k = m+1$. In this case,

$$h(i_k, j_k ; i_k, m) = h(i_k, j_k ; i_k, j_k) ,$$

as observed in Case 1.

We have thus established formula (5) in all cases.



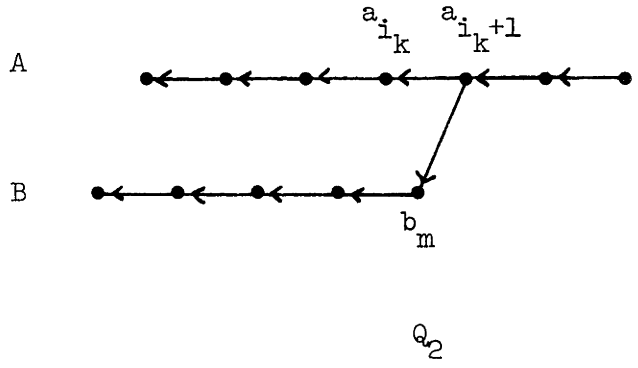


Figure 6. Partial orders Q_1 and Q_2 for $i_k < t$ and $j_k = m+1$.

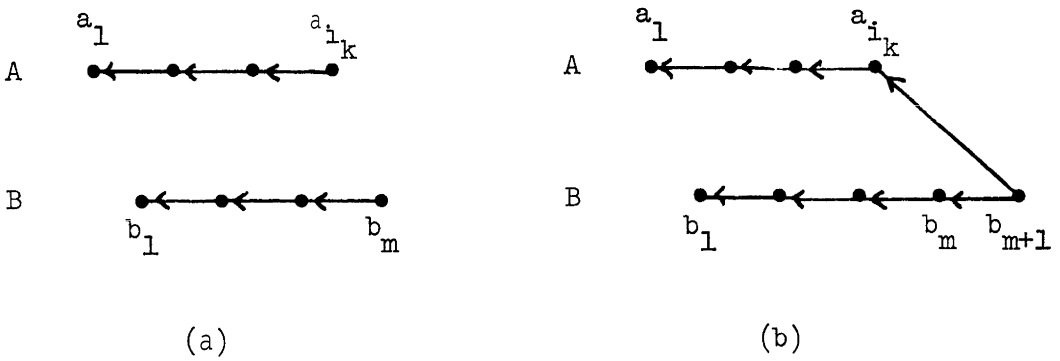


Figure 7. $h(i_k, j_k ; i_k, m) = h(i_k, j_k ; i_k, j_k)$ when $jk = m+1$, as the former is the probability of $b_m < a_{i_k}$ in (a) and the latter is the probability of $b_m < a_{i_k}$ in (b).

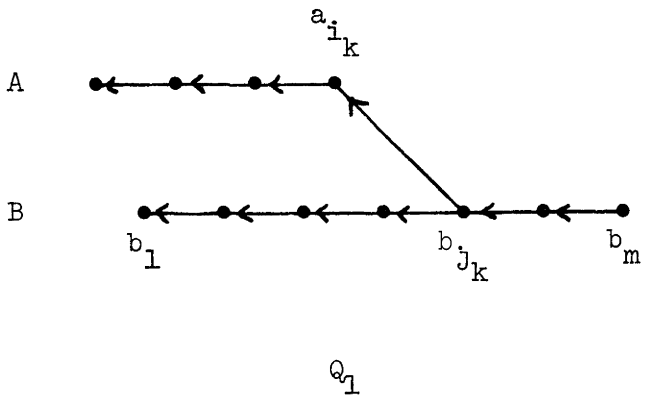


Figure 8. The partial order Q_1 when $t = ik$ and $j_k < m+1$.

From the definition of h , we obtain

$$h(i_k, j_k ; i_k, j_k) = \frac{\binom{i_k+j_k-2}{j_k-1}}{\binom{i_k+j_k-1}{j_k-1}}$$

$$= \frac{i_k}{i_k+j_k-1} \quad . \tag{6}$$

Formulas (4), (5), and (6) lead to

$$h(i_1,\ldots,i_k ; j_1,\ldots,j_k ; t,m) \leq \frac{\bar{}k}{i_k+j_k-1} \tag{7}$$

Formula (7) is the purpose of this digression; note that it is valid for all permissible values of the i 's, j 's, t , and m ,

We now return to formulas (2) and (3), and continue the proof of Lemma 2. From (2), (3), and (7), we obtain (noting that in (3), $j_{\ell+1} = 1$ implies $j_\ell = 1$)

$$\Pr\left\{E_\ell \mid P \wedge \left(\bigwedge_{\ell < s \leq k} E_s\right)^7\right\} \leq \frac{i_\ell}{i_\ell+j_\ell} \leq \frac{i_\ell}{r_\ell} \quad \text{for } 1 \leq \ell \leq k \ . \tag{8}$$

Substituting (8) into (1) gives

$$\alpha(i_1,\ldots,i_k ; r_1,\ldots,r_k ; P) \leq \prod_{1 \leq \ell \leq k} \frac{i_\ell}{r_\ell} \quad .$$

This completes the proof of Lemma 2. □

The Corollary follows immediately from Lemma 2 as $i_\ell < t$ for all ℓ .

Proof of Lemma 3. Let Λ_A and Λ_B denote the sets of all linear orders on A and B , respective*. Then, using the Corollary to Lemma 2, we obtain

$$\Pr\left\{ \bigwedge_{1 \le \ell \le k} (\mathrm{rank}(a_{i_\ell}) = r_\ell) \mid P \right\}$$

$$= \sum_{\substack{\lambda_A \in \Lambda_A \\ \lambda_B \in \Lambda_B}} \Pr\{\lambda_A \wedge \lambda_B \mid P\} \Pr\left\{ \bigwedge_{1 \le \ell \le k} (\mathrm{rank}(a_{i_\ell}) = r_\ell) \mid P \wedge \lambda_A \wedge \lambda_B \right\}$$

$$\le \left(\frac{t}{\min_\ell r_\ell} \right)^k \sum_{\substack{\lambda_A \in \Lambda_A \\ \lambda_B \in \Lambda_B}} \Pr\{\lambda_A \wedge \lambda_B \mid P\} \quad .$$

Lemma 3 follows, as $\sum_{\lambda_A, \lambda_B} \Pr\{\lambda_A \wedge \lambda_B \mid P\} = 1$. □

4. Proof of Theorem 1.

The -proof follows the same general outline as the corresponding -proof in [1], aside from stylistic changes. The main modification is in the use of more sophisticated results on partial orders developed in Section 3.

We begin by discussing a property of general MID-programs. Let $n > 0$ and $1 \leq k$, $n_0 < n$ be integers, τ be a MIN-branching program of time $t > 0$. Clearly, the output for any input vector $(x_1, x_2, \ldots, x_n)$ depends only on the permutation and not the actual values. From now on, in this section, we only consider inputs $(x_1, x_2, \ldots, x_n)$ that are permutations of $(1, 2, \ldots, n)$. Let us say an input permutation to be (k, n_0) -respected by τ, if all the output -pairs (r_{pip}) are correct (i.e., $\text{rank}(x_{i_\ell}) = r_\ell$) and if there are at least k distinct r_ℓ with $r_\ell > n_0$. Let $\mathcal{A}(\tau)$ be the set of input permutations (k, n_0) -respected by τ.

Lemma 4. $\quad |\mathcal{A}(\tau)| < n! \left((t+k)/n_0\right)^k$.

Proof. The lemma is trivially true when $t+k \geq n$. We shall, therefore, assume $t+k < n$.

Because of Proposition 3 in Section 2, we can assume that τ is a MIN-tree program of time t. For each leaf ψ that can be reached by some input, let P_ψ be the partial order at ψ that represents all the information gathered along the path from the root to ψ. Then P_ψ is generated by a collection of inequalities $\{x_{\ell_i} < x_j$ for $j \in R_i - \{\ell_i\}$, $1 \leq i \leq t_\psi\}$, where $\min R_i = \ell_i$ is the response to the i-th query on the path and t_ψ is the distance of ψ from the root. Clearly $t_\psi \leq t$.

Let Φ be the set of reachable leaves ψ for which there are at least k output pairs (r_ℓ, i_ℓ) with all r_ℓ distinct and greater than n_0. For each $\psi \in \Phi$, define $A_\psi = \{x_{\ell_i} \mid 1 \le i \le t_\psi\} \cup \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ and $B_\psi = \{x_1, x_2, \ldots, x_n\} - A_\psi$. Clearly, both A_ψ and B_ψ are non-empty, and P_ψ is slanted on (A_ψ, B_ψ). Let α_ψ denote the set of input permutations leading to ψ, and $\alpha_\psi' \subseteq \alpha_\psi$ the subset of those (k, n_0)-respected by τ. By Lemma 3,

$$\frac{|\alpha_\psi'|}{|\alpha_\psi|} \le \left(\frac{t_\psi + k}{n_0}\right)^k \le \left(\frac{t+k}{n_0}\right)^k .$$

Therefore,

$$\begin{aligned} |\alpha(\tau)| &= \sum_{\psi \in \Phi} |\alpha_\psi'| \\ &\le \left(\frac{t+k}{n_0}\right)^k \sum_{\psi \in \Phi} |\alpha_\psi| \\ &\le \left(\frac{t+k}{n_0}\right)^k n! \quad . \quad \square \end{aligned}$$

We now proceed to prove Theorem 1. Assume $n > 20$. Let τ be any MIX-branching program (in normal form) for sorting n numbers with time T and capacity S. Since τ has to identify the element x_i with rank n, we must have $T > n-1$, because all other elements x_j have to be shown less than some elements and each Min R = ? query can only supply such a certificate for one x_j. By Proposition 1 in Section 2 and the fact $T \ge n-1$, we have

$$S > 1 . \tag{9}$$

Without loss of generality, we also assume that $S < n/20$, as $TS = \Omega(n^2)$ otherwise.

Let $n_0 = \lceil n/4 \rceil$, $t = \lfloor n/20 \rfloor$, and $g = \lfloor T/t \rfloor$. As $T \geq n-1$, we have $g \geq 2$. We wish to prove

$$S \geq \lceil (n-n_0)/(g+1) \rceil , \tag{10}$$

which will imply the theorem by the following argument. From (10) and the definition of g , we have $S(T/t) \geq (n-n_0)g/(g+1)$, implying $ST = \Omega(n^2)$ and hence the theorem.

It remains to prove (10). We assume that $S < \lceil (n-n_0)/(g+1) \rceil$ and will show that it leads to a contradiction.

Let V_ℓ be the set of nodes on level ℓ , $0 \leq \ell \leq T$ (the root being on level 0). Define $V' = \bigcup_{0 \leq j \leq g} V_{jt}$. For each $v \in V'$, let τ_v be the sub-branching program rooted at v and of height $\leq t$, such that all nodes of τ at a distance $> t$ are chopped off and all descendants of v at exactly a distance t are converted to leaves of τ_v . Thus, τ is divided by level into $g+1$ consecutive groups, with the j -th group being the (usually non-disjoint) union of τ_v , $v \in V_{(j-1)t}$. Any path in τ (from the root down) is divided into no more than $g+1$ intervals, each starting at a $v \in V'$ and tracing a path in τ_v .

Let σ be any input permutation. There must be $n-n_0$ distinct output pairs (r,i) with $r > n_0$ along the path it follows in τ . Thus, the interval of the path between levels jt and $(j+1)t$ for some j must have output $\lceil (n-n_0)/(g+1) \rceil > S$ such pairs. By the previous discussions, that means the existence of a τ_v with $v \in V'$ that (S,n_0) -respects σ . Therefore,

$$\sum_{v \in V'} |\beta(v)| \geq n! , \tag{11}$$

where $\mathcal{P}(v)$ denotes the set of permutations that are (S, n_0) -respected by τ_v .

By Lemma 4, $|\mathcal{P}(v)| \leq ((t+S)/n_0)^S n!$. As $|V'| \leq 2^S$, $t \leq n/20$, $1 < S \leq n/20$, and $n_0 \geq n/4$, we have

$$\sum_{v \in V'} |\mathcal{P}(v)| \leq \left(\frac{2(t+S)}{n_0}\right)^S n!$$

$$< \frac{4}{5} n! \quad .$$

This contradicts formula (11). We have thus shown that (10) must be true.

This completes the proof of Theorem 1. □

5. Proof of Theorem 2,

Let L be a LINEAR-branching program (in normal form) for sorting any n distinct input numbers $x_1, x_2, \ldots, x_n$. We shall construct a MIN-branching program τ for sorting any n distinct numbers with the same required time and capacity. Theorem 2 then follows from Theorem 1 immediately.

We first chop off the " = " branches of L at all nodes. Then we replace each internal node v of L by a new node $\xi(v)$ in the following way (see Figure 9). Let $\ell(\vec{x}) = \sum_{i \in O_1} \lambda_i x_i + \sum_{i \in O_2} \lambda_i x_i - c : 0$, with $O_1 \cap O_2 = \emptyset$, $\lambda_i > 0$ for $i \in O_1$ and $\lambda_i < 0$ for $i \in O_2$, be the linear query at v . We replace it with a Min query " $\mathrm{Min}(O_1 \cup O_2) = ?$ ". The $|O_1 \cup O_2|$ outgoing edges of the new node $\xi(v)$ are divided into two groups B_1 and B_2 . Each edge in B_1 corresponds to a response $\mathrm{Min}(O_1 \cup O_2) = i$ with $i \in O_1$; it goes into the leftson of v , and outputs the same output pairs as the original left-branch edge of v , Similarly, each edge in B_2 corresponds to a response $i \in O_2$, goes into the rightson of v , and has the same output pairs, if any, of the original right-branch edges of v in L . (By convention the left-branch edge of v in L corresponds to the response $\ell(\vec{x}) < 0$.) This defines τ , Clearly, τ has the same required time and capacity as L . It remains to prove that τ actually sorts, for any n distinct inputs $x_1, x_2, \ldots, x_n$.

For each internal node v of L , let $\ell_v(\vec{x}) : 0$ be the linear query where $\ell_v(\vec{x}) \equiv \sum_{1 \le i \le n} \lambda_{vi} x_i - c_v$. Define $\eta = \min_{v,i} \{|\lambda_{vi}| \mid \lambda_{vi} \neq 0\}$, $\beta = \max_{v,i} \{|\lambda_{vi}| \mid \lambda_{vi} \neq 0\}$, and $\gamma = \max_{V} |c_v|$; clearly η , β exist and are strictly positive. Let $\delta_1, \delta_2, \ldots, \delta_n$ be defined by

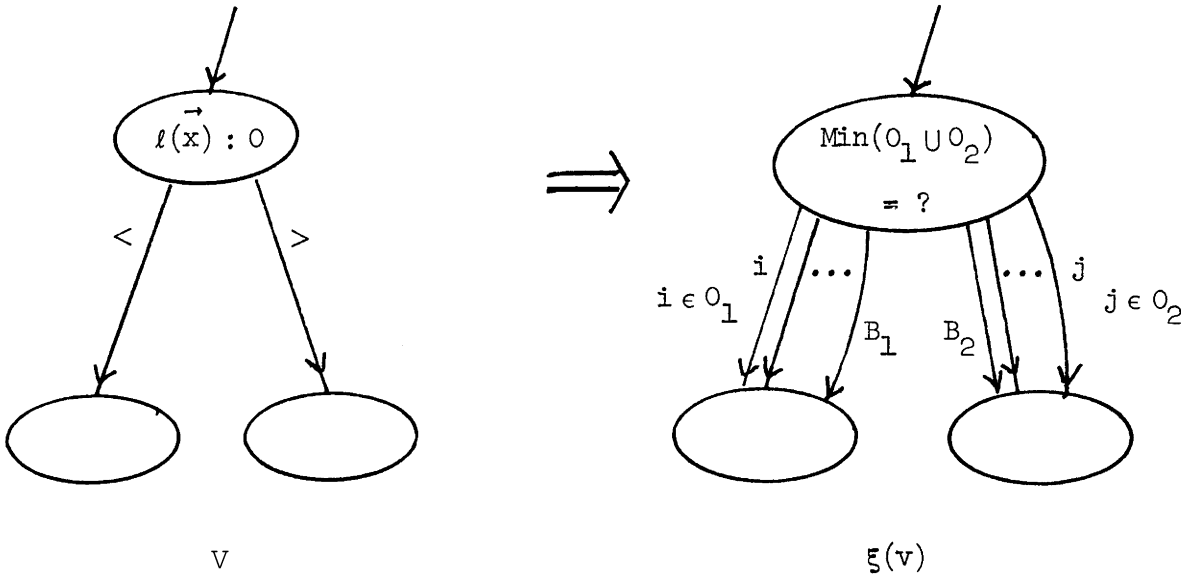


Figure 9. Transforming L into τ ; the replacement of v by $\xi(v)$.

$$\begin{cases} \delta_n = -1 , \\ \delta_j = -\frac{1}{\eta}(n\beta|\delta_{j+1}| + \gamma + 1) \quad \text{for } j = n-1, n-2, \ldots, 1 . \end{cases} \tag{12}$$

It is easy to check that $\delta_1 < \delta_2 < \ldots < \delta_n < 0$ and hence are all distinct.

For each internal node v in L , let $0_{v1} = \{i \mid \lambda_{vi} > 0 , 1 \leq i \leq n\}$ and $0_{v2} = \{i \mid \lambda_{vi} < 0 , 1 \leq i \leq n\}$. Let Λ be the set of input vectors $\vec{x}$ defined by:

$$\Lambda = \{(x_1, x_2, \ldots, x_n) \mid \exists \text{ a permutation } \sigma \text{ such that } x_1 \, ? \, \delta_{\sigma(1)}, \ldots \, ? \, \delta_{\sigma(n)}\} .$$

Clearly, all components x_i are distinct for any $(x_1, x_2, \ldots, x_n) \in \Lambda$,

Lemma 5. For any $\vec{x} \in \Lambda$ and any internal node v in L, $\ell_v(\vec{x}) < 0$ if $\text{Min}(0_{v1} \cup 0_{v2}) \in 0_{v1}$, and $\ell_v(\vec{x}) > 0$ if $\text{Min}(0_{v1} \cup 0_{v2}) \in 0_{v2}$.

Proof. Suppose $x_i = \delta_{\sigma(i)}$ for $1 \leq i \leq n$. Define $Q'_{v\alpha} = \{\sigma(i) \mid i \in Q_{v\alpha}\}$ for $\alpha = 1, 2$. If $\text{Min}(0_{v1} \cup 0_{v2}) \in 0_{v1}$, then there exists $j \in 0'_{v1}$ such that $j < i$ for all other $i \in 0'_{v1} \cup 0'_{v2}$, Thus,

$$\begin{aligned} \ell(\vec{x}) &= \sum_{i \in 0_1} \lambda_{vi}\delta_{\sigma(i)} + \sum_{i \in 0_2} \lambda_{vi}\delta_{\sigma(i)} - c_v \\ &\leq \lambda_{vi'}\delta_j + \sum_{i \in 0_2} |\lambda_{vi}| \cdot |\delta_{j+1}| - c_v , \end{aligned} \tag{13}$$

where $i' = \sigma^{-1}(j)$.

We now use formula (12) in (13) to obtain

$$\begin{aligned} \ell(x) &\leq -n\beta|\delta_{j+1}| - \gamma - 1 + |0_2|\beta|\delta_{j+1}| - c_v \\ &< 0 . \end{aligned}$$

The case when $\text{Min}(O_{v1} \cup O_{v2}) \in O_{v2}$ can be similarly treated. □

The above lemma implies that, for each input vector $\vec{x} \in \Lambda$, the path followed in τ is exactly the image of the path followed in L. Thus, τ gives the same set of output pairs as L. As L computes the sorted output vector $\vec{f}(\vec{x}) = (\sigma^{-1}(1), \sigma^{-1}(2), \ldots, \sigma^{-1}(n))$ by definition, so does τ. Since Λ contains all n! permutations, τ is a MIN-branching program for sorting any n distinct numbers.

We have completed the proof of Theorem 2. □

6. Concluding Remarks.

In this paper we have extended the time-space tradeoff result of Borodin, et. al. [1] to programs using linear queries. It is perhaps worth noting that a major step in the proof is to show lower bounds for programs with MIN-queries. This is a somewhat unexpected technique, as the MIN-queries look too powerful to be used for lower bound proofs (e.g. due to the O(n) -way branching of a MIX-query, one can sort n elements in n-1 MIN-queries in the decision tree model). Aside from the direct comparisons $x_i : x_j$, linear queries are the most-studied primitives for sorting-related problems (e.g. [2][3][7]). The approach used here offers yet another technique for dealing with such questions.

Acknowledgement. I wish to thank Nancy Lynch for critical comments on an earlier draft.

References

[1] A. Borodin, M. J. Fischer, D. G. Kirkpatrick, N. A. Lynch, and M. Tompa, "A Time-Space Tradeoff for Sorting and Related Non-Oblivous Computations," Technical Report No. 79-01-01, January 1979, Department of Computer Science, University of Washington, Seattle, (See also Proc. 20-th Annual IEEE Symp. on Foundations of Computer Science, to appear, 1979.)

[2] D. Dobkin and R. J. Lipton, "On the Complexity of Computations under Varying Sets of Primitive Operations," in Automata Theory and Formal Languages, Springer-Verlag Lecture Notes in Computer Science, No. 33, Springer-Verlag, Berlin/New York, 1975.

[3] F. Fussenegger and H. N. Gabow, "A Counting Approach to Lower Bounds for Selection Problems," Journal ACM 26 (1979), 227-238.

[4] R. L. Graham, A. C. Yao, and F. F. Yao, "Some Monotonicity Properties of Partial Orders," Stanford Computer Science Department Report, 1979, to appear.

[5] D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 2nd printing, 1975.

[6] M. Tompa, 'Time-Space Tradeoffs for Straight-Line and Branching Programs," Technical Report 122/78, University of Toronto, July 1978.

[7] A. C. Yao, "On the Complexity of Comparison Problems Using Linear Functions," Proc. 16-th Annual IEEE Symp. on Foundations of Computer Science, Berkeley, 1975, 85-89.