ELSEVIER

Information Processing Letters 79 (2001) 223–227

Information Processing Letters

www.elsevier.com/locate/ipl

Analyzing variants of Shellsort

Bronislava Brejová [1]

Department of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Received 30 January 2000; received in revised form 24 November 2000
Communicated by P.M.B. Vitányi

Abstract

We consider two variants of Shellsort — Dobosiewicz sort and Shaker sort. Not much is known about the running time of these algorithms. We prove that the worst-case time of Shaker sort is $O(n^{3/2}\log^3 n)$ for certain sequences of increments and the average-case time for both variants is $\Omega(n^2/c^p)$ where p is the number of increments and $c = 2$ for Dobosiewicz sort and $c = 4$ for Shaker sort. In our proofs we adapt techniques and results from analysis of Shellsort.

Keywords: Analysis of algorithms; Sorting; Kolmogorov complexity

1. Introduction

Shellsort is an algorithm that sorts a sequence in several stages. Each stage uses some given distance h. The sequence is divided into subsequences, each containing the elements that were in the sequence h elements apart. These subsequences are sorted using Insertion sort. Time complexity of Shellsort depends on a choice of distances used in individual phases (these distances are often called increments).

Several other variants of Shellsort were proposed in literature. Dobosiewicz [1] proposed to use only one pass of Bubblesort for each subsequence instead of sorting the subsequences in each stage. [2] Incerpi and Sedgewick [2] suggest using two passes of Bubblesort in each stage, one going from left to right and the other from right to left. This variant is more symmetrical and probably has better properties (we will call it Shaker sort).

Shellsort is guaranteed to sort each sequence as far as the last increment is 1. The variants, on the contrary, may leave a sequence unsorted even if the last increment is 1. Therefore a special "mop-up" phase is required to finish the sort. We will assume that Insertion sort is used as the last phase. Other solutions were also considered in the literature.

Unlike Shellsort, which has been widely studied, there is hardly any work concerning the two variants. Early papers [1,2,9] as well as survey [8] bring mainly empirical results. These results suggest that the variants have good running time (comparable with Shellsort) on random sequences of a moderate length and that the worst-case time might be quadratic for increment sequences that perform well in tests with random sequences. Poonen in [7] proves lower bound $\Omega(n^{1+c/\sqrt{p}})$ for a sequence of p increments. This bound holds for a class of "Shellsort-type" algorithms including Shellsort, Shaker sort and Dobosiewicz

E-mail address: bbrejova@uwaterloo.ca (B. Brejová).

[1] Supported in part by NSERC Research Grant OGP0046506.

[2] This idea was presented already by Knuth [5, exercise 5.2.1.40, p. 105].

0020-0190/01/$ – see front matter
PII: S0020-0190(00)00223-4

sort. Moreover for nearly geometric sequences of increments (increments of the form $\Theta(\alpha^i)$) Poonen proves the lower bound $\Omega(n^2)$ which holds for Shaker sort and Dobosiewicz sort.

In this article we show that Shaker sort runs in $O(n^{3/2}\log^3 n)$ time for certain sequence of increments. This is the first known sub-quadratic upper bound for this algorithm. In the construction we use known results for Shellsort.

We also prove an $\Omega(n^2/4^p)$ lower bound for the average case of Shaker sort and $\Omega(n^2/2^p)$ for Dobosiewicz sort where p is the number of increments. The proof uses incompressibility method similarly as it is done in [4,3].

Throughout the paper we use $\log n$ to denote the logarithm with base 2 (i.e., $\log_2 n$).

2. Upper bound for Shaker sort

Here we will show how to use results for Shellsort to construct an increment sequence for Shaker sort that has a sub-quadratic worst-case running time. Let 1-shake denote one stage of Shaker sort using increment $h = 1$ (i.e., sequence of compare-exchange operations for pairs (x_1, x_2), (x_2, x_3), …, (x_{n-2}, x_{n-1}), (x_{n-1}, x_n), (x_{n-2}, x_{n-1}), …, (x_2, x_3), (x_1, x_2)). Inversion in a sequence $x_1, x_2, \ldots, x_n$ is any pair (i, j) such that $i < j$ and $x_i > x_j$.

Lemma 1. *Let X be a sequence of 0's and 1's containing m inversions. Then $\sqrt{m}$ passes of 1-shake are sufficient to sort it.*

Proof. Let us assume that sequence X requires exactly k passes of 1-shake to become sorted. In each of these passes the leftmost 1 is exchanged with the rightmost 0, the rest of sequence remaining unchanged (see [9]). Consider the situation in the ith pass. There are some 0's at the beginning followed by the leftmost 1. Then there is some arbitrary subsequence of length x followed by the right-most 0 and some sequence of 1's. Neither 0's at the beginning nor 1's at the end are involved in exchanges in the following $k - i$ passes. Still in each pass we need to exchange 2 elements and therefore there are at least $2(k - i)$ elements between the left-most 1 and the right-most 0 (more precisely there are at least $k - i$ zeroes and at least $k - i$ ones).

When we exchange an inversion pair in a sequence of 0's and 1's having x elements between them, we remove exactly $x + 1$ inversions. Thus the ith pass removes at least $2(k - i) + 1$ inversions, and together in all k passes we remove at least

$$(2k - 1) + (2k - 3) + (2k - 5) + \cdots + 1 = k^2$$

inversions. Therefore if the original sequence has m inversions, $\sqrt{m}$ passes suffice to sort it. □

Note, that the previous lemma does not hold for Dobosiewicz sort. Sequence containing $n - 1$ occurrences of 1 followed by one zero has $n - 1$ inversions and requires $n - 1$ passes of Dobosiewicz sort with increment 1 to become sorted.

Theorem 2. *Let H be a sequence of p increments such that Shellsort for this sequence uses at most m exchanges to sort any input sequence. Then there is an increment sequence H' for which Shaker sort works in $O(pn\sqrt{m})$ time.*

Proof. We will construct H' very simply: each h in H is replaced by $\lceil\sqrt{m}\rceil$ occurrences of h. Such sequence H' has length $\lceil\sqrt{m}\rceil p$ and thus all stages of Shaker sort together require $O(np\sqrt{m})$ time. We will prove that after using this sequence we always obtain sorted sequence and the final Insertion sort is not needed (and if it is used it works in $O(n)$ time). Therefore we have time $O(np\sqrt{m})$.

In order to prove that sequence H' sorts every input we will use 0–1 principle. 0–1 principle states that if a comparator network sorts every input sequence of zeroes and ones, then it sorts every input sequence (for the proof see, for example, in [5, p. 224]). We can easily express Shaker sort (without the final Insertion sort) as a comparator network and therefore 0–1 principle applies to Shaker-sort as well (this statement was proved in [9]).

Therefore we need to prove that Shaker sort with increments H' sorts every 0–1 sequence. Assume that such sequence is used as an input to Shellsort with increments H. We know that at most m exchanges are done for this sequence during Shellsort. Take a stage of Shellsort using some increment h. In this stage h subsequences are sorted independently and in each such sort at most m exchanges are done. It means that each such subsequence contains at most m inversions.

Therefore it can be sorted using $\sqrt{m}$ passes of 1-shake (Lemma 1). Using $\sqrt{m}$ stages of Shaker sort with increment h therefore sorts every subsequence achieving the same result as was achieved by Shellsort in one stage with increment h. Shellsort obtains a sorted sequence after all stages, Shaker sort therefore also obtains a sorted sequence. □

The best known upper bound for Shellsort is $O(n\log^2 n)$ for the sequence of the length $p = \Theta(\log^2 n)$ containing all numbers of the form $2^i 3^j$ smaller than n. This sequence was proposed by Vaughan Pratt in 1969, for details see, for example, [7]. Let us apply the theorem for Pratt's increments (where $p = O(\log^2 n)$ and $m = O(n\log^2 n)$). We obtain sequence H' of the length $O(\sqrt{n}\log^3 n)$ which gives us $O(n^{3/2}\log^3 n)$ time for Shaker sort.

Theorem 3. *There is a sequence of increments for Shaker sort such that Shaker sort works in* $O(n^{3/2}\log^3 n)$ *time in the worst case.*

3. Lower bound for average case

We prove a lower bound on the average case time complexity of Dobosiewicz sort and Shaker sort using incompressibility method. The argument is similar to the lower bound proof for the average case of Shellsort in [4]. We use conditional Kolmogorov complexity of a string, denoted $C(x|y)$. It is the length of the shortest binary string w such that if strings y and w are used as an input for a fixed universal Turing machine, string x will be the result of the computation. For more details about the Kolmogorov complexity see [6].

Theorem 4. *Let H be a sequence of p increments. Dobosiewicz sort with increments H has the average-case running time $\Omega(n^2/2^p)$, and Shaker sort with increments H has the average-case running time $\Omega(n^2/4^p)$.*

Proof. First let us prove the result for Dobosiewicz sort. Each comparison-based sorting algorithm needs $\Omega(n\log n)$ time in average case. For $p > \log n - \log\log n$ we get the bound $\Omega(n^2/2^p)$ directly from this trivial lower bound. Therefore further we will assume that $p < \log n - \log\log n$.

Assume that the input sequence is some permutation π of the set $\{1, 2, \ldots, n\}$. Let π' be a permutation which we obtain after all p stages of Dobosiewicz sort (before the final Insertion sort). Let X be the number of inversions in π'. Then the final Insertion sort works in $\Omega(X)$ time. We will prove that $X = \Omega(n^2/2^p)$ in the average case.

We will give a description that uniquely describes any permutation π of n elements. The description is divided into two parts, the first one provides information which enables us to construct π once we know π' and the second one describes permutation π'. The first part consists of p strings of length n. Bit j of the ith string determines whether x_j was exchanged with x_{j-h_i} during the ith stage of the algorithm (where h_i is the ith increment). Given this information and permutation π' we can simulate the stages backwards and we can construct permutation π.

Permutation π' is described using its inversion table $a_1, a_2, \ldots, a_n$ where a_i is the number of elements greater than π'_i which are to the left of π'_i in permutation π'. Sum $\sum_{i=1}^{n} a_i$ is exactly X. There are $D(X) = \binom{X+n-1}{n-1}$ sequences of n non-negative numbers with sum X and therefore π' can be described as number X followed by the index of $a_1, a_2, \ldots, a_n$ in some fixed enumeration of all sequences with sum X. To get prefix-free description we need $\log X + \log D(X) + 2\log\log X$ bits. Together we need $np + \log X + \log D(X) + 2\log\log X$ bits to encode any permutation π (denote this value $L(\pi)$). Using the formula for $\log\binom{a}{b}$ given in [4] we get

$$\begin{aligned}\log D(X) &= \log\binom{X+n-1}{n-1}\\ &\leqslant \log\binom{X+n}{n}\\ &= n\log\frac{X+n}{n} + X\log\frac{X+n}{X}\\ &\quad + \frac{1}{2}\log\frac{X+n}{Xn} + O(1).\end{aligned}$$

The second term is equal to $\log(1+\frac{n}{X})^X$ and it is at most $n\log e$. The third term is $O(\log n)$ because $X \leqslant n^2$. Using this we obtain the following bounds for $\log D(X)$ and $L(\pi)$

$$\log D(X) \leqslant n\log\left(\frac{X}{n}+1\right) + O(n),$$

$$L(\pi) \leqslant np + \log X + \log D(X) + 2\log\log X$$
$$\leqslant np + n\log\left(\frac{X}{n} + 1\right) + \mathrm{O}(n).$$

Now let P be a program that produces permutation π given n, p, H and our encoding. Let $C(\pi|n, p, H, P)$ be the conditional Kolmogorov complexity of π given n, p, H and P. From the definition of Kolmogorov complexity we have that $L(\pi) \geqslant C(\pi|n, p, H, P)$. Now we will use the property, that most permutations do not have a short description (they are incompressible). More precisely, it is possible to prove by a simple counting argument, that there are at least $(1 - 1/n)n!$ permutations such that $C(\pi|n, p, H, P) \geqslant \log(n!) - \log n$ (see [4]). Let π be such permutation. Then our description must have at least $\log(n!) - \log n = n\log n - \Theta(n)$ bits and therefore

$$np + n\log\left(\frac{X}{n} + 1\right) + \mathrm{O}(n) \geqslant n\log n - \Theta(n),$$

$$X \geqslant \frac{n^2}{2^p\Theta(1)} - n = \Omega\left(\frac{n^2}{2^p}\right).$$

The last step holds because $p \leqslant \log n - \log\log n$ and which means $n^2/2^p \geqslant n\log n$.

The running time for permutation π is therefore $\Omega(n^2/2^p)$. This lower bound applies to at least $(1 - 1/n)n!$ permutations and therefore the average complexity is at least

$$\left(1 - \frac{1}{n}\right)\Omega\left(\frac{n^2}{2^p}\right) = \Omega\left(\frac{n^2}{2^p}\right).$$

The lower bound for Shaker sort can be proved analogously, only we need $2n$ bits to encode one stage of the algorithm instead of n bits used for Dobosiewicz sort. Thus we get lower bound of the form $\Omega(n^2/4^p)$. □

4. Conclusion

We have proved lower bound for the average-case of the form $\Omega(n^2/c^p)$ for both Shaker sort and Dobosiewicz sort. This lower bound gives non-trivial results only for

$$p \leqslant \log n - \log\log n$$

in case of Dobosiewicz sort, and

$$p \leqslant \tfrac{1}{2}(\log n - \log\log n)$$

in case of Shaker sort. On the other hand, for p sufficiently small it gives even better lower bounds that the best known lower bounds for the worst case $\Omega(n^{1+c/\sqrt{p}})$ [7].

We have also shown that the worst-case running time of Shaker sort is sub-quadratic for some increment sequences. Unfortunately, the sequences constructed in the proof are not useful in practice, because Shaker sort with such sequences runs for most inputs much longer than Shellsort.

Our results provide at least some non-trivial bounds for the two variants of Shellsort, however there is still a huge gap between the lower and upper bound in the worst case and no upper bound whatsoever is known for the average case. Also we have no upper bound for Dobosiewicz sort leaving the possibility that it has worst-case complexity $\Omega(n^2)$ for any increments.

Comparing Shellsort with its variants, it seems that the variants do not have any advantage. They are about as difficult to implement as Shellsort and practical tests give comparable results. Best known lower bounds for variants are slightly worse than bounds for Shellsort and Shellsort has much better upper bound.

However increment sequences for Shellsort were much more studied that sequences for its variants. Therefore it might be possible that there is a sequence for which Shaker sort outperforms the best increments of Shellsort either in experimental results or in the theoretical analysis of the average case.

Acknowledgement

I thank Ming Li for many useful comments.

References

[1] W. Dobosiewicz, An efficient variation of bubble sort, Inform. Process. Lett. 11 (1) (1980) 5–6.
[2] J. Incerpi, R. Sedgewick, Practical variations of Shellsort, Inform. Process. Lett. 26 (1) (1987) 37–43.
[3] T. Jiang, M. Li, P. Vitányi, Average-case complexity of Shellsort, in: Proc. 26th International Colloquium on Automata, Languages, and Programming, Prague, Czech Republic, 11–15 July, 1999, Lecture Notes in Comput. Sci., Vol. 1644, Springer, Berlin, 1999, pp. 453–462.
[4] T. Jiang, M. Li, P. Vitányi, A lower bound on the average-case complexity of Shellsort, J. ACM 47 (5) (2000) 905–911.

[5] D.E. Knuth, Sorting and Searching, The Art of Computer Programming, Vol. 2, Addison-Wesley, Reading, MA, 1973.
[6] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, 2nd edn., Springer, New York, 1997.
[7] B. Poonen, The worst case in Shellsort and related algorithms, J. Algorithms 15 (1) (1993) 101–124.
[8] R. Sedgewick, Analysis of Shellsort and related algorithms, in: J. Díaz, M. Serna (Eds.), Algorithms—ESA'96, Fourth Annual European Symposium, Lecture Notes in Comput. Sci., Vol. 1136, Barcelona, Spain, 25–27 September 1996, Springer, Berlin, 1996, pp. 1–11.
[9] M.A. Weiss, R. Sedgewick, Bad cases for Shaker-sort, Inform. Process. Lett. 28 (3) (1988) 133–136.