Theoretical Computer Science 1 (1976) 355–361. © North-Holland Publishing Company

HOW GOOD IS THE INFORMATION THEORY BOUND IN SORTING?

Michael L. FREDMAN

Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Mass., USA

Communicated by A. Meyer
Received November 1974

Abstract. We define a sorting problem on an n element set S to be a family $\langle A_1, ..., A_r \rangle$ of disjoint subsets of the set of $n!$ linear orderings on S. Given an ordering $\omega \in \bigcup_j A_j$, we want to determine to which subset A_j the ordering ω belongs by performing a sequence of comparisons between the elements of S. The classical sorting problem corresponds to the case where the subsets A_j comprise the $n!$ singleton sets of orderings.

If a sorting problem is defined by r nonempty subsets A_j, then the information theory bound states that at least $\log_2 r$ comparisons are required to solve that problem in the worst case. The purpose of this paper is to investigate the accuracy of this bound. While we show that it is usually very weak, we are nevertheless able to define a large class of problems for which this bound is good. As an application, we show that if X and Y are n element sets of real numbers, then the n^2 element set $X+Y$ can be sorted with $O(n^2)$ comparisons, improving upon the $n^2 \log_2 n$ bound established by Harper et al. The problem of sorting $X+Y$ was posed by Berlekamp.

1. Introduction

Let $\{X_1, X_2, ..., X_n\}$ be an n element set. We define a *sorting problem* on these elements to be a pair (Γ, P), where Γ is a subset of the $n!$ possible linear orderings on $\{X_1, ..., X_n\}$, and P is a family of disjoint nonempty sets $\{A_1, ..., A_r\}$ that partition Γ, so that $\Gamma = A_1 \cup ... \cup A_r$. Given an ordering $\omega \in \Gamma$, we want to determine to which set A_j the ordering ω belongs by performing a sequence of comparisons between pairs of elements, $(X_i : X_j)$. A comparison algorithm is said to *solve* (Γ, P) if the following condition is satisfied. Upon representing the algorithm as a comparison tree T, we associate with each leaf L of T the subset Γ_L consisting of the orderings in Γ that define the path through T ending at L. For each L we must have $\Gamma_L \subseteq A_j$ for some $A_j \in P$.

One of the fundamental questions that can be posed about a problem (Γ, P) is the determination of the best worst case number of comparisons required for its solution, which we denote by $N(\Gamma, P)$. Any given algorithm that solves (Γ, P) performs a certain maximal number of comparisons defined by its worst case, and among all such algorithms we are asking for the minimum value of these worst case

numbers. If P partitions Γ into r nonempty sets, then any comparison tree that solves (Γ, P) must have at least r leaves, and therefore must contain a path of length $\geq \log_2 r$. We conclude that $N(\Gamma, P) \geq \lceil \log_2 r \rceil$. This bound is known as the information theory bound (ITB).

In this paper we discuss the strength of the information theory bound, showing that it is almost always terribly weak, in a sense to be made precise. However, we also describe a class of problems for which it is fairly good. When P is the partition of Γ into singleton sets, let $N(\Gamma)$ denote $N(\Gamma, P)$. We show that $N(\Gamma) = \log_2 |\Gamma| + O(n)$, where n is the number of elements we are sorting on. For example, if $|\Gamma| = (n!)^{1/2}$, then $N(\Gamma) = \frac{1}{2} n \log_2 n + O(n)$. We apply this result to a problem posed by Berlekamp.

2. The pitiful ITB

In this section we indicate just how poor the ITB can be, and in fact usually is. We first observe that the most difficult sorting problems, having the largest value $N(\Gamma, P)$, include the usual complete sorting problem where Γ is the entire set of $n!$ orderings and P partitions Γ into singleton sets. Letting $S(n)$ denote $N(\Gamma, P)$ in this case, it is known that $S(n) = n \log_2 n + O(n)$. (See [4, Section 5.3.1] for a detailed discussion about $S(n)$.) Now we describe a problem due to Chase (see [4, p. 198]) where Γ is again the set of $n!$ orderings, but where P partitions Γ into only two sets: $P = \{A_1, A_2\}$, $A_1 = \{X_{\pi(1)} < \ldots < X_{\pi(n)} : \pi \text{ has odd parity}\}$, $A_2 = \{X_{\pi(1)} < \ldots < X_{\pi(n)} : \pi \text{ has even parity}\}$. It is easily verified that $N(\Gamma, P) = S(n)$ in this case. Yet the ITB $= 1$. Now consider all problems of the form (Γ, P), where Γ is the set of all $n!$ orderings, and $P = \{A_1, A_2\}$ is a partition of Γ into two sets. (There are $2^{n!} - 2$ such P.) Our first theorem shows that $N(\Gamma, P) = n \log_2 n + O(n)$ for almost all of these problems.

Theorem 2.1. *Let Γ be the set of all $n!$ orderings on $\{X_1, \ldots, X_n\}$. Let P partition Γ into two nonempty sets. As $n \to \infty$,*

$$N(\Gamma, P) > \log_2 n! - \log_2 \log_2 n - 2$$

for almost all of these partitions P.

Proof. Let $S(n, K)$ denote the number of P such that $N(\Gamma, P) \leq K$. We will show that

$$S(n, K) \leq n^{2^{K+1}}. \tag{2}$$

Assuming (2) for the moment, we note that when $K = \log_2 n! - \log_2 \log_2 n - 2$, $S(n, K) \leq 2^{n!/2} = o(2^{n!})$, which establishes the theorem. To demonstrate (2), let T_K denote the complete binary tree of height K. Consider a labelling of the $2^K - 1$ internal nodes of T_K with the $\binom{n}{2}$ possible comparisons, $(X_i : X_j)$, $1 \leq i$

$< j \leqslant n$, where we branch to the left if $X_i < X_j$, and to the right if $X_i > X_j$, and a labelling of the 2^K leaves of T_K with the labels A_1 or A_2. Clearly, every problem $(\Gamma, P = \{A_1, A_2\})$ with $N(\Gamma, P) \leqslant K$ can be solved by a comparison tree arising from such a labelling of T_K. The number of such labellings is

$$\binom{n}{2}^{2^K-1} 2^{2^K} \leqslant n^{2^{K+1}}.$$

This establishes (2) and completes the proof. □

Apart from the parity problem mentioned above, and contrived modifications of it, the author knows of no other problem $(\Gamma, P = \{A_1, A_2\})$ with $N(\Gamma, P) \geqslant \log_2 n! - O(n)$. This is not suprising since "real problems" would have "structure", causing them to be exceptional. There is one reasonable problem, however, that almost qualifies. Let L denote the length of the longest increasing subsequence of a sequence of n terms from an ordered set, and let $K = \lfloor n/\log_2 n \rfloor$. Let A_1 be the set of orderings for which $L \leqslant K$, and A_2 those for which $L > K$. Then

$$N(\Gamma, P = \{A_1, A_2\}) = n \log_2 n - n \log_2 \log_2 n + O(n)$$

as shown in [2].

3. The powerful ITB

We now prove our claim that the ITB is good for the case when P is a partition of Γ into singleton sets: $N(\Gamma) = \log_2 |\Gamma| + O(n)$. First, we demonstrate that this $O(n)$ error term is best possible as $n \to \infty$ when $|\Gamma| \leqslant (n!)^{1-\varepsilon}$ for fixed $\varepsilon > 0$. (We cannot replace $O(n)$ by $o(n)$.) If Γ is the set of n orderings, $\Gamma = \{X_{\pi(1)} < \ldots < X_{\pi(n)} : \pi$ has at most one inversion$\}$, then $N(\Gamma) = n - 1$, whereas $\log_2 |\Gamma| = \log_2 n$. We generalize this construction to establish our "best possible" claim when $|\Gamma| \leqslant (n!)^{1-\varepsilon}$. Let

$$\Gamma_1 = \{X_{\pi(1)} < \ldots < X_{\pi(n)} : \pi(j) = j \text{ for } 1 \leqslant j \leqslant \lfloor \varepsilon n \rfloor\},$$

$$\Gamma_2 = \{X_{\pi(1)} < \ldots < X_{\pi(n)} : \pi \text{ when restricted to } \{1, \ldots, \lfloor \varepsilon n \rfloor\} \text{ is a permutation with at most one inversion}\}.$$

Observe that $N(\Gamma_2) = N(\Gamma_1) + \lfloor \varepsilon n - 1 \rfloor$, that $\log_2 |\Gamma_2| - \log_2 |\Gamma_1| = \log_2 n$, and that $\log_2 |\Gamma_1| = (1 - \varepsilon) n \log_2 n + O(n)$. Our "best possible" claim follows easily.

Theorem 3.1. *Let Γ be a subset of the $n!$ orderings on $x_1, \ldots, x_n$. Then $N(\Gamma) \leqslant \log_2 |\Gamma| + 2n$.*

Proof. We construct an algorithm that performs $\log_2 |\Gamma| + O(n)$ comparisons in its worst case. Our algorithm is an insertion sort, with a biased insertion: having determined the relative ordering of $X_1, \ldots, X_{K-1} : X_{i_1} < \ldots < X_{i_{K-1}}$, we insert X_K into its appropriate location in this sorted list, in the manner decribed below. First, we decribe an equivalent formulation of this insertion process. Let $b_K = |\{X_j : j \leqslant K,$

$X_j \leqslant X_K\}|$. Clearly $1 \leqslant b_K \leqslant K$. We observe that b_K is the location number of X_K after it is insered into $X_{i_1} < \ldots < X_{i_{K-1}}$, and this gives us a natural correspondence between orderings on $X_1, \ldots, X_n$ and n-tuples $(b_1, \ldots, b_n)$, with $1 \leqslant b_i \leqslant i$. This n-tuple is essentially the inversion table of the permutation π such that $X_{\pi(1)} < \ldots < X_{\pi(n)}$ (see [4, Section 5.1.1]). With this correspondence in mind, we can regard Γ as a set of n-tuples of b_i's. The insertion of X_K is tantamount to the determination of b_K. A comparison between X_K and X_{i_j} is equivalent to asking if $b_K \leqslant J$. We now state and prove a more general result which implies our theorem.

Lemma 3.2. *Given $n \geqslant 1$, let S be a finite set of n-tuples with unrestricted positive integer coordinates. Given an unknown n-tuple in S, we can determine its components, $b_1, b_2, \ldots$, in that order, by making queries of the form, is $b_i \leqslant J$, and with a total of no more than $\log_2|S|+2n$ such queries.*

Proof. Let $S(k)$ denote the number of n-tuples in S with $b_1 = k$. There are finitely many values, $i_1 < i_2 < \ldots < i_l$ such that $S(i_j) > 0$. Letting $N_j = S(i_j)$, we have that $\sum_{j=1}^{l} N_j = |S|$. By invoking, if necessary, the reversible order preserving transformation of S defined by replacing each vector of S with $b_1 = i_j$ by the same vector except with $b_1 = j$, $1 \leqslant j \leqslant l$, we can assume that $i_j = j$ without loss of generality. Now we describe a procedure for constructing queries "is $b_1 \leqslant j$" having the property that if we have to perform r such queries before the value of b_1 can be determined, then $N_{b_1} \leqslant 4|S|/2^r$. Our lemma then follows by induction on n.

Take the interval from 0 to 1, and starting with $j = 1$, mark off successively from the left adjacent intervals of lenght $N_j/|S|$. For each $r \in [0, 1]$ define $J(r)$ to be the number of interval midpoints lying to the left of r, so that $0 \leqslant J(r) \leqslant l$. Having made this transformation of scale, our search for b_1 now proceeds as the usual binary search: We ask if $b_1 \leqslant J(\frac{1}{2})$. If so, then we ask if $b_1 \leqslant J(\frac{1}{4})$. If $b_1 > J(\frac{1}{2})$, we ask if $b_1 \leqslant J(\frac{3}{4})$, etc.

Suppose that $b_1 = i$ and that $N_i/|S| > 4/2^r$. Then the i th interval has radius $> 2/2^r$, and therefore contains $m_1/2^{r-1}$ and $m_2/2^{r-1}$ such that $J(m_1/2^{r-1}) = i-1$ and $J(m_2/2^{r-1}) = i$. If the search for b_1 has not previously ended, then by the time $r-1$ queries have been performed, we will have ascertained that $J(c/2^{r-1}) < b_1 \leqslant J((c+1)/2^{r-1})$ for some c with $0 \leqslant c \leqslant 2^{r-1}$. But then $m_1 \leqslant c \leqslant m_2 - 1$, for if $c \geqslant m_2$, then

$$i = b_1 > J(c/2^{r-1}) \geqslant J(m_2/2^{r-1}) = i,$$

a contradiction; and if $c < m_1$, then

$$i = b_1 \leqslant J((c+1)/2^{r-1}) \leqslant J(m_1/2^{r-1}) = i-1,$$

a contradiction. Hence we are forced to conclude that

$$i-1 = J(m_1/2^{r-1}) \leqslant J(c/2^{r-1}) < b_1 \leqslant J((c+1)/2^{r-1}) < J(m_2/2^{r-1}) = i,$$

and the search for b_1 ends within $r-1$ queries. This completes our proof.□

The reader may recognize the search procedure for b_1 in the above proof as being equivalent to the binary code of Gilbert and Moore (see [4, p.445]). We have used this code to establish a bound on a worst case measure as opposed to an average case measure, which was the original intended use of this code.

The algorithm in Theorem 3.1 requires considerable enumerative information about the set Γ for its construction. To some extent this is necessary since, as Theorem 3.1 shows, a good estimate for $N(\Gamma)$ provides a rough estimate for $|\Gamma|$. The following example, however, shows that in certain cases it is not too difficult to "guess" a decent algorithm.

Let $\Gamma = \{X_{\pi(1)} < \ldots < X_{\pi(n)} : \pi \text{ has } \geqslant \binom{n}{2} - n^{3/2} \text{ inversions}\}$. In terms of the correspondence in Theorem 3.1, these orderings correspond to n-tuples $(b_1, \ldots, b_n)$ such that $\sum_{j=1}^{n} b_j \leqslant n^{3/2}+n$, and $1 \leqslant b_j \leqslant j$. If $1 \leqslant b_j \leqslant j^{1/2}$, then automatically $\sum_{j=1}^{n} b_j \leqslant n^{3/2}+n$, and we conclude that $|\Gamma| \geqslant \prod_{j=1}^{n} \lfloor j^{1/2} \rfloor$, and that $N(\Gamma) \geqslant \frac{1}{2} n \log_2 n + O(n)$.

Now we describe an insertion algorithm for sorting among the orderings of Γ. In terms of the notation in the proof of Theorem 3.2, when determining b_i, we perform the queries, $Q_1, Q_2, \ldots$, where Q_K asks if $b_i \leqslant K\sqrt{n}$, until an affirmative answer is obtained, thereby confining b_i to an interval of length $n^{1/2}$. Then we perform the usual binary search within this interval, which requires potentially $\frac{1}{2}\log_2 n + 1$ further queries. Each insertion therefore requires

$$\leqslant \tfrac{1}{2}\log_2 n + b_i/n^{1/2} + O(1)$$

comparisons, and the total number is

$$\leqslant \tfrac{1}{2} n \log_2 n + n^{-1/2} \sum_{i=1}^{n} b_i + O(n) = \tfrac{1}{2} n \log_2 n + O(n)$$

comparisons. We conclude that $N(\Gamma) = \frac{1}{2} n \log_2 n + O(n)$.

4. An application

Let $\{X_1, \ldots, X_n\}$ and $\{Y_1, \ldots, Y_n\}$ be sets of real numbers, and let $Z = \{X_i + Y_j : 1 \leqslant i,j \leqslant n\}$. Berlekamp posed the problem of determining how fast the n^2 elements of Z can be sorted. For each number $u \in Z$, we assume that the i,j indices, such that $u = X_i + Y_j$, are known. It has been shown [3] that $n^2 \log_2 n$ comparisons suffice to sort Z, thereby saving a factor of 2 over sorting without making use of the structure of Z. However, because the ITB is only $O(n \log n)$ for this case as shown in [3], it follows from Theorem 3.1 that $O(n \log n) + O(n^2) = O(n^2)$ compa-

risons suffice to sort the elements of Z. For the sake of completeness, we describe the proof that the ITB $= O(n \log n)$.

We show that the number of possible orderings of the n^2 elements in Z is $O(n^{8n})$, from which it follows that the ITB $\leqslant 8n \log_2 n + O(1)$. Consider the set of all $2n$-tuples of real numbers $(X_1, \ldots, X_n, Y_1, \ldots, Y_n)$ which are consistent with a fixed ordering of Z. This set forms a convex cone bounded by hyperplanes of the form $X_{i_1} + Y_{i_2} - X_{j_1} - Y_{j_2} = 0$. Because k hyperplanes partition m dimensional space into at most $\binom{k}{m} + \binom{k}{m-1} + \ldots + \binom{k}{0}$ regions, (see [1]), it follows that there are at most $O\left(\binom{n^4}{2n}\right) = O(n^{8n})$ such cones, and therefore a like number of possible orderings of Z.

Note that we have only established the existence of a comparison tree that sorts $X + Y$ with $O(n^2)$ comparisons. This does not imply the existence of a fixed algorithm that sorts $X + Y$ in time $O(n^2)$ for *each* n. Such an algorithm has to generate the relevant portions of this tree which we have shown exists. This generation of the tree is what is usually referred to as the data management aspect of a comparison algorithm, as distinguished from the comparisons themselves. Such an algorithm, if it exists, would be of use in computing the product of sparse polynomials.

If we restrict ourselves as above to performing comparisons between the elements of Z, as opposed to comparing other expressions involving X_i's and Y_j's, then our $O(n^2)$ bound is best possible. Specifically, we show that at least $(n-1)^2$ comparisons are required.

Let ω be the following ordering of the elements of Z:

$$X_{i_1} + Y_{i_2} \leqslant X_{j_1} + Y_{j_2}$$

if and only if $i_1 + i_2 < j_1 + j_2$, or $i_1 + i_2 = j_1 + j_2$ and $i_1 \leqslant j_1$.

If we choose ε so that $0 < \varepsilon < 1/n$ and define $x_j = j + j\varepsilon$ and $Y_j = j$ for $1 \leqslant j \leqslant n$, then Z satisfies ω. For $1 \leqslant r \leqslant n-1$, $2 \leqslant s \leqslant n$, let ω_{rs} be the ordering of Z identical to ω except that

$$X_r + Y_s > X_{r+1} + Y_{s-1}.$$

Z satisfies ω_{rs} if we define

$$X_j = \begin{cases} j + j\varepsilon & \text{if } 1 \leqslant j \leqslant r, \\ j + j\varepsilon - \frac{2}{3}\varepsilon & \text{if } r+1 \leqslant j \leqslant n, \end{cases}$$

$$Y_j = \begin{cases} j & \text{if } 1 \leqslant j \leqslant s-1, \\ j + \frac{2}{3}\varepsilon & \text{if } s \leqslant j \leqslant n. \end{cases}$$

If Z satisfies ω, then just to verify this requires that we perform the $(n-1)^2$ comparisons between the pairs, $X_r + Y_s$ and $X_{r+1} + Y_{s-1}$, for $1 \leqslant r \leqslant n-1$, $2 \leqslant s \leqslant n$. Because if we omit one of these comparisons, with say $r = u$ and $s = v$, then Z could conceivably satisfy ω_{uv}.

References

[1] R. C. Buck, Partition of space, Am. Math. Monthly 50 (1943) 541–544.
[2] M. L. Fredman, On computing the length of longest increasing subsequencies, Discrete Math. 11 (1975) 29–35.
[3] L. H. Harper, T. H. Payne, J. E. Savage and E. Strauss, Sorting $X+Y$, Comm. ACM, to appear.
[4] D. E. Knuth, The Art of Computer Programming, Vol. 3 (Addison-Wesley, Reading, Mass., 1973).