An Old Sub-Quadratic Algorithm for Finding Extremal Sets

Paul Pritchard*

Abstract

Some previously proposed algorithms of the author are re-examined. They were designed to find all sets in a collection that have no subset in the collection, but are easily modified to find all sets that have no supersets. One is shown to have a worst-case running-time of $O(N^2/\log N)$, where N is the sum of the sizes of all the sets. This is lower than the only previously known sub-quadratic worst-case upper bound for this problem.

1 Introduction

Yellin and Jutla [2] tackled the following problem, for applications of which see [1].

> Given is a collection $\mathcal{F} = \{S_1, \ldots, S_k\}$, where each S_i is a set over the same domain. A set is a *minimal* (resp. *maximal*) set of $\mathcal{F}$ iff it has no strict subset (resp. superset) in $\mathcal{F}$. Find the *extremal* sets of $\mathcal{F}$, i.e., those that are minimal or maximal.

With the problem size chosen as $N = \sum_i S_i$, Yellin and Jutla presented an algorithm with worst-case running-time $O(N^2/\sqrt{\log N})$ and expected time $O(N^2/\log N)$, and stated that they knew of "no previous works that show a sub-quadratic complexity".

In [1] we had previously developed algorithms for finding the minimal sets in a collection. The main algorithms were analyzed with N denoting the number (rather than total size) of sets, and were observed to involve $O(N^2)$ subset tests or other operations of the same order of complexity (rather than RAM operations). The most complex algorithms were analyzed in complex terms of multiple parameters of the input. All of which mitigates against a direct comparison with Yellin and Jutla's algorithm.

*On leave from School of Computing and Information Technology, Griffith University, Queensland, Australia 4111

In this paper we re-analyze our algorithms in the terms and notation of [2]. It is shown that our main algorithms are indeed quadratic in the worst-case, but that one of the variants (only briefly discussed in [1]) based on an opportunistic subset test has sub-quadratic worst-case complexity — indeed, at $O(N^2/\log N)$, superior to the worst-case bound obtained in [2].

2 Our Previous Algorithms and Analyses

Our algorithms derive from a simple algorithmic ancestor:

Algorithm A_0:
  $\mathcal{F}' := \mathcal{F}$;
  forall (x, y) **in** $\mathcal{F} \times \mathcal{F}$ **do**
    if $y \subset x$ **then** $\mathcal{F}' := \mathcal{F}' - \{y\}$ **fi**
  od $\{\mathcal{F}' =$ the minimal sets of $\mathcal{F}\}$

A_0 is quadratic in the worst case. Since there are k sets, the cost is $\Theta(k^2)$ subset tests. With the sets represented as ordered linked lists or arrays of their elements, a subset test can be done in time bounded by the size of the larger set involved. Since each set is compared with every set in $\mathcal{F}$, the total complexity is $O(kN)$, which is $O(N^2)$. Since $k = \Theta(N)$ is possible, this worst-case bound is tight.

This basic algorithm is easily modified to compute maximal sets, by deleting x instead of y, or to compute both kinds of extremal sets, by recording for each set whether a subset and whether a superset has been found. Its descendants can be similarly modified.

In [1], algorithm A_0 was progressively refined with a view to increasing its efficiency. The presentation's high levels of new notation and attention to detail unfortunately act to obscure the algorithmic ideas, which are essentially quite simple, so we proceed to give here an informal presentation.

The first refinement avoids obviously redundant subset tests, by partitioning $\mathcal{F}$ into the sub-collections $\mathcal{F}_n$ of the sets of size n, and only evaluating $y \subset x$, where $x \in \mathcal{F}_n$ and $y \in \mathcal{F}_m$, when $m < n$. The resulting algorithm A_1 saves at least half the subset tests, but retains the quadratic worst-case complexity.

The second refinement avoids further subset tests, by imposing a particular total order $\prec$ (called *precedes*) on $\mathcal{F}$ (and all finite sets over its domain). With the domain treated as a totally ordered alphabet, and each set mapped to the string of its members concatenated in increasing order, the order on sets is just lexicographic order on the associated strings. [1, Corollary 2] shows that if $y \subset x$ then either $x \prec y$ or the string associated with y is a strict prefix of that associated with x. All the latter "prefix" subsets can be discovered in a preliminary scan of $\mathcal{F}$ in time $O(N)$, after which subset tests can be restricted to the former kind. On average about half the remaining tests are avoided in the resulting algorithm A_2, leaving its worst-case complexity quadratic.

The remaining refinements exploit an "opportunistic" algorithm for finding all matching pairs of elements in two ordered sequences. If the sequences have lengths n and m, where $m \leq n$, the algorithm requires $O(m \log(1 + n/m))$ comparisons of elements, because each of the $O(m)$ steps along each sequence requires a number of comparisons that is logarithmic in the size of the step. Note that the complexity is (at worst) $O(n)$.

The third refinement gives algorithm A_3. It aims to avoid subset tests by finding, for each x in $\mathcal{F}_n$, *all* y in $\mathcal{F}_m$ for which $y \subset x$ (or just whether there is such a y, if only minimal sets are sought), without necessarily examining every member of $\mathcal{F}_m$ that x precedes. It does so by treating this problem as that of finding all matching pairs of elements in (a) the sequence of members of $\mathcal{F}_m$ that x precedes and (b) the sequence of subsets of x of size m, both ordered by $\prec$.

Each step along sequence (b) is accomplished with a *single* subset-like operation, and since there are no more steps than the members of sequence (a), the cost of this refinement is no more than that of individually checking all the y's. Each step along sequence (a) is accomplished with a logarithmic number of comparisons (with $\prec$) of sets of size m. Since the number of subsets of size m of a set of size n is given by the binomial co-efficient C^n_m, the total cost of processing all of $\mathcal{F}_n \times \mathcal{F}_m$ is

$$O(|\mathcal{F}_n|(|\mathcal{F}_m| \ \mathbf{min}\ C^n_m)(t^n_m + m \log \frac{|\mathcal{F}_m|}{|\mathcal{F}_m| \ \mathbf{min}\ C^n_m})) \quad (1)$$

where t^n_m is the worst-case cost of the subset-like operation on sets of size n and m, $n > m$; $t^n_m = \Omega(m)$ and $t^n_m = O(n)$.

In [1, Section 7] we showed that if each set in $\mathcal{F}$ has at most K members, then the worst-case complexity of algorithm A_3 is $O(2^K N \log k)$. (Recall that k is the number of sets in $\mathcal{F}$). It follows that if K is a constant, then the complexity is $O(N \log N)$.

The fourth refinement uses the opportunistic sequence matching algorithm to speed up the subset-like operation involved in stepping along sequence (b) in the context above, so that its cost is $O(m \log(1 + n/m))$ rather than $O(n)$. This possibility was briefly discussed in [1, Section 8], where an informal analysis classified it as a '"theoretical"' improvement. We analyze algorithm A_4, obtained by incorporating this refinement in A_3, in the next section.

3 Further Complexity Analyses

Despite its good performance on sets of bounded size, it is not difficult to see that the worst-case complexity of A_3 is still quadratic. Suppose, for example, that $\mathcal{F}$ comprises a single set of size $N/2$ and $N/4$ sets of size 2. Then because $C^{N/2}_2 > N/4$ for N sufficiently large, there is nothing gained by testing the small sets *en masse*, and $N/4$ comparisons are performed, each capable of requiring all of the large set to be scanned.

An opportunistic subset test can be used in algorithms A_0, A_1 and A_2. The above counter-example is then processed in time $O(N \log N)$, but the algorithms remain quadratic in their worst-cases, when all sets are small.

However, it transpires that the two techniques of batch subset testing and opportunistic subset testing handle complementary subsets of the input efficiently, and acting in concert give a sub-quadratic worst-case complexity for algorithm A_4.

Let $\mathcal{F}_m^n$ denote $\bigcup_{m \leq i \leq n} \mathcal{F}_i$. Partition $\mathcal{F}$ into the collection $\mathcal{F}_0^D$ of "smaller" sets and $\mathcal{F}_{D+1}^K$ of "larger" sets, where the cutoff D is to be determined later. The cost of finding all subsets of the smaller sets is $O(2^D N \log N)$, by the result given earlier for A_3, since this sub-problem is equivalent to the full problem with $\mathcal{F}_0^D$ as input.

To find the subsets of the larger sets, both partitions must be examined. Finding the subsets *from* the larger sets is equivalent to the full problem with $\mathcal{F}_{D+1}^K$ as input, and thus costs $O(N^2/D)$, as there are $O(N/D)$ larger sets.

It remains to consider the cost of searching the smaller sets for subsets of the larger sets. We first partition the larger sets into $\mathcal{F}_{D+1}^{D \log D}$ and $\mathcal{F}_{D \log D+1}^K$. The cost of processing the latter collection is bounded by that of applying the opportunistic subset-like operation to each x, y where $x \in \mathcal{F}_{D \log D+1}^K$ and $y \in \mathcal{F}_0^D$. It is therefore of order

$$\begin{aligned} & \sum_x \sum_y |y|(1 + \log(|x|/|y|)) \\ \leq & \sum_x \sum_y |y|(1 + \log |x|) \\ = & \sum_x (1 + \log |x|) \sum_y |y| \\ = & N \sum_x 1 + N \sum_x \log |x|. \end{aligned}$$

Since each x has more than $D \log D$ members, there are at most $N/(D \log D)$, and the first summand is $O(N^2/(D \log D))$. The second sum is the log of a product, which is maximized when each x is as small as possible (provided $D > 2$). So the second summand is $O(N(N/(D \log D)) \log(D \log D)) = O(N^2/D)$.

It remains now to determine the cost of searching the smaller sets for subsets of $\mathcal{F}_{D+1}^{D \log D}$. We partition the former into $\mathcal{F}_0^d$ and $\mathcal{F}_{d+1}^D$, where the cutoff d is to be determined later. The cost of searching the latter sets is of order

$$\begin{aligned} & \sum_x \sum_y |y|(1 + \log(|x|/|y|)) \\ \leq & \sum_x (N/d)(1 + \log(|x|/d)) \end{aligned}$$

since $\sum_y |y|(1 + \log(|x|/|y|))$ is maximized when each y is as small as possible (provided $d > 2$). Continuing, the latter expression

$$= \quad N/d \sum_x 1 + N/d \sum_x \log(|x|/d)$$

$$
\begin{aligned}
&< (N/d)(N/D) + (N/d)(N/D)\log(D/d) \\
&= O(N^2 \log(D/d)/(Dd)).
\end{aligned}
$$

Finally, consider the cost of searching $\mathcal{F}_0^d$ for subsets of sets in $\mathcal{F}_{D+1}^{D\log D}$. (The reader will be relieved to see no further partitioning!) Applying (1), the cost is of order

$$
\begin{aligned}
& \sum_{n=D+1}^{D\log D} \sum_{m=0}^{d} |\mathcal{F}_n| C_m^n (n + m\log k) \\
< & \sum_{n=D+1}^{D\log D} |\mathcal{F}_n| \sum_{m=0}^{d} n^m (n + m\log N) \\
\leq & \sum_{n=D+1}^{D\log D} |\mathcal{F}_n| \sum_{m=0}^{d} (D\log D)^d (D\log D + d\log N) \\
\leq & \; N(d+1)(D\log D)^d (D\log D + d\log N).
\end{aligned}
$$

The above calculations have bounded the complexity of A_4 at

$$
\begin{aligned}
& O(2^D N\log N) + O(N^2/D) + O(N^2\log D/(Dd)) + \\
& O(Nd(D\log D)^d(D\log D + d\log N)).
\end{aligned} \tag{2}
$$

Now with all logarithms taken to base 2, put $D = \log(N/\log^2 N)$ and $d = \log D$, so that $D = \Theta(\log N)$, $d = \Theta(\log\log N)$ and $d < \log\log N$. The first term of (2) becomes $O(N^2/\log N)$, as do the second and third. The fourth term is seen to be $O(N\log\log N(\log N\log\log N)^{\log\log N+1})$. Since $N = \log N^{\log N/\log\log N}$, this last term is seen to be $O(N^{1+\epsilon})$ for any $\epsilon > 0$.

Therefore the worst-case complexity of A_4 is $O(N^2/\log N)$.

This is the best-possible worst-case bound for A_4, because there are inputs which require $\Theta(N^2/\log N)$ operations. An example is the family of sets constructed by Yellin and Jutla in [2, Theorem 4] to give a lower bound on the size of the subset graph constructed by their algorithm. There are $\Theta(N/\log N)$ sets of size s, and $\Theta(N/\log N)$ sets of size $2s+1$, where $s = \Theta(\log N)$. Equation (1) reveals that $\Theta(N^2/\log^2 N)$ subset-like operations are performed by A_4 with this family of sets as input, and each has cost $\Theta(\log N)$.

4 Final Remarks

Although our worst-case bound is superior, it is equaled by the expected time bound of Yellin and Jutla's algorithm. However, the latter requires the use of sophisticated hashing techniques with much larger constants implicit in the big-oh upper bound.

Nevertheless, the approach in [2] is conceptually cleaner than ours, being founded on a natural mathematical structure, and perhaps offers better prospects for improvement.

The respective approaches in [1, 2] seem to have nothing in common except the problem. It remains to be seen whether their respective insights can be mutually beneficial.

Acknowledgments

I wish to thank the Department of Computer Science at the University of Edinburgh for kindly hosting my sabbatical visit.

References

[1] P. Pritchard, Opportunistic algorithms for eliminating supersets. *Acta Informatica* ***28*** (1991) 733–754.

[2] D. M. Yellin and C. S. Jutla, Finding extremal sets in less than quadratic time. *Inform. Process. Lett.* ***48*** (1993) 29–34.