THE WORST CASE IN SHELLSORT AND RELATED ALGORITHMS

BJORN POONEN

ABSTRACT. We show that sorting a sufficiently long list of length N using Shellsort with m increments (not necessarily decreasing) requires at least $N^{1+c/\sqrt{m}}$ comparisons in the worst case, for some constant $c > 0$. For $m \leq (\log N/\log\log N)^2$ we obtain an upper bound of the same form. We also prove that $\Omega(N(\log N/\log\log N)^2)$ comparisons are needed regardless of the number of increments. Our approach is general enough to apply to other sorting algorithms, including Shaker-sort, for which an even stronger result is proved.

1. Introduction

Shellsort is a general-purpose sorting algorithm that was invented by Shell in 1959 [14]. Empirical results show that it is competitive with the fastest sorting algorithms, especially when N, the number of elements to be sorted, is not too large. The algorithm is very simple to describe. If $h \geq 1$, to *h-sort* a list $L[1], \ldots, L[n]$ is to apply insertion sort to each of the sublists $L[i], L[i+h], \ldots, L[i+2h]$, etc., where $1 \leq i \leq h$. A list for which these sublists are in order is said to be *h-ordered*. Shellsort sorts a list by performing an h_1-sort, an h_2-sort, ..., and finally an h_m-sort, where $h_1, h_2, \ldots, h_m$ are positive integers called the *increments* and $h_m = 1$. The choice $h_m = 1$ guarantees that the resulting list is in order, because 1-sorting a list is the same as sorting it using insertion sort.

As far as complexity is concerned, insertion sort is a series of compare-exchange operations between pairs of adjacent elements, where a compare-exchange consists of comparing two elements and exchanging them, if necessary, to put them in order. Thus Shellsort also can be considered a series of compare-exchanges, between pairs of elements $L[i]$ and $L[i+h]$ for various h's (the increments).

Shaker-sort, proposed by Incerpi and Sedgewick [5], is another general-purpose sorting algorithm which sorts a list in a number of passes, each having an associated increment h. To *h-shake* a list $L[1], \ldots, L[n]$ is to perform compare-exchanges on the following pairs of positions, in the order listed:

$$(1, 1+h), (2, 2+h), \ldots, (N-h-1, N-1), (N-h, N),$$
$$(N-h-1, N-1), (N-h-2, N-2), \ldots, (2, 2+h), (1, 1+h).$$

Note that the distance between the elements of each pair is h. Shaker-sort sorts a list by performing an h_1-shake, an h_2-shake, ..., and an h_m-shake, for some sequence $h_1, h_2, \ldots, h_m$. Instead of assuming $h_m = 1$, which would not guarantee that the result was sorted, we assume that insertion sort is applied at the end, after the h_m-shake. (Alternatively, we could perform 1-shakes repeatedly at the end until the list is sorted.)

Date: October 11, 1992.

For the final version of this article, see *J. Algorithms* **15**, no. 1 (July 1993), 101–124.

Often one defines an infinite sequence of integers and takes the integers less than N, in decreasing order, as the increments. Such increment sequences are said to be *uniform*. But in general, the increment sequence need not be decreasing, and it may depend on N.

Choosing the increment sequence wisely is crucial for good performance, both in theory and in practice, so much research has gone into finding the best sequence. Shell [14] suggested using $(\lfloor N/2 \rfloor, \lfloor N/4 \rfloor, \lfloor N/8 \rfloor, \ldots, 1)$ in his algorithm, but when N is a power of 2, this sequence requires $\Theta(N^2)$. (All bounds listed here are for the worst case.) Lazarus and Frank [7], Hibbard [3], and Knuth (Section 5.2.1 in [6]) proposed modifications to Shell's sequence, and Papernov and Stasevich [9] showed that Hibbard's sequence required only $O(N^{3/2})$ comparisons. All of these sequences were within a constant of a geometric sequence with integer ratio, and Pratt [10] showed that Shellsort with any such sequence would require $\Omega(N^{3/2})$ comparisons. Pratt also gave a sequence with $\Theta((\log N)^2)$ increments that required only $\Theta(N(\log N)^2)$ comparisons. (See Proposition 4 in this paper.) But this sequence gave a slow algorithm in practice, because of the large number of increments involved.

For a while it was believed, because of Pratt's lower bound, that Shellsort with any sequence of $O(\log N)$ increments would require $\Omega(N^{3/2})$ comparisons. But then Sedgewick [11] gave an example of such a sequence which required only $O(N^{4/3})$ comparisons, and Incerpi and Sedgewick [4] later reduced the upper bound to $O(N^{1+\epsilon/\sqrt{\log N}})$, for any $\epsilon > 0$. Weiss and Sedgewick [16] made a conjecture about inversions which implied that this bound was best possible. We prove a version of this conjecture, and hence, using an argument similar to theirs, we can deduce that this is the best possible bound for $O(\log N)$ increments.

Very recently, Cypher [2] proved that all Shellsort sorting networks based on monotonically decreasing increment sequences require $\Omega(N(\log N)^2/\log\log N)$ comparators. The differences between this lower bound and ours are the following:

- Most importantly, his lower bound applies only to Shellsort *networks*. This means that the entire sequence of comparisons must be specified before looking at the given list. By contrast, when Shellsort is used as a general-purpose sorting algorithm, the results of previous comparisons are used in choosing the comparisons to be done next. For example, the part of the algorithm that inserts an element into a sorted list moves the element only as far as it needs to be moved. With a network, one would have to allow for the possibility that the element would move the entire length of the list, unless something was already known about the size of the element.
- Our results apply to other algorithms, such as Shaker-sort.
- We do not require that the increments be decreasing.
- We do not require that the h-sort be done by insertion sort. It may be done by any series of compare-exchanges on pairs separated by h.
- We have a bound as a function of the number of increments, in addition to a general bound.
- Cypher's bound is slightly stronger than ours: $\Omega(N(\log N)^2/\log\log N)$ vs. $\Omega(N(\log N/\log\log N)^2)$.

We now outline the rest of our paper. First, in Section 2, we discuss the generalizations of Shellsort to be considered. In Section 3, we describe the Frobenius problem, and prove a result showing its connection with Shellsort. This result is used in Section 4 to obtain upper bounds on the worst case of Shellsort, as a function of the number of increments used.

The rest of the paper is devoted to proving the corresponding lower bound theorems. Section 5 defines inversions, which are used to measure how unordered a list is. Sections 6 and 7 discuss the Frobenius problem in greater depth. These two sections contain the hardest part of the argument. The results of those two sections are used in Section 8 to show that that a certain list, the Frobenius pattern, has many inversions even though it is h-ordered for many h's. Section 9 uses the Frobenius pattern to build other such lists. Because Sections 6, 7, 8, and 9 involve so much detail, it may be best for the reader to skip them on a first reading, and to look at only the important Lemma 18, which is the only result from those sections that is used later.

Finally, in Section 10 we prove three lower bound theorems. Theorem 3 says that if not enough (only m) increments are used, many ($N^{1+c_2/\sqrt{m}}$ for some constant c_2) compare-exchanges will have to be done in the worst case. This is proved by dividing $[\sqrt{N}, N/2]$ into intervals and showing that if some interval contains few increments, then Lemma 18 can be applied to give a list with many inversions so that many compare-exchanges are needed to sort the list. Theorem 4 gives a lower bound on the worst case which is independent of the number of increments. It follows easily from Theorem 3. The argument is as follows: the number of comparisons per increment is always at least $\Omega(N)$. So if the number of increments m is large, many comparisons are needed. On the other hand, if m is small,, again many comparisons are needed by Theorem 3. The last result of the section, Theorem 5, is a strengthening of the lower bound for Shaker-sort for certain increment sequences. This was proved by Weiss and Sedgewick [15], assuming their inversion conjecture.

In proving our theorems, we do not attempt to obtain the best possible constants. Also, wherever convenient, we assume N is sufficiently large. (At the very end of the proof of Theorem 3, we need $N \geq 2^{2^{36}}$, for example!) We could certainly make our theorems work for smaller N and improve our constants by making more precise estimates in the course of the proof, but it is probably not worth the effort, because it seems nearly impossible to improve the result so much that it gives a meaningful general lower bound for any $N < 10^{12}$. Nevertheless, we feel that the existence of an $\Omega(N(\log N/\log\log N)^2)$ bound suggests that even in practical situations, Shellsort may not be the best algorithm to use when N is large, say $N > 10^7$.

We end our paper by mentioning a few unanswered questions.

2. Shellsort-type Algorithms

We now describe the general class of algorithms we will consider. A sorting algorithm is a *Shellsort-type* algorithm if it satisfies the following:

(1) The algorithm sorts a list $L[1], \ldots, L[N]$ in a number of passes, each with an associated increment h. During a pass with increment h, the algorithm performs compare-exchanges (one at a time) on pairs $L[i], L[i+h]$ for $1 \leq i \leq N-h$.
(2) At least $\Omega(N)$ comparisons are done during each pass with increment $h < N/2$.
(3) If the list is k-ordered at the beginning of a pass, the list is k-ordered at the end of the pass also.

The sequence of increments need not be decreasing, and it need not end in 1, so long as the algorithm actually sorts. It may contain duplicates. The sequence also may depend on N, but it must be fixed before looking at the list to be sorted.

Because of condition 3, it is not obvious that even Shellsort is a Shellsort-type algorithm. (That Shellsort satisfies 3 is Theorem K in Section 5.2.1 of [6].) Even more surprising is that Shaker-sort satisfies 3. To prove this, we will use the following easy result, which is Exercise 21 in Section 5.3.4 of [6].

Proposition 1. *If L, L' are lists of the same length N, and $L[i] \le L'[i]$ for all i, then the inequalities still hold after the same series of compare-exchanges is performed on both lists.*

Proof. It suffices to consider a single compare-exchange, say on (j,k), with $j < k$. Let M and M' be the lists after the compare-exchange. For $i \ne j,k$, $M[i] = L[i] \le L'[i] = M'[i]$. Also

$$M[j] = \min\{L[j], L[k]\} \le \min\{L'[j], L'[k]\} = M'[j]$$

$$M[k] = \max\{L[j], L[k]\} \le \max\{L'[j], L'[k]\} = M'[k]$$

so the result follows. □

Proposition 2. *Shellsort and Shaker-sort are Shellsort-type algorithms.*

Proof. It is clear that condition 1 is satisfied, and that 2 is satisfied by Shaker-sort. Condition 2 is also true for Shellsort, because $N - h$ comparisons are necessary in an h-sort, even if only to check that the list is h-ordered. (This is $\Omega(N)$ when $h < N/2$.) So it remains to show that 3 holds.

Let L denote the k-ordered list at the beginning of a pass with increment h. Consider the following 2 by $(N+k)$ array:

$-\infty$	$\cdots$	$-\infty$	$L[1]$	$\cdots$	$L[N-k]$	$L[N-k+1]$	$\cdots$	$L[N]$
$L[1]$	$\cdots$	$L[k]$	$L[k+1]$	$\cdots$	$L[N]$	$+\infty$	$\cdots$	$+\infty$

Since L is k-ordered, each entry in the first row is less than or equal to the entry below it. Consider performing an h-sort on both rows or an h-shake on both rows. This has the same effect as performing it on the two copies of L within the array, because the $-\infty$ and $+\infty$ terms stay where they are. But this can be done by a series of compare-exchanges, so by Proposition 1, in the resulting array each entry of the first row is less than or equal to the entry below it. So the resulting list is k-ordered. □

Proposition 2 holds for Shellsort even if the method used to do the h-sort is not insertion sort, as long as conditions 1 and 2 still hold. This is because the result of an h-sort done by compare-exchanges at distance h is independent of the method. Shellsort-type algorithms have the following property, which will be used in proving our lower bound theorems.

Lemma 1. *If a Shellsort-type algorithm is applied to a list which is h-ordered for all increments $h \ge \beta$, where β is some fixed cutoff, then the passes with increments $h \ge \beta$ do nothing to the list.*

Proof. By condition 3, the list will be h-ordered at the beginning of every pass, for each increment $h \ge \beta$. So the compare-exchanges at distance h done during a pass with increment $h \ge \beta$ can't do anything. □

If we dropped condition 3 from the definition of Shellsort-type algorithms, we could still prove Lemma 1 under the assumption that the increments were decreasing, because then at the beginning of a pass with increment $h \ge \beta$, the list would be in its initial state, and would hence be h-ordered already.

3. The Frobenius Problem

Suppose $a_1, \dots, a_r$ is a sequence of positive integers with $\gcd(a_1, \dots, a_r) = 1$. Then every integer k can be expressed in the form $n_1a_1 + \cdots + n_ra_r$ for some integers $n_1, \dots n_r$. We say k is *expressible* if the n_i's can be chosen nonnegative, and *inexpressible* otherwise. It is not hard to show that if k is sufficiently positive, k is expressible. So a natural question to ask is, what is the largest inexpressible integer $g = g(a_1, \dots, a_r)$? For example, suppose $r = 2$, $a_1 = 3$, $a_2 = 5$. Then the expressible integers are $0, 3, 5, 6, 8, 9, 10, 11, 12, \dots$, so $g = 7$. (Note that if a_1 consecutive integers are expressible, then all larger integers will be expressible.)

The problem of determining g is known as the Frobenius problem, because according to Brauer [1], Frobenius mentioned it in his lectures. Although the problem dates back to the nineteenth century [13], no general formula for g in terms of $a_1, \dots, a_r$ has been found. The problem has been solved in some special cases, however. (See [12] for some of these.) For example, the following result is well-known. For a proof, see Exercise 21 in Section 5.2.1 of [6] or Example 1 on page 4 of [12].

Proposition 3. *If* $\gcd(a_1, a_2) = 1$, $g(a_1, a_2) = a_1a_2 - a_1 - a_2$.

Our next result shows how knowledge of g can be useful in obtaining upper bounds on the time required to do an h-sort.

Lemma 2. *Let $a_1, \dots, a_r$ be positive integers with greatest common divisor 1. The number of steps required to h-sort a list $L[1], \dots, L[N]$ which is already (a_1h)-ordered, ..., (a_rh)-ordered, is $O(Ng)$, where $g = g(a_1, \dots, a_r)$.*

Proof. By definition of g, every multiple of h greater than gh is a sum $n_1(a_1h) + \cdots + n_r(a_rh)$ for some $n_i \geq 0$. Since L is (a_1h)-ordered, ..., (a_rh)-ordered, it follows from transitivity that $L[j - bh] \leq L[j]$ for all $b > g$. So during the h-sort, inserting each element $L[j]$ into its proper position in its sublist will involve moving it at most g steps of h. There are N elements to put into place, so the total number of steps required is at most Ng. □

4. Upper Bounds on the Worst Case

We now prove the best known upper bound for Shellsort, due to Pratt [10]. All logarithms in this paper are to the base 2 unless explicitly written otherwise.

Proposition 4. *If the numbers of the form 2^a3^b less than N are used in decreasing order as increments in Shellsort, then the worst case and average case running times are $\Theta(N(\log N)^2)$.*

Proof. By the time the list is to be h-sorted, the list will have already been $(2h)$-sorted and $(3h)$-sorted, and it will still be $(2h)$-ordered and $(3h)$-ordered by condition 3 in the definition of Shellsort-type algorithms. So by Lemma 2, the h-sort requires only $O(Ng(2, 3)) = O(N)$ steps. There are $\lceil \log_2 N \rceil$ powers of 2 less than N and $\lceil \log_3 N \rceil$ powers of 3 less than N, so there are at most $\lceil \log_2 N \rceil \lceil \log_3 N \rceil = O((\log N)^2)$ increments. Thus the total number of steps is always $O(N(\log N)^2)$.

By condition 2 in the definition of Shellsort-type algorithms, $\Omega(N)$ comparisons are required for each increment $h < N/2$. And there are $\Omega((\log N)^2)$ increments less than $N/2$, since there are $\lceil \log_2 \sqrt{N/2} \rceil$ powers of 2 less than $\sqrt{N/2}$, which may be paired with the $\lceil \log_3 \sqrt{N/2} \rceil$ powers of 3 less than $\sqrt{N/2}$. So the total number of steps is $\Omega(N(\log N)^2)$ also. □

Even though Pratt's increment sequence gives the best known theoretical bound, it does not give the fastest algorithm in practice. (See [4] for some empirical results.) This may be because the other sequences actually have an average case better than $\Theta(N(\log N)^2)$. Or it may be that N is not sufficiently large yet. But certainly one reason is that too many increments are being used. (There are $\Theta((\log N)^2)$ of them.) In order to get reasonable running times in practice, it seems necessary to limit the number of increments to $O(\log N)$. So what time bounds can we get under this restriction? We shall see that the asymptotic worst case running time worsens as we reduce the number of increments we are allowed to use.

For now, we give upper bounds where the number of increments is restricted. A result very similar to the following (the special case $m = \Theta(\log N)$) was proved by Chazelle. (See Theorem 3 (non-uniform case) in [4].)

Theorem 1. *For all m, there exists a sequence of at most m increments for which Shellsort requires $O(mN^{1+O(1/\sqrt{m})})$ steps in the worst case.*

Proof. The proof is a generalization of the proof of Pratt's result. If $m \geq (\log N)^2$ we are done, since we can take the sequence of Proposition 4. Otherwise, pick the smallest integer $\alpha \geq 3$ such that $\lfloor \log_{\alpha-1} N \rfloor \lfloor \log_\alpha N \rfloor \leq m$. There are at most m numbers of the form $(\alpha-1)^a \alpha^b$ less than N, so we may use them as increments, in decreasing order. As in the proof of Proposition 4, each h-sort will require $O(Ng(\alpha-1, \alpha))$ steps. By Proposition 3, this is $O(N\alpha^2)$. Since there are at most m increments, the total running time will be $O(mN\alpha^2)$. Now $m < (\log N)^2$, so by definition of α,

$$\begin{aligned} \left(\frac{\log N}{\log \alpha}\right)^2 &= \Theta(m) \\ \alpha &= N^{\Theta(1/\sqrt{m})} \\ O(mN\alpha^2) &= O(mN^{1+O(1/\sqrt{m})}) \end{aligned}$$

as desired. □

Theorem 2. *There is a constant c_1 such that for all sufficiently large N and all $m \leq (\log N / \log\log N)^2$, there exists a sequence of at most m increments for which Shellsort requires at most $N^{1+c_1/\sqrt{m}}$ steps.*

Proof. By Theorem 1, there are constants c, c' such that we can find a sequence of at most m increments for which Shellsort requires at most $cmN^{1+c'/\sqrt{m}}$ steps. But for sufficiently large N,

$$\log N^{1/\sqrt{m}} = \frac{\log N}{\sqrt{m}} \geq \log\log N \geq \log c$$

and

$$\log N^{2/\sqrt{m}} \geq 2\log\log N \geq \log m$$

so

$$N^{1+(3+c')/\sqrt{m}} \geq cmN^{1+c'/\sqrt{m}}$$

and we may take $c_1 = 3 + c'$. □

In contrast with Proposition 4, the sequences produced by Theorems 1 and 2 are non-uniform. (The sequences depend on α which depends on N.) But it should be possible to obtain the same upper bounds for uniform sequences by generalizing Theorem 3 in [4], by choosing the sequence of primes $a_1, a_2, a_3, \ldots$ (of the proof there) with an appropriate rate of growth (not necessarily exponential).

5. Inversions

An *inversion* in a list $L[1], \ldots, L[N]$ is an ordered pair (i, j) where $i < j$ and $L[i] > L[j]$. The number of inversions in a list is a measure of how much work must be done to sort it. For example, an already sorted list has no inversions, a list of length N in reverse order (such as $N, N-1, \ldots, 1$) has $N(N-1)/2$ inversions, and a random list of length N is expected to have $N(N-1)/4$ inversions, since each of the $N(N-1)/2$ pairs (i, j) with $i < j$ is an inversion with probability $1/2$.

Counting inversions is especially useful when analyzing algorithms which sort by exchanges because of the following proposition:

Proposition 5. *If $i < j$, exchanging elements $L[i]$ and $L[j]$ in a list changes the number of inversions by less than $2(j-i)$.*

Proof. We need only consider the pairs which involve either i or j. If $k < i$, then (k, i) is an inversion in the new list *iff* (k, j) was an inversion in the original list, and (k, j) is an inversion in the new list *iff* (k, i) was an inversion in the original list. So there is no change in the number of inversions involving k where $k < i$. Similarly we need not consider the pairs involving k where $k > j$. The only pairs that remain are (i, k) and (k, j) for $i < k < j$, and (i, j) itself. But there are only $2(j-i)-1$ of these. $\square$

We also need a result about inversions of a sublist. In proving this we will use the following inequality:

Lemma 3. *If $k \geq 2$ and $0 \leq b_1, \ldots, b_k \leq t$, then there exists s, $1 \leq s \leq k-1$, such that $b_s(t - b_{s+1}) \geq b_1(t - b_k)/4$.*

Proof. Without loss of generality we may replace b_1 with t. This can only shrink the set of s which satisfy the inequality. Similarly we may assume $b_k = 0$. Choose the largest s such that $b_s \geq t/2$, so $1 \leq s \leq k-1$. Then $b_{s+1} < t/2$, so $b_s(t - b_{s+1}) \geq t^2/4 = b_1(t - b_k)/4$. $\square$

The following result says (roughly) that in a list of N zeros and ones where a fraction p of the pairs are inversions, one can find a sublist of given length $n \leq N$ where at least a fraction $p/256$ of the pairs are inversions.

Proposition 6. *If $2 \leq n \leq N$ and $L[1], \ldots, L[N]$ is a list of zeros and ones with I inversions, then some sublist $L[k], \ldots, L[k+n-1]$ of length n has at least $\frac{n^2 I}{256 N^2}$ inversions.*

Proof. Let $t = \lfloor n/2 \rfloor \geq 1$ and $q = \lceil N/t \rceil \geq 2$. Then $t \geq n/4$, and $q \leq 2N/t \leq 8N/n$. Make a list of length $N' = qt$ with I inversions by appending $N' - N$ ones to the end of L. Divide this list into q blocks B_i of length t. Let b_i be the number of ones in B_i, and let n_i be the number of inversions within B_i. Then

$$I = \sum_{i<j} b_i(t - b_j) + \sum_{i=1}^{q} n_i,$$

since the first sum counts the number of inversions involving a pair of elements from different blocks. If $n_i \geq I/2q$ for some i, then any sublist of length n containing the part of B_i belonging to the original list L will have at least

$$\frac{I}{2q} \geq \frac{I}{2}\left(\frac{n}{8N}\right) \geq \frac{n^2 I}{256N^2}$$

inversions. Otherwise

$$\sum_{i<j} b_i(t-b_j) = I - \sum_{i=1}^{q} n_i \geq I - q\left(\frac{I}{2q}\right) = \frac{I}{2},$$

so for some $i < j$,

$$b_i(t-b_j) \geq \frac{I/2}{q(q-1)/2} \geq \frac{I}{q^2} \geq \frac{n^2 I}{64N^2}.$$

By the previous lemma applied to $b_i, b_{i+1}, \ldots, b_j$, we get $b_s(t-b_{s+1}) \geq \frac{n^2 I}{256N^2}$ for some s. So any sublist of length n containing the part of $B_s \cup B_{s+1}$ belonging to the original list L will have at least $\frac{n^2 I}{256N^2}$ inversions. □

6. A Geometric Interpretation of the Frobenius Problem

Throughout the next three sections, $a_1, \ldots, a_r$ will be a fixed sequence of integers with $2 \leq a_1 \leq \cdots \leq a_r$ and $\gcd(a_1, \ldots, a_r) = 1$. (So $r \geq 2$.) Define $a = a_1 + \cdots + a_r$. We let $g = g(a_1, \ldots, a_r)$ as in Section 3. We will also be interested in a related function $\psi(x)$, defined as the number of (nonnegative) expressible integers less than or equal to x. In our example $r = 2$, $a_1 = 3$, and $a_2 = 5$ from Section 3, we have $\psi(6) = 4$, for instance.

To get information about $\psi(x)$ and g, we will express $\psi(x)$ as a volume of a certain geometrical figure, and approximate the figure. First we need a few definitions. Label each point $(x_1, \ldots, x_r) \in \mathbb{R}^r$ with the integer $a_1\lfloor x_1 \rfloor + \cdots + a_r\lfloor x_r \rfloor$. Let $\mathcal{L}$ be the set of lattice points with label zero. So $\mathcal{L}$ is a subgroup of $\mathbb{Z}^r$ isomorphic (as an abstract group) to $\mathbb{Z}^{r-1}$. For $x \geq 0$, define regions

$$\begin{aligned}
\mathcal{P} &= \{\, (x_1, \ldots, x_r) \in \mathbb{R}^r \mid x_i \geq 0 \text{ for all } i \,\} \\
\mathcal{S} &= \mathcal{L} + \mathcal{P} \overset{\text{def}}{=} \{\, v + w \mid v \in \mathcal{L}, w \in \mathcal{P} \,\} \\
\mathcal{Q}(x) &= \{\, (x_1, \ldots, x_r) \in \mathbb{R}^r \mid 0 \leq a_1x_1 + \cdots a_rx_r \leq x \,\} \\
\mathcal{S}(x) &= \{\, v \in \mathcal{S} \text{ with label at most } x \,\} \\
\mathcal{T}(x) &= \mathcal{S} \cap \mathcal{Q}(x) = \{\, (x_1, \ldots, x_r) \in \mathcal{S} \mid a_1x_1 + \cdots a_rx_r \leq x \,\}
\end{aligned}$$

Lemma 4. *The region $\mathcal{S}$ is the set of points with expressible label.*

Proof. If $w \in \mathcal{P}$ the label of w is obviously expressible. But for any $v \in \mathcal{L}$, w and $v + w$ have the same label. Thus every point in $\mathcal{S}$ has expressible label. Conversely, if $w = (w_1, \ldots, w_r)$ has an expressible label, equal to $a_1n_1 + \cdots + a_rn_r$, where $n_i \in \mathbb{Z}$, $n_i \geq 0$, then $w \in (v + \mathcal{P})$, where $v = (\lfloor w_1 \rfloor - n_1, \ldots, \lfloor w_r \rfloor - n_r) \in \mathcal{L}$. □

We think of $\mathcal{T}(x)$ as an approximation to $\mathcal{S}(x)$, because of the following lemma. Recall that $a = a_1 + \cdots + a_r$. Think of a as an error term. We will later impose conditions that make a small compared to other relevant quantities.

Lemma 5. *For all $x \geq 0$, $\mathcal{T}(x) \subseteq \mathcal{S}(x) \subseteq \mathcal{T}(x+a)$.*

Proof. The first inclusion follows, because if $a_1x_1 + \cdots + a_rx_r \leq x$, then $a_1\lfloor x_1 \rfloor + \cdots + a_r\lfloor x_r \rfloor \leq x$. The second inclusion follows, because if $a_1\lfloor x_1 \rfloor + \cdots + a_r\lfloor x_r \rfloor \leq x$, then $a_1x_1 + \cdots + a_rx_r \leq x + a$. □

Now for any region $\mathcal{U}$ invariant under translation by elements of $\mathcal{L}$ (such as $\mathcal{S}$, $\mathcal{Q}(x)$, $\mathcal{S}(x)$, or $\mathcal{T}(x)$), we define the *volume of $\mathcal{U}$ per point of $\mathcal{L}$* as the r-dimensional volume of any measurable region $\mathcal{U}_0$ such that $\mathcal{U}$ is the disjoint union of the translates $v + \mathcal{U}_0$ where $v \in \mathcal{L}$. (This is independent of the choice of $\mathcal{U}_0$.)

Lemma 6. *The volume of $\mathcal{Q}(x)$ per point of $\mathcal{L}$ is x.*

Proof. If $\{e_1, \ldots, e_{r-1}\}$ is a basis for $\mathcal{L}$ (as a $\mathbb{Z}$-module), and e_r is any lattice point in $\mathbb{Z}^r$ with label 1, then $\{e_1, \ldots, e_{r-1}, e_r\}$ is a basis for $\mathbb{Z}^r$, because for each $v \in \mathbb{Z}^r$ there is a unique integer k such that $v - ke_r \in \mathcal{L}$, namely the label of v. So the change of coordinates to $\{e_1, \ldots, e_r\}$ has determinant ± 1, and hence preserves volumes. Under this transformation, $\mathcal{L}$ becomes $\{\, (x_1, \ldots, x_{r-1}, 0)\} \in \mathbb{Z}^r \,\}$, and $\mathcal{Q}(x)$ becomes $\{\, (x_1, \ldots, x_r) \in \mathbb{R}^r \mid 0 \leq x_r \leq x \,\}$, so the result is now clear. □

Lemma 7. *The volume of $\mathcal{S}(x)$ per point of $\mathcal{L}$ is $\psi(x)$.*

Proof. For each expressible integer $k \leq x$, pick a cube $[n_1, n_1 + 1) \times \cdots \times [n_r, n_r + 1)$ where $(n_1, \ldots, n_r)$ is a lattice point with label k. Let $\mathcal{U}_0$ be their union. Then $\mathcal{S}(x)$ is the disjoint union of $(v + \mathcal{U}_0)$ for $v \in \mathcal{L}$ (because of Lemma 4), and the volume of $\mathcal{U}_0$ is clearly $\psi(x)$. □

To approximate $\psi(x)$, we define $\Psi(x)$ as the volume of $\mathcal{T}(x)$ per point of $\mathcal{L}$. To check this is well-defined, we must exhibit a measurable region $\mathcal{U}_0$ such that $\mathcal{T}(x)$ is the disjoint union of the translates of $\mathcal{U}_0$. First, we can make $\mathcal{L}$ into an ordered group by identifying it with a subgroup of the additive group of $\mathbb{R}$ isomorphic to $\mathbb{Z}^{r-1}$. Then for $v \in \mathcal{L}$, we can define

$$\mathcal{P}_v = (v + \mathcal{P}) \setminus \bigcup_{\substack{w \in \mathcal{L} \\ w < v}} (w + \mathcal{P}).$$

Each $z \in \mathcal{S}$ is in finitely many $w + \mathcal{P}$ for $w \in \mathcal{L}$, so $\mathcal{S}$ is the disjoint union of the $\mathcal{P}_v$. Also $\mathcal{P}_v = v + \mathcal{P}_0$ for each $v \in \mathcal{L}$. Thus $\mathcal{T}(x)$ is the disjoint union of the translates of $\mathcal{P}_0 \cap \mathcal{Q}(x)$, so $\Psi(x)$ is the volume of $\mathcal{P}_0 \cap \mathcal{Q}(x)$.

The advantage of working with $\Psi(x)$ instead of $\psi(x)$ is that the region $\mathcal{T}(x)$ is much simpler than $\mathcal{S}(x)$. We shall see that whereas $\psi(x)$ is a step function (stepping up 1 at each expressible integer), $\Psi(x)$ is continuous and even differentiable.

We also want to approximate g. By Lemma 4, g is the supremum of the labels of points in $\mathbb{R}^r \setminus \mathcal{S}$. So we define G as the supremum of $a_1x_1 + \cdots + a_rx_r$ for $(x_1, \ldots, x_r) \in \mathbb{R}^r \setminus \mathcal{S}$.

Lemma 8. *$G = g + a$.*

Proof. By Lemma 4, $\mathbb{R}^r \setminus \mathcal{S}$ is the union of the cubes $[n_1, n_1 + 1) \times \cdots \times [n_r, n_r + 1)$ where $(n_1, \ldots, n_r)$ ranges over lattice points with inexpressible label. So we maximize $a_1x_1 + \cdots a_rx_r$ in $\mathbb{R}^r \setminus \mathcal{S}$ by picking a lattice point $(n_1, \ldots, n_r)$ with maximal inexpressible label and looking at the value $a_1x_1 + \cdots + a_rx_r$ at $(n_1 + \epsilon, \ldots, n_r + \epsilon)$ as ϵ approaches 1. Thus we get $G = g + a$. □

7. Properties of $\Psi(x)$ and G

We will derive a large number of results about $\Psi(x)$ and G, and use these to get information about $\psi(x)$ and g. But first we need a lemma.

Lemma 9. *If $r \geq 2$, $(1-1/r)^r \geq 1/4$.*

Proof. For $r \geq 2$ we can apply the arithmetic-geometric mean inequality to get

$$1 - 1/(r+1) = \frac{1 + \overbrace{(1-1/r) + \cdots + (1-1/r)}^{r \text{ terms}}}{r+1} \geq \sqrt[r+1]{1 \cdot \underbrace{(1-1/r)\cdots(1-1/r)}_{r \text{ terms}}}.$$

So $(1-1/(r+1))^{r+1} \geq (1-1/r)^r$ for all $r \geq 2$. Thus

$$(1-1/r)^r \geq (1-1/(r-1))^{r-1} \geq \cdots \geq (1-1/2)^2 = 1/4.$$

□

In the following proposition, the first six properties are proved using geometrical arguments. The rest are simply corollaries of these six.

Proposition 7.

(1) *$\Psi(x)$ is increasing for $x \geq 0$*
(2) *$\Psi(x) \leq \frac{x^r}{r! a_1 a_2 \cdots a_r}$ with equality for sufficiently small $x \geq 0$*
(3) *$\Psi(G) \leq G/2$*
(4) *If $x \geq y \geq 0$, then $\Psi(x) - \Psi(y) \leq x - y$, and equality holds if $x \geq y \geq G$*
(5) *Ψ is differentiable*
(6) *$\Psi'(\alpha x) \geq \alpha^{r-1}\Psi'(x)$ for $x \geq 0, 0 \leq \alpha \leq 1$*
(7) *$|\Psi(x) - \Psi(y)| \geq |x - y|$ for all $x, y \geq 0$.*
(8) *$0 \leq \Psi'(x) \leq 1$ for all $x \geq 0$*
(9) *$\Psi'(x) = 1$ for $x \geq G$*
(10) *$0 \leq \Psi(x) \leq x$ for all $x \geq 0$*
(11) *$x - \Psi(x)$ is increasing for $x \geq 0$*
(12) *$\Psi(x) \geq x\Psi'(x)/r$*
(13) *$\Psi(x)/x^r$ is decreasing for $x > 0$*
(14) *$\Psi(\alpha x) \geq \alpha^r \Psi(x)$ for $x \geq 0, 0 \leq \alpha \leq 1$*
(15) *$\Psi(G) \geq G/r$*
(16) *$G \geq a_1^{1+1/r}$*
(17) *$\Psi(G - \Psi(G)) \geq G/8r$*

Proof. If $x \geq y \geq 0$, then $\mathcal{T}(x) \supseteq \mathcal{T}(y)$, so $\Psi(x) \geq \Psi(y)$. So (1) holds. For all $x \geq 0$, $\mathcal{P}_0 \cap \mathcal{Q}(x)$ is contained in the r-dimensional simplex $\mathcal{P} \cap \mathcal{Q}(x)$, and they are equal for sufficiently small x, by definition of $\mathcal{P}_0$. The lengths of the r edges of this simplex meeting orthogonally at 0 are $x/a_1, \ldots, x/a_r$, so the volume is $(1/r!)(x/a_1)\cdots(x/a_r)$. Thus (2) holds.

Statement (3) is analogous to the result that $\psi(g) \leq (g+1)/2$ (due to Nijenhuis and Wilf [8]), which is proved by noting that if k is expressible, $g - k$ cannot be expressible. The proof is analogous as well, although it is harder to state. By definition of G, we can pick $v = (v_1, \ldots, v_r) \in \mathbb{R}^r \setminus int(\mathcal{S})$ with $a_1 v_1 + \cdots + a_r v_r = G$. The interior $int(\mathcal{S})$ of $\mathcal{S}$ is closed under addition, since $int(\mathcal{S}) = \mathcal{L} + int(\mathcal{P})$, where $\mathcal{L}$ and $int(\mathcal{P})$ are closed under addition.

If the union $int(\mathcal{T}(G)) \cup (v - int(\mathcal{T}(G)))$ inside $\mathcal{Q}(G)$ were not disjoint, then v would be a sum of two vectors in $int(\mathcal{S})$, which contradicts $v \notin int(\mathcal{S})$. Thus this union is disjoint, and by taking the volume per point of $\mathcal{L}$, we deduce from the definition of $\Psi(G)$ and from Lemma 6 that $2\Psi(G) \leq G$. (Note that as far as volumes are concerned, it does not matter that we have $int(\mathcal{T}(G))$ instead of $\mathcal{T}(G)$.) Thus $\Psi(G) \leq G/2$.

If $x \geq y \geq 0$, then $\mathcal{T}(x) \setminus \mathcal{T}(y) \subseteq \mathcal{Q}(x) \setminus \mathcal{Q}(y)$, by their definitions, and equality holds for $x \geq y \geq G$, by definition of G. By taking the volume per point of both sides and applying Lemma 6, we get (4).

It is clear from the geometry that Ψ is differentiable and that $\Psi'(x)$ is proportional to the $(r-1)$-dimensional volume per point of $\mathcal{L}$ of the face of $\mathcal{T}(x)$ contained in the hyperplane $a_1x_1+\ldots+a_rx_r = x$. (This is the part of $\mathcal{T}(x)$ that grows with x.) Since $\mathcal{T}(x)$ is the disjoint union of the translates of $\mathcal{P}_0 \cap \mathcal{Q}(x)$, this can also be expressed as the $(r-1)$-dimensional volume of the face $\mathcal{F}(x)$ of $\mathcal{P}_0 \cap \mathcal{Q}(x)$ contained in the hyperplane $a_1x_1 + \ldots + a_rx_r = x$. If $0 \leq \alpha \leq 1$, and $v \in \mathcal{P}_0 \cap \mathcal{Q}(x)$, then $\alpha v \in \mathcal{P}_0 \cap \mathcal{Q}(\alpha x)$, so $\alpha\mathcal{F}(x) \subseteq \mathcal{F}(\alpha x)$. Taking $(r-1)$-dimensional volumes, we get $\alpha^{r-1}\Psi'(x) \leq \Psi'(\alpha x)$.

Now (7), (8), and (9) follow from (1), (4), and (5). By (2), $\Psi(0) = 0$, so (10) and (11) follow from (8). Integrating (6) with respect to α from 0 to 1 yields $(1/x)(\Psi(x) - \Psi(0)) \geq (1/r)\Psi'(x)$, so (12) holds. Since

$$\frac{d}{dx}\left(\frac{\Psi(x)}{x^r}\right) = \frac{\Psi'(x)}{x^r} - \frac{r\Psi(x)}{x^{r+1}} = -\frac{r}{x^{r+1}}\left(\Psi(x) - \frac{x\Psi'(x)}{r}\right),$$

(13) follows from (12). Then (14) follows from (13). Set $x = G$ in (9) and (12) to deduce (15). By (15) and (2),

$$\frac{G}{r} \leq \Psi(G) \leq \frac{G^r}{r!a_1a_2\cdots a_r} \leq \frac{G^r}{ra_1^r},$$

so $G \geq a_1^{1+1/(r-1)}$. This implies (16).

By (3), $G - \Psi(G) \geq G/2$. So if $\Psi(G - \Psi(G)) \geq (G - \Psi(G))/r$, (17) follows. Otherwise, by (15), the continuous function $\Phi(x) - x/r$ changes sign in $[G - \Psi(G), G]$, so $\Psi(H) = H/r$ for some $H \in [G - \Psi(G), G]$. Then

$$\begin{array}{rcll}
G - \Psi(G) & \geq & H - \Psi(H) & \text{(by (11))} \\
\Psi(G - \Psi(G)) & \geq & \Psi(H - \Psi(H)) & \text{(by (1))} \\
& = & \Psi((1-1/r)H) & \\
& \geq & (1-1/r)^r\Psi(H) & \text{(by (14))} \\
& = & (1-1/r)^r H/r & \\
& \geq & H/4r, &
\end{array}$$

by Lemma 9. But $H \geq G - \Psi(G) \geq G/2$, so $\Psi(G - \Psi(G)) \geq G/8r$. □

Our goal is to produce a result similar to (17) for $\psi(x)$ and g.

Lemma 10. $|\psi(x) - \Psi(x)| \leq a$.

Proof. Taking volumes per point of $\mathcal{L}$ in Lemma 5 gives $\Psi(x) \leq \psi(x) \leq \Psi(x+a)$. But $\Psi(x+a) \leq \Psi(x) + a$, by (4) in Proposition 7, so the result follows. □

Lemma 11. $\psi(g - \psi(g)) \geq G/8r - 4a$.

Proof. First we will apply Lemma 10, (7) from Proposition 7, and Lemma 8 repeatedly to show $\psi(g-\psi(g))$ is close to $\Psi(G-\Psi(G))$. From $|\psi(g)-\Psi(g)| \le a$ and $|\Psi(g)-\Psi(G)| \le |g-G| = a$, we get $|\psi(g)-\Psi(G)| \le 2a$. Similarly from $|\psi(g-\psi(g))-\Psi(g-\psi(g))| \le a$ and

$$\begin{aligned} |\Psi(g-\psi(g))-\Psi(G-\Psi(G))| &\le |(g-\psi(g))-(G-\Psi(G)| \\ &\le |g-G|+|\psi(G)-\Psi(G)| \\ &\le a+2a \\ &= 3a, \end{aligned}$$

we get $|\psi(g-\psi(g))-\Psi(G-\Psi(G))| \le 4a$. Thus

$$\psi(g-\psi(g)) \ge \Psi(G-\Psi(G))-4a \ge G/8r-4a,$$

by (17) in Proposition 7. $\square$

8. Inversions in the Frobenius Pattern

The *Frobenius pattern* is the list $L[0],\dots,L[g]$, where $L[i]$ is 1 if i is expressible, and 0 otherwise. In our example $r=2$, $a_1=3$, and $a_2=5$ from Section 3, the Frobenius pattern is

i	0	1	2	3	4	5	6	7
$L[i]$	1	0	0	1	0	1	1	0

Lemma 12. *The Frobenius pattern is a_1-ordered, ..., a_r-ordered.*

Proof. Suppose $0 \le k < k+a_i \le g$. If $L[k]=1$, then k is expressible, so $k+a_i$ is expressible, and $L[k+a_i]=1$. So $L[k] \le L[k+a_i]$ in any case. $\square$

Lemma 13. *The Frobenius pattern has at least $(\psi(g-\psi(g)))^2$ inversions.*

Proof. The Frobenius pattern has $\psi(g)$ ones and $g+1-\psi(g)$ zeros. There are $\psi(g-\psi(g))$ ones in the first $g+1-\psi(g)$ positions (where the zeros belong), so there must also be $\psi(g-\psi(g))$ zeros in the last $\psi(g)$ positions (where the ones belong). Each of these ones occurs before each of these zeros, so we get $(\psi(g-\psi(g)))^2$ inversions. $\square$

The next lemma says roughly that the fraction of pairs in the Frobenius pattern which are inversions is at least $\Omega(r^{-2})$. To prove it, we need a few hypotheses to guarantee that a will be small compared to G/r.

Lemma 14. *If $a_r \le 2a_1$ and $a_1 \ge r^{9r}$, then the Frobenius pattern has length $N \ge \frac{1}{2}a_1^{1+1/r}$ and has at least $N^2/256r^2$ inversions.*

Proof. First,

$$\begin{aligned} \frac{G}{64r} &\ge \frac{a_1^{1+1/r}}{64r} && \text{(by (16) in Proposition 7)} \\ &\ge \frac{r^9 a_1}{64r} \\ &\ge 2ra_1 && \text{(since } r \ge 2\text{)} \\ &= \overbrace{2a_1+\cdots+2a_1}^{r \text{ terms}} \\ &\ge a_1+\cdots a_r, \end{aligned}$$

so $a \leq G/64r$. Thus

$$\begin{aligned} N &= (G-a)+1 \quad \text{(by Lemma 8)} \\ &\geq G/2 \\ &\geq \frac{1}{2} a_1^{1+1/r} \end{aligned}$$

by (16) in Proposition 7. Also, by Lemma 11,

$$\psi(g-\psi(g)) \geq G/8r - 4a \geq G/8r - 4(G/64r) = G/16r \geq N/16r,$$

since $G \geq G-a+1 = N$. By Lemma 13, the Frobenius pattern contains at least $(N/16r)^2 = N^2/256r^2$ inversions. □

9. Making Lists with Lots of Inversions

In this section, we show how to use Frobenius patterns to build other lists which are h-ordered for various h's but which still have many inversions. Our first lemma is similar in spirit to Exercise 16 in Section 5.2.1 of [6].

Lemma 15. *If $N, h \geq 2$, there exists an h-ordered list of N zeros and ones having at least $N^2/256$ inversions.*

Proof. If $N < h$ then the list of length N in reverse order is h-ordered and has $N(N-1)/2 \geq N^2/4$ inversions. Otherwise let $q = \lfloor N/h \rfloor$, so $q \geq N/2h$. For $0 \leq i \leq h-1$, let

$$L[i] = \begin{cases} 1 & \text{if } (i \bmod h) < h/2, \\ 0 & \text{otherwise.} \end{cases}$$

For $1 \leq j \leq k \leq q$, we get $\lceil h/2 \rceil \lfloor h/2 \rfloor$ inversions by pairing the $\lceil h/2 \rceil$ ones in the j^{th} block of h elements with the $\lfloor h/2 \rfloor$ zeros in the k^{th} block of h elements, so there are at least $\frac{1}{2}q(q+1)\lceil h/2 \rceil \lfloor h/2 \rfloor$ inversions in L. But $\frac{1}{2}q(q+1) \geq q^2/2 \geq N^2/8h^2$ and $\lceil h/2 \rceil \lfloor h/2 \rfloor \geq (h/2)(h/4)$, so this is at least $N^2/256$ inversions. □

Lemma 16. *If $a_1, \ldots, a_r$ are arbitrary integers satisfying $2 \leq a_1 \leq \cdots \leq a_r$ and $a_1 \geq r^{9r}$, then there exists $N \geq \frac{1}{2}a_1^{1+1/r}$ and a list of N zeros and ones which is a_1-ordered,…,a_r-ordered, and which has at least $N^2/256r^2$ inversions.*

Proof. We can assume $a_r < 2a_1$, because otherwise we can replace each a_i with $b_i = a_1 + (a_i \bmod a_1)$ (and then order the b_i's). A list which is b_i-ordered for all i will be a_i-ordered for all i, since $a_1 = b_1$ and each other a_i is b_i plus some nonnegative multiple of a_1.

Let $h = \gcd(a_1, \ldots, a_r)$. If $h = 1$, the result follows from Lemma 14. If $h \geq 2$, then for any $N \geq 2$ there is an h-ordered list with at least $N^2/256$ inversions, by Lemma 15. This list is a_1-ordered,…,a_r-ordered, so we are done. □

Lemma 17. *If $2 \leq a_1 \leq \cdots \leq a_r$, $a_1 \geq r^{9r}$, and $2 \leq l \leq \frac{1}{2}a_1^{1+1/r}$, there exists a list of l zeros and ones which is a_1-ordered,…,a_r-ordered, and which has at least $2^{-16}l^2/r^2$ inversions.*

Proof. By the previous lemma, for some $N \geq l$ there is a list of N zeros and ones which is a_1-ordered,…,a_r-ordered, and which has at least $N^2/256r^2$ inversions. By Proposition 6, we can take a sublist of length l having at least $(l^2/256N^2)(N^2/256r^2) = 2^{-16}l^2/r^2$ inversions, and this sublist is also a_1-ordered,…,a_r-ordered. □

Everything so far has been building up to the following lemma, which is the only result from the past four sections which we use later.

Lemma 18. *If $2 \le a_1 \le \cdots \le a_r$, $a_1 \ge r^{9r}$, $2 \le l \le \frac{1}{2}a_1^{1+1/r}$, and $N \ge l$, then there exists a list of length N which is a_1-ordered,…,a_r-ordered, and h-ordered for all $h \ge l$, and which has at least $2^{-17}Nl/r^2$ inversions.*

Proof. Let $L[0], L[1], \ldots, L[l-1]$ be the list of l zeros and ones given by the previous lemma. For $0 \le i \le N-1$, define $L'[i] = 10\lfloor i/l \rfloor + L[i \bmod l]$. Then L' contains $\lfloor N/l \rfloor \ge N/2l$ copies of L (with entries displaced by various multiples of 10), so L' contains at least $(N/2l)(2^{-16}l^2/r^2) = 2^{-17}Nl/r^2$ inversions. Also L' is clearly a_1-ordered,…,a_r-ordered, and h-ordered for all $h \ge l$. □

10. Lower Bounds on the Worst Case

We are now ready to prove a lower bound corresponding to the upper bound of Theorem 2. The situation is as follows: we are sorting a list of length N using a Shellsort-type algorithm. Let $c_2 = 1/432$, and let m be the number of increments less than $N/2$.

Theorem 3. *If N is sufficiently large, there is a list of length N on which the algorithm must perform at least $N^{1+c_2/\sqrt{m}}$ compare-exchange operations.*

Before proving the theorem, let us remark that if we defined m as the total number of increments, we would be proving a weaker result. So the theorem would still hold. But we will need the stronger version when we prove our next theorem.

Proof. If $N^{c_2/\sqrt{m}} < \sqrt{\log N}$, then

$$\begin{aligned} N^{1+c_2/\sqrt{m}} &< N\sqrt{\log N} \\ &\le \log N!, \end{aligned}$$

for sufficiently large N. We would then be done, since this is the information-theoretic lower bound on the number of comparisons required. So we may assume

$$\begin{aligned} N^{c_2/\sqrt{m}} &\ge \sqrt{\log N} && (1) \\ \frac{c_2}{\sqrt{m}}\log N &\ge \frac{1}{2}\log\log N \\ m &\le \left(\frac{\log N}{\log\log N}\right)^2, && (2) \end{aligned}$$

since $c_2 < 1/2$.

For $k \ge 0$, define $\beta_k = N^{\frac{1}{2}+\frac{k}{72\sqrt{m}}}$. If $k \in \mathbb{Z}$ and $0 \le k \le \lceil 18\sqrt{m} \rceil$, then

$$k \le 18\sqrt{m} + 1 \le 24\sqrt{m}$$

$$\beta_k \le N^{\frac{1}{2}+\frac{24\sqrt{m}}{72\sqrt{m}}} = N^{5/6} \le N/2,$$

provided N is sufficiently large. So we have at least $18\sqrt{m}$ disjoint intervals $[\beta_i, \beta_{i+1})$ with $\beta_{i+1} \le N/2$, which together contain at most m increments. Thus some interval $[\beta_i, \beta_{i+1})$ contains r increments $a_1 \le \cdots \le a_r$, where

$$r \le \frac{m}{18\sqrt{m}} = \frac{\sqrt{m}}{18}. \qquad (3)$$

By (2),

$$r \leq \frac{1}{18}\left(\frac{\log N}{\log\log N}\right) \tag{4}$$

$$r \leq \log N, \tag{5}$$

for sufficiently large N.

The next two paragraphs are devoted to showing that the conditions of Lemma 18 hold. We have

$$a_1 \geq \beta_i \geq \sqrt{N}, \tag{6}$$

so

$$\begin{aligned} \log a_1 &\geq \frac{1}{2}\log N \\ &= 9\left[\frac{1}{18}\left(\frac{\log N}{\log\log N}\right)\right](\log\log N) \\ &\geq 9r\log r, \end{aligned}$$

by (4) and (5). Thus $a_1 \geq r^{9r}$.

Let $l = \lceil \beta_{i+1} \rceil$. Then $2 \leq l \leq N$, and

$$\begin{aligned} l &\leq 2\beta_{i+1} \\ &= 2N^{\frac{1}{72\sqrt{m}}}\beta_i \\ &\leq \left(\frac{1}{2}\sqrt{\log N}\right)N^{\frac{1}{72\sqrt{m}}}\beta_i && \text{(for sufficiently large } N) \\ &\leq \frac{1}{2}N^{c_2/\sqrt{m}}N^{\frac{1}{72\sqrt{m}}}\beta_i && \text{(by (1))} \\ &\leq \frac{1}{2}N^{9/\sqrt{m}}\beta_i && \text{(since } c_2 + 1/72 \leq 9) \\ &\leq \frac{1}{2}a_1^{18/\sqrt{m}}a_1 && \text{(by (6))} \\ &\leq \frac{1}{2}a_1^{1+1/r} && \text{(by (3)).} \end{aligned}$$

So we may apply Lemma 18 to obtain a list L of length N which is a_1-ordered, $\ldots$, a_r-ordered, and h-ordered for all $h \geq l$, and which has at least $2^{-17}Nl/r^2$ inversions. In particular, L is h-ordered for each increment $h \geq \beta_i$. By Lemma 1, no exchanges are done during the passes corresponding to these large increments, so the burden of removing the inversions is laid upon the passes with increments less than β_i. By Proposition 5, each exchange during those passes reduces the number of inversions by less than $2\beta_i$. To reduce this number to zero (to sort the list), the number of exchanges needed is at least

$$\begin{aligned}
\frac{1}{2\beta_i}\left(\frac{Nl}{2^{17}r^2}\right) &\geq \frac{N\beta_{i+1}}{2^{18}\beta_i(\log N)^2} && \text{(by (5))}\\
&= \frac{N \cdot N^{\frac{1}{72\sqrt{m}}}}{2^{18}(\log N)^2}\\
&= \frac{N^{1+c_2/\sqrt{m}}N^{5c_2/\sqrt{m}}}{2^{18}(\log N)^2} && \text{(since } 6c_2 = 1/72\text{)}\\
&\geq \frac{N^{1+c_2/\sqrt{m}}(\sqrt{\log N})^5}{2^{18}(\log N)^2} && \text{(by (1))}\\
&\geq N^{1+c_2/\sqrt{m}},
\end{aligned}$$

for sufficiently large N. □

The next theorem follows quite easily.

Theorem 4. *Any Shellsort-type algorithm makes* $\Omega(N(\log N/\log\log N)^2)$ *comparisons in the worst case.*

Proof. As in Theorem 3, let m be the number of increments less than $N/2$. If

$$m > \frac{c_2^2}{4}\left(\frac{\log N}{\log\log N}\right)^2,$$

then we are done, since each pass with increment $h < N/2$ makes $\Omega(N)$ comparisons. Otherwise

$$\begin{aligned}
\sqrt{m} &\leq \frac{c_2}{2}\left(\frac{\log N}{\log\log N}\right)\\
\frac{c_2}{\sqrt{m}}\log N &\geq 2\log\log N\\
N^{c_2/\sqrt{m}} &\geq (\log N)^2\\
N^{1+c_2/\sqrt{m}} &\geq N(\log N)^2,
\end{aligned}$$

so by Theorem 3, at least $N(\log N)^2$ comparisons will be made in the worst case, for sufficiently large N. □

For the particular case of Shaker-sort, we can prove an even stronger result for some increment sequences.

Theorem 5. *If* $h_j = \Theta(\alpha^j)$ *for some fixed real* $\alpha > 1$*, and the increments* h_j *less than* N *are used in any order in Shaker-sort, then* $\Omega(N^2)$ *steps are required in the worst case.*

Proof. Let h_m be the largest increment used and let $a_i = h_{m-r+i}, 1 \leq i \leq r$, be the r largest increments in increasing order. (We'll choose r later.) For fixed r, $a_1 = \Theta(N/\alpha^r) = \Theta(N)$, so $N \leq \frac{1}{2}a_1^{1+1/r}$ for large N. Thus by Lemma 18 with $l = N$, there exists a list L of length N which is a_1-ordered, …, a_r-ordered and which has at least $2^{-17}N^2/r^2$ inversions.

An h-shake involves at most $2N$ compare-exchanges at a distance of h, so by Proposition 5, the increments $h_1, \dots, h_{m-r}$ can remove at most

$$\begin{aligned} 4N(h_1 + h_2 + \cdots + h_{m-r}) &= \Theta(N(\alpha + \alpha^2 + \cdots + \alpha^{m-r})) \\ &= \Theta(N\alpha^{m-r}) \\ &= \Theta(\alpha^{-r}N^2) \end{aligned}$$

inversions. Pick r large enough that this is less than or equal to $2^{-18}N^2/r^2$. Then when Shaker-sort is applied to L, the increments $a_1, \dots, a_r$ do no work (by Lemma 1), so after all the h-shakes, at least

$$2^{-17}N^2/r^2 - 2^{-18}N^2/r^2 = 2^{-18}N^2/r^2$$

inversions remain to removed by the insertion sort at the end, and this will take at least $2^{-18}N^2/r^2$ steps. Since r is independent of N, this is $\Omega(N^2)$. $\square$

11. Unsolved Problems

We now mention some things we would have liked to prove. It would be nice to have a tight bound on the worst case performance. There is still a small gap between our upper and lower bounds, when the number of increments exceeds $(\log N/\log\log N)^2$, and when the number of increments is unrestricted. We know that the complexity of Shellsort, for instance is $\Omega(N(\log N/\log\log N)^2)$ and $O(N(\log N)^2)$. What is the actual complexity? Probably the answer is $\Theta(N(\log N)^2)$, so Pratt's increments are the best asymptotically, but it seems impossible to obtain this lower bound simply by refining the techniques used above. Making more careful estimates seems only to improve the constants.

An even more interesting problem, which we have not considered at all in this paper, is the determination of the *average* case behavior of Shellsort or Shaker-sort. This is irrelevant when constructing sorting networks, but it is important when using either method as a general-purpose sorting algorithm. Virtually nothing is known about the average case. The only increment sequences for which the average running time of Shellsort has been calculated are $(h, k, 1)$ (see [17]), and sequences in which each increment is a multiple of the next (see the Corollary to Theorem H in Section 5.2.1 of [6]). None of our results rule out an $O(N \log N)$ average case Shellsort or Shaker-sort, although perhaps they make it a little less likely.

References

[1] Brauer, A., On a Problem of Partitions, *Amer. J. Math.* **64** (1942), 299–312.

[2] Cypher, R., A Lower Bound on the Size of Shellsort Sorting Networks, *Proceedings of the 1989 International Conference on Parallel Processing*, 58–63.

[3] Hibbard, T., An Empirical Study of Minimal Storage Sorting, *Comm. of the ACM* **6**, no. 5 (May 1963), 206–213.

[4] Incerpi, J. and R. Sedgewick, Improved Upper Bounds on Shellsort, *J. Comput. System. Sci.* **31**, no. 2 (1985), 210–224.

[5] Incerpi, J. and R. Sedgewick, Practical Variations on Shellsort, *Inform. Process. Lett.* **26** (1987), 37–43.

[6] Knuth, D., "The Art of Computer Programming. 3. Sorting and Searching", Addison-Wesley, Reading, Mass., 1973.

[7] Lazarus, R. and R. Frank, A High-Speed Sorting Procedure, *Comm. of the ACM* **3**, no. 1 (1960), 20–22.

[8] Nijenhuis, A. and H. Wilf, Representations of Integers by Linear Forms in Nonnegative Integers, *J. Number Theory* **4** (1972), 98–106.

[9] Papernov, A. and G. Stasevich, A Method of Information Sorting in Computer Memories, *Problems of Inform. Transmission* **1**, no. 3 (1965), 63–75.
[10] Pratt, V., "Shellsort and Sorting Networks", Garland Publishing, New York, 1979. (Originally presented as the author's Ph. D. thesis, Stanford University, 1971.)
[11] Sedgewick, R., A New Upper Bound for Shellsort, *J. of Algorithms* **2** (1986), 159–173.
[12] Selmer, E., On the Linear Diophantine Problem of Frobenius, *J. reine angew. Math.* **294** (1977), 1–17.
[13] Sharp, W. J. Curran, Solution to Problem 7382 (Mathematics) proposed by J. J. Sylvester, *Ed. Times* **41** (1884), London.
[14] Shell, D., A High-Speed Sorting Procedure, *Comm. of the ACM* **2**, no. 7 (1959), 30–32.
[15] Weiss, M. and R. Sedgewick, Bad Cases for Shaker Sort, *Inform. Process. Lett.* **28** (1988), 133–136.
[16] Weiss, M. and R. Sedgewick, Tight Lower Bounds for Shellsort, *J. of Algorithms* **11** (1990), 242–251.
[17] Yao, A., An Analysis of $(h,k,1)$-Shellsort, *J. of Algorithms* **1** (1980), 14–50.

Department of Mathematics, University of California at Berkeley, Berkeley, CA 94720
E-mail address: poonen@math.berkeley.edu