JOURNAL OF COMPUTER AND SYSTEM SCIENCES 31, 210–224 (1985)

Improved Upper Bounds on Shellsort*

JANET INCERPI†

Department of Computer Science, Brown University, Providence, Rhode Island

AND

ROBERT SEDGEWICK‡

INRIA, 78150 Rocquencourt, France

Received April 17, 1984; revised November 30, 1984

The running time of Shellsort, with the number of passes restricted to $O(\log N)$, was thought for some time to be $\Theta(N^{3/2})$, due to general results of Pratt. Sedgewick recently gave an $O(N^{4/3})$ bound, but extensions of his method to provide better bounds seem to require new results on a classical problem in number theory. In this paper, we use a different approach to achieve $O(N^{1+\varepsilon/\sqrt{\lg N}})$, for any $\varepsilon > 0$.

INTRODUCTION

Shellsort is a fundamental, but little-understood, sorting algorithm. A brief description of the algorithm is given below. It is based on a table $h_1, h_2, \ldots$, of integers called an *increment sequence*. In practice, increment sequences are chosen heuristically based on partial analytic results which have been derived for some specific increment sequences. This algorithm is an attractive candidate for detailed study because it is closely related to classical problems in number theory and because theoretical results translate directly to practice. (A practioner can make immediate use of a good increment sequence, no matter how intricate the analysis.) It is difficult to deny the existence of increment sequences that would make Shellsort the sorting method of choice, for most situations. Moreover, relatively few types of increment sequences have been tried. Some references for Shellsort and some of the analysis that has been done are [6, 9, 10, and 11]; some of this information is summarized below.

The Shellsort algorithm works as follows: given an increment sequence $h_1, h_2, \ldots$, a file is sorted by successively *h_j-sorting* it, for j from some integer t down to 1. An

* This research was supported in part by NSF Grant MCS-83-08806 and in part by the Office of Naval Research and DARPA under Contract N00014-83-K-0146 and ARPA Order 4786.

† Current address: INRIA, Sophia Antipolis, 06560 Valbonne, France.

‡ Current address: Dept. of Computer Science, Princeton University, Princeton, N.J. 08544.

0022-0000/85 $3.00

array $a[1],..., a[N]$ is defined to be h_j-sorted if $a[i-h_j] \leqslant a[i]$ for i from h_j+1 to N. The method used for h_j-sorting is *insertion sort*: for i from h_j+1 to N, we sort the sequence ..., $a[i-2h_j]$, $a[i-h_j]$, $a[i]$ by taking advantage of the fact that the sequence ..., $a[i-2h_j]$, $a[i-h_j]$ is already sorted, so $a[i]$ can be inserted by moving larger elements one position to the right in the sequence, then putting $a[i]$ in the place vacated.

A fundamental property of this process is that, if we h-sort a file which is k-sorted, then the file remains k-sorted. Thus, when we come to h_j-sort the file during Shellsort, we know that it is h_{j+1}-, h_{j+2}-,..., h_t-sorted. This ordering makes the h_j-sort less expensive than if we were to h_j-sort a randomly ordered file.

Shellsort sorts properly whenever the increment sequence ends with $h_1 = 1$, but the running time of the algorithm clearly is quite dependent on the specific increment sequence used. Unfortunately, we have little guidance on how to pick the "best" increment sequences. All the results that we have relate to specific sequences (from a quite larger universe) and leave open the possibility of an undiscovered increment sequence with far better performance characteristics than those that have been tried to date.

From a practical standpoint, Shellsort leads to a simple and compact sorting program which works well for small files and for files which are already partially ordered. It is the practical method of choice for files with less than several hundred elements, and each new increment sequence that we discover raises this bound. Empirical tests by several researchers indicate that there might exist increment sequences for which the average running time is $O(N \log N)$ (e.g., see [4]).

From a theoretical standpoint, the study of increment sequences for Shellsort is important because of the potential for a simple constructive proof of the existence of an $O(N \log N)$ sorting network. (An increment sequence of length $O(\log N)$ for which each insertion requires a constant number of steps would imply this.) This was an open problem in the theory of sorting for some time; the existence of such a network was recently presented by Ajtai, Komlos, and Szemerdi [1] but their construction is hardly practical. (Further refinements have been made by Leighton [7], but his networks are still far more complex than a Shellsort-based network would be.) These results make the search for a short proof based on Shellsort even more appealing. Weaker results (e.g., an $O(N \log N)$ average case) are also worth pursuing because of the practical implications.

In this paper, we are interested in worst-case bounds for the total running time of Shellsort for practicular increment sequences. Specifically, we are most interested in increment sequences of length $O(\log N)$; this would be required for an optimal sorting network, and such sequences are the most viable from a practical standpoint. Even with this restriction, the space of possible increment sequences is quite large. For simplicity, in this paper we assume that the sequence increases (although there is no particular requirement for this). Further, we make the following distinction:

DEFINITION. A Shellsort implementation is said to be *uniform* if the increments

used to sort N items are all the numbers less than N (taken in decreasing order) from a fixed infinite increasing sequence $h_1, h_2,\ldots$.

A non-uniform Shellsort might use a different increment sequence for each file size. Both types are used in practice, though uniform implementations have been studied more heavily. For example, Knuth [6] recommends using a uniform implementation based on the sequence 1, 4, 13, 40,..., $\frac{1}{2}(3^k-1)$,.... On the other hand, in order to use a uniform sequence one must calculate an appropriate starting place and/or save the sequence, so some practitioners find it more convenient to use non-uniform sequences such as $\lfloor N/2 \rfloor$, $\lfloor \lfloor N/2 \rfloor /2 \rfloor$, etc. Unless designed with care, non-uniform sequences are susceptible to bad worst-case performance for some file sizes. Consequently, uniform implementations are more widely used and studied. We use the terminology "uniform $f(N)$-sequence" to refer to an infinite sequence for which the number of integers less than N is $f(N)$.

Shellsort and the Frobenius Problem

To prove upper bounds on the number of steps required for Shellsort, we are interested specifically in the following function:

Definition. $n_d(a_1, a_2,\ldots,a_k) \equiv$ the number of multiples of d which cannot be represented as linear combinations (with non-negative integer coefficients) of $a_1, a_2,\ldots, a_k$.

We assume that $a_1, a_2,\ldots, a_k$ are >1 (otherwise all integers could be represented) and that $a_1, a_2,\ldots, a_k$ are *independent*: that none can be represented as a linear combination with non-negative integer coefficients of the others (otherwise it could be deleted from the list without affecting the result). More important, for $n_d(a_1, a_2,\ldots, a_k)$ to be defined, it must be the case that $a_1, a_2,\ldots, a_k$ do not have a common factor which is not shared by d (otherwise, only those multiples of d which share that common factor could be represented as linear combinations of $a_1, a_2,\ldots, a_k$, and there are an infinite number of multiples of d which do not).

This function is related to Shellsort by the following lemma:

Lemma 1. *The number of steps required to h_j-sort a file which is h_{j+1}-, h_{j+2}-,..., h_t-sorted is*

$$O(Nn_{h_j}(h_{j+1}, h_{j+2},\ldots, h_t)).$$

Proof. The number of steps required to insert element $a[i]$ is the number of elements among $a[i-h_j]$, $a[i-2h_j]$,..., which are greater than $a[i]$. Any element $a[i-x]$ with x a linear combination of $h_{j+1},\ldots, h_t$ must be less than $a[i]$ since the file is h_{j+1}-, h_{j+2}-,..., h_t-sorted. Thus, an upper bound on the number of steps to insert $a[i]$, for $1 \leqslant i \leqslant N$, is the number of multiples of h_j which are not expressible as linear combinations of $h_{j+1}, h_{j+2},\ldots, h_t$ or $n_{h_j}(h_{j+1}, h_{j+2},\ldots, h_t)$. ∎

When $d=1$, we have $n_1(a_1, a_2,..., a_k)$ (or just $n(a_1, a_2,..., a_k)$) which is the number of positive integers which cannot be represented as linear combinations with nonnegative coefficients of $a_1, a_2,..., a_k$. A closely related function is $g(a_1, a_2,..., a_k)$, the largest integer which cannot be so represented. These functions are well-studied in number theory [5, 11, 12]: to find $g(a_1,..., a_k)$ is the so-called *Frobenius problem.*

The function which arises in Shellsort is related to the standard Frobenius function by the following lemma:

LEMMA 2. *For $a_1, a_2,..., a_k$ relatively prime,*

$$n_d(a_1, a_2,..., a_k) < \frac{g(a_1, a_2,..., a_k)}{d}.$$

Proof. (Note that $g(a_1, a_2,..., a_k)$ is undefined unless $a_1, a_2,..., a_k$ are relatively prime.) Every number greater than $g(a_1, a_2,..., a_k)$ can be represented as a linear combination of $a_1, a_2,..., a_k$; in the worst case all multiples of d less than $g(a_1, a_2,..., a_k)$ cannot. ∎

Previous upper bound proofs for Shellsort have used a combined version of these lemmas:

LEMMA 3. *The number of steps required to h_j-sort a file which is h_{j+1}-, h_{j+2}-,..., h_t-sorted is*

$$O(N\, g(h_{j+1}, h_{j+2},..., h_t)/h_j).$$

Proof. Immediate from Lemmas 1 and 2. ∎

Specific bounds are obtained by solving the Frobenius problem for specific increment sequences. For $k=2$, we have the original Frobenius problem whose solution dates at least to 1884:

LEMMA 4. *If a_1 and a_2 are relatively prime, then $g(a_1, a_2) = (a_1 - 1)(a_2 - 1)$.*

Proof. See Knuth [6, Ex. 5.2.1–21, or 2]. ∎

For example, this leads directly to an upper bound for h_j-sorting of $O(N h_j)$ when $h_j = 2^j + 1$ since

$$\begin{aligned} N\frac{g(h_{j+1},..., h_t)}{h_j} &\leqslant N\frac{g(h_{j+1}, h_{j+2})}{h_j} \\ &= O\left(N\frac{h_{j+1}h_{j+2}}{h_j}\right) \\ &= O(N h_j). \end{aligned}$$

This is the bound of Papernov and Stasevich [8], which was generalized by Pratt [9] to cover a large family of "almost geometric" increment sequences.

Upper bounds on h_j-sorting for sequences with geometric growth translate to upper bounds on the total number of steps required by Shellsort as follows:

LEMMA 5. *Suppose that an increment sequence $h_1, h_2, \ldots,$ is used to Shellsort a file of size N, with $h_j = \Theta(\alpha^j)$ for some constant α. If the number of steps for h_j-sorting is $O(N h_j^{1/c})$. then the total running time for Shellsort is $O(N^{1+1/(c+1)})$.*

Proof. The increments used are $h_1, \ldots, h_t$, where t is the largest integer such that h_t is less than N. We use the bound $O(N^2/h_j)$ for large h_j (this comes from considering h_j independent subfiles of size N/h_j, each of which could require $O((N/h_j)^2)$ steps) and the bound $O(N h_j^{1/c})$ for small h_j, switching at $h_j = \Theta(N^{c/(c+1)})$, when both bounds are $O(N^{1+1/(c+1)})$. The total number of steps for Shellsort is

$$\sum_{1 \leqslant j \leqslant t_o} O(N h_j^{1/c}) + \sum_{t_o \leqslant j \leqslant t} O\left(\frac{N^2}{h_j}\right),$$

where t_o is such that $h_{t_o} = \Theta(N^{c/(c+1)})$. Both sums are geometric and are bounded by their largest term $O(N^{1+1/(c+1)})$ in both cases. ∎

For example, Lemma 5, with $c = 1$, gives the $O(N^{3/2})$ upper bound for Shellsort of Papernov and Stasevich [8] and Pratt [9]. In fact, Pratt showed this bound to be tight for a large family of increment sequences (encompassing most of those that have been proposed), where the increments are within an additive constant of a geometric progression.

Sedgewick [11] used general results of Selmer [12] and Johnson [5] for the Frobenius problem for $k = 3$ to develop increment sequences that grow geometrically and the upper bound for h_j-sorting is $O(N h_j^{1/2})$. This leads to an $O(N^{4/3})$ upper bound for Shellsort (Lemma 5 with $c = 2$). (These sequences are of the form $\alpha 4^j + \beta 2^j + \upsilon$, not within an additive constant of a geometric progression.) Unfortunately, there are few available results on the Frobenius problem for $k > 3$, and such results would seem to be required to get better upper bounds using this approach.

Furthermore, we can show that a bound of the type $O(N^{1+1/(c+1)})$ is the best that can be achieved with this approach because we have a lower bound on the Frobenius function:

LEMMA 6. *For $a_1, a_2, \ldots, a_k$ increasing,*

$$n(a_1, a_2, \ldots, a_k) = \Omega(a_1^{1+1/(k-1)}).$$

Proof. Define $L(m) = \{x | x = c_1 a_1 + \cdots + c_k a_k$ with $c_1, c_2, \ldots, c_k \geqslant 0$ and $c_1 + \cdots + c_k = m\}$. Then if $x \in L(m)$ we know that $x \geqslant m a_1$. The cardinality of $L(m)$

is at most the number of ways to choose $c_1,..., c_k$ satisfying $c_1 + \cdots + c_k = m$. This is precisely the number of different outcomes possible if you have an urn with k different colors balls and you select m balls with replacement. There are $\binom{m+k-1}{m}$ possible outcomes. Thus, $|L(m)| \leqslant \binom{m+k-1}{k-1}$.

Now, for any constant $m_0 \geqslant 1$, we know that the number of integers which cannot be represented as a linear combination of $a_1, a_2,..., a_k$ is greater than or equal to the number of integers which are less than $(m_0+1)\,a_1$ minus the number of integers we know can be represented as a linear combination of $a_1, a_2,..., a_k$. The number of such integers is certainly less than $\sum_{1 \leqslant m \leqslant m_0} |L(m)|$, so

$$n(a_1, a_2,..., a_k) \geqslant (m_0+1)\,a_1 - \sum_{1 \leqslant m \leqslant m_0} |L(m)|$$

$$\geqslant (m_0+1)\,a_1 - \binom{m_0+k}{k}.$$

Let $f(m_0) = (m_0+1)\,a_1 - \binom{m_0+k}{k}$. Then differencing and setting the result equal to zero, we have

$$f(m_0+1) - f(m_0) = a_1 - \binom{m_0+1+k}{k} + \binom{m_0+k}{k}$$

$$= a_1 - \binom{m_0+k}{k-1}.$$

This function is maximized at m_0 such that $a_1 \sim \binom{m_0+k}{k-1} \sim {m_0}^{k-1}/(k-1)!$. Thus m_0 is approximately ${a_1}^{1/(k-1)}$. When this occurs we have

$$n(a_1, a_2,..., a_k) = \Omega(a_1^{1+1/(k-1)}). \quad \blacksquare$$

If we only consider the effects of $c+1$ increments $h_{j+1},..., h_{j+c+1}$ when h_j-sorting and we use the approach of Lemma 3 (the standard approach), then Lemma 6 says that the best bound that we can hope for for h_j-sorting is $O(h_j^{1/c})$, which translates to an $O(N^{1+1/(c+1)})$ Shellsort bound by Lemma 5. Thus, the bounds of Papernov and Stasevich ($c=1$) and Sedgewick ($c=2$) are best possible in this sense. Below, we show how to achieve $O(N^{1+1/(c+1)})$ with $(c \log N)$-uniform sequences for any c, though we do so by circumventing the standard approach of Lemma 3 and using Lemma 1 directly, not by developing new results on the Frobenius problem. Furthermore, we show how this method extends to provide even better bounds. We do so by turning attention to increments which have large common divisors, then by exploiting specific properties of the generalized Frobenius function with two arguments, $n_d(r, s)$.

Generalized Frobenius Problem

It is possible to completely characterize the generalized Frobenius function with two arguments. We have:

THEOREM 1. *For any positive integers r, s, and d with* $\gcd(d, r, s) = \gcd(r, s)$:

$$n_d(r, s) = n_{d \gcd(d, r, s)/\gcd(d, r) \gcd(d, s)} \left(\frac{r}{\gcd(d, r)}, \frac{s}{\gcd(d, s)} \right).$$

If d, r, and s are pairwise relatively prime, then

$$n_d(r, s) = n\left(r, s, \frac{r + b_1 s}{d}, \ldots, \frac{(d-1)\, r + b_{d-1} s}{d}\right)$$

where b_i is the unique integer between 1 *and* $d-1$ *such that* $ir + b_i s \equiv 0 \pmod{d}$. *Note that $n_d(r, s)$ is undefined if* $\gcd(d, r, s) \neq \gcd(r, s)$.

Proof. (See Appendix.)

Although this characterization is not needed in its full generality for the results in this paper, it does indicate that divisibility properties among the increments can be instrumental in lowering bounds on the generalized Frobenius function. For the constructions of the next section, we actually use a very special case of this theorem which can be proved directly from the definition.

COROLLARY 1. *For integer* $z > 1$, $n_{dz}(rz, sz) = n_d(r, s)$.

This property holds for more than two arguments: we have $n_{dz}(a_1 z, a_2 z, \ldots, a_k z) = n_d(a_1, a_2, \ldots, a_k)$, but a full characterization such as Theorem 1 for more than two arguments seems complicated.

Applying Lemmas 2 and 4 with the corollary to Theorem 1, we have $n_{dz}(rz, sz) = n_d(r, s) < rs/d$ (if r and s are relatively prime), which is less by a factor of z than the bound $rzsz/dz$ which derives from direct application of Lemmas 2 and 4 (although Lemma 4 could not be applied since rz and sz are not relatively prime).

Increment Sequences

Our increment sequences represent a compromise between two classical increment sequences that have been proposed for Shellsort. The first, proposed by Shell, is the geometric sequence 1, 2, 4, 8, 16,.... The problem with this sequence is that the generalized Frobenius function is always undefined, since even after the application of Theorem 1, $h_{j+1}, h_{j+2}, \ldots$, have a common factor (2) which is not shared by h_j. The practical effect of this is that the worst case is $\Theta(N^2)$, for example, for a shuffled file with the $N/2$ smallest elements in the odd positions and the $N/2$

largest elements in the even positions. Because of this effect, Shellsort increment sequences are normally designed to have successive increments relatively prime.

A notable exception is the sequence 1, 2, 3, 4, 6, 9,..., given by Pratt, which is defined by appending $2z$ and $3z$ to the sequence for every element z in the sequence. Thus, by the corollary to Theorem 1, the running time for each increment is $O(N n_1(2, 3))$. Unfortunately, there are $\Theta(\log^2 N)$ increments less than N, and even after applying tradeoffs as in Lemma 5, the running time is always $\Theta(N \log^2 N)$. Increment sequences with $O(\log N)$ increments are of more interest because in principle, the running time for such sequences could be $O(N \log N)$ on the average (or even in the worst case), and in practice, the large number of passes required for Pratt's sequence makes it slower than typical $O(\log N)$-pass Shellsorts.

Thus, our goal is to design a geometrically increasing sequence in which successive increments have both large common factors and small relatively prime factors. Our method for doing so is to build up increments by multiplying together selected terms of a "base" sequence $a_1, a_2, \ldots$.

Given a constant c, we associate c increments with each term of the base sequence, each increment formed by multiplying together c terms of the base sequence. To simplify the discussion, we first consider explicitly the increment sequence formed for $c = 3$; the extension to larger c follows directly. Specifically, for $c = 3$, we form an increment sequence by interleaving the three sequences

$$a_1 a_2 a_3, a_2 a_3 a_4, \ldots, a_i a_{i+1} a_{i+2}, \ldots$$

$$a_1 a_2 a_4, a_2 a_3 a_5, \ldots, a_i a_{i+1} a_{i+3}, \ldots$$

$$a_1 a_3 a_4, a_2 a_4 a_5, \ldots, a_i a_{i+2} a_{i+3}, \ldots$$

(and, of course, prepending 1). Now, each increment has exactly two "a" factors in common with two increments that appear later in the sequence, which leads directly to an application of the corollary to Theorem 1. We have

$$n_{a_i a_{i+1} a_{i+2}}(a_{i+1} a_{i+2} a_{i+3}, a_{i+1} a_{i+2} a_{i+4}) = n_{a_i}(a_{i+3}, a_{i+4})$$

$$n_{a_i a_{i+1} a_{i+3}}(a_{i+1} a_{i+2} a_{i+3}, a_{i+1} a_{i+3} a_{i+4}) = n_{a_i}(a_{i+2}, a_{i+4})$$

$$n_{a_i a_{i+2} a_{i+3}}(a_{i+1} a_{i+2} a_{i+3}, a_{i+2} a_{i+3} a_{i+4}) = n_{a_i}(a_{i+1}, a_{i+4}).$$

If the elements $a_{i+1}, a_{i+2}, a_{i+3}$, and a_{i+4} are all relatively prime, and if each term is within a constant factor of the previous, then these are all $O(a_i)$, by Lemmas 2 and 4. Therefore, by Lemma 3, the number of steps to h-sort is $O(N h^{1/3})$ for each increment h in this sequence. (For 1-sorting, we must argue separately that the running time is $O(1)$ if $a_1, a_2, \ldots, a_6$ are all $O(1)$: the running time for 1-sorting is $O(N n(a_1 a_2 a_3, a_4 a_5 a_6))$ since those two increments are relatively prime.) Now, by Lemma 5, we get an $O(N^{5/4})$ bound for this sequence.

The extension of this argument to general c is straightforward:

THEOREM 2. *Given a constant c, there exists a uniform $(c \log N)$-sequence of increments for which the running time of Shellsort is $O(N^{1+1/(c+1)})$.*

Proof. As before, the increment sequence is 1 followed by an interleaving of the c sequences

$$\left\{\left(\prod_{0 \leqslant k \leqslant c} a_{i+k}\right) \Big/ a_{i+c_0}\right\} \; i \geqslant 1$$

where c_0 ranges from c down to 1. For example, for $c = 5$ we have

$$a_1 a_2 a_3 a_4 a_5, \ldots, a_i a_{i+1} a_{i+2} a_{i+3} a_{i+4}, \ldots$$

$$a_1 a_2 a_3 a_4 a_6, \ldots, a_i a_{i+1} a_{i+2} a_{i+3} a_{i+5}, \ldots$$

$$a_1 a_2 a_3 a_5 a_6, \ldots, a_i a_{i+1} a_{i+2} a_{i+4} a_{i+5}, \ldots$$

$$a_1 a_2 a_4 a_5 a_6, \ldots, a_i a_{i+1} a_{i+3} a_{i+4} a_{i+5}, \ldots$$

$$a_1 a_3 a_4 a_5 a_6, \ldots, a_i a_{i+2} a_{i+3} a_{i+4} a_{i+5}, \ldots.$$

Now, we note that each increment has exactly $c-1$ factors in common with two increments that appear later in the sequence, which allows application of the corollary to Theorem 1. When

$$d = \frac{1}{a_{i+c_0}} \prod_{0 \leqslant k \leqslant c} a_{i+k},$$

$$r = \prod_{0 \leqslant k < c} a_{i+1+k},$$

$$s = \frac{1}{a_{(i+1)+(c_0-1)}} \prod_{0 \leqslant k \leqslant c} a_{i+1+k},$$

then

$$n_d(r, s) = n_{a_i}(a_{i+c_0}, a_{i+c+1}).$$

(For example, when $c = 5$, $n_{a_1 a_2 a_3 a_4 a_6}(a_2 a_3 a_4 a_5 a_6, a_2 a_3 a_4 a_6 a_7) = n_{a_1}(a_5, a_7)$.) This works except for $c_0 = 1$, when we take $s = (\prod_{0 \leqslant k < c} a_{i+2+k})$ which still gives $n_d(r, s) = n_{a_i}(a_{i+1}, a_{i+c+1})$. Again, if all $a_{i+1}, \ldots, a_{i+c+1}$ are relatively prime and related by a constant factor then these are all $O(a_i)$ which leads to a bound of $O(N h^{1/c})$ for each increment in this sequence. This gives a Shellsort bound of $O(N^{1+1/(c+1)})$ by Lemma 5, using the same argument as before for 1-sorting.

The proof is completed by exhibiting a sequence $a_1, a_2, \ldots$, that satisfies the conditions above; this is easy because of the density of primes. For example, we can take a_i to be the smallest prime greater than or equal to 2^i, to get a geometrically increasing sequence of primes 1, 2, 5, 11, 17,..., which satisfies the conditions. ∎

Note that the constant implied by the O-notation in Theorem 2 is exponential in c. This makes the increment sequences hardly of practical use. Next, we examine sequences built according to the same principle as those above but which have good practical performance and even better asymptotic bounds:

THEOREM 3. *For any $\varepsilon > 0$, there exists a uniform $(\log N)$-sequence for which the running time of Shellsort is $O(N^{1+\varepsilon/\sqrt{\lg N}})$.*

Proof. As above, we start with a base sequence $a_1, a_2, a_3, \ldots$, of relatively prime integers. In this case, we construct the sequence as follows:

$$\begin{array}{cccc} a_1 & a_1a_2 & a_1a_2a_3 & a_1a_2a_3a_4 \ \ldots \\ & a_1a_3 & a_1a_2a_4 & a_1a_2a_3a_5 \ \ldots \\ & & a_1a_3a_4 & a_1a_2a_4a_5 \ \ldots \\ & & & a_1a_3a_4a_5 \ \ldots \\ & & & \ldots \end{array}$$

The cth column in the table is formed by starting with $\prod_{1 \leqslant i \leqslant c} a_i$, then multiplying each element in the previous column by a_{c+1}. This ensures that each increment exactly divides two increments which appear later in the sequence. The following table gives the upper bound for the increment appearing in the corresponding position in the above sequence:

$$\begin{array}{cccc} n(a_2, a_3) & n(a_3, a_4) & n(a_4, a_5) & n(a_5, a_6) \ \ldots \\ & n(a_2, a_4) & n(a_3, a_5) & n(a_4, a_6) \ \ldots \\ & & n(a_2, a_5) & n(a_3, a_6) \ \ldots \\ & & & n(a_2, a_6) \ \ldots \\ & & & \ldots \end{array}$$

If we use c columns in the table, then we use $\frac{1}{2}(c^2 + c)$ increments, all less than $\prod_{1 \leqslant i \leqslant c} a_i$ with a total cost of less than $N(\sum_{1 \leqslant i \leqslant c} a_i)^2$. (This bound follows quickly from the fact that $n(r, s) < rs$.) Once again, we achieve good asymptotics by proper choice of the base sequence. Specifically, we take a_i to be the smallest prime greater than or equal to α^i, so that

$$a_i = O(\alpha^i)$$

$$\prod_{1 \leqslant i \leqslant c} a_i = O(\alpha^{\frac{1}{2}(c^2+c)})$$

$$\prod_{1 \leqslant i \leqslant c} a_i = O(\alpha^c).$$

Using all the increments less than N corresponds to taking c equal to the closest integer $\sqrt{2 \log_\alpha N}$, we have a total cost of

$$O(N \alpha^{2\sqrt{2 \log_\alpha N}}) = O(N^{1+(2\sqrt{2 \lg \alpha})/\sqrt{\lg N}}).$$

with $\log_\alpha N$ increments. Given $\varepsilon > 0$, take $\alpha = 2^{\varepsilon^2/8}$ to obtain the stated result. ∎

There is a quite simple proof of the same asymtotic result for non-uniform sequences, due to Chazelle [3]. This result actually motivated the search for the sequence of Theorem 3.

Proof of Theorem 3 (*non-uniform case* [3]). Simply use Pratt's method, starting with $(\alpha - 1)$ and α for an appropriately chosen α (instead of 2 and 3). The running time is bounded by $N\alpha^2$ for each of the $O((\log_\alpha N)^2)$ increments, for a total of

$$N(\lg N)^2 \frac{\alpha^2}{(\lg \alpha)^2}.$$

Now, take α such that $(\lg \alpha)^2 = \alpha^* \lg N$, or $\alpha = 2^{\sqrt{\alpha^* \lg N}}$, for a total cost of

$$N \frac{\lg N}{\alpha^*} 2^{2\sqrt{\alpha^* \lg N}} = O\left(\frac{\lg N}{\alpha^*} N^{1+2\sqrt{\alpha^*}/\sqrt{\lg N}}\right).$$

Again, proper choice of the α^* gives $O(N^{1+\varepsilon/\sqrt{\lg N}})$ for any $\varepsilon > 0$. However, ε does affect the number of increments. There are $O((\lg N/\lg \alpha)^2)$ increments. ∎

The table below shows the number of exchanges required by Knuth's sequence, the uniform sequence suggested from Theorem 3 with $a_i =$ the smallest prime greater than or equal to 2^i, and the non-uniform sequence with $\alpha^* = 1$, averaged over a few random files for various file sizes.

	10000	20000	40000	80000
Knuth	242110	556142	1317825	2898495
Theorem 3 (uniform)	219536	489187	1054873	2288179
Theorem 3 (non-uniform)	242248	545801	1153723	2755272
Pratt	473900	1018642	2177565	4688691

Note that, even though the best worst-case results we have been able to prove for Theorem 3's increments are asymptotically worse than the $O(N(\log N)^2)$ for Pratt's method, the average case appears to be significantly better. This has little relevance when comparing the methods as networks, but it is significance when comparing them as sorts on a general-purpose computer.

CONCLUSIONS

Despite the substantial improvements that we have been able to make in upper bounds for Shellsort, the results still pertain to particular increment sequences of somewhat artificial construction and there seems to be room for improvement. Furthermore, even the bounds derived for the given sequences are not tight. For example, they only derive from the effects of a few of the previous passes and they don't take into account obvious correlations in insertion costs of successive elements.

It seems likely that better bounds can be obtained by taking such effects into account, and these are worth exploring because of the direct practical benefits that accrue. The question of whether there exists an increment sequence of $O(\log N)$ numbers which produces an $O(N \log^2 N)$ or $O(N \log N)$ Shellsort still remains open.

APPENDIX

Proof of Theorem 1. By definition, $n_d(r, s)$ is the number of multiples of d which cannot be represented as linear combinations of r and s. The following facts about the gcd function will prove useful later on. (It is straightforward to verify these.)

FACT 1. *For positive integers r, s, and d, if* $\gcd(d, r, s) = 1$ *then* $\gcd(d/\gcd(d, r), s) = \gcd(d, s)$.

FACT 2. *For positive integers r, s, and d,*

$$\frac{r}{\gcd(d, r)} = r/\gcd(d, r, s) \Big/ \gcd\left(\frac{d}{\gcd(d, r, s)}, \frac{r}{\gcd(d, r, s)}\right).$$

Fact 2 can be rewritten, giving us the following identity for positive integers r, s, and d:

$$\gcd\left(\frac{d}{\gcd(d, r, s)}, \frac{r}{\gcd(d, r, s)}\right) = \frac{\gcd(d, r)}{\gcd(d, r, s)}. \tag{1}$$

We will first look at $n_d(r, s)$ when $\gcd(d, r, s) = 1$ and then prove the desired result.

Claim. Given positive integers r, s, and d such that $\gcd(d, r, s) = 1$,

$$n_d(r, s) = n_{d/\gcd(d, r)}\left(\frac{r}{\gcd(d, r)}, s\right). \tag{2}$$

Recall $n_d(r, s) =$ the number of c_1 such that there exists no $c_2, c_3 \geqslant 0$ with $c_1 d = c_2 r + c_3 s$. Let $y = \gcd(d, r)$, denote d/y (resp. r/y by x_d (resp. x_r), it is clear that $\gcd(x_d, x_r) = 1$. Since $\gcd(d, r, s) = 1$ we know that $\gcd(y, s) = 1$.

If we let C represent the condition "there exists no $c_2, c_3 \geqslant 0$ with $c_1 d = c_2 r + c_3 s$" then we know that C is equivalent to the following:

$$\begin{aligned} C &\equiv \text{there exists no } c_2, c_3 \text{ with } c_1 x_d y = c_2 x_r y + c_3 s \\ &\equiv \text{there exists no } c_2, c_3 \text{ with } c_1 x_d = c_2 x_r + (c_3/y)\, s \\ &\equiv \text{there exists no } c_2, c_3' \text{ with } c_1 x_d = c_2 x_r + c_3' s. \end{aligned}$$

Thus,

$$n_d(r, s) = n_{x_d}(x_r, s) = n_{d/\gcd(d, r)}\left(\frac{r}{\gcd(d, r)}, s\right)$$

proving the claim stated above. Notice that this equation is symmetric in r and s. We could, in the same manner, have derived

$$n_d(r, s) = n_{d/\gcd(d, s)}\left(r, \frac{s}{\gcd(d, s)}\right).$$

Also notice that if $\gcd(d, r, s) = 1$ then $\gcd(d/\gcd(d, r), r/\gcd(d, r), s) = 1$ as well. If we let $D = d/\gcd(d, r)$, $R = r/\gcd(d, r)$, and $S = s$ then we may apply Eq. (2) again since $\gcd(D, R, S) = 1$. This gives us the following:

$$\begin{aligned} n_d(r, s) &= n_D(R, S) \\ &= n_{D/\gcd(D, S)}\left(R, \frac{S}{\gcd(D, S)}\right). \end{aligned}$$

By Fact 1, we know that $\gcd(D, S) = \gcd(d/\gcd(d, r), s)$ is simply $\gcd(d, s)$. Substituting into the above equation, we get

$$n_d(r, s) = n_{d/\gcd(d, r)\gcd(d, s)}\left(\frac{r}{\gcd(d, r)}, \frac{s}{\gcd(d, s)}\right) \tag{3}$$

whenever $\gcd(d, r, s) = 1$.

Next we consider $n_d(r, s)$ without the gcd constraint. Let $k = \gcd(d, r, s)$ then $d = x_d k$, $r = x_r k$, and $s = x_s k$. $n_d(r, s) =$ the number of c_1 such that there exists no $c_2, c_3 \geqslant 0$ with $c_1 d = c_2 r + c_3 s$. If we let C represent the condition "there exists no $c_2, c_3 \geqslant 0$ with$c_1 d = c_2 r + c_3 s$" then we know that C is equivalent to the following:

$$\begin{aligned} C &\equiv \text{there exists no } c_2, c_3 \text{ with } c_1 x_d k = c_2 x_r k + c_3 x_s k \\ &\equiv \text{there exists no } c_2, c_3 \text{ with } c_1 x_d = c_2 x_r + c_3 x_s. \end{aligned}$$

So we have

$$n_d(r, s) = n_{x_d}(x_r, x_s) = n_{d/\gcd(d, r, s)}\left(\frac{r}{\gcd(d, r, s)}, \frac{s}{\gcd(d, r, s)}\right).$$

Recall that from the definition of gcd we know that $\gcd(x_d, x_r, x_s) = 1$. We can now apply the result in equation (3) and obtain the following

$$n_d(r, s) = n_{x_d}(x_r, x_s) = n_{x_d/\gcd(x_d, x_r)\gcd(x_d, x_s)}\left(\frac{x_r}{\gcd(x_d, x_r)}, \frac{x_s}{\gcd(x_d, x_s)}\right).$$

By Fact 2 we have $x_r/\gcd(x_d, x_r) = r/\gcd(d, r)$. We can apply this three times in the above equation, to x_r, x_s, and also to x_d with one of the gcd's in the denominator. This give us

$$n_d(r, s) = n_{d/\gcd(x_d, x_r)\gcd(d, s)}\left(\frac{r}{\gcd(d, r)}, \frac{s}{\gcd(d, s)}\right).$$

Finally, since $\gcd(x_d, x_r) = \gcd(d/\gcd(d, r, s), r/\gcd(d, r, s))$ we can use Eq. (1),

$$\gcd(x_d, x_r) = \frac{\gcd(d, r)}{\gcd(d, r, s)}.$$

This leads to the desired result,

$$n_d(r, s) = n_{d\gcd(d, r, s)/\gcd(d, r)\gcd(d, s)}\left(\frac{r}{\gcd(d, r)}, \frac{s}{\gcd(d, s)}\right).$$

Note that for any r, s, and d this allows us to express $n_d(r, s)$ in terms of a generalized Frobenius function with the three arguments pairwise relatively prime. The second part of this theorem allows us to express this in terms of the standard Frobenius function. We wish to show that if d, r, and s are pairwise relatively prime, then

$$n_d(r, s) = n\left(r, s, \frac{r + b_1 s}{d}, \cdots, \frac{(d-1)r + b_{d-1}s}{d}\right)$$

where b_i is the unique integer between 1 and $d-1$ such that $ir + b_i s \equiv 0 \pmod{d}$.

Again, $n_d(r, s)$ = the number of c_1 such that there exists no $c_2, c_3 \geqslant 0$ with $c_1 d = c_2 r + c_3 s$. We must show that this equals the number of integers which cannot be written as a linear combination of r, s, $(r + b_1 s)/d$,..., $((d-1)r + b_{d-1}s)/d$, where the numerators are congruent to 0 mod d.

If we let C represent the condition "there exists no $c_2, c_3 \geqslant 0$ with $c_1 d = c_2 r + c_3 s$" then we know that C is equivalent to the following: "there exists no c_2, c_3 with

$c_1 = (c_2 r + c_3 s)/d$." Let $c_2 = x_2 d + y_2$ and $c_3 = x_3 d + y_3$ where $0 < y_2, y_3 < d$, then we have the following

$$C \equiv \text{there exists no } x_2, x_3, y_2, y_3 \text{ with } c_1 = x_2 r + x_3 s + \frac{y_2 r + y_3 s}{d}.$$

If $y_2 r + y_3 s$ is congruent to zero mod d, then the last term above is divisible by d. We know since $\gcd(r, d) = \gcd(s, d) = 1$ that both r and s have inverses r' and s' such that $rr' \equiv ss' \equiv 1 \pmod{d}$. We can use this equivalence to show that the inverse must also be relatively prime to d.

Notice that for $j = 1,..., d-1$ that $(jr')\, r + ((d-j)\, s')\, s \equiv 0 \pmod{d}$. Let $i = jr' \bmod d$, then $b_i = (d-j)\, s' \bmod d$. But since r' and s' are both relatively prime to d we know that i and b_i take on every value from the set $\{1,..., d-1\}$ if we let $j = 1,..., d-1$. These are the only times $y_2 r + y_3 s \equiv 0 \pmod{d}$, so we have $C \equiv$ there exists no $x_2, x_3, z_1,..., z_{d-1}$ with

$$c_1 = x_2 r + x_3 s + z_1 \frac{r + b_1 s}{d} + \cdots \oplus z_{d-1} \frac{(d-1)\, r + b_{d-1} s}{d}.$$

Thus $n_d(r, s) = n(r, s, (r + b_1 s)/d,..., ((d-1)\, r + b_{d-1} s)/d)$. ∎

References

1. M. Ajtai, J. Komlos, and E. Szemerdi, An $O(n \log n)$ sorting network, *in* "Proceedings 15th Annual ACM Symposium of Theory of Computing," April 1983, Boston, Mass.
2. W. J. Curran-Sharp, Solution to Problem 7382 (Mathematics), *Ed. Times (London)* **1** (1884).
3. B. Chazelle, private communication, 1983.
4. W. Dobosiewicz, An efficient variation of bubble sort, *Inform. Process. Lett.* **11**, No. 1 (1980), 5–6.
5. S. M. Johnson, A linear diophantine problem, *Canad. J. Math.* **12** (1960), 390–398.
6. D. E. Knuth, "The Art of Computer Programming. Volume 3: Sorting and Searching," Addison–Wesley, Reading, Mass., 1973.
7. T. Leighton, Tight bounds on the complexity of parallel sorting, *in* "Proceedings 16th Annual ACM Symposium of Theory of Computing," April 1984, Washington, D.C.
8. A. A. Papernov and G. V. Stasevich, "A method of information sorting in computer memories," *Probl. Inform. Transmiss.* **1**, No. 3 (1965), 63–75.
9. V. Pratt, "Shellsort and Sorting Networks," Garland, New York, 1979; Ph. D. thesis, Stanford University, 1971.
10. R. Sedgewick, "Algorithms," Addison–Wesley, Reading, Mass., 1983.
11. R. Sedgewick, A new upper bound for Shellsort, *J. Algorithms*, in press.
12. E. S. Selmer, On the linear diophantine problem of Frobenius, *J. Reine Angew. Math.* **294** (1977), 1–17.
13. D. L. Shell, A high-speed sorting procedure, *Comm. ACM* **2**, No. 7 (1959), 30–32.