JOURNAL OF ALGORITHMS 7, 159–173 (1986)

A New Upper Bound for Shellsort

ROBERT SEDGEWICK*

Department of Computer Science, Princeton University, Princeton, New Jersey 08544

Received October 21, 1982

A direct relationship between Shellsort and the classical "problem of Frobenius" from additive number theory is used to derive a sequence of $O(\log N)$ increments for Shellsort for which the worst case running time is $O(N^{4/3})$. The previous best-known upper bound for sequences of $O(\log N)$ increments was $O(N^{3/2})$, which was shown by Pratt to be tight for a large family of sequences, including those commonly used in practice. The new upper bound is of theoretical interest because it suggests that increment sequences might exist which admit even better upper bounds, and of practical interest because the increment sequences which arise outperform those common used, even for random files.

1. INTRODUCTION

Shellsort [15] is a classical sorting algorithm which, despite its practical utility and unusual simplicity, has yet to submit to analysis. The algorithm may be implemented as follows:

```
repeat
  k := h[t]; t := t - 1;
  for i := k + 1 to N do
    begin
    v := a[i]; j := i - k;
    repeat
      if v ≥ a[j] then goto 0;
      a[j + k] := a[j]; j := j - k
    until j < 1;
    0: a[j + k] := v
  end;
until k = 1;
```

*This research was supported in part by NSF Grant MCS-80-17579 while the author was at Brown University, and in part while the author was visiting the Institute for Defense Analyses, Princeton, N.J.

0196-6774/86 $3.00

The algorithm is based on a sequence $h_t, h_{t-1}, \ldots, h_1$ of *increments* which are successively assigned, in decreasing order, to the variable k. When $k = 1$, the method is a simple *insertion sort*: each element is inserted into place among the elements to its left by moving those that are larger right one position. We require that $h_1 = 1$ to ensure that the algorithm fully sorts the file. When $k > 1$, the file becomes *k-sorted*: the k subfiles of elements spaced k apart are all sorted. By successive h_j-sorting passes for decreasing h_j, we hope to move elements close to their final position, each pass improving things for subsequent passes. The following table shows how a sample file is sorted by Shellsort with the increments 1, 3, 7, 15:

	2	7	1	8	2	8	1	8	2	8	4	5	9	0	4	5
15-sorted	2	7	1	8	2	8	1	8	2	8	4	5	9	0	4	5
7-sorted	2	2	1	4	2	8	0	4	5	8	8	5	9	1	8	7
3-sorted	0	1	1	2	2	5	4	2	5	7	4	8	8	8	8	9
Sorted	0	1	1	2	2	2	4	4	5	5	7	8	8	8	8	9

The performance of the algorithm clearly depends on the choice of increments, and it is natural to appeal to mathematical analysis to help choose the best sequence of increments.

Below are listed some sequences which have been suggested for use:

1	2	4	8	$\cdots$	2^k	$\cdots$	(Shell)
1	3	7	15	$\cdots$	$2^k - 1$	$\cdots$	(Hibbard)
1	3	5	9	$\cdots$	$2^k + 1$	$\cdots$	(Papernov–Stasevich)
1	4	13	40	$\cdots$	$\frac{1}{2}(3^k - 1)$	$\cdots$	(Knuth)

Typically, the sequences used are "almost" geometric sequences: geometric sequences with a small constant added to or subtracted from each term. For simplicity, we will work with increasing infinite sequences and assume that all increments less than N are used, in decreasing order, to sort a file of N elements. Then, for fixed N, t is defined to be the largest integer such that $h_t < N$.

Unfortunately, the algorithm has been analyzed only for some special cases. Furthermore, these results indicate that the dependence on the choice of increments can be dramatic. If $t = 1$, then Shellsort is equivalent to insertion sort, an algorithm whose performance is well understood. For insertion sort, the running time is known to be proportional to the number of *inversions* in the input (each element must be moved past the elements which are greater than it and to the left): the worst case running time is $O(N^2)$ [8]. For $t > 1$ the running time of Shellsort is known to be $O(N^{3/2})$ (on the average and in the worst case) for the special case where each increment divides the previous increment [8]. On the other hand, Pratt [12] gives a set of $O((\log N)^2)$ increments for which the running time is $\Theta(N(\log N)^2)$. Between these extremes, a host of open problems remain.

For example, not even the asymptotic growth of the average case performance is known, for the types of sequences used in practice, even though empirical studies show Shellsort to be among the most efficient sorting methods available.

In this paper, we consider upper bounds on the worst case running time. The first results for this problem are due to Papernov and Stasevich [11]. Their results were extended by Pratt [12], who showed that the worst case running time is $\Theta(N^{3/2})$ for sequences that approximate geometric progressions whose common ratio is an integer, a property which holds for sequences commonly used in practice. It turns out that this is a significant restriction, for we are able to exhibit sequences which do not satisfy Pratt's property for which the worst case running time is $O(N^{4/3})$. The method used to prove the upper bound is an extension of the previous methods which leads to a classical problem in number theory, the problem of Frobenius, and results due to Selmer [13].

This result suggests that even better upper bounds might be possible (though new results on the Frobenius problem might be needed), and it suggests sequences of increments that perform better on the average than those commonly used (though this can only be verified empirically).

Section 2 states Pratt's results and describes the upper bound proof of Papernov and Stasevich. Section 3 deals with the Frobenius problem and Selmer's results. Section 4 combines these to give the new upper bound proof. Concluding remarks are offered in Section 5.

2. Previous Bounds

The general argument used for upper bound proofs was given in 1965 by Papernov and Stasevich [11]. It involves bounding the time to h_j-sort in two ways, then picking the smaller bound for each j and summing on j.

The first bound is simple: when h_j-sorting, we are dealing with h_j independent files of about N/h_j elements each. The worst case running time for each of these files is $O((N/h_j)^2)$ (this tim is required, for example, when they are in reverse order); therefore h_j-sorting the entire file requires $O(N^2/h_j)$ steps in the worst case.

The second bound requires more subtle reasoning. First we need the fundamental result:

Lemma 1. *If a k-sorted file is h-sorted, it remains k-sorted.*

This result goes back at least to Boerner [2]: a proof is given in [8]. □

Lemma 1 implies that when we come to h_j-sort a file, it is already h_{j+1}-sorted and h_{j+2}-sorted. But this means that when we come to any particular element $a[k]$ during the h_j-sort, there are many elements which

are guaranteed to be smaller, and the h_j-sort will require fewer exchanges. To bound the number of such elements, we begin with the following observation:

LEMMA 2. *If a file is h-sorted and k-sorted, then, for each* i_0, $a[i_0 - i] \leq a[i_0]$ *whenever* i *can be expressed as a linear combination with nonnegative coefficients of* h *and* k.

Proof. If $i = sh + tk$, then $a[i_0] \geq a[i_0 - h] \geq \cdots \geq a[i_0 - sh]$ since the file is h-sorted, and $a[i_0 - sh] \geq a[i_0 - sh - k] \geq \cdots \geq a[i_0 - sh - tk] = a[i_0 - i]$ since the file is k-sorted. □

The key fact which limits the number of exchanges required by Shellsort is that if h and k are relatively prime, eventually every integer can be expressed as a linear combination of h and k. We have

LEMMA 3. *If* h *and* k *are relatively prime, then every integer greater than* $(h - 1)(k - 1) - 1$ *can be represented as a linear combination of* h *and* k *with nonnegative coefficients.*

Proof. See Knuth [8, Ex. 5.2.1–5.2.2], and discussion in Section 3. □

From these lemmas, we can now prove a second upper bound on the worst case for h_j-sorting. From Lemma 1, when we come to h_j-sort a file, it is already h_{j+1}-sorted and h_{j+2}-sorted. From Lemmas 2 and 3, the elements which are greater than any particular element i_0 must be among those elements within the first $(h_{j+2} - 1)(h_{j+1} - 1)$ positions to the right of i_0. But only one out of each h_j of these elements are examined when h_j-sorting, so the time to process i_0 is $O(h_{j+1}h_{j+2}/h_j)$. This holds for $1 \leq i_0 \leq N$, so the total time to h_j-sort the whole file is $O(Nh_{j+1}h_{j+2}/h_j)$. This is the "second bound" that allows the derivation of $O(N^{3/2})$ upper bounds for many Shellsorts.

If h_{j+1} and h_{j+2} are $O(h_j)$, which holds for the types of increment sequences which have been used for Shellsort, then we have a simple tradeoff between the "first bound" of $O(N^2/h_j)$ and the "second bound" of $O(Nh_j)$. For example, we have

THEOREM 1 (Papernov–Stasevich). *The running time of Shellsort is* $O(N^{3/2})$ *for the increments* $1, 3, 7, 15, 31, 63, 127, 255, \ldots, 2^j - 1, \ldots$.

Proof. Let $h_j = 2^j - 1$. For $h_j = O(N^{1/2})$, use the second bound; for large h_j use the first bound. This gives a bound of

$$N \sum_{1 \leq j < t/2} (2^j - 1) + \sum_{t/2 \leq j \leq t} \frac{N^2}{(2^j - 1)} = O(N2^{t/2}) = O(N^{3/2}).$$

(In these formulas, we have $2^{t/2} = \Theta(N^{1/2})$ by the definition of t.) □

Pratt extended this theorem to cover most "almost geometric" sequences of the type used in practical applications. It is clear that $O(N^{3/2})$ is the best lower bound available using the lemmas above, because if even one increment is $O(N^{1/2})$, then the two bounds are roughly equal and the running time for that increment is $O(N^{3/2})$. Pratt's result shows that the contribution from other increments does not raise the asymptotic worst case running time for a large class of increments.

Indeed, Pratt shows that $O(N^{3/2})$ is the best possible upper bound for many increment sequences:

THEOREM 2 (Pratt–Knuth). *The running time of Shellsort is* $\Omega(N^{3/2})$ *for the increments* $1, 3, 7, 15, 31, 63, 127, 255, \ldots, 2^j - 1, \ldots$.

Proof. A construction of a permutation for which the running time is $\Omega(N^{3/2})$ is given in Knuth [8, Ex. 5.2.1–24]. □

Pratt [12] extended this result to cover a large family of sequences. The most important property required of an increment sequence for the $\Theta(N^{3/2})$ bound to hold is an *integer ratio* condition: there must exist k so that $h_k = \Theta(N^{1/2})$ and for each $j > k$, there exists an integer m so that $h_j = mh_k + \Theta(1)$. (Besides this condition, Pratt's proof involves certain technical requirements on h_{k+1}.) This property is held by the type of increment sequences which have been tried in practice: those which approximate geometric sequences with an integer common ratio.

The generalizations of Theorems 1 and 2 by Pratt seem to suggest that the worst case asymptotic performance of Shellsort is $\Theta(N^{3/2})$ for the increment sequences of interest. In Section 4, we show that this bound can be improved for some sequences which violate the "integer ratio" condition. Before doing so, we need to examine extensions of Lemma 3 in some detail.

3. THE FROBENIUS PROBLEM

Suppose that a country wishes to issue stamps in only a limited number of denominations $a_1, a_2, \ldots, a_k$. What is the largest value which cannot be achieved using only those denominations, and how many values cannot be achieved? This problem is named after the German mathematician Frobenius, apparently because of a comment by Brauer [3] that "Frobenius mentioned it occasionally in his lectures". Despite its simple formulation, there are few results available on the problem in its general form.

To be precise, define $g(a_1, a_2, \ldots, a_k)$ to be the largest integer which cannot be represented as a linear combination with nonnegative integer coefficients of $a_1, a_2, \ldots, a_k$, and define $n(a_1, a_2, \ldots, a_k)$ to be the number

of integers with no such representation. For the problem to make sense, we assume that $\gcd(a_1, a_2, \ldots, a_k) = 1$ (otherwise an infinite number of integers would have no representation) and that all $a_1, a_2, \ldots, a_k$ are > 1 (otherwise all integers could be represented). Also, we assume that $a_1, a_2, \ldots, a_k$ are *independent*: that none can be represented as a linear combination with nonnegative integer coefficients of the others (otherwise it could be deleted from the list without affecting the result).

For $k = 2$, we have Lemma 3 from the previous section. Sylvester posed this as a problem for solution in 1884: the solution as given by Curran Sharp [14],

$$g(a_1, a_2) = (a_1 - 1)(a_2 - 1) - 1,$$
$$n(a_1, a_2) = \tfrac{1}{2}(a_1 - 1)(a_2 - 1),$$

for a_1, a_2 relatively prime.

Obviously, we could apply any results for $k > 2$ in the same way as we applied Lemma 3 in the previous section to get a "second bound" on the worst case of $O(Ng(h_{j+1}, h_{j+2}, \ldots, h_{j+k})/h_j)$ for h_j-sorting. (It is tempting to consider a more complicated argument which would involve n, but this couldn't improve the asymptotic result because it is known that $g/2 < n \leq g$.)

Unfortunately, few general results are available for the Frobenius problem. A series of papers beginning with Brauer [3, 10, 5] deal chiefly with exact formulas for g and n for various special cases, most of them not apparently applicable to Shellsort. A complete survey of available results along with a method which may be useful for obtaining new results is given by Selmer [13]. Selmer does give a quite general result for $k = 3$:

THEOREM 3 (SELMER, 1977). *If a_1, a_2, a_3 are independent and relatively prime in pairs, then*

$$g(a_1, a_2, a_3) \leq \max[(s - 1)a_2 + (q - 1)a_3, (r - 1)a_2 + qa_3] - a_1,$$

where s is determined by

$$a_3 \equiv sa_2 \bmod a_1, \qquad 1 < s < a_1$$

and q and r are determined by

$$a_1 = gs + r, \qquad 0 < r < s.$$

Proof. See Selmer [13] for a proof, a condition for equality, and a formula for $n(a_1, a_2, a_3)$. □

If the increments are not pairwise relatively prime, it is possible to eliminate common divisors using a result of Johnson:

THEOREM 4 (Johnson, 1960). *If a_1, a_2, a_3 are independent, then*

$$g(a_1, a_2, a_3) = d \cdot g\left(\frac{a_1}{d}, \frac{a_2}{d}, a_3\right) + (d-1)a_3,$$

where $d = \gcd(a_1, a_2)$.

Proof. See Johnson [7]. Coupled with Theorem 3, this result can be developed into a procedure for computing $g(a_1, a_2, a_3)$ whenever it is defined. □

These theorems open the possibility that adding a third value can drastically decrease g. If a_2 and a_3 are $O(a_1)$ and q, s, and r in Theorem 3 or d in Theorem 4 are $O(a_1^{1/2})$, then $g(a_1, a_2, a_3) = O(a_1^{3/2})$ not $O(a_1^2)$ as in the previous bound. In the next section, we exhibit triples that satisfy these conditions (and the conditions of the theorem) and use them to derive an improved upper bound for Shellsort.

4. THE NEW UPPER BOUND

We are now ready to prove the main result of this paper. The proof involves a particular sequence of increments that satisfies a host of conditions. After the proof we will discuss how this sequence was discovered and how others might be found.

THEOREM 5. *The running time of Shellsort is $O(N^{4/3})$ for the increments* $1, 8, 23, 77, 281, 1073, 4193, 16577, \ldots, 4^{j+1} + 3 \cdot 2^j + 1, \ldots$.

Proof. Let $h_j = 4^{j+1} + 3 \cdot 2^j + 1$. Below, we use Selmer's theorem to show that the running time for h_j-sorting is $O(Nh_j^{1/2})$. Then, as in the proof of Theorem 1, we can apply this bound for small h_j and the $O(N^2/h_j)$ bound for large h_j; switching at $h_j = \Theta(N^{2/3})$ when both bounds are $O(N^{4/3})$. The total running time is thus bounded by

$$N \sum_{1 \le j < 2t/3} (4^{j+1} + 3 \cdot 2^j + 1)^{1/2} + \sum_{2t/3 \le j \le t} \frac{N^2}{4^{j+1} + 3 \cdot 2^j + 1} = O(N^{4/3}).$$

To bound the time required for h_j-sorting, we need to apply Selmer's theorem to $h_{j+1}, h_{j+2}, h_{j+3}$.

The reader may verify that

$$(4 \cdot 2^{j+1} + 7)h_{j+2} - (16 \cdot 2^{j+1} + 6)h_{j+1} = h_{j+3}$$

so

$$h_{j+3} \equiv (4 \cdot 2^{j+1} + 7)h_{j+2} \bmod h_{j+1},$$

which means that we can take $s = 4 \cdot 2^{j+1} + 7$ in Theorem 3. From this we can calculate q and r: we have

$$h_{j+1} = (4 \cdot 2^{j+1} + 7)(2^{j+1} - 1) + 8$$

so that we can take $q = 2^{j+1} - 1$ and $r = 8$ in Theorem 3. Substituting, we find that

$$g(h_{j+1}, h_{j+2}, h_{j+3}) = O(8^j) = O(h_j^{3/2}).$$

By the same argument as in the proof of Theorem 1, this leads to an upper bound of $O(Nh_j^{1/2})$ for the running time when h_j-sorting, as desired.

Now, to complete the proof of Theorem 5, we need to show that Selmer's theorem applies for all increment triples: we must have $h_{j+1}, h_{j+2}, h_{j+3}$ independent and pairwise relatively prime for $1 \le j \le t - 3$.

The proofs of pairwise relative primeness involve a symbolic method based on Euclid's algorithm. For example,the following table proves that h_{j+1} is always relatively prime to h_{j+2}: in each line, (u, v) from the previous line is replaced either by $(v, u - qv)$, where qv is the largest multiple of v less than u or by (u, v'), where v' is the largest divisor of v not divisible by 2 or 3. These operations preserve common divisors of the pair of numbers: the first is always valid as in Euclid's algorithm; the second is valid in this case because the original numbers are not divisible by 2 or 3 ($h_j \equiv 1 \bmod 2$ and $h_j \equiv 2 \bmod 3$ for $j \ge 1$), so no common divisor could be.

u	v
$4^{j+3} + 3 \cdot 2^{j+2} + 1$	$4^{j+2} + 3 \cdot 2^{j+1} + 1$
$4^{j+2} + 3 \cdot 2^{j+1} + 1$	$4^{j+2} - 3 \cdot 2^{j+1} - 2$
$4^{j+2} - 3 \cdot 2^{j+1} - 2$	$3 \cdot 2^{j+2} + 3$
$4^{j+2} - 3 \cdot 2^{j+1} - 2$	$2^{j+2} + 1$
$2^{j+2} + 1$	$2^{j+1} + 1$
$2^{j+1} + 1$	2^{j+1}
2^{j+1}	1

The pairs of numbers on each line of this table have the same greatest common divisor, so we have proved that $\gcd(h_{j+1}, h_{j+2}) = 1$ (and that $\gcd(h_{j+2}, h_{j+3}) = 1$) for all j. The proof for (h_{j+1}, h_{j+3}) is a similar table:

u	v
$4^{j+4} + 3 \cdot 2^{j+3} + 1$	$4^{j+2} + 3 \cdot 2^{j+1} + 1$
$4^{j+2} + 3 \cdot 2^{j+1} + 1$	$4^{j+2} - 33 \cdot 2^{j+1} - 14$
$4^{j+2} - 33 \cdot 2^{j+1} - 14$	$9 \cdot 2^{j+3} + 15$
$3 \cdot 4^{j+2} - 99 \cdot 2^{j+1} - 42$	$3 \cdot 2^{j+3} + 5$
$3 \cdot 2^{j+3} + 5$	$2^{j+3} + 3$
$2^{j+3} + 3$	$2^{j+3} - 1$
$2^{j+3} - 1$	4
$2^{j+3} - 1$	1

In the fourth line of this table, u is replaced by $3u$, which simplifies the calculations substantially but which cannot affect the result. This completes the proof that $(h_{j+1}, h_{j+2}, h_{j+3})$ are pairwise relatively prime.

To prove independence of $(h_{j+1}, h_{j+2}, h_{j+3})$, assume that $h_{j+3} = c_0 h_{j+1} + c_1 h_{j+2}$ for $c_0, c_1 > 0$. (This is the only possibility, since h_{j+3} is the largest of the three). Clearly $c_0 < 16$ and $c_1 < 4$. Substituting and rearranging terms, we have

$$c_0(4^{j+2} + 3 \cdot 2^{j+1}) + c_1(4^{j+3} + 3 \cdot 2^{j+2})$$

$$-(4^{j+4} + 3 \cdot 2^{j+3}) = 1 - c_0 - c_1.$$

All terms on the left in this equation are divisible by 2^{j+1}, and the right side cannot be 0, so this implies that 2^{j+1} divides $c_0 + c_1 - 1$. But this is impossible for $j > 3$, so we must have independence for all $(h_{j+1}, h_{j+2}, h_{j+3})$ with $j > 3$. (In fact, we do not have independence for $j = 1$, since $77 = 3 \cdot 23 + 8$.)

Independence for $j > 3$ is sufficient to prove the asymptotic result, since the contribution of h_1, h_2, and h_3 to the total running time is $O(N)$. This follows from the fact that they are relatively prime in pairs, so the "second bound" of $O(Nh_j)$ used in Theorem 1 applies. □

There certainly are other increment sequences for which Shellsort is $O(N^{4/3})$, but construction of sequences which satisfy all the requisite conditions can be difficult. The sequence of Theorem 5 was found by an ad hoc method: we want a_1, a_2, and a_3 to be members of a geometric sequence; we want a_2 and a_3 to be $O(a_1)$; and we want q, s, r to be $O(a_1^{1/2})$. These considerations lead directly to sequences of the form $w4^k + x2^k + y$: the coefficients w, x, and y are determined from the

conditions and constraints of Theorem 3. If we take

$$a_1 = w4^k + x2^k + y,$$
$$a_2 = w4^{k+1} + x2^{k+1} + y,$$
$$a_3 = w4^{k+2} + x2^{k+2} + y,$$

then we need only find s and t of the form

$$s = s_1 2^k = s_0,$$
$$t = t_1 2^k = t_0,$$

such that

$$sa_2 - ta_1 = a_3.$$

(Or, in other words, $a_3 \equiv sa_2 \bmod a_1$.) Substituting and setting coefficients of 2^{3k}, 2^{2k}, 2^k, and 1 equal leads to nonlinear simultaneous equations in these variables which are not difficult to solve. For example, it turns out that

$$s_0 = \frac{45yw - 6x^2}{9yw - 2x^2}.$$

There are similar formulas for s_1, t_0, and t_1. Integer values for w, x, and y need to be chosen to make these integers, with the additional constraints that w, s_1, and t_1 must be positive. For example, the choice $w = 1$, $x = -3$, and $y = 1$ meets this requirement, as does the choice which is used in Theorem 5, $w = 4$, $x = 3$, and $y = 1$. Finally, the sequences must be checked for independence and relative primeness of consecutive triples. For example, the sequence which derives from the first choice above, $4^k - 3 \cdot 2^k + 1$, fails to satisfy the condition that all consecutive triples must be pairwise relatively prime. (There are occasional pairs divisible by 17.)

The same asymptotic result is also available for sequences at the other end of the spectrum, where successive pairs have very large common divisors.

THEOREM 6. *The running time of Shellsort is* $O(N^{4/3})$ *for the increments* $1, 5, 65, 377, 1769, \ldots, 2 \cdot 4^j - 9 \cdot 2^j + 9, \ldots$.

Proof. Let $h_j = 2 \cdot 4^j - 9 \cdot 2^j + 9$. This sequence was invented by multiplying successive terms of the sequence $1, 5, 13, 29, 61, \ldots, 2^j - 3, \ldots$:

we have $h_j = (2^j - 3)(2^{j+1} - 3)$. This construction ensures that

$$\gcd(h_{j+1}, h_{j+2}) = 2^{j+2} - 3$$

so that Theorem 4 can be applied directly. We have

$$\begin{aligned} g(h_{j+1}, h_{j+2}, h_{j+3}) &= (2^{j+2} - 3)g\left(\frac{h_{j+1}}{2^{j+2} - 3}, \frac{h_{j+2}}{2^{j+2} - 3}, h_{j+3}\right) + (2^{j+2} - 4)h_{j+3} \\ &< \frac{h_{j+1}h_{j+2}}{2^{j+2} - 3} + (2^{j+2} - 4)h_{j+3} \\ &= O(h_j^{3/2}). \end{aligned}$$

The second equation follows from the fact that $g(a_1, a_2, a_3) \le g(a_1, a_2) < a_1a_2$ if $a_1, a_2 < a_3$. The rest of the proof proceeds exactly as for Theorem 5. □

The sequences in Theorems 5 and 6 violate Pratt's integer ratio condition in the same way: a geometric sequence is modified by adding a slowly growing term (but not a constant). It turns out that sequences of this type not only lead to good upper bounds but also they lead to good average case performance. This is discussed in more detail in the next section.

5. Conclusion

There is a clear possibility that increment sequences exist which lead to an even better upper bound. The most obvious weakness in Theorem 5 is that it only takes into account, for each increment, the effects of the previous three increments, not all those that have been used. However, results like Theorem 3 for more than three increments are rare. Several results for special cases are available [13], but they seem difficult to apply in the way that we used Theorem 3. The main difficulty is embedding the special sequences within a sequence of $O(\log N)$ increments so that a good bound can be derived for each increment.

The connection with the Frobenius problem not only guides us in searching for new increment sequences, it also helps us to study old ones. Selmer gives a method which might lead to an analytic derivation for some increment sequences, and which can be adapted to a procedure [4] that can compute the exact value of $g(h_{j+1}, \ldots, h_t)$ for any sequence $h_1, \ldots, h_t$, and any j. This gives the best bound available using the general method of Theorems 1 and 5. The table below shows how the various bounds are

related for the increment sequence suggested by Knuth [8], the sequence in Theorem 5, and a more conventional sequence with values close to the sequence of Theorem 5. For each increment h_j, three bounds are given on the number of exchanges required for h_j-sorting: the bound from Theorem 1, $g(h_{j+1}, h_{j+2})/h_j$, which is labeled Ω_2; the bound from Theorem 5, $g(h_{j+1}, h_{j+2}, h_{j+3})/h_j$, which is labeled Ω_3; and $g(h_{j+1}, \ldots, h_{j+t})/h_j$, which is labeled Ω_t.

$\frac{1}{2}(3^j - 1)$

	1	4	13	40	121	364	1093	3280	9841
Ω_2	35	117	360	1089	3276	9837	29529	88569	265716
Ω_3	35	88	277	817	2464	7378	22147	66427	199294
Ω_t	35	88	249	735	2193	6567	19689	59055	177153

$4^{j+1} + 1$

	1	5	17	65	257	1025	4097	16385
Ω_2	63	205	964	4033	16320	65472	262080	1048512
Ω_3	63	164	780	3224	13066	52378	209675	838810
Ω_t	63	164	734	3037	12252	49116	—	—

$4^{j+1} + 3 \cdot 2^j + 1$

	1	8	23	77	281	1073	4193	16577
Ω_2	153	209	925	3898	15992	64759	260599	1045494
Ω_3	153	132	359	853	1867	3909	8003	16193
Ω_t	153	132	359	798	1540	3044	5865	—

The numbers in the corresponding table for the sequence from Theorem 6 are undefined for the Ω_2 row and about a factor of two larger than the corresponding numbers from the table for the sequence from Theorem 5 for the other rows. These tables clearly indicate the dramatic reductions in the upper bound achieved by Theorem 5, and show that some further reduction is possible by considering more increments.

A second major weakness of the bounds of Theorems 1 and 4 is that they ignore the fact that substantially less time is required to h_j-sort when the file is already sorted in multiples of h_j. For example, a file which is 24-sorted and 36-sorted can be 12-sorted in linear time (using at most N exchanges). (Note carefully, however, that the time required to, say, 11-sort a file which is 12-, 24-, and 36-sorted could be quadratic in the worst case.) Pratt devised a sequence which exploits this property: in his sequence, twice and thrice each increment is also in the sequence. This guarantees that the running time is proportional to N times the number of increments, which is $O((\log N)^2)$. It may be possible to devise a sequence for which some elements have a low bound because of this divisibility property and others have a low bound because of "nondivisibility" as suggested by Theorem 5. Also, it is an interesting problem to extend the "best bound" calculation to incorporate this divisibility property.

Simulation results show that the type of sequence suggested by Theorem 5 is of practical interest. The table below shows the number of exchanges

required by Knuth's sequence and a sequence similar to the sequence of Theorem 5, averaged over three random files, for various values of N:

1, 4, 13, 40, 121, 364, 1093, 3280, 9841, 29524, ...,

N	1024	2048	4096	8192	16384	32768	65536
exchanges	14325	32970	75786	171472	386531	858447	1884863

1, 5, 19, 41, 109, 209, 505, 929, 2161, 3905, 8749, 16001, 36449, 64769, ...,

N	1024	2048	4096	8192	16384	32768	65536
exchanges	14495	32141	72585	162078	354045	771812	1665118

The particular sequence used here is a merge of the sequences with $h_j = 4^j - 3 \cdot 2^j + 1$ and $h_j = 9 \cdot 4^j - 9 \cdot 2^j + 1$. These occasionally have triples that are not relatively prime, but the combination does better on random inputs than the sequence of Theorem 5 because it has more smaller increments. Of course, there is no clear relationship between average case performance and the worst-case results being considered in this paper, except that studying the worst case did lead us to sequences which violate Pratt's integer divisibility condition and which turn out to outperform previous sequences on the average. These simulations were part of an extensive study on a wide range of increment sequences that will be reported on in a future paper.

Shellsort is already the method of choice in many practical situations (especially when the number of elements to be sorted is not large), and an increment sequence which improves the running time by, say, 50% would have very significant practical impact. The methods of this paper show significant promise in the search for such a sequence.

The major unsolved analytic problem in the study of Shellsort is to determine the asymptotic behavior of the average running time for any "almost geometric" sequence of $O(\log N)$ increments which do not necessarily divide each other. The connection with the Frobenius problem described in this paper gives a set of analytic and computational tools that have the potential to lead to significant progress in this area.

Finally, the methods of this paper have potential applicability to one of the major problems in the theory of sorting. How many comparators are required in a sorting network which sorts according to a fixed predetermined sequence of comparisons? (See, for example, Knuth [8] for a description of this problem and its history.) The existence of a sequence of $O(\log N)$ increments for Shellsort each with a "second upper bound" of $O(N)$ would imply the existence of a sorting network with $O(N(\log N))$ comparators. Pratt's method with $O((\log N)^2)$ increments corresponds to a network with $O(N(\log N)^2)$ comparators, and it was long thought that no asymptotically network exists (see below). The machinery from the

literature of the Frobenius problem may help to shed further light on this fundamental problem.

Note added in proof. A number of related results have been developed in the interval between original submission and final publication of this paper.

Of most significance is the paper by Ajtai, Komlos, and Szemeredi [1] which settles the question referred to in the last paragraph: they exhibit a sorting network with just $O(N \log N)$ comparators. Other networks for various models are given by Leighton [9]. These results make the search for a Shellsort-based network even more appealing, because such a network would be far simpler (and likely to be of direct practical utility) and the existence proofs of [1] and [9] lend credence to the conjecture that an $O(N \log N)$ Shellsort might exist.

Incerpi and Sedgewich [6] have extended the results of this paper to show that the running time of Shellsort can be made to be $O(N^{1+\varepsilon})$ for any $\varepsilon > 0$, still using only $O(\log N)$ increments. Also, they show that this result is the best possible if only a constant number of previous increments is taken into account in the proof, so that improved results on the Frobenius problem in the form of extensions to Theorem 3 for four or any constant number of increments (which seems very difficult) still could not lead to better asymptotic results for Shellsort. The paper shows further improvements to $O(N^{1+\varepsilon/\sqrt{N}})$. The increment sequences used for these results involve large common divisors in successive increments, as in Theorem 6. However, it still seems as though the best increment sequences will involve a combination of "Frobenius effects" (as in Theorem 5) and "large common divisor effects" (as in Theorem 6), so new results on the Frobenius problem will be of interest, especially in developing increment sequences for practical use.

ACKNOWLEDGMENT

Janet Incerpi's assistance in the preparation of this paper was invaluable.

REFERENCES

1. AJTAI, KOMLOS, AND SZMEREDI, "An $O(n \log n)$ Sorting Network," Proceedings 15th Annual ACM Symposium of Theory of Computing, Boston, Mass., April 1983.
2. H. BOERNER, "Darstellung von Gruppen," Springer-Verlag, Berlin, 1955.
3. A. BRAUER, On a problem of partitions, *Amer. J. Math.* **64** (1942), 299–312.
4. H. GREENBERG, An algorithm for a linear diophantine equation and a problem of Frobenius, *Numer. Math.* **34** (1980), 349–352.
5. G. R. HOFMEISTER, Zu einem Problem von Frobenius, *Norske Vid. Selsk. Skr.* **5** (1966), 1–37.
6. J. INCERPI AND R. SEDGEWICK, Improved upper bounds on Shellsort, *J. Comput. System Sci.* **31** (1985), 210–224.
7. S. M. JOHNSON, A linear diophantine problem, *Canad. J. Math.* **12** (1960), 390–398.
8. D. E. KNUTH, "The Art of Computer Programming. 3. Sorting and Searching," Addison–Wesley, Reading, Mass., 1973.
9. T. LEIGHTON, "Tight Bounds on the Complexity of Parallel Sorting," Proceedings 16th Annual ACM Symposium of Theory of Computing, Washington D.C., April 1984.
10. M. LEWIN, On a linear diophantine problem, *Bull. London Math. Soc.* **5** (1973), 75–78.
11. A. A. PAPERNOV AND G. V. STASEVICH, A method of information sorting in computer memories, *Problems Inform. Transmission* **1**(3) (1965), 63–75.

12. V. PRATT, "Shellsort and Sorting Networks," Garland, New York, 1979; Originally presented as the author's Ph.D. thesis, Stanford University, 1971.
13. E. S. SELMER, On the linear diophantine problem of Frobenius, *J. Reine Angew. Math.* **294** (1977), 1–17.
14. W. J. CURRAN SHARP, "Solution to Problem 7382 (Mathematics)," *Educational Times*, London, 1884.
15. D. L. SHELL, A high-speed sorting procedure, *Comm. Assoc. Comput. Mach.* **2**(7) (1959), 30–32.