Lower Bounds for Shellsort *

C. Greg Plaxton [1] *Torsten Suel* [2]

April 20, 1996

Abstract

We show lower bounds on the worst-case complexity of Shellsort. In particular, we give a fairly simple proof of an $\Omega(n \lg^2 n/(\lg\lg n)^2)$ lower bound for the size of Shellsort sorting networks, for arbitrary increment sequences. We also show an identical lower bound for the running time of Shellsort algorithms, again for arbitrary increment sequences. Our lower bounds establish an almost tight trade-off between the running time of a Shellsort algorithm and the length of the underlying increment sequence.

1 Introduction

Shellsort is a classical sorting algorithm introduced by Shell in 1959 [15]. The algorithm is based on a sequence $H = h_0, \ldots, h_{m-1}$ of positive integers called an *increment sequence*. An input file $A = A[0], \ldots, A[n-1]$ of elements is sorted by performing an h_j-sort for every increment h_j in H, starting with h_{m-1} and going down to h_0. Every h_j-sort partitions the positions of the input array into congruence classes modulo h_j, and then performs Insertion Sort on each of these classes. It is not difficult to see that at least one of the h_j's must be equal to 1 in order for the algorithm to sort all input files properly. Furthermore, once some increment equal to 1 has been processed, the file will certainly be sorted. Hence, we may assume without loss of generality that $h_0 = 1$ and $h_j > 1$ for all $j > 0$.

The running time of Shellsort varies heavily depending on the choice of the increment sequence H. Most practical Shellsort algorithms set H to the prefix of a single, monotonically increasing infinite sequence of integers, using only the increments that are less than n. Shellsort algorithms based on such increment sequences are called *uniform*. In a *nonuniform* Shellsort algorithm, H may depend on the input size n in an arbitrary fashion.

---

*A preliminary version of these results appears in [11].

[1] Department of Computer Science, University of Texas at Austin, Austin, TX 78712. Supported by NSF Research Initiation Award CCR–9111591, and by the Texas Advanced Research Program under Grant No. 91–003658–480. Email: plaxton@cs.utexas.edu.

[2] Department of Computer Science, University of Texas at Austin, Austin, TX 78712. Supported by the Texas Advanced Research Program under Grant No. 91–003658–480, and by an MCD Fellowship of the University of Texas at Austin. Email: torsten@cs.utexas.edu.

A general analysis of the running time of Shellsort is difficult because of the vast number of possible increment sequences, each of which can lead to a different running time and behavior of the resulting algorithm. Consequently, many important questions concerning general upper and lower bounds for Shellsort have remained open, in spite of a number of attempts to solve them. Apart from pure mathematical curiosity, the interest in Shellsort is motivated by the good performance of many of the known increment sequences. The algorithm is very easy to implement and outperforms most other sorting methods on small or nearly sorted input files. Moreover, Shellsort is an in-place sorting algorithm, so it is very space-efficient.

1.1 Previous Results on Shellsort

The original algorithm proposed by Shell was based on the increment sequence given by $h_{m-1} = \lfloor n/2 \rfloor$, $h_{m-2} = \lfloor n/4 \rfloor, \ldots, h_0 = 1$. However, this choice of H leads to a worst case running time of $\Theta(n^2)$ if n is a power of 2. Subsequently, several authors proposed modifications to Shell's original sequence [9, 5, 8] in the hope of obtaining a better running time. Papernov and Stasevich [10] showed that the sequence of Hibbard [5], consisting of the increments of the form $2^k - 1$, achieves a running time of $O(n^{3/2})$. A common feature of all of these sequences is that they are *nearly geometric*, meaning that they approximate a geometric sequence within an additive constant.

An exception is the sequence designed by Pratt [13], which consists of all increments of the form $2^i 3^j$. This sequence gives a running time of $O(n \lg^2 n)$, which still represents the best asymptotic bound known for any increment sequence. In practice, the sequence is not popular because it has length $\Theta(\lg^2 n)$; implementations of Shellsort tend to use $O(\lg n)$-length increment sequences because these result in better running times for files of moderate size [6]. In addition, there is no hope of getting an $O(n \lg n)$-time algorithm based on a sequence of length $\omega(\lg n)$.

Pratt [13] also showed an $\Omega(n^{3/2})$ lower bound for all nearly geometric sequences. Partly due to this result, it was conjectured for quite a while that $\Theta(n^{3/2})$ is the best worst-case running time achievable by increment sequences of length $O(\lg n)$. However, in 1982, Sedgewick [14] improved this upper bound to $O(n^{4/3})$, using an approximation of a geometric sequence that is not nearly geometric in the above sense. Subsequently, Incerpi and Sedgewick [6] designed a family of $O(\lg n)$-length increment sequences with running times $O(n^{1+\epsilon/\sqrt{\lg n}})$, for all $\epsilon > 0$. Chazelle achieves a similar running time with a class of nonuniform sequences [6]; his construction is based on a generalization of Pratt's sequence.

The sequences proposed by Incerpi and Sedgewick are all within a constant factor of a geometric sequence, that is, they satisfy $h_j = \Theta(\alpha^j)$ for some constant $\alpha > 0$. Weiss [17, 20] showed that all sequences of this type take time $\Omega(n^{1+\epsilon/\sqrt{\lg n}})$, but his proof assumed an as yet unproven conjecture on the number of inversions in the Frobenius pattern. Based on this so-called Inversion Conjecture, he also showed an $\Omega(n^{1+\epsilon/\sqrt{\lg n}})$ lower bound for the $O(\lg n)$-length increment sequences of Chazelle. The question of existence of Shellsort algorithms with running time $O(n \lg n)$ remained unresolved.

The two classes of increment sequences given by Incerpi and Sedgewick and by Chazelle are of particular interest because they not only establish an improved upper bound for sequences of length $O(\lg n)$, but also indicate an interesting trade-off between the running time and the length of an increment sequence. Specifically, using a construction described in [6], it is possible to achieve better asymptotic running times by allowing longer increment sequences.

Another goal in the study of Shellsort is the construction of sorting networks of small depth and size. A Shellsort sorting network of depth $\approx 0.6 \lg^2 n$ based on increments of the form $2^i 3^j$ was given by Pratt [13]. Thus, his network came very close to the fastest known network at that time, due to Batcher [2], with depth $\approx 0.5 \lg^2 n$. In 1983, Ajtai, Komlós, and Szemerédi [1] designed a sorting network of depth $O(\lg n)$; however, their construction suffers from an irregular topology and a large constant hidden by the O-notation. This situation has motivated the search for $O(\lg n)$-depth sorting networks with simpler topologies or a smaller multiplicative constant. Shellsort has been considered a potential candidate for such a network [16], due to the rich variety of possible increment sequences and the lack of nontrivial general lower bounds. The lower bounds of Pratt and Weiss also apply to network size, but they only hold for very restricted classes of increment sequences.

Cypher [3] has established an $\Omega(n \lg^2 n / \lg \lg n)$ lower bound for the size of Shellsort networks. However, his proof technique only works for *monotone* increment sequences, that is, sequences that are monotonically increasing. Though this captures a very general class of sequences, it does not rule out the possibility of an $O(\lg n)$-depth network based on some nonmonotone sequence.

Recently, and independent of this paper, Poonen [12] has shown a lower bound of $\Omega(n \lg^2 n / (\lg \lg n)^2)$ that holds for arbitrary Shellsort algorithms. His lower bound has the form of a trade-off between the running time of a Shellsort algorithm and the length of the underlying increment sequence. The proof uses techniques from solid geometry and is quite intricate. A comparison of Poonen's results and the results in this paper will be given in the next subsection.

1.2 Overview of the Paper

In this paper we show lower bounds on the worst-case complexity of Shellsort. In particular, we give a fairly simple proof of an $\Omega(n \lg^2 n / (\lg \lg n)^2)$ lower bound for the size of Shellsort networks, for arbitrary increment sequences. We also establish an identical lower bound for the running time of Shellsort algorithms, again for arbitrary increment sequences. As in Poonen's paper, our lower bounds establish a trade-off between the running time of an algorithm and the length of the underlying increment sequence. This gives lower bounds for increment sequences of length $O(\lg n)$ that come very close to the best known upper bounds. At the other end of the spectrum, the trade-off implies that no increment sequence can match Pratt's upper bound with significantly fewer increments.

Our proof technique is based on purely combinatorial arguments, and we believe that it is significantly simpler than the technique used by Poonen. The technique also leads to certain improvements in the lower bounds, particularly in the trade-off between the running time and the length of the increment sequence. The result by Poonen, on the other hand, is of independent interest, since it establishes a variant of the Inversion Conjecture of Weiss [20] using a new geometric approach to the Frobenius Problem. The technique used in this paper is not based on a proof of the Inversion Conjecture. Instead, it shows how to "combine" Frobenius patterns to construct permutations with a large number of inversions. This result, together with the idea of dividing an increment sequence into "stages" (also called "intervals" in [12]), leads to the strong lower bounds of this paper.

Throughout this paper, we will limit our attention to increment sequences of length $O(\lg^2 n/(\lg\lg n)^2)$. Lower bounds for longer increment sequences are implied by the fact that Shellsort performs at least $\Omega(n)$ comparisons for every increment less than $n/2$. The results of this paper are presented in an "incremental" fashion, starting with a very basic argument for a restricted class of algorithms, and extending the lower bounds to more general classes in each of the subsequent sections.

The paper is organized as follows. Section 2 illustrates our proof technique by giving a simple and informal argument showing a lower bound for the depth of Shellsort networks based on monotone increment sequences. Section 3 introduces a number of definitions and simple lemmas, and then proceeds to give a formal proof of a general lower bound for the depth and size of arbitrary Shellsort networks. Section 4 then establishes a lower bound on the running time of adaptive Shellsort algorithms based on arbitrary increment sequences. Section 5 contains a discussion of our results and a comparison with the best known upper bounds. Finally, Section 6 lists some open questions for future research.

2 The Basic Proof Idea

In this section we illustrate our proof idea by giving a very simple and informal argument showing a polylogarithmic lower bound for the depth of any Shellsort network based on a monotone increment sequence of length at most $c\lg^2 n/(\lg\lg n)^2$, for some small c. In the following sections, we will then formalize and extend this technique to obtain more general lower bounds.

Let H be a monotone increment sequence with $m \leq c\lg^2 n/(\lg\lg n)^2$ increments. We now divide the increment sequence H into a number of *stages* $S_0, \ldots, S_{t-1}$. Every stage S_i is a set consisting of all increments h_j of H such that $n_i \geq h_j > n_{i+1}$, where $n_0, \ldots, n_t$ are chosen appropriately. We define the n_i by $n_0 = n$ and $n_{i+1} = n_i/\lg^k n_i$, for $i \geq 0$ and some fixed integer k. In this informal argument, we will not be concerned about the integrality of the expressions obtained. Note that the n_i divide the increment sequence into at least $\frac{\lg n}{k\lg\lg n}$ disjoint stages. There are at least $s \stackrel{\text{def}}{=} \frac{\lg n}{2k\lg\lg n}$ disjoint stages consisting of increments $h_j \geq n^{1/2}$.

By averaging, one of these stages, say S_i, will contain at most $m/s \leq \frac{2ck\lg n}{\lg\lg n}$ increments. Now suppose there exists an input permutation A such that, after sorting A by all increments in stages S_0 to S_i, some element is still $\Omega(n_i)$ positions away from its final position in the sorted file. Since H is monotone, we know that from now on only comparisons over a distance of at most n_{i+1} positions will be performed. Hence, we can conclude that the element has to pass through at least $\Omega(n_i/n_{i+1}) = \Omega(\lg^k n)$ comparators in order to reach its final, correct position.

To complete the proof we have to show the existence of a permutation A such that some element is still "far out of place" after sorting A by all increments in S_0 to S_i. We will only give an informal argument at this point; a formal proof will be given in the next subsection. Consider all permutations of length n of the following form: Every element is in its correct, final position, except for the elements in a block of size n_i, ranging from some position a to position $a + n_i - 1$ in the permutation. The elements in this block are allowed to be scrambled up in an arbitrary way. It is easy to see that a permutation of this form is already sorted by all increments greater than n_i, that is, all increments in stages S_0 to S_{i-1}. Hence, no exchanges will occur during these stages.

We now look at what happens in the block of size n_i during stage S_i. Note that no element outside the block will have an impact on the elements in the block. Thus, when we sort the permutation by some increment h_j with $n_i \geq h_j > n_{i+1}$, the new position of any element only depends on its previous position and on the elements in the at most $\frac{n_i}{h_j} \leq \lg^k n_i$ other positions in the block that are in its h_j-class. By our assumption, there are at most $m/s \leq \frac{2ck\lg n}{\lg\lg n}$ increments in stage S_i. Hence, the position of an element after stage S_i only depends on its position before the stage, which can be arbitrary, and on the elements in at most

$$\left(\lg^k n_i\right)^{m/s} \leq (\lg n_i)^{\frac{2ck^2 \lg n}{\lg\lg n}}$$

other positions. If we choose c such that $4ck^2 < 1 - \epsilon$, for some $\epsilon > 0$, then we get

$$\begin{aligned} (\lg n_i)^{\frac{2ck^2 \lg n}{\lg\lg n}} &\leq 2^{2ck^2 \lg n} \\ &\leq 2^{4ck^2 \lg n_i} \\ &= o(n_i), \end{aligned}$$

where the second inequality follows from $n_i \geq n^{1/2}$. This means that for large n, the position of an element in the block after sorting by all increments in S_i will only depend on the elements in $o(n_i)$ other positions in the block. If we assign the smallest elements in the block to these positions, then an element that is larger than these, but smaller than all other elements will end up in a position close to the largest elements after stage S_i. Hence, this element is $\Omega(n_i)$ positions away from its final position. All in all, we get the following result:

Theorem 2.1 *Let H be a monotone increment sequence of length at most $c\lg^2 n/(\lg\lg n)^2$, and let k be such that $4ck^2 < 1 - \epsilon$, for some $\epsilon > 0$. Then any sorting network based on H has depth $\Omega\left(\lg^k n\right)$.*

The above argument is quite informal and does not make use of the full potential of our proof technique; it has mainly been given to illustrate the basic proof idea and to demonstrate its simplicity. The above result implies that we cannot hope to match the $O(\lg^2 n)$-depth upper bound of Pratt [13] with any increment sequence of fewer than $\frac{(1-\epsilon)\lg^2 n}{16(\lg\lg n)^2}$ increments, thus answering a question left open by Cypher's lower bound [3]. It also implies that we cannot achieve polylogarithmic depth with increment sequences of length $o(\lg^2 n/(\lg\lg n)^2)$.

By extending the argument we will be able to show much stronger lower bounds for shorter increment sequences. More precisely, we can get a trade-off between depth and increment sequence length by choosing appropriate values for the integers n_i that divide the increment sequence into stages. We can also extend the result to adaptive Shellsort algorithms by showing the existence of an input such that not just one, but "a large number" of elements are "far out of place" after the sparse stage S_i.

3 Lower Bounds for Networks

In this section, we will show general lower bounds for the depth and size of Shellsort sorting networks. We start off by giving a number of definitions and simple lemmas. We then show how to formalize and generalize the argument of Section 2 to obtain a trade-off between the depth of a Shellsort network and the length of the underlying increment sequence. Next, we explain how the results on network depth imply lower bounds on the size of Shellsort networks. We conclude this section by extending our results to nonmonotone increment sequences.

3.1 Definitions and Simple Lemmas

This section contains a number of basic definitions and associated lemmas. All of the lemmas are quite straightforward and so their proofs have been omitted.

We will use $\Pi(n)$ to denote the set of $n!$ permutations over $\{0,\ldots,n-1\}$. A 0-1 permutation of length n is an n-tuple over $\{0,1\}$. Thus $\{0,1\}^n$ denotes the set of 2^n 0-1 permutations.

Throughout this paper we will assume that the input files are drawn from $\Pi(n)$. We will use the letters A, B, and C to denote elements from $\Pi(n)$, and we will use X, Y, and Z to denote 0-1 permutations. We say that a file A is *h-sorted* if $A[i] \leq A[i+h]$, for $0 \leq i < n-h$. The following trivial lemma arises as a special case of the last definition.

Lemma 3.1 *Every file of length n is h-sorted for any $h \geq n$.*

In the following, let $H = h_0, \ldots, h_{m-1}$ be an increment sequence of length $m \geq 1$. Let $\min(H)$ denote the smallest increment in H. We say that a file is *H-sorted* if and only if it is h_i-sorted for all i such that $0 \leq i < m$.

Definition 3.1 *Let template(H,n) denote the 0-1 permutation X obtained by setting $X[i]$ to 1 if and only if there exist nonnegative integers $a_0,\ldots,a_{m-1}$ such that*

$$i = \sum_{0 \leq j < m} a_j \cdot h_j.$$

Lemma 3.2 *The 0-1 permutation template(H,n) is H-sorted.*

Lemma 3.3 *The number of 1's in the 0-1 permutation template(H,n) is at most*

$$\left\lceil \frac{n}{\min(H)} \right\rceil^m .$$

Definition 3.2 *For any 0-1 permutation X of length n' with $0 \leq n' \leq n$, let pad(X,n) denote the 0-1 permutation Y of length n obtained by setting*

$$Y[i] = \begin{cases} X[i] & 0 \leq i < n', \text{ and} \\ 1 & n' \leq i < n. \end{cases}$$

Lemma 3.4 *Let X be an arbitrary 0-1 permutation of length n' with $0 \leq n' \leq n$. Then X is H-sorted if and only if pad(X,n) is H-sorted.*

Definition 3.3 *For any 0-1 permutation X of length $n \geq 0$, and any integer k, let shift(X,k) denote the 0-1 permutation Y of length n obtained by setting*

$$Y[i] = \begin{cases} 0 & \text{if } i < k, \text{ and} \\ X[i-k] & \text{if } k \leq i < n. \end{cases}$$

Lemma 3.5 *For any 0-1 permutation X and any integer k, if X is H-sorted, then shift(X,k) is also H-sorted.*

Definition 3.4 *For any 0-1 permutation X of length $n \geq 0$, let perm(X) denote the permutation Y in $\Pi(n)$ obtained by setting*

$$Y[i] = |\{j : X[j] < X[i] \ \vee \ ((X[j] = X[i] \ \wedge \ j < i)\}|.$$

Lemma 3.6 *Let X be an arbitrary 0-1 permutation. Then X is H-sorted if and only if perm(X) is H-sorted.*

We say that an element $A[j]$ of a permutation A in $\Pi(n)$ is k places out of position if $|A[j] - j| \geq k$. The following lemma will be used in Section 3.2 to obtain a simple lower bound on the depth of any Shellsort sorting network.

Lemma 3.7 *Let X denote any H-sorted 0-1 permutation of length n, let i denote the number of 1's in X, and let j denote the least index such that $X[j] = 1$ (if $i = 0$ then set $j = n$). Then element perm$(X)[j]$ is $n - i - j$ places out of position.*

3.2 A More General Lower Bound

We will now generalize the proof technique presented in the previous section to obtain a trade-off between the length of an increment sequence H and the lower bound for the depth of a sorting network based on H. For the sake of simplicity, we assume H to be monotone. It will be shown later that this assumption is not really necessary.

As before, we divide the increment sequence into stages $S_0, \ldots, S_{t-1}$, such that stage S_i contains all increments h_j with $n_i \geq h_j > n_{i+1}$. We define the n_i by $n_0 = n$ and $n_{i+1} = \lfloor n_i / \lg^k n_i \rfloor$, but we now assume k to be a function of the input size n and the increment sequence length m. Note that the number of stages t is determined by our choice of k. In particular, if we choose k such that

$$\left(\lg^k n\right)^s \leq n^{1/2},$$

then we get at least s stages that contain only elements that are greater than $n^{1/2}$. Solving this inequality we get

$$k = \left\lfloor \frac{\lg n}{2s \lg\lg n} \right\rfloor. \qquad (1)$$

as a possible choice of k. We will now formalize our earlier observation that an element can be "far out of place" after sorting by all increments up to stage S_i, provided that S_i contains "few" increments.

Lemma 3.8 *Let H be an increment sequence for permutations of length n, and suppose that for some integers ν, ν' with $0 < \nu' < \nu \leq n$ there are at most μ increments h_j with $\nu' < h_j \leq \nu$ in H. If $\left\lceil \frac{\nu}{\nu'} \right\rceil^{\mu} = o(\nu)$, then there exists an input file A such that: (i) A is sorted by all $h_j > \nu'$, and (ii) there exists an element in A that is $\Omega(\nu)$ places away from its final position.*

Proof: Let H' denote the subsequence of H consisting of all increments h_j such that $h_j > \nu'$, let H'' denote the subsequence of H' consisting of all increments h_j such that $h_j \leq \nu$, and let $X = \mathit{template}(H'', \nu)$. We know that X is H''-sorted by Lemma 3.2. Lemma 3.1 then implies that X is also H'-sorted. Note that $|H''| \leq \mu$ and $\min(H'') > \nu'$. Hence, by Lemma 3.3, the number of 1's in X is at most

$$\left\lceil \frac{\nu}{\nu'} \right\rceil^{\mu} = o(\nu).$$

Now let $Y = \mathit{pad}(X, n)$. By Lemma 3.4, the 0-1 permutation Y is H'-sorted. Furthermore, since the number of 1's in Y is exactly $n - \nu$ greater than the number of 1's in X, and $X[0] = Y[0] = 1$, Lemma 3.6 implies that the permutation $A \stackrel{\text{def}}{=} \mathit{perm}(Y)$ is H'-sorted, and by Lemma 3.7 some element of A is $\Omega(\nu)$ places out of position. □

Note that, in the preceding argument, we could have defined Y as $\mathit{shift}(\mathit{pad}(X, n), j)$ for any integer j with $0 \leq j \leq n - \nu$. We will make use of this observation to establish Corollary 3.1.1 below.

Theorem 3.1 *Any Shellsort sorting network based on a monotone increment sequence of length m has depth*

$$\Omega\left(2^{\frac{\lg n}{(2+\epsilon)\sqrt{m}}}\right),$$

for all $\epsilon > 0$.

Proof: We will partition the increment sequence H into at least $s = (1+\epsilon_0)\sqrt{m}$ disjoint stages consisting of increments h_j with $h_j \geq n^{1/2}$, for some $\epsilon_0 > 0$. By averaging, one of these stages, say S_i, will contain at most

$$\mu \stackrel{\text{def}}{=} \left\lfloor \frac{m}{s} \right\rfloor = \left\lfloor \frac{\sqrt{m}}{1+\epsilon_0} \right\rfloor$$

increments. Using Equation 1 we determine k as:

$$k = \left\lfloor \frac{\lg n}{2(1+\epsilon_0)\sqrt{m}\lg\lg n} \right\rfloor \tag{2}$$

Define $\nu = n_i$ and $\nu' = n_{i+1}$. We now have

$$\begin{aligned}
\left\lceil \frac{\nu}{\nu'} \right\rceil^{\mu} &\leq \left\lceil \lg^k \nu \right\rceil^{\frac{\sqrt{m}}{1+\epsilon_0}} \\
&\leq (\lg n)^{\frac{\lg n}{2(1+\epsilon_0)^2 \lg\lg n}} \\
&= 2^{\frac{\lg n}{2(1+\epsilon_0)^2}} \\
&\leq 2^{\frac{\lg \nu}{(1+\epsilon_0)^2}} \\
&= o(\nu).
\end{aligned}$$

Thus, we can apply Lemma 3.8. According to the lemma, there exists a permutation such that an element is $\Omega(n_i)$ positions away from its final position after stage S_i. Since all subsequent increments are less than or equal to n_{i+1}, this element must pass through at least $\Omega(n_i/n_{i+1})$ comparators. Using $n_i \geq n^{1/2}$ and Equation 2, we obtain for sufficiently large n

$$\begin{aligned}
\frac{n_i}{n_{i+1}} &\geq \lg^k n_i \\
&= (\lg n_i)^{\left\lfloor \frac{\lg n}{2(1+\epsilon_0)\sqrt{m}\lg\lg n} \right\rfloor} \\
&\geq (\lg n_i)^{\frac{\lg n}{2(1+\epsilon_0)(1+\epsilon_1)\sqrt{m}\lg\lg n}} \\
&\geq \left(2^{\frac{\lg\lg n_i}{\lg\lg n}}\right)^{\frac{\lg n}{2(1+\epsilon_0)(1+\epsilon_1)\sqrt{m}}} \\
&\geq \left(2^{\frac{1}{1+\epsilon_2}}\right)^{\frac{\lg n}{2(1+\epsilon_0)(1+\epsilon_1)\sqrt{m}}} \\
&= 2^{\frac{\lg n}{(2+\epsilon)\sqrt{m}}},
\end{aligned}$$

where ϵ is chosen to satisfy the inequality $(2+\epsilon) \geq 2(1+\epsilon_0)(1+\epsilon_1)(1+\epsilon_2)$. $\square$

3.3 A Lower Bound for Network Size

The depth lower bound of Theorem 3.1 also implies a lower bound on the size of any Shellsort network based on a monotone increment sequence. We will not give a formal proof of this result, since it arises as a special case of the lower bound for the running time of adaptive Shellsort established in the next section. Instead, we will briefly describe the main idea.

Lemma 3.8 shows how to construct an input file A that is sorted under all increments in stages S_0 to S_i of an increment sequence H such that one element $A[z]$ in A is "far out of place". In fact, as discussed immediately after the proof of Lemma 3.8, we can use the method of the lemma to construct a set of $n - n_i$ "shifted" versions of such an input file A. In particular, let A_j, $0 \leq j < n - n_i$, denote the input file obtained by setting Y to $\mathit{shift}(\mathit{pad}(X,n), j)$ instead of $\mathit{pad}(X,n)$. Note that $A_0 = A$. Let $A_0[z]$ be the element proven to be far out of place in A_0. By construction, the element $A_j[z+j]$ is far out of place in A_j. Due to the common structure of the input files, element $A_j[z+j]$ in file A_j will never pass through the same comparator as element $A_k[z+k]$ in A_k, for any $j \neq k$. Instead, the two elements will always be exactly $k - j$ positions apart at each level of the sorting network. This implies the result.

Corollary 3.1.1 *Any sorting network based on a monotone increment sequence of length m has size*

$$\Omega\left(n \cdot 2^{\frac{\lg n}{(2+\epsilon)\sqrt{m}}}\right),$$

for all $\epsilon > 0$.

We can now compare our result to the lower bound of $\Omega(n \lg^2 n / \lg\lg n)$ for network size given by Cypher [3]. The main difference between the two results is that Cypher gets a lower bound that is independent of the length of the increment sequence, while we get a trade-off between network size and increment sequence length. This makes our lower bound much stronger for short increment sequences. Our method also implies a lower bound of $\Omega(n \lg^2 n / (\lg\lg n)^2)$ for increment sequences of arbitrary length, since every increment increases the size of a Shellsort network by at least n. This is slightly weaker than Cypher's lower bound. However, Cypher's bound only applies to monotone increment sequences, while our result also holds for nonmonotone sequences, as will be shown in the next subsection. Another strength of our method is its simplicity and flexibility, which will make it possible to extend our lower bound to adaptive Shellsort algorithms and certain variations of Shellsort.

3.4 Nonmonotone Increment Sequences

So far, we have restricted our attention to monotone increment sequences. We will now show that this restriction is really unnecessary, and that the same lower bounds also apply to nonmonotone sequences. Recall that we obtained the depth lower bound by showing the existence of an input permutation such that an element is "far out of place" after the "sparse" stage S_i. More precisely, Lemma 3.8 showed the existence of a permutation A that is already sorted by all increments in stages S_0 through S_i and that contains such an element.

Thus, no exchanges are performed by the increments in stages S_0 through S_i on input A, and the lower bound follows. We will make use of the following well-known lemma (see, for example, [13]) in order to extend this argument to nonmonotonic increment sequences.

Lemma 3.9 *For any two increments h, h', if we h'-sort an h-sorted file, it stays h-sorted.*

Now suppose we have a nonmonotone increment sequence H. We can divide H into stages $S_0, \ldots, S_{t-1}$ as before, with stage S_i containing all increments h_j with $n_i \geq h_j > n_{i+1}$. Again, there exists a "sparse" stage S_i with few increments, and a permutation sorted by all increments in $S \stackrel{\text{def}}{=} S_0 \cup \cdots \cup S_i$ such that some element is "far out of place". If we take A as the input permutation, then by Lemma 3.9 A will stay sorted by all increments in S throughout the network. Hence, no exchanges will take place during the applications of Insertion Sort corresponding to increments in S. This implies that all of the exchanges needed to move the "out-of-place" element to its final position are performed by increments $h_j \leq n_{i+1}$, and the lower bound follows. The same reasoning also applies to the lower bound for network size, and to the results obtained in the next section. This gives us the following result:

Corollary 3.1.2 *Any sorting network based on an increment sequence of length m has size*

$$\Omega\left(n \cdot 2^{\frac{\lg n}{(2+\epsilon)\sqrt{m}}}\right),$$

for all $\epsilon > 0$.

Note that this result does not rule out the existence of nonmonotone increment sequences that perform better than the "corresponding" monotone sequences (that is, the sequences obtained by sorting the nonmonotone sequences into increasing order). It is an open question whether such sequences exist.

4 Adaptive Shellsort Algorithms

The results obtained so far all rely on the fact, established in Lemma 3.8, that we can construct an input file such that one element is "far away" from its final position in the sorted file. We were able to extend the lower bounds to network size due to the nonadaptive nature of sorting networks. However, the results for network size do not imply a lower bound for the running time of Shellsort algorithms that are adaptive.

In this subsection, we will establish such a lower bound. The high-level structure of the proof is the same as that of the depth lower bound in the last section; we only have to substitute Lemma 3.8 by a stronger lemma showing that there exists an input file A such that not just one, but "a large number" of the elements in A are "far away" from their final position. This result is formalized in the following lemma, which we will prove later in this subsection.

Lemma 4.1 *Let H be an increment sequence applied to input files of length n, and suppose that for some integers ν, ν' with $4 \leq \nu' < \nu \leq n$ there are at most μ increments h_j with $\nu' < h_j \leq \nu$ in H. If $\left\lceil \frac{\nu}{\nu'} \right\rceil^{\mu} \leq \nu / \lg^3 \nu$, then there exists an input file A such that: (i) A is sorted by all $h_j > \nu'$, and (ii) there exist $\Omega(n/\lg^3 \nu)$ elements in A that are $\Omega(\nu/\lg^2 \nu)$ places away from their final position.*

Given an increment sequence H, we can establish the lower bound for adaptive Shellsort algorithms by dividing H into stages in the same way as in the proof of Theorem 3.1, and then applying the above Lemma 4.1 instead of Lemma 3.8. The lower bound obtained is slightly weaker than the one for network size, since Lemma 4.1 only shows that a polylog fraction of the elements are a polylog fraction of n_{i-1} out of place. This gives the following theorem:

Theorem 4.1 *Any Shellsort algorithm based on an increment sequence of length m has running time*

$$\Omega\left(\frac{n}{\lg^5 n} \cdot 2^{\frac{\lg n}{(2+\epsilon)\sqrt{m}}}\right),$$

for all $\epsilon > 0$.

We remark that the exponent "5" in the preceding theorem is not the best possible. It results from summing the exponents "3" and "2" appearing in the statement of Lemma 4.1, which can be improved to "2" and "1", respectively. We have chosen to weaken these constants in order to simplify the proof of Lemma 4.1.

Comparing the bound of Theorem 4.1 to previous results we note that the lower bounds of Pratt [13] and Weiss [17] only hold for increment sequences approximating a geometric sequence, while the lower bound of Theorem 4.1 applies to all increment sequences. Also, the bound given by Weiss, which holds for a more general class than Pratt's bound, is based on an unproven conjecture about the number of inversions in certain input files.

The remainder of this subsection contains the proof of Lemma 4.1. To establish the result, we will need a few technical lemmas. The first two lemmas are straightforward and their proofs will be omitted. In particular, Lemma 4.2 is a straightforward generalization of Lemma 3.7.

Lemma 4.2 *Let X denote any H-sorted 0-1 permutation of length n, let i denote the number of 1's in X, let n' be such that $0 \leq n' < n - 2i$, and let $j = \sum_{0 \leq k < n'} X[k]$. Then at least j elements of perm(X) are $n - n' - i$ places out of position.*

Definition 4.1 *For any 0-1 permutation X of length n' such that $0 \leq n' \leq n$, let perm$^*(X, n)$ denote the permutation Y in $\Pi(n)$ obtained from $Z \stackrel{\text{def}}{=}$ perm(X) by setting*

$$Y[i] = Z[i \bmod n'] + \left\lfloor \frac{i}{n'} \right\rfloor \cdot n'.$$

Lemma 4.3 *Let X be any 0-1 permutation of length n' such that $0 \le n' \le n$. Then X is H-sorted if and only if perm$^*(X, n)$ is H-sorted. If i elements of perm(X) are j places out of position, then at least $i \cdot \lfloor n/n' \rfloor$ elements of perm$^*(X, n)$ are j places out of position.*

In the following, let H be an arbitrary increment sequence. Let ν be an integer such that $\nu \ge 4$, and define $\alpha \stackrel{\mathrm{def}}{=} \nu - 2\nu/\lg^2 \nu$ and $\beta \stackrel{\mathrm{def}}{=} \nu - \nu/\lg^2 \nu$.

Lemma 4.4 *Let X denote a 0-1 permutation of length $\nu \ge 4$ with $X[0] = 1$ and $\sum_{0 \le i < \nu} X[i] \le \nu/\lg^3 \nu$. Then there exists an integer k, $0 \le k \le (\nu - \alpha)\lfloor \lg \nu \rfloor$, such that the 0-1 permutation $Y \stackrel{\mathrm{def}}{=} \mathit{shift}(X, k)$ satisfies*

$$\sum_{0 \le i < \alpha} Y[i] \ge \sum_{\alpha \le i < \nu} Y[i].$$

Proof: Suppose, for the sake of contradiction, that

$$\sum_{0 \le i < \alpha} \mathit{shift}(X, k)[i] < \sum_{\alpha \le i < \nu} \mathit{shift}(X, k)[i]$$

holds for all k with $0 \le k \le (\nu - \alpha)\lfloor \lg \nu \rfloor$. This implies that $\sum_{0 \le i < \alpha - k} X[i] < \sum_{\alpha - k \le i < \nu - k} X[i]$. Using $R_k \stackrel{\mathrm{def}}{=} \sum_{0 \le i < \alpha - k} X[i]$, this can be rewritten as $R_k < R_{k-(\nu-\alpha)} - R_k$, or $R_k < \frac{1}{2} R_{k-(\nu-\alpha)}$. Hence,

$$R_{(\nu-\alpha)\lfloor \lg \nu \rfloor} < 2^{-\lfloor \lg \nu \rfloor} R_0,$$

and from $R_0 \le \sum_{0 \le i < \nu} X[i] \le \nu/\lg^3 \nu$ we get

$$R_{(\nu-\alpha)\lfloor \lg \nu \rfloor} < \frac{2}{\lg^3 \nu} < 1.$$

This is clearly a contradiction, since $X[0] = 1$ implies $R_{(\nu-\alpha)\lfloor \lg \nu \rfloor} \ge 1$. □

In the next lemma, given 0-1 permutations X and Y of length n, we will use $or(X, Y)$ to denote the 0-1 permutation Z obtained by setting bit $Z[i]$ to the logical OR of bits $X[i]$ and $Y[i]$, $0 \le i < n$. Clearly, if X and Y are H-sorted, then $or(X, Y)$ is also H-sorted.

Lemma 4.5 *Let X be an H-sorted 0-1 permutation of length ν with $x \stackrel{\mathrm{def}}{=} \sum_{0 \le i < \nu} X[i] \le \nu/\lg^3 \nu$. Let $x' \stackrel{\mathrm{def}}{=} \sum_{0 \le i < \alpha + k} X[i]$ where $(x \lg \nu)/2 \le k \le \nu/(2\lg^2 \nu)$. Then there exists an H-sorted 0-1 permutation Y of length ν with $\sum_{0 \le i < \nu} Y[i] \le 2x$ and*

$$\sum_{0 \le i < \alpha + 2k} Y[i] \ge 2\left(1 - \frac{1}{\lg \nu}\right) x'.$$

Proof: We will set Y to $Y_j \stackrel{\mathrm{def}}{=} or(X, \mathit{shift}(X, j))$ for some appropriately chosen integer j, $1 \le j \le k$. Note that by Lemma 3.5, any such 0-1 permutation Y_j is H-sorted, and it is easy to see that $\sum_{0 \le i < \nu} Y_j[i] \le 2x$ holds. Let $y'_j = \sum_{0 \le i < \alpha + 2k} Y_j[i]$. It remains to show the existence of an integer j_0, $1 \le j_0 \le k$, such that $y'_{j_0} \ge 2(1 - 1/\lg \nu)x'$. We will accomplish this by means of an averaging argument. We have

$$\sum_{1 \le j \le k} y'_j \ge 2kx' - \binom{x'}{2}.$$

(Here, $\binom{x'}{2}$ is the sum, over all i, of the number of ways the ith "1" in X can coincide with a "1" in some *shift*(X, j).) Hence, by averaging, there exists a j_0, $1 \le j_0 \le k$, such that

$$\begin{aligned} y'_{j_0} &\ge \frac{2kx' - \binom{x'}{2}}{k} \\ &\ge 2x' - \frac{x'-1}{\lg \nu} \\ &\ge 2\left(1 - \frac{1}{\lg \nu}\right) x', \end{aligned}$$

where the second inequality follows from $k \ge (x \lg \nu)/2$ and $x' \le x$. Now choose $Y = Y_{j_0}$. □

Lemma 4.6 *Let Y be an H-sorted 0-1 permutation of length ν with $\sum_{0 \le i < \nu} Y[i] \le \nu/\lg^3 \nu$ and*

$$\sum_{0 \le i < \alpha} Y[i] \ge \sum_{\alpha \le i < \nu} Y[i].$$

Then there exists an H-sorted 0-1 permutation Z of length ν such that $\sum_{0 \le i < \nu} Z[i] \le \nu/\lg^3 \nu$ and

$$\sum_{0 \le i < \beta} Z[i] = \Omega(\nu/\lg^3 \nu).$$

Proof: We will "transform" the given 0-1 permutation Y into a 0-1 permutation of the desired form by a sequence of applications of Lemma 4.5. Let $Y_0 \stackrel{\text{def}}{=} Y$. The jth application of Lemma 4.5 will be used to obtain Y_j from Y_{j-1}, $j \ge 1$. Let $y_j = \sum_{0 \le i < \nu} Y_j[i]$, and let $y'_j = \sum_{0 \le i < \alpha + 2^j y_0 \lg \nu} Y_j[i]$. Note that $y'_0 \ge \sum_{0 \le i < \alpha} Y[i] \ge y_0/2$. Then Lemma 4.5 implies that $y_j \le 2^j y_0$, and

$$y'_j \ge \left[2\left(1 - \frac{1}{\lg \nu}\right)\right]^j y'_0$$

for $j \le \lg \nu - \lg y_0 - 3 \lg \lg \nu$ (the latter inequality ensures that $\alpha + 2^j y_0 \lg \nu \le \beta$). Setting j_0 to $\lfloor \lg \nu - \lg y_0 - 3 \lg \lg \nu \rfloor$, we find that $y_{j_0} \le \nu/\lg^3 \nu$ and

$$y'_{j_0} \;\ge\; \left[2\left(1 - \frac{1}{\lg \nu}\right)\right]^{j_0} y'_0 \;=\; \Theta(2^{j_0} y'_0) \;=\; \Omega(2^{j_0} y_0).$$

Thus $y'_{j_0} = \Omega(\nu/\lg^3 \nu)$ and we can choose $Z = Y_{j_0}$. □

Given the above lemmas, we are now ready to proceed with the proof of Lemma 4.1.

Proof: Let H' denote the subsequence of H consisting of exactly those increments h_j such that $h_j > \nu'$, let H'' denote the subsequence of H' consisting of exactly those increments h_j such that $h_j \leq \nu$, and let $X = \mathit{template}(H'', \nu)$. We know that X is H''-sorted by Lemma 3.2. Note that $|H''| \leq \mu$ and $\min(H'') > \nu'$. Hence, by Lemma 3.3, we have

$$\sum_{0 \leq i < \nu} X[i] \leq \left\lceil \frac{\nu}{\nu'} \right\rceil^{\mu} \leq \frac{\nu}{\lg^3 \nu}.$$

By Lemmas 3.5 and 4.4, the existence of X implies the existence of a 0-1 permutation Y of length ν such that Y is H''-sorted and

$$\sum_{0 \leq i < \alpha} Y[i] \geq \sum_{\alpha \leq i < \nu} Y[i].$$

The existence of Y then establishes, via Lemma 4.6, the existence of a 0-1 permutation Z of length ν such that:

- Z is H''-sorted,
- $\sum_{0 \leq i < \nu} Z[i] \leq \nu / \lg^3 \nu$, and
- $\sum_{0 \leq i < \beta} Z[i] = \Omega(\nu / \lg^3 \nu)$.

By Lemma 3.1 and Lemma 3.6 we know that $B = \mathit{perm}(Z)$ is H'-sorted, and Lemma 4.2 implies that B contains $\Omega(\nu / \lg^3 \nu)$ elements that are $\Omega(\nu / \lg^2 \nu)$ places out of position. Let $A = \mathit{perm}^*(Z, n)$. By Lemma 4.3, A is H'-sorted and contains $\Omega(n / \lg^3 \nu)$ elements that are $\Omega(\nu / \lg^2 \nu)$ places out of position. $\square$

5 Discussion

In this paper, we have given a fairly simple proof of a lower bound of $\Omega(n \lg^2 n / (\lg \lg n)^2)$ for the size of any Shellsort network, thus ruling out the existence of a network of size $O(n \lg n)$ based on a nonmonotone increment sequence. By extending our argument to the case of adaptive algorithms, we have also established a general lower bound for Shellsort that holds for arbitrary increment sequences.

Our lower bound can be further generalized to a fairly large class of "Shellsort-like" algorithms, including the *Shaker Sort* algorithm of Incerpi and Sedgewick [7, 19] as well as other algorithms proposed by Knuth [8] and Dobosiewicz [4]. Poonen [12] has formally defined a class of such algorithms, called *Shellsort-type* algorithms, and has shown how to extend his lower bound to this class. We will not elaborate further on such possible extensions, and instead refer the reader to the presentation in [12].

The lower bound of Theorem 4.1 establishes a trade-off between the running time of a Shellsort algorithm and the length of the underlying increment sequence. We will now compare this lower bound trade-off with the best known upper bound trade-off given by the nonuniform increment sequences of Chazelle (see the non-uniform case of Theorem 3 in [6]).

Expressing the running time as a function of the increment sequence length m we obtain the following bounds:

$$\begin{array}{lrcl} \text{Lower Bound:} & T & \geq & \frac{n}{\lg^5 n} \cdot n^{\frac{1}{(2+\epsilon)\sqrt{m}}} \\ \text{Upper Bound:} & T & \leq & mn \cdot n^{\frac{2}{\sqrt{m}}} \end{array}$$

Note that both the factor $1/\lg^5 n$ in the lower bound and the factor m in the upper bound are only significant for increment sequences of length $\Omega(\lg^2 n/(\lg\lg n)^2)$. In every other case, the upper and lower bounds differ only by a factor of $4+\epsilon$ in the exponent. In the lower bound trade-off shown by Poonen, the constant in the exponent is $1/432$ instead of $1/(2+\epsilon)$.

We can also express the length of the increment sequence as a function of the running time. In this case, for $m = o(\lg^2 n/(\lg\lg n)^2)$, the lower and upper bounds are only a constant factor apart. This means that, for a given T, the length of the increment sequence of Chazelle that achieves running time T is only a factor of $16+\epsilon$ larger than the minimum length possible under our lower bound trade-off. (For Poonen's result, this factor would be much larger.) In other words, one cannot hope to match the running time of Chazelle's sequences with significantly shorter increment sequences.

6 Open Questions

The primary remaining challenge in the study of Shellsort seems to be the virtual nonexistence of both upper and lower bounds for the average case complexity. A result for a particular increment sequence is given by Knuth [8], who determines an average case running time of $\Omega(n^{3/2})$ for Shell's original sequence. Increment sequences of the form $(h,1)$ and $(h,k,1)$ were investigated by Knuth [8] and Yao [21], respectively. Weiss [18] conducted an extensive empirical study and conjectured that Shellsort will on average not perform significantly better than in the worst case. Any general upper and lower bound for the average case would certainly be very interesting.

It would be nice to close the remaining gap between the upper and lower bounds. Our lower bound trade-off comes quite close to the known upper bounds, but there is certainly still room for improvement.

Finally, one might try to find interesting "Shellsort-like" algorithms that are not covered by our proof technique, and that lead to improved running times.

Acknowledgements

We thank Mark Weiss and the anonymous referee for helpful comments.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c\log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.

[2] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference,* vol. 32, pages 307–314, 1968.

[3] R. E. Cypher. A lower bound on the size of Shellsort sorting networks. *SIAM Journal on Computing*, 22:62–71, 1993.

[4] W. Dobosiewicz. An efficient variation of Bubble Sort. *Information Processing Letters*, 11:5–6, 1980.

[5] T. N. Hibbard. An empirical study of minimal storage sorting. *Communications of the ACM*, 6:206–213, 1963.

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

[7] J. Incerpi and R. Sedgewick. Practical variations of Shellsort. *Information Processing Letters*, 26:37–43, 1987.

[8] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, MA, 1973.

[9] R. Lazarus and R. Frank. A high-speed sorting procedure. *Communications of the ACM*, 3:20–22, 1960.

[10] A. Papernov and G. Stasevich. A method for information sorting in computer memories. *Problems of Information Transmission*, 1:63–75, 1965.

[11] C. G. Plaxton, B. Poonen, and T. Suel. Improved lower bounds for Shellsort. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 226–235, October 1992.

[12] B. Poonen. The worst case in Shellsort and related algorithms. *Journal of Algorithms*, 15:101–124, 1993.

[13] V. R. Pratt. *Shellsort and Sorting Networks.* PhD thesis, Department of Computer Science, Stanford University, December 1971. Also published by Garland, New York, 1979.

[14] R. Sedgewick. A new upper bound for Shellsort. *Journal of Algorithms*, 7:159–173, 1986.

[15] D. L. Shell. A high-speed sorting procedure. *Communications of the ACM*, 2:30–32, 1959.

[16] J. S. Vitter and P. Flajolet. Average-case analysis of algorithms and data structures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 431–524. Elsevier/MIT Press, 1990.

[17] M. A. Weiss. *Lower Bounds for Shellsort.* PhD thesis, Princeton University, Department of Computer Science, June 1987.

[18] M. A. Weiss. Empirical study of the expected running time of Shellsort. *The Computer Journal*, 34:88–91, 1991.

[19] M. A. Weiss and R. Sedgewick. Bad cases for Shaker-sort. *Information Processing Letters*, 28:133–136, 1988.

[20] M. A. Weiss and R. Sedgewick. Tight lower bounds for Shellsort. *Journal of Algorithms*, 11:242–251, 1990.

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