Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 106 (2013)

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. The algorithm uses less comparisons for more probable inputs: For every permutation π, the algorithm sorts π by using at most $\log_2(\frac{1}{\Pr_\mu(\pi)})+2n$ comparisons. A lower bound on the expected number of comparisons 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 $\log_2(n!)$ 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[1] about the input, it will still make at least $\log_2(n!)$ 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

*Department of Computer Science, Technion-IIT, Israel and Max Planck Institute for Informatics, Saarbrücken, Germany, . 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.

[1] A question of the form "is $\pi \in \Gamma$?″ where π is the input permutation and $\Gamma \subseteq S_n$ is any subset of permutations.

ISSN 1433-8092

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. On the other hand, Huffman coding implies that there are algorithms that ask at most $H(\mu)+1$ yes/no questions (see, e.g. [1]). 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 $\pi \in S_n$ such that $\Pr_\mu(\pi) > 0$ asks $Q(\pi)$ questions of the form "is $\pi(i) < \pi(j)$?" for some $i, j \in [n]$ and always finds π so that*

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

Moreover, for every $\pi \in S_n$ such that $\Pr_\mu(\pi) > 0$, we have $Q(\pi) \leq \log(\frac{1}{\Pr_\mu(\pi)}) + 2n$.

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[2] 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.

It is worth noting, however, that there are interesting distributions for which the linear dependence on n is not necessary. For example, let P be a partial order on $[n]$ and let $E \subseteq S_n$ be the set of all linear orders that extend P. Kahn and Kim [4] provided a comparison based algorithm that uses only $O(\log(|E|))$ comparisons. Specifically, if μ is the uniform distribution over E then $O(H(\mu))$ comparisons suffice.

Worst-case complexity: Can we hope for an algorithm whose worst-case number of comparisons depends on $H(\mu)$? If it is required that the algorithm never errs then if μ has a full support then the worst-case number of comparisons for any comparison based algorithm for inputs from μ is at least $\log_2(n!)$. Thus, there are full-support distributions with entropy say 2 for which the worst-case number of comparisons is at least $\log_2(n!)$. However, by allowing the probability of error to be at most $\epsilon > 0$, 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. Indeed, stop the algorithm from Theorem 1 after $\frac{H(\mu)+2n}{\epsilon}$ many

[2] Distributions which give a positive probability for every permutation in S_n.

comparisons, and report “error” if the algorithm did not yet terminate. By Markov inequality, the resulting algorithm errs with probability at most ϵ over μ and has the desired worst-case guarantee.

Another interesting aspect is that 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 weight then the entropy of the distribution is not full (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 $b \in B$ so that $\Pr_\nu(b) > 0$, finds b by asking $C(b)$ questions of the form “is $b \leq b'$?” for some $b' \in B$, and it holds that*

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

Moreover, for every $b \in B$ so that $\Pr_\nu(b) > 0$, we have $C(b) \leq \log(\frac{1}{\Pr_\nu(b)}) + 2$.

Lemma 2 is proved in Section 2.1. 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 2.2, 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.

Previous work: Our results are related to the results of Fredman in [2] which show that for any $\Gamma \subseteq S_n$ there exists a comparison based algorithm which sorts any $\pi \in \Gamma$

using at most $\log_2(|\Gamma|) + 2n$ comparisons. These results provide worst-case guarantees on the number of comparisons, and in this aspect, they are incomparable with our average case analysis. 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 $\log_2(n!)$ but the average-case guarantee is only $3n$. Also, 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, nevertheless, quite natural given Fredman's one. The context of average-case complexity allows for a clean statement and provides additional insight.

We note that both Fredman's algorithm and ours require knowledge of the underlying structure/distribution. For the algorithms to be implemented efficiently, we should be able to get efficiently answers to queries of the form "what is the μ-probability that $\pi(j_1) < \pi(j_2) < \ldots < \pi(j_i)$?" for some $j_1 < j_2 < \ldots < j_i$ in $[n]$.

2 Algorithms

2.1 Searching

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

$$\nu_j = \Pr_{\nu}[b = j].$$

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. It is also technically convenient to define $m_0 = 0$. Thus $0 = m_0 \leq m_1 \leq \ldots \leq 1$, and if $m_j = m_{j'}$ for $j < j'$ then $\nu_j = \nu_{j+1} = \ldots = \nu_{j'} = 0$. 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 $r \in [0,1]$,

$$j \leq J(r) \quad \Leftrightarrow \quad m_j \leq r.$$

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 can find $J(r)$ since we know ν.

Let $b \in B$ be so that $\Pr_\nu(b) > 0$. 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 $m_b \in L_q$. Let

$$S_q = \left\{ j \in B : \Pr_{\nu}(j) > 0,\ m_j \in L_q \right\}.$$

Stop the search at the first time that $|S_q| \leq 1$, and then decide that b is the only $j \in S_q$.

We claim that the number of questions $C = C(b)$ satisfies $C \leq \lceil \log_2 \frac{2}{\nu_b} \rceil \leq 2 + \log_2 \frac{1}{\nu_b}$, and that the answer is always correct. First, if $q \geq \log_2 \frac{2}{\nu_b}$ then $|L_q| \leq \frac{\nu_b}{2}$. Second, for every $j \neq b$ so that $\Pr_\nu(j) > 0$, we have $|m_j - m_b| > \frac{\nu_b}{2}$. Third, we always have $m_b \in L_q$. Thus, for $q \geq \log_2 \frac{2}{\nu_b}$, we have $S_q = \{b\}$, which indeed implies $C \leq \lceil \log_2 \frac{2}{\nu_b} \rceil$.

The average number of comparisons can therefore be bounded by

$$\mathbb{E}C \leq \sum_{j \in B} \nu_j \left(2 + \log_2 \frac{1}{\nu_j} \right) = H(\nu) + 2.$$

2.2 Sorting

We use Lemma 2 to prove Theorem 1. Let $\pi \in S_n$ such that $\mu(\pi) > 0$, as in [6], define the *inversion table* $t = t(\pi) = (t_1, \ldots, t_n)$ of π by

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

One way to think of t_i is as the position in which i is inserted to in an insertion sort algorithm with inputs $1, 2, \ldots, n$. For example, if $\pi(2) < \pi(1)$ then $t_2 = 1$ and 2 is inserted to the first position, and if $\pi(2) > \pi(1)$ then $t_2 = 2$ and 2 is inserted to the second position.

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

Proof sketch. The permutation π can be thought of as an order on $[n]$. It follows by induction on i that knowledge of $t_1, \ldots, t_i$ implies knowledge of the π-order restricted to $[i]$. In particular $t_1, \ldots, t_n$ uniquely define π. □

Let μ be a distribution on S_n and let $T = (T_1, ..., T_n)$ denote the random inversion table. Fix $\pi \in S_n$ so that $\Pr_\mu(\pi) > 0$ and let $t = t(\pi) = (t_1, \dots, t_n)$. Observe that:

$$\begin{aligned}
\Pr_\mu(\pi) &= \Pr_\mu(T = t(\pi)) && (\pi \mapsto t(\pi) \text{ is one-to-one})\\
&= \Pr_\mu(T_1 = t_1, \dots, T_n = t_n)\\
&= \Pr_\mu(T_1 = t_1) \Pr_\mu(T_2 = t_2 | T_1 = t_1) \dots \Pr_\mu(T_n = t_n | T_1 = t_1, \dots, T_{n-1} = t_{n-1}).
\end{aligned}$$

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 $[i]$ that are not more than i according to π. In other words, given $T_1 = t_1, \dots, T_{i-1} = t_{i-1}$ we wish to find T_i. Think of $B = B_i$ as the set $\{1, \dots, i\}$ of possible values for T_i. The distribution μ induces a distribution ν on B: For every $b \in B$,

$$\Pr_\nu(b) = \Pr_\mu(T_i = b | T_1 = t_1, \dots, T_{i-1} = t_{i-1}).$$

Using the search algorithm from Lemma 2, we may find t_i with number of comparisons of the form "is $t_i \le j$?" for some $j \in B$ that is at most

$$\log\left(\frac{1}{\Pr_\nu(t_i)}\right) + 2 = \log\left(\frac{1}{Pr_\mu(T_i = t_i | T_1 = t_1, \dots, T_{i-1} = t_{i-1})}\right) + 2.$$

These questions can be simulated by comparisons of the form "is $\pi(i) \le \pi(k)$?" 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 total number of comparisons required to find π is at most:

$$\begin{aligned}
Q(\pi) &\le \sum_{i=1}^{n} \left(\log\left(\frac{1}{\Pr_\mu(T_i = t_i | T_1 = t_1, \dots, T_{i-1} = t_{i-1})}\right) + 2 \right)\\
&= \log\left(\frac{1}{\Pi_{i=1}^{n} \Pr_\mu(T_i = t_i | T_1 = t_1, \dots, T_{i-1} = t_{i-1})}\right) + 2n\\
&= \log\left(\frac{1}{\Pr_\mu(\pi)}\right) + 2n.
\end{aligned}$$

Finally, the overall expected number of comparisons is

$$\mathbb{E}Q = \sum_{\pi \in S_n} \Pr_\mu(\pi) Q(\pi) \le \sum_{\pi \in S_n} \Pr_\mu(\pi) \log\left(\frac{1}{\Pr_\mu(\pi)}\right) + 2n = H(\mu) + 2n.$$

Acknowledgements

We thank Jeff Kahn, Kurt Mehlhorn, Shlomo Moran, Amir Shpilka, Manfred Warmuth and Avi Wigderson for helpful conversations. We also thank the anonymous referees for useful comments on an earlier version of this text.

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.* J. Comput. Syst. Sci. 51(3), pages 390–399, 1995.

[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.