On the Average-Case Complexity of Shellsort

Paul Vitányi*
CWI and University of Amsterdam

Abstract

We prove a lower bound expressed in the increment sequence on the average-case complexity of the number of inversions of Shellsort. This lower bound is sharp in every case where it could be checked. A special case of this lower bound yields the general Jiang-Li-Vitányi lower bound. We obtain new results e.g. determining the average-case complexity precisely in the Yao-Janson-Knuth 3-pass case.

1 Introduction

The question of a tight general lower bound or upper bound on the average-case complexity of Shellsort (due to D.L. Shell [11]) has been open for more than five decades [5]. We use "average" throughout in the sense of "arithmetic mean of a uniform distribution," and the average-case complexity is the average number of inversions. We present an average-case lower bound on the number of inversions for a p-pass Shellsort with increments $h_1, h_2, \ldots, h_p$ for every number of passes and increment sequences.

Shellsort sorts *in situ* a list of n keys in p passes using a sequence of increments $h_1, \ldots, h_p$ with $n > h_1 > \cdots > h_p$. In the kth pass the main list is divided in h_k separate sublists of length about n/h_k: if $n = \lfloor n/h_k \rfloor + r$ then the initial r sublists have length $\lfloor n/h_k \rfloor + 1$ and the remaining $h_k - r$ sublists have length $\lfloor n/h_k \rfloor$. The hth sublist consists of the keys in positions $j \bmod h_k = h$ of the main list $j \in [1, n] := \{1, \ldots, n\}$. Every sublist is sorted using a straightforward insertion sort. The efficiency of the method is governed by the number of passes p and the selected increment sequence $h_1, \ldots, h_p$ satisfying $h_p = 1$ to ensure sortedness of the final list. Shellsort can be implemented using little code and does not use a call stack and therefore some implementations of the qsort function of the C standard library

*Address: CWI, Science Park 123, 1098 XG Amsterdam, The Netherlands. Email: paulv@cwi.nl

targeted at embedded systems use it instead of quicksort, it is used in the uClibc library, Linux kernel, and bzip2 compressor [12]. A complete description of the Shellsort algorithm, together with animation and examples, is provided in the last reference.

1.1 Previous Work

Let log denote the binary logarithm $\log_2$. All results below concern a permutation of n keys (items) to be sorted. For the *worst-case complexity* of the number of inversions the following is known. The original $\log n$-pass increment sequence $\lfloor n/2 \rfloor, \lfloor n/4 \rfloor, \ldots, 1$ of Shell [11] uses in the worst case $\Theta(n^2)$ inversions, but Papernov and Stasevich [8] showed that another related increment sequence uses a worst-case number of inversions equal to $\Theta(n^{3/2})$. Pratt [10] extended the argument to a class of all nearly geometric increment sequences and proved that there are permutations of n keys to be sorted that require $\Theta(n^{3/2})$ inversions for such an increment sequence and all such permutations can be sorted with Shellsort using such an increment sequence in an upper bound of $\Theta(n^{3/2})$ inversions. Therefore the lower bound is equal to the upper bound. Incerpi and Sedgewick [2] constructed a family of $(8/\epsilon^2) \log n$-length increment sequences for which Shellsort sorts all permutations of n keys in an upper bound of $O(n^{1+\epsilon/\sqrt{\log n}})$ inversions for every $\epsilon > 0$. Poonen [9] proved a $\Omega(n^{1+c/\sqrt{p}})$ lower bound for any number p of passes of Shellsort using any increment sequence for some $c > 0$ and showed that this lower bound is tight for the Incerpi/Sedgewick increment sequence (and one due to B. Chazelle) for $p = \Omega(\log n)$. Since the lower bound drops to order $n \log n$ for $\log^2 n/(\log \log n)^2$ passes and every pass takes at least n steps this shows in fact a $\Omega(n \log^2 n/(\log \log n)^2)$ lower bound on the worst-case number of inversions of Shellsort for every increment sequence. The currently best asymptotic method was found by Pratt [10]. It uses all $\log^2 n$ increments of the form $2^i 3^j < \lfloor n/2 \rfloor$ to obtain a number of inversions of $\Theta(n \log^2 n)$ in the worst case. Therefore the only possibility for Shellsort to sort in $O(n \log n)$ inversions for some number of passes and increment sequence is on the average. For the *average-case complexity* little is known. In Pratt's [10] method with $\log^2 n$ increments the average-case complexity is $\Theta(n \log^2 n)$. Knuth [5] shows $\Theta(n^{5/3})$ for the average-case of $p = 2$ passes and Yao [14] derives an expression for the average case for $p = 3$ that does not give a direct bound but was used by Janson and Knuth to derive an upper bound of $O(n^{23/15})$ on the average-case complexity of 3-pass Shellsort for a particular increment sequence. In [4] Jiang, Li and Vitányi derived a general lower bound of $\Omega(pn^{1+1/p})$ on the average-case complexity of p-pass

Shellsort. This lower bound shows that the only possibility of Shellsort to run on the average in $O(n \log n)$ inversions is for the number of passes p to satisfy $p = \Theta(\log n)$. Apart from this, no nontrivial results were known for the average case until the results presented here. A more detailed history can be found in [5].

1.2 Present Work

We show a lower bound on the average number of inversions of Shellsort expressed in the increment sequence used (Theorem 1). The proof uses the fact that most permutations of n keys have high Kolmogorov complexity. Since the number of inversions in the Shellsort process is not easily amenable to analysis, we analyze a simpler process. The lower bound on the number of unit moves of the simpler process gives a lower bound on the number of inversions of the original process. We show that the largest number of unit moves of each key in the kth pass of the simpler process is less than h_{k-1}/h_k where $h_1, \ldots, h_p$ is the increment sequence and $h_0 = n$ (Claim 2). Subsequently it is shown using the high Kolmogorov complexity of the permutation that most keys in each pass have a number of unit moves close to the maximum. This gives a lower bound on the total number of unit moves of the simpler process (Claim 3) and hence a lower bound on the number of inversions of the original process. This holds for the chosen single permutation. Since all permutations but for a vanishing fraction (with growing n) have this high Kolmogorov complexity, the lower bound on the total number of inversions holds for the average-case of the original Shellsort process (Theorem 1). The lower bound is possibly tight since it coincides with all known bounds. For 2-pass Shellsort Knuth in [5] determined the average-case complexity and the new lower bound on the average complexity coincides with it (Corollary 1). For 3-pass Shellsort Knuth and Janson [3], building on the work of Yao [14], gave an upper bound on the average-case complexity for a particular increment sequence and the new lower bound coincides with this upper bound (Corollary 1). This yields the new result that the average-case complexity of Shellsort for this increment sequence is now determined. They [3, Section 10] conjecture an upper bound on the average-case complexity for another increment sequence. The lower bound on the average-case complexity established here for this sequence coincides with this upper bound (Corollary 1). For the logarithmic increment sequences of Shell [11], Papernov and Stasevich [8], Hibbard [1], and Pratt [10] (also reported in [5]), the lower bound on the average-case complexity for the respective increment sequences is $\Omega(n \log n)$ (Corollary 2). No upper

bound on the average-case complexity is known for any of these increment sequences. For the square logarithmic increment sequence of Pratt [10] the average-case complexity is known. Again, the lower bound given here coincides with it (Corollary 3). A special case of the lower bound gives the Jiang-Li-Vitányi general lower bound (Corollary 4).

2 Preliminaries

We use the plain Kolmogorov complexity defined in [6] and denoted by C in the text [7]. It deals with finite binary strings, *strings* for short. Other finite objects can be encoded into single strings in natural ways. The following notions and notation may not be familiar to the reader so we briefly discuss them. The length of a string x is denoted by $l(x)$. The *empty string* of 0 bits is denoted by ϵ. Thus $l(\epsilon) = 0$. Let x be a natural number or finite binary string according to the correspondence

$$(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), \ldots .$$

Then $l(x) = \lfloor \log(x + 1) \rfloor$. The Kolmogorov complexity $C(x)$ of x is the length of a shortest string x^* such that x can be computed from x^* by a fixed universal Turing machine (of a special type called "optimal" to exclude undesirable such machines). In this way $C(x)$ is a definite natural number associated with x and a lower bound on the length of a compressed version of it by any known or as yet unknown compression algorithm. We also use the conditional version $C(x|y)$.

A *pairing function* uniquely encodes two natural numbers into a single natural number by a primitive recursive bijection. One of the best-known is the computationally invertible Cantor pairing function defined by $\gamma(a, b) = \frac{1}{2}(a+b)(a+b+1)+b$. This pairing function is inductively generalized to the Cantor tuple function $\gamma^n(a, \ldots, y, z) := \gamma(\gamma^{n-1}(a, \ldots, y), z)$. We use this to encode finite sequences of finite objects in a single natural number.

Let $\mathcal{A}$ be a finite set of objects. We denote the cardinality of $\mathcal{A}$ by $|\mathcal{A}|$ (confusion with the absolute value notation is avoided by the context). The incompressibility method [7, Chapter 6] is used here as follows. Let n be a positive integer and f an integer function such that $f(n) = \lceil \log |\mathcal{A}| \rceil$. Fix a y (possibly $y \notin \mathcal{A}$). We prove a certain property for a particular $x \in \mathcal{A}$ using (only) its high Kolmogorov complexity $C(x|\mathcal{A}, y) \geq f(n) - g(n)$ for a function $g(n) = o(f(n))$ and $\lim_{n\to\infty} g(n) = \infty$. How many $x \in \mathcal{A}$ are there such that $C(x|\mathcal{A}, y)$ satisfies this lower bound? Since there are at most $\sum_{i=0}^{f(n)-g(n)-1} 2^i = 2^{f(n)-g(n)} - 1$ binary programs of length less than

$f(n) - g(n)$ there are at least $(1 - 2^{-g(n)})2^{f(n)} + 1$ objects $x \in \mathcal{A}$ such that $C(x|\mathcal{A}, y) \geq f(n) - g(n)$. Since $\lim_{n\to\infty}(1 - 2^{-g(n)})2^{f(n)} = 2^{f(n)}$ all but a vanishing fraction of the objects in $\mathcal{A}$ possess the property involved with growing n.

It is customary to use "additive constant c" or equivalently "additive $O(1)$ term" to mean a constant, accounting for the length of a fixed binary program, independent from every variable or parameter in the expression in which it occurs.

3 The Lower Bound

A Shellsort computation consists essentially of a sequence of comparison and inversion (swapping) operations. We count the total number of data movements (here inversions). The lower bound obtained below holds *a fortiori* for the number of comparisons—the algorithm must compare a pair of keys to decide whether or not to swap them. In practice the running time of the algorithm is proportional to the number of inversions [5]. Keys in the input permutation go by inversions to their final destination. The sequences of inversions constitute insertion paths. The proof is based on the following intuition. There are $n!$ different permutations of n keys. Given the sorting process (the insertion paths in the right order) one can recover the original permutation from the sorted list. The length of a computable description of the sorting process must be at least as great as the Kolmogorov complexity of the starting permutation. The overwhelming majority of permutations have high Kolmogorov complexity. Hence the overwhelming majority of sorting processes must have computable descriptions of at least a certain length. Therefore the average sorting process has a computable description of that length which translates in the number of inversions. The average number of inversions below is the expectation of the number of inversions in the Shellsort sorting process when the permutations of n keys are uniformly distributed.

THEOREM 1. *Let for a Shellsort algorithm the sequence $h_1, \ldots, h_p$ be the increment sequence and n be the number of keys in the list to be sorted. The average number of inversions is $\Omega\left(n \sum_{k=1}^{p} h_{k-1}/h_k\right)$ where $h_0 = n$. (The proof shows this lower bound for all permutations of n keys with probability going to 1 for $n \to \infty$).*

Proof. Let the list to be sorted consist of a permutation σ_0 of the keys $1, \ldots, n$. Let A be a p-pass Shellsort algorithm with increments $h_1, \ldots, h_p$

such that h_k is the increment in the kth pass and $h_p = 1$. Denote the permutation resulting from pass k by σ_k. In each permutation the keys are ordered left-to-right. In the final permutation $\sigma_p = 12 \dots n$ the least key 1 is on the left end and the greatest key n is on the right end.

For $k = 1, 2, \dots, p$, the kth pass starts from σ_{k-1} and this list (or permutation) is divided into h_k separate sublists or h_k-*chains* of length about n/h_k: if $n = \lfloor n/h_k \rfloor + r$ then the initial r sublists have length $\lfloor n/h_k \rfloor + 1$ and the remaining $h_k - r$ sublists have length $\lfloor n/h_k \rfloor$. The hth h_k-chain ($1 \leq h \leq h_k$) consists of the keys in positions $j \bmod h_k = h$ where j is a position in the main list $j \in [1, n]$. The insertion sort of an h_k-chain goes as follows. We start at the left end. If the second key is less than the first key then the second key is swapped with the first key. Otherwise nothing happens. This creates a new h_k-chain. If the third key is smaller than the first key or the second key in the new h_k-chain, then the third key is inserted in its correct position in the $<$-order before the first key or in between the first key and the second key. Otherwise nothing happens. We continue this way. The ith key is inserted in its correct position in the $<$-order in the initial segment of the current h_k-chain consisting of the first key through the $(i-1)$th key. All keys greater than the ith key in this initial segment move one position to the right. This is possible since the inserted key left a gap at the ith position of the current h_k-chain. An *inversion* is a swap of key i with key j which changes list $\dots ji \dots$ to list $\dots ij \dots$. We can view the insertion above as the ith key changing place with the key before it (an inversion), then changing place with the key before that (a second inversion), and so on, until it ends up in its correct position. The inversions involved are called its *insertion path*. By the time the final key is inserted in its correct position in the $<$-order the h_k-chain involved is sorted.

All keys $i = 1, 2, \dots, n$ reside in a h_k-chain. Let $m_{i,k}$ be the number of inversions of key i in its h_k-chain in this sorting process. At the end of the sorting process the h_k-many h_k-chains are merged to establish permutation σ_k by putting the jth key of the hth sorted h_k-chain into position $h+(j-1)h_k$ of permutation σ_k ($1 \leq h \leq h_k$). This process takes place for passes $k \in [1, p]$ resulting in the final permutation $\sigma_p = 12 \dots n$. The sum

$$T = \sum_{i=1}^{n} \sum_{k=1}^{p} m_{i,k} \tag{1}$$

is the total number of inversions that algorithm A performs.

DEFINITION 1. Let n, σ_0, and the increment sequence $h_1, \dots, h_p$ with $h_p = 1$ be as described above. At the start let key $i \in [1, n]$ be in position $p(i) \in$

$[1,n]$ of σ_0. For each i define the $n_{i,k}$'s ($k \in [1,p]$) by the displacement $p(i)-i$:

- If $p(i)-i > 0$ then $\sum_{k=1}^{p} n_{i,k}h_k = p(i)-i$ with each $n_{i,k} \geq 0$ and $\sum_{k=1}^{p} n_{i,k}$ minimal. In this way $p(i)-i$ is represented in a mixed radix system as $\sum_{k=1}^{p} n_{i,k}h_k$.
- If $p(i)-i < 0$ then $\sum_{k=1}^{p} n_{i,k}h_k = p(i)-i$ with each $n_{i,k} \leq 0$ and $\sum_{k=1}^{p} |n_{i,k}|$ minimal. In this way $p(i)-i$ is represented in a mixed radix system as $\sum_{k=1}^{p} n_{i,k}h_k$ with non-positive coefficients $n_{i,k}$.
- If $p(i)-i = 0$ then $n_{i,k} = 0$ for all k ($k \in [1,p]$).

The sequence of integers $n_{1,1},\ldots,n_{n,p}$ is the *minor sequence*. We define $N_i = \sum_{k=1}^{p} n_{i,k}$ for all $i \in [1,n]$ and $N = \sum_{i=1}^{n} |N_i|$.

Claim 1. *Given n, A and the minor sequence we can computably reconstruct the original permutation σ_0.*

Proof. From the minor sequence and algorithm A (containing the increment sequence we need) we can compute the displacements $p(1)-1, p(2)-2, \ldots, p(n)-n$ and therefore the permutation σ_0 from $12\ldots n$. □

Claim 2. (i) $\frac{1}{2}N \leq T$.

(ii) With $h_0 = n$ we have $|n_{i,k}| < h_{k-1}/h_k$ for all i and k ($i \in [1,n]$, $k \in [1,p]$).

(iii) For every $i \in [1,n]$ one can compute the $n_{i,k}$'s in the order $n_{i,1},\ldots,n_{i,p}$ from distance $p(i)-i$ with an algorithm of $O(1)$ bits.

Proof. (i) By Definition 1 the quantity $\sum_{i=1}^{n} |p(i)-i|$ is the required sum of the distances the keys have to travel from their positions in σ_0 to their final positions in $\sigma_p = 12\ldots n$. Each $p(i)-i = \sum_{k=1}^{p} n_{i,k}h_k$ is expressed as the sum of terms with coefficients $n_{i,k}$ ($k \in [1,p]$) of a mixed radix representation with radices $h_1,\ldots,h_p$. Because of Definition 1 for every $i \in [1,n]$ we have the sum $|N_i| = \sum_{k=1}^{p} |n_{i,k}|$ minimal for the coefficients of such a radix representation for each distance $|p(i)-i|$. The number of inversions of each key $i \in [1,n]$ in the sorting process of Shellsort consists of $\sum_{k=1}^{p} m_{i,k}$ of the coefficients in $\sum_{k=1}^{p} m_{i,k}h_k$. A *unit move* of key i is the absolute value of a unit of an integer $n_{i,k}$. In the Shellsort sorting process keys move by inversions. Since every inversion moves one key one position forward and an adjacent key one position backward in its h_k-chain ($k \in [1,p]$) it is a pair of dependent unit moves equal to at most two independent unit moves. Hence $N = \sum_{i=1}^{n} |N_i|$ is smaller or equal to $2T = 2\sum_{i=1}^{n}\sum_{k=1}^{p} m_{i,k}$.

(ii) Assume by way of contradiction that there exist i, k ($i \in [1, n]$, $k \in [1, p]$) such that $|n_{i,k}| \geq h_{k-1}/h_k$. Suppose $k = 1$. Since $h_0 = n$ we have $|n_{i,1}|h_1 \geq n$. Therefore $|p(i) - i| \geq n$ which is impossible. Hence $k \in [2, p]$. Let $n_{i,k}$ be positive. Since $n_{i,k}$ is integer this implies $n_{i,k} \geq \lceil h_{k-1}/h_k \rceil$. Define $n'_{i,k-1} := n_{i,k-1} + 1$ and $n'_{i,k} := n_{i,k} - \lceil h_{k-1}/h_k \rceil$ while $n'_{i,h} = n_{i,h}$ otherwise. Since $h_{k-1} > h_k$ we have $\lceil h_{k-1}/h_k \rceil > 1$ and therefore $\sum_{j=1}^{p} n'_{i,j} < \sum_{j=1}^{p} n_{i,j} = N_i$ contradicting the minimality of N_i. Let $n_{i,k}$ be negative. Since $n_{i,k}$ is integer this implies $n_{i,k} \leq -\lceil h_{k-1}/h_k \rceil$. Define $n'_{i,k-1} := n_{i,k-1} - 1$ and $n'_{i,k} := n_{i,k} + \lceil h_{k-1}/h_k \rceil$ while $n'_{i,h} = n_{i,h}$ otherwise. Since $h_{k-1} > h_k$ we have $\lceil h_{k-1}/h_k \rceil > 1$ and therefore $\sum_{j=1}^{p} |n'_{i,j}| < \sum_{j=1}^{p} |n_{i,j}| = |N_i|$ contradicting the minimality of $|N_i|$.

(iii) The representation $\sum_{k=1}^{p} n_{i,k}h_k$ of $p(i) - i$ for every i ($i \in [1, n]$) is represented in a mixed radix system with radices $h_1, h_2, \ldots, h_p$ and the $n_{i,k}$ of the same sign (or 0) for all $k \in [1, p]$. The question asked is whether this representation can be uniquely retrieved from $p(i) - i$. Computing the minimum of $\sum_{k=1}^{p} n_{i,k}$ for sequences $n_{i,1}, \ldots, n_{i,p}$ satisfying $\sum_{k=1}^{p} n_{i,k}h_k = p(i) - i$ given $h_1, h_2, \ldots, h_p$ can be done by a program of constant length by trying all finitely many possibilities. □

There are $n! \approx \sqrt{2\pi n}\binom{n}{e}^n$ permutations of n keys by Stirling's approximation. This implies $\log n! \approx n \log n - 1.5n$. Choose the permutation σ_0 such that its conditional Kolmogorov complexity (Section 2) satisfies

$$C(\sigma_0 | n, A, P) \geq n \log n - 3n, \tag{2}$$

with fixed n, A, P, where from n we determine the set $\mathcal{A}$ of all permutations of n keys such that $\sigma_0 \in \mathcal{A}$, the algorithm A is used in this p-pass Shellsort (including the increment sequence), and P is a constant-size algorithm to process all the information and to output σ_0. We use a pairing function to encode the conditional in a single natural number (Section 2).

Denote the minor sequence $n_{1,1}, \ldots, n_{n,p}$ by S_n. The description of S_n comprises the displacements $p(1) - 1, p(2) - 2, \ldots, p(n) - n$ from which the minor sequence can be extracted. A computable description of S_n, given n, A and P, requires at most

$$l(descr(S_n)) = (\sum_{i=1}^{n} \sum_{k=1}^{p} \log |n_{i,k}|) + D \tag{3}$$

bits. Here D is the number of bits required to be able to parse the main part of $descr(S_n)$ into its constituent parts. By Claim 1 we can compute

permutation σ_0 from $descr(S_n)$, given n, A and P. Hence

$$l(descr(S_n)) \geq C(\sigma_0|n, A, P). \tag{4}$$

From (2) and (4) it follows that

$$l(descr(S_n)) \geq n \log(n/8). \tag{5}$$

CLAIM 3. Writing $h_0 = n$ we have

$$\sum_{i=1}^{n} \sum_{k=1}^{p} |n_{i,k}| = \Omega\left(n \sum_{k=1}^{p} h_{k-1}/h_k\right).$$

Proof. By Claim 2 item (ii) for every $i \in [1, n]$ and every pass $k \in [1, p]$ we have $|n_{i,k}| < h_{k-1}/h_k$. Since $\prod_{k=1}^{p} h_{k-1}/h_k = h_0/h_p = n$ we have by (3) and (5) that

$$\begin{aligned} \frac{l(descr(S_n))}{n} &= \frac{\sum_{k=1}^{p} n(\log(h_{k-1}/h_k) - a_k)}{n} + \frac{D}{n} \\ &= \log n - \sum_{k=1}^{p} a_k + \frac{D}{n} \geq \log \frac{n}{8}, \end{aligned} \tag{6}$$

where $a_k = \log(h_{k-1}/h_k) - 1/n \sum_{i=1}^{n} \log |n_{i,k}|$ for $k = 1, 2, \ldots, p$ and $a_k > 0$ by Claim 2 item (ii).

DEFINITION 2. The *self-delimiting* encoding of string x is $1^{|x|}0|x|x$. If the length of x is equal $\log n$ then its self-delimiting encoding has length $\log n + 2 \log \log n + 1$.

For each $i \in [1, n]$ we have $|p(i) - i| < n$ (the displacement of a key cannot be as great or greater than the length of the list) and the sequence $n_{i,1}, \ldots, n_{i,p}$ can be extracted from the self-delimiting encoding of $p(i) - i$ using the information in D. We now show that $D/n = o(\log n)$.

- The information D accounts for the at most $(2 \log \log n + 1)$-length part of the self-delimiting encoding of $|p(i) - i|$ $(i \in [1, n])$.
- We require one time $O(1)$ bits for a self-delimiting program to extract the sequences $|n_{i,1}|, \ldots, |n_{i,p}|$ from the $|p(i) - i|$ $(i \in [1, n])$. The extraction can be done for all $i \in [1, n]$ by a single $O(1)$-bit program by Claim 2 item (iii).
- Since all $n_{i,1}, \ldots, n_{i,p}$ have the same sign for each $i \in [1, n]$ we require $O(n)$ self-delimiting bits to encode them all.

To parse $descr(S_n)$ it therefore suffices that the quantity $D \leq 2\sum_{i=1}^{n} \log\log n + O(n) = 2n\log\log n + O(n)$. The total of the description D is $o(n\log n)$ bits.

Hence up to lower order terms the last inequality of (6) is rewritten as $\sum_{k=1}^{p} a_k \leq 3$. Since $a_k > 0$ for every $k \in [1,p]$ we have $0 < a_k \leq 3$. Writing a_k out and reordering this gives up to lower order terms

$$\log(h_{k-1}/h_k) \leq 1/n \sum_{i=1}^{n} \log|n_{i,k}| + 3,$$

and by exponentiation of both sides of the inequality one obtains

$$h_{k-1}/h_k = O((\prod_{i=1}^{n} |n_{i,k}|)^{1/n}).$$

By the inequality of the arithmetic and geometric means and rearranging we obtain $\sum_{i=1}^{n} |n_{i,k}| = \Omega(nh_{k-1}/h_k)$ for every $1 \leq k \leq p$. Therefore, $N = \sum_{k=1}^{p}\sum_{i=1}^{n} |n_{i,k}| = \Omega(n\sum_{k=1}^{p} h_{k-1}/h_k)$. □

Since $\frac{1}{2}N \leq T$ by Claim 2 item (i), a lower bound for $\frac{1}{2}N$ is also a lower bound for T. Therefore Claim 3 proves the statement of the theorem for the particular σ_0.

By Stirling's approximation $\log n! \approx n\log(n/e) + \frac{1}{2}\log n + O(1) \approx n\log n - 1.44n + \frac{1}{2}\log n + O(1)$. Therefore $n\log n - 1.5n \leq \log n! \leq n\log n - n$ for large n. Therefore by [7, Theorem 2.2.1] which uses a simple counting argument, (see also Section 2) at least a $(1-2^{-n})$-fraction of all permutations σ on n keys satisfy (2). Hence the desired lower bound holds on the average (expected over the uniform distribution) number of inversions. In fact, it holds for all permutations of $1,\ldots,n$ with probability going to 1 with growing n. □

Corollary 1. Set $h_0 = n$. For $p = 2$ with $h_1 = n^{1/3}$, and $h_2 = 1$ this yields

$$T = \Omega(n(n^{1-1/3} + n^{1/3}) = \Omega(n^{5/3}),$$

which coincides with the best number of inversions for 2-pass Shellsort $T = \Theta(n^{5/3})$ using the same increment sequence $h_1 = n^{1/3}, h_2 = 1$ as given by [5].

For $p = 3$ with $h_1 = n^{7/15}$, $h_2 = n^{1/5}$, and $h_3 = 1$ this yields

$$T = \Omega(n(n^{1-7/15} + n^{7/15-1/5} + n^{1/5}) = \Omega(n^{1+8/15}) = \Omega(n^{23/15}).$$

The upper bound of $O(n^{23/15})$ for 3-pass Shellsort using the same increment sequence $h_1 = \Theta(n^{7/15}), h_2 = \Theta(n^{1/5}), h_3 = 1$ with the additional restriction

that $\gcd(h_1, h_2) = 1$ is given in [3]. This reference uses a complicated probabilistic analysis based on the still more complicated combinatorial characterization in [14]. Together with the lower bound we establish the new fact that the average number of inversions of 3-pass Shellsort with this increment sequence is $\Theta(n^{23/15})$.

In [3, Section 10] it is conjectured that with $h_1 \approx n^{1/2}$ and $h_2 \approx n^{1/4}$ ($h_3 = 1$) one may obtain an average-case number of inversions of $O(n^{3/2})$. Using the theorem above shows that $T = \Omega(n(n^{1-1/2} + n^{1/2-1/4} + n^{1/4}) = \Omega(n^{3/2})$. Therefore, if the conjecture on the upper bound is true then 3-pass Shellsort has an average-case number of inversions of $\Theta(n^{3/2})$ for this increment sequence.

Corollary 2. The increment sequence $h_1, \ldots, h_p$ with $p = \lfloor \log n \rfloor$ of Papernov and Stasevich in [8] is $h_1 = n/2 + 1, h_2 = n/2^2 + 1, \ldots, h_p = n/2^{\lfloor \log n \rfloor} + 1$. The worst-case number of inversions reported by [8] is $\Theta(n^{3/2})$. Since $h_{k-1}/h_k \approx 2$, the theorem above gives a lower bound on the average number of inversions of $T = \Omega(n \sum_{k=1}^{\Theta(\log n)} \Omega(1)) = \Omega(n \log n)$.

The increment sequence of Hibbard [1] with increment sequence $2^k - 1$ until it passes n has a worst-case number of inversions $\Theta(n^{3/2})$. With a similar analysis as before it gives a lower bound on the average-case of $T = \Omega(n \log n)$. It is conjectured in [13] to lead to an average-case number of inversions of $O(n^{5/4})$ as reported in [5]. This conjecture is difficult to settle empirically since for $n = 100,000$ we have $\log n \approx n^{1/4}$.

Pratt's logarithmic increment sequence (one of his "hypergeometric" sequences) in [10] also reported by [5] is $h_1, \ldots, h_p$ with $h_k = (3^k - 1)/2$ not greater than $\lceil n \rceil$. This increment sequence leads to a worst-case number of inversions of $\Theta(n^{3/2})$. In this case $h_{k-1}/h_k \approx 3$ and the number of passes is $p = \log_3 n$. The theorem above gives a lower bound on the average number of inversions of $T = \Omega(n \sum_{k=1}^{\Theta(\log n)} \Omega(1)) = \Omega(n \log n)$.

The original increment sequence used by Shell [11] was $\lfloor n/2 \rfloor, \lfloor n/2^2 \rfloor$, and so on for $\log n$ passes. Knuth [5] remarks that this is undesirable when the binary representation of n contains a long string of zeroes and has the effect that Shellsort runs in worst-case time $\Theta(n^2)$. By the same analysis as given above for the Papernov-Stasevich increment sequence the lower bound on the average number of inversions is $\Omega(n \log n)$.

By [4] the average number of inversions of Shellsort can be $\Theta(n \log n)$ only for an increment sequence $h_1, \ldots, h_p$ with $p = \Theta(\log n)$. We have shown here that the lower bound on the average number of inversions is $\Omega(n \log n)$ for many increment sequences of this length. It is an open problem whether it can be proved that for some such increment sequence the average number

of inversions is $O(n \log n)$.

COROLLARY 3. For Pratt's square logarithmic increment sequence $h_1, \ldots, h_p$ with $p = \Theta((\log n)^2)$, the average-case number of inversions is lower bounded by $T = \Omega(n \sum_{k=1}^{\Theta((\log n)^2)} \Omega(1)) = \Omega(n(\log n)^2)$. The precise average-case number (and worst-case number) of inversions is $\Theta(n(\log n)^2)$ in [10], and therefore the lower bound is tight.

COROLLARY 4. The theorem enables us to establish asymptotic lower bounds for $n \to \infty$ keys and p passes. For example choose for a p-pass Shellsort the increment sequence with identical ratios between increments $h_1 = n^{1-1/p}, h_2 = n^{1-2/p}, \ldots, h_p = n^{1-p/p} = 1$. With $h_0 = n$ the sum becomes $\sum_{k=1}^{p} h_{k-1}/h_k = pn^{1/p}$. The lower bound for p passes becomes $T = \Omega(pn^{1+1/p})$, that is, the lower bound of [4]. This lower bound is a greatest lower bound which holds for all increment sequences of p passes. This can be seen as follows. For increment sequences we express the increments as real powers of n. If the ratios between successive increments are unequal then there is one of those ratios which is greater than other ratios. If h_{k_0-1}/h_{k_0} is such a maximum ratio ($k_0 \in [1, p]$) which means that for some $\epsilon > 0$ we have $h_{k_0-1}/h_{k_0} = n^{1/p+\epsilon} > n^{1/p}$, then the lower bound becomes $T = \Omega(nh_{k_0-1}/h_{k_0})) \neq \Omega(pn^{1+1/p})$.

We give an example of an increment sequence for $p = 4$ which has a lower bound greater than $\Omega(pn^{1+1/p}) = \Omega(n^{5/4})$. We choose the increment sequence $h_1 = n^{11/16}, h_2 = n^{7/16}, h_3 = n^{3/16}, h_4 = 1$. The lower bound becomes $T = \Omega(n \cdot (n^{1-11/16} + n^{11/16-7/16} + n^{7/16-3/16} + n^{3/16}) = \Omega(n^{1+5/16}) = \Omega(n^{21/16}) \neq \Omega(n^{20/16})$.

4 Conclusion

A first nontrivial general lower bound on the average-case number of inversions of Shellsort using p passes was given in [4]. Here we gave a lower bound on the average-case complexity for each increment sequence separately. The lower bound of above reference turns out to be the greatest lower bound which holds for all increment sequences. In fact, the lower bound given here seems to be possibly tight as follows from the corollaries. A tantalizing prospect is to obtain a lower bound for the best increment sequences expressed only in the number of keys to be sorted and the number of passes in the sorting process, and which is tighter than the lower bound in the quoted reference.

Acknowledgment

I thank Ronald de Wolf for comments on a preliminary draft of this paper and the referees for constructive suggestions.

References

[1] T.N. Hibbard, An Empirical Study of Minimal Storage Sorting, *Commun. ACM*, 6:5(1963), 206-213.

[2] J. Incerpi and R. Sedgewick, Improved upper bounds on Shellsort, *Journal of Computer and System Sciences*, 31(1985), 210–224.

[3] S. Janson, D. Knuth, Shellsort with three increments, *Random Structures Algorithms*, 10:1-2(1997), 125–142.

[4] T. Jiang, M. Li, P.M.B. Vitányi, A lower bound on the average-case complexity of Shellsort, *J. Assoc. Comp. Mach.*, 47:5(2000), 905–911.

[5] D.E. Knuth, *The Art of Computer Programming, Vol.3: Sorting and Searching*, Addison-Wesley, 1973 (1st Edition), 1998 (2nd Edition).

[6] A.N. Kolmogorov, Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1:1(1965), 1–7.

[7] M. Li and P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 3rd Edition, 2008.

[8] A. Papernov and G. Stasevich, A method for information sorting in computer memories, *Problems Inform. Transmission*, 1:3(1965), 63–75.

[9] B. Poonen, The worst case in Shellsort and related algorithms, *J. of Algorithms*, 15(1993), 101-124.

[10] V.R. Pratt, *Shellsort and Sorting Networks*, Ph.D. Thesis, Stanford University, 1972.

[11] D.L. Shell, A high-speed sorting procedure, *Commun. ACM*, 2:7(1959), 30–32.

[12] Shellsort in Wikipedia in December 2016, http://en.wikipedia.org/wiki/Shellsort, 2016.

[13] M.A. Weiss, Empirical study of the expected running time of Shellsort, *Comput. J.*, 34(1991), 88–91

[14] A.C.C. Yao, An analysis of $(h, k, 1)$-Shellsort, *J. of Algorithms*, 1(1980), 14–50.