

A note on average-case sorting

Shay Moran* Amir Yehudayoff†

Abstract

This note studies the average-case comparison-complexity of sorting n elements when there is a known distribution on inputs and the goal is to minimize the expected number of comparisons. We generalize Fredman's algorithm which is a variant of insertion sort and provide a basically tight upper bound: If μ is a distribution on permutations on n elements, then one may sort inputs from μ with expected number of comparisons that is at most $H(\mu) + 2n$, where H is the entropy function. A lower bound of $H(\mu)$ always holds, and a linear dependence on n is also required.

1 Introduction

Sorting n elements is one of the most studied and fundamental problems in the theory of computing. The information theoretic lower bound for comparison-based sorting of n elements states that any comparison-based sorting algorithm must make at least $\lg_2(n!)$ comparisons. The idea of the proof is to represent such an algorithm by a binary comparison tree and observe that the leaves in the tree define a prefix-free binary encoding of the $n!$ permutations/orders. This argument actually shows that even if the algorithm is allowed to ask any yes/no questions about the input, it will still make at least $\lg_2(n!)$ questions.

Sorting has also been studied when there is additional partial information on the input e.g. [5, 4]. We consider the *average-case* comparison complexity of sorting. The input is a random permutation π that is drawn from an arbitrary known distribution μ . The goal is to minimize the expected number of comparisons required for sorting. Information theory provides a general lower bound of $H(\mu)$ on the expected number of any yes/no questions required, where H is Shannon's entropy (a formal definition

*Max Planck Institute for Informatics, Saarbrücken, Germany, and Technion-IIT, Israel. smoran@mpi-inf.mpg.de.

†Department of Mathematics, Technion-IIT, Israel. amir.yehudayoff@gmail.com. Horev fellow – supported by the Taub foundation. Research also supported by ISF and BSF.

is given in Section 2.1). On the other hand, Huffman coding implies that there are algorithms that ask at most $H(\mu) + 1$ yes/no questions. It is natural to ask whether there are comparison-based algorithms that are (roughly) as efficient. We show that there are:

Theorem 1. *Let S_n be the set of permutations of $[n]$. For every distribution μ on S_n , there is a comparison algorithm that for every $\pi \in S_n$ asks $Q(\pi)$ questions of the form “is $\pi(i) < \pi(j)$?” for some $i, j \in [n]$ and always finds π so that*

$$\mathbb{E}_{\pi \sim \mu} Q(\pi) \leq H(\mu) + 2n.$$

The algorithm is a generalization of an algorithm by Fredman [2] which is a variant of insertion sort.

The upper bound is tight up to constants: As mentioned, a lower bound of $H(\mu)$ always holds. The linear dependence on n follows since there are full support distributions μ with say $H(\mu) = 2$. Such distributions require at least $n - 1$ comparisons on expectation: Indeed, after $n - 2$ comparisons the comparison-graph is disconnected (The comparison graph has $[n]$ as vertices and two vertices are connected by an edge if they were compared during the execution of the algorithm). So, there are at least two permutations that are consistent with the comparisons made so far. This means that a comparison-based algorithm can never stop before making $n - 1$ comparisons.

Worst-case complexity: If it is required that the algorithm never errs, then the worst-case number of comparisons is at least $\lg_2(n!)$. This is because there are full-support distributions with entropy say 2. However, by allowing an ϵ probability error, one may clearly obtain a worst-case guarantee. For every μ , there is an algorithm that asks for at most $\frac{H(\mu)+2n}{\epsilon}$ comparisons in the worst-case and succeeds with probability at least $1 - \epsilon$ over μ .

Another interesting aspect of this result is the following. The number of yes/no questions is $2^{n!}$, whereas the number of comparisons is only $n(n - 1)$. The class of comparison-based algorithms is therefore a tiny subset of the class of algorithms that are allowed to ask any yes/no question. Yet, comparison-based algorithms are rich enough to (almost) achieve the optimal bound.

A natural approach toward proving Theorem 1 is to iteratively choose a comparison that roughly halves the weight of the distribution. This approach, however, can not work since there are distributions for which such a comparison does not exist. An example is a distribution with a large mass on a single permutation. A related structural result from [7] is that if there is no comparison that is close to halving the weight¹ then the entropy of the distribution is not full. Somewhat surprisingly together with Theorem 1 we get a slight improvement over $\lg_2(n!)$ comparisons for such distributions.

¹In a consistent way, as in the example. See [7] for more details.

The algorithm from Theorem 1 is a variant of insertion sort and iteratively solves the following search problem: Assume we have a distribution ν on a linearly ordered set B , that b is chosen at random from ν , and that we wish to find the unknown b using as few B -comparisons as possible on average. We show that few comparisons always suffice:

Lemma 2. *For every distribution ν on a finite linearly ordered set B , there is an algorithm that for every $b \in B$ asks $C(b)$ questions of the form “is $b \leq b'?$ ” for some $b' \in B$ and always finds b so that*

$$\mathbb{E}_{b \sim \nu} C(b) \leq H(\nu) + 2.$$

Lemma 2 is proved in Section 2.2. The proof uses an alphabetical code of Gilbert and Moore [3]. The way the search algorithm of Lemma 2 is used in the sorting algorithm of Theorem 1 is explained in Section 2.3, where it is also explained what the ordered set B is and how the distribution μ on S_n induces a distribution ν on B (there are in fact several sets B and distributions on them).

This upper bound is tight: Again, the information theoretic lower bound says that $H(\nu)$ comparisons are necessary. To see why the additive factor of 2 is necessary, let ν be a distribution on $\{1 < 2 < 3\}$ such that $\nu(1) = \nu(3) = \epsilon$ where $\epsilon > 0$ is arbitrarily small. If the input is 2 then at least two comparisons are needed to verify it. The average number of comparisons required is thus at least $2(1 - 2\epsilon)$. However, $H(\nu)$ approaches zero as ϵ approaches zero.

Finally, a discussion of the relation to Fredman’s algorithm [2] seems in place. First, Fredman considered only worst-case guarantees on the number of comparisons and we consider average-case ones. In this aspect, the two results are incomparable. On one hand, a worst-case guarantee is always better than an average-case one. On the other hand, if e.g. μ has full support and entropy n then the worst-case guarantee is $\lg_2(n!)$ but the average-case guarantee is only $3n$. Second, Fredman’s result can be interpreted as over distributions that are uniform on their support, whereas we consider arbitrary distributions. His algorithm is based on weighting according to set-sizes, and the variant we present uses weighting according to the underlying distribution. The weighting we consider is quite natural given Fredman’s one. The context of average-case complexity allows for a clean statement and provides additional insight. Third, both Fredman’s algorithm and ours require knowledge of the underlying structure/distribution. For the algorithms to be implemented efficiently, we should allow oracle access to μ of the form “what is the μ -probability that $\pi(j_1) < \pi(j_2) < \dots < \pi(j_i)?$ ” for some $j_1 < j_2 < \dots < j_i$ in $[n]$.

2 Algorithms

2.1 Preliminaries: entropy

We briefly provide definitions that are related to entropy. For more details about entropy, see [1]. For a distribution μ on a finite set U , the *entropy* of μ is

$$H(\mu) = \sum_{u \in U} \mu(u) \lg_2 \frac{1}{\mu(u)},$$

where $0 \lg_2 0 = 0$. If b is a random variable chosen according to μ , it is convenient to denote $H(b) = H(\mu)$. For two random variables b_1, b_2 , the *conditional entropy* is defined as

$$H(b_1|b_2) = H(b_1, b_2) - H(b_2).$$

The *chain rule* for entropy states that if $b = (b_1, \dots, b_n)$ is a random variable, then

$$H(b) = H(b_1) + H(b_2|b_1) + \dots + H(b_n|b_1, \dots, b_{n-1}).$$

The last property we use is that $H(f(b)) \leq H(b)$ for any function f .

2.2 Searching

We now prove Lemma 2. Think of B as the set $\{1, 2, \dots, |B|\}$. For every $j \in B$, let

$$\nu_j = \Pr[b = j].$$

We map B into the interval $[0, 1]$ in an order-preserving manner using the weights defined by ν : Let

$$m_j = \frac{\nu_j}{2} + \sum_{i=1}^{j-1} \nu_i,$$

where the empty sum is zero. Thus $0 \leq m_1 \leq m_2 \leq m_3 \leq \dots \leq 1$. For every $r \in [0, 1]$, define

$$J(r) = \max\{j \geq 0 : m_j \leq r\}.$$

The following observation is useful: For every $j \geq 0$ and $r \in [0, 1]$,

$$j \leq J(r) \iff m_j \leq r.$$

This implies that the question “is $m_b \leq r$?” is equivalent to the question “is $b \leq J(r)$?” and for each r we know $J(r) \in B$ since we know ν .

Find b using a binary search as follows. Ask “is $m_b \leq 1/2$?” and if the answer is “yes” then ask “is $m_b \leq 1/4$?” and so forth. In other words, we start with $L_0 = [0, 1]$ and after asking q questions we have an interval $L_q \subset [0, 1]$ of length 2^{-q} so that $m_b \in L_q$. Stop the search at the first time that

$$|\{j \in B : m_j \in L_q\}| \leq 1$$

and then decide that b is the only j so that $m_j \in L_q$.

We claim that the number of questions q in the case that $b = j$ satisfies $q \leq 1 + \lg_2 \frac{2}{\nu_j}$, and that the answer is always correct. First, observe that in this case $\nu_j > 0$. Second, if $q > \lg_2 \frac{2}{\nu_j}$ then $|L_q| < \frac{\nu_j}{2}$. Third, for every $j' \neq j$, we have $|m_{j'} - m_j| \geq \frac{\nu_j}{2}$. Fourth, we always have $m_j \in L_q$. Thus, for $q > \lg_2 \frac{2}{\nu_j}$, we have $\{j' \geq 0 : m_{j'} \in L_q\} = \{j\}$. The search therefore stops and outputs the correct answer as claimed.

The average number of comparisons can therefore be bounded by

$$\mathbb{E}C \leq \sum_{j \in B} \nu_j \left(2 + \lg_2 \frac{1}{\nu_j} \right) = H(\nu) + 2.$$

2.3 Sorting

We use Lemma 2 to prove Theorem 1. Given $\pi \in S_n$, as in [6], define the *inversion table* $t = t(\pi) = (t_1, \dots, t_n)$ of π by

$$t_i = |\{j \in \{1, 2, \dots, i\} : \pi(i) \leq \pi(j)\}|.$$

Claim 3. *The map $\pi \mapsto t$ is one-to-one.*

Proof. The permutation π can be thought of as an order on $[n]$. It follows by induction on i that knowledge of t_1, \dots, t_i implies knowledge of the π -order on $[i]$. \square

The sorting algorithm is a variant of insertion sort. It runs in n iterations. At iteration $i \in [n]$, we use the already known π -order on $[i - 1]$ to find out the π -order of $[i]$. In order to do so it is enough to determine t_i , the number of elements in $[i - 1]$ which are π -smaller than i . In other words, given t_1, \dots, t_{i-1} we wish to find t_i . Think of B as the set $[i]$ of possible values for t_i . For fixed t_1, \dots, t_{i-1} that occur with positive μ -probability, the distribution μ induces a distribution ν on B . Using the search algorithm from Lemma 2, we may find t_i with expected number of comparisons of the form “is $t_i \leq j$?” for some $j \in B$ that is at most $H(\nu) + 2$. These questions can be simulated by comparisons of the form “is $\pi(i) \leq \pi(k)$?” for $k \in [i - 1]$ since we already know the π -order on $[i - 1]$.

So we have used comparisons to find t and therefore π . The expected number of comparisons is

$$\begin{aligned}\mathbb{E}Q &\leq H(t_1) + 2 + H(t_2|t_1) + 2 + H(t_3|t_1, t_2) + 2 + \dots + H(t_n|t_1, \dots, t_{n-1}) + 2 \\ &= H(t) + 2n && \text{(the chain rule)} \\ &\leq H(\mu) + 2n. && (t \text{ is determined by } \pi)\end{aligned}$$

Acknowledgements

We would like to thank Jeff Kahn, Shlomo Moran, Amir Shpilka and Avi Wigderson for helpful conversations.

References

- [1] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley 2006, ISBN 978-0-471-24195-9, pages 1–748.
- [2] M. L. Fredman. *How good is the information theory bound in sorting?* Theoretical Computer Science 1, pages 355–361, 1976.
- [3] E. N. Gilbert and E. F. Moore. *Variable-length binary encodings*. Bell Syst. Tech. J. 38(4), pages 933–968, 1959.
- [4] J. Kahn and J. H. Kim. *Entropy and sorting*. STOC, pages 178–187, 1992.
- [5] J. Kahn and M. Saks. *Balancing poset extensions*. Order 1, pages 113–126, 1984.
- [6] D. E. Knuth. *The art of computer programming*. Vol. 3, Addison-Wesley, Reading, Mass., 1973.
- [7] T. Leighton and A. Moitra. *On entropy and extensions of posets*. Manuscript, 2011.