Algorithmica (1995) 14: 154–168

Algorithmica

© 1995 Springer-Verlag New York Inc.

Lower Bounds for Set Intersection Queries

P. Dietz,[1] K. Mehlhorn,[1] R. Raman,[1] and C. Uhrig[1]

Abstract. We consider the following *set intersection reporting* problem. We have a collection of initially empty sets and would like to process an intermixed sequence of n updates (insertions into and deletions from individual sets) and q queries (reporting the intersection of two sets). We cast this problem in the *arithmetic* model of computation of Fredman [F1] and Yao [Ya2] and show that any algorithm that fits in this model must take time $\Omega(q + n\sqrt{q})$ to process a sequence of n updates and q queries, ignoring factors that are polynomial in log n. We also show that this bound is tight in this model of computation, again to within a polynomial in log n factor, improving upon a result of Yellin [Ye]. Furthermore, we consider the case $q = O(n)$ with an additional space restriction. We only allow the use of m memory locations, where $m \leq n^{3/2}$. We show a tight bound of $\Theta(n^2/m^{1/3})$ for a sequence of n operations, again ignoring the polynomial in log n factors.

Key Words. Algorithms, Arithmetic model, Data structures, Intersection reporting, Lower bounds, Memory restriction, Set handling, Upper bounds.

1. Introduction. We consider the complexity of maintaining a collection of sets with a very simple but fundamental set of operations: we would like to support updates, which are insertions into and deletions from individual sets and intersection queries reporting the intersection of two sets. Other variations could include returning the size of the intersection, or retrieving some values associated with the elements in the intersection. A unifying way of studying these problems is as follows: we are given a universe $\mathcal{U}$ of *keys*, a set M of *information* items that will be associated with elements of $\mathcal{U}$, a function $I: \mathcal{U} \to M$ that associates values from M with keys in $\mathcal{U}$, and a collection $\mathcal{C}$ of initially empty subsets of $\mathcal{U}$. Assume further that $M = (M, +, 0)$ is a *monoid*, i.e., that M is closed under some associative and commutative operator $+$ and that 0 is the identity element for $+$.

We want to maintain $\mathcal{C}$ while processing a sequence of update operations of the form *insert*(x, A) and *delete*(x, A), for $x \in \mathcal{U}$ and $A \in \mathcal{C}$, so as to answer efficiently queries of the form *intersect*(A, B), $A, B \in \mathcal{C}$, which returns $\sum_{x \in A \cap B} I(x)$, where the sum is taken with respect to $+$. We also consider a variant of the problem, where the number of memory locations available is restricted. It is easy to cast the intersection problem and its variants in this framework. The basic problem defined above can be obtained by letting $M = (2^{\mathcal{U}}, \cup, \{\})$ and $I(x) = \{x\}$ for all x, and the problem where one merely has to report the size of the intersection can be obtained by setting $M = (\mathbb{N}, +, 0)$ and $I(x) = 1$ for all x, where here $+$ is arithmetic addition.

[1] Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany.

Received January 5, 1993; revised July 19, 1993. Communicated by B. Chazelle.

The *arithmetic* model of computation was proposed by Fredman [F1] and Yao [Ya2] as a framework for studying "information retrieval" problems of the above kind. A set of data points is to be stored, each with some associated information; queries return some combination of the values associated with some subset of the data points which matches some query specification. Many geometric retrieval problems have been previously studied in this framework (see [C], [F1], [F2], and Section VII.2.3 of [M]). One of the assumptions made in this framework is that the algorithm must be general enough to work for any choice of the combining operator, provided only that the associated values form a monoid under this operator. In particular, the algorithm is not permitted to assume that the combining operation is invertible (a more detailed description of this framework is given in Section 2). This makes it easier to prove lower bounds and is justified by the fact that the best known algorithms for most problems of interest do not effectively make use of the potential invertibility of the combining operator.

Yellin [Ye] gives a data structure for the set intersection problem which fits in the arithmetic model of computation. Yellin's algorithm processes a series of n *insert* and *delete* operations and q *intersect* operations in time $\tilde{O}(n \cdot n^{1/k} + qn^{(1-1/k)})$ time for any fixed k, where $\tilde{O}(f(n)) = \bigcup_{c=0}^{\infty} O(f(n) \log^c n)$, i.e., polylogarithmic factors are ignored (a similar convention is used for the Ω notation, with inverse polylogarithmic factors being ignored).

In order to process a sequence of n updates and q queries, the value of k that yields the smallest running time of $\tilde{O}(q + n\sqrt{q})$ is defined by $n^{1/k} = \min\{\sqrt{q}, n\}$. We prove that this bound is tight; i.e., any algorithm that fits within the arithmetic model must take $\tilde{\Omega}(q + n\sqrt{q})$ time to process a sequence of n updates and q queries. This is true even for the expected running time of randomized algorithms for this problem. Further, we prove that this bound can be achieved on-line by an algorithm that does not know either n or q in advance. Yellin obtains a weaker bound of $\tilde{O}(n^{1/2}(n + q))$ by using a "doubling" technique. We refer to this problem in the following as Problem 1.

Our lower bound applies to algorithms that know the values of n and q, as well as the series of queries and updates, in advance, and do not handle deletions; the upper bound can be achieved by an algorithm that knows neither parameter in advance and handles on-line an intermixed sequence of queries and updates (both insertions and deletions). That is, the semidynamic off-line version of this problem is not easier than the general case within this model of computation.

We also consider a variant of the problem which we call Problem 2. Here we want to process a sequence of n operations (updates as well as queries). The number of memory locations available is limited by m, $m \leq n^{3/2}$. For this variant we show a lower bound of $\tilde{\Omega}(n^2/m^{1/3})$ time to process the sequence. Again we show that this bound is tight by giving an algorithm that needs time $\tilde{O}(n^2/m^{1/3})$.

2. The Lower bounds.

We now give a somewhat simplified description of the lower-bound model which conveys the essential aspects: the interested reader is referred to Section VII.2.3 of [M] for further details. The data structure is modeled as a collection of variables $v_0, v_1, \ldots,$ which take values in M, and initially contain

0. In Problem 1 this collection is considered to be unlimited whereas in Problem 2 the number of variables is at most m. After receiving the input to each operation, the algorithm executes a sequence of operations of the form $v_i \leftarrow$ INPUT, $v_i \leftarrow v_i + v_k$, or OUTPUT $\leftarrow v_i$. Here INPUT refers to the information associated with a key that is the argument to an *insert* or *delete* operation. The algorithm must be correct for all choices of M, thus in particular it cannot assume that the operator $+$ is invertible. The cost of an algorithm in processing a sequence of operations is the total number of such instructions executed by it. All other computation is given for free.

The intuition for the lower bounds for the two problems considered is essentially as follows. We consider the problem where the elements in the intersection are to be reported, i.e., we take $M = (2^{\mathscr{U}}, \cup, \{\,\})$ and $I(x) = \{x\}$. We construct a collection of sets the (sums of) sizes of whose pairwise intersections are large, and query all possible intersections of the sets. If the answers to all the queries were to be obtained by unioning together singleton sets, then the lower bound would follow.

However, this is too simplistic: subsets obtained as temporary values during the computation of one answer may be reused to answer another query. An important observation to make at this point is that a subset that is used to compute the answer to several intersection queries must lie in the common intersection of all the sets involved. Therefore, we also ensure that the common intersection of sufficiently many (this is still a small quantity) of the sets we construct is small, from which it follows that no large subset obtained during the computation of one answer can be used to answer very many different queries.

In the proof to Problem 2 we destroy the useful subsets by some deletions and afterward perform hard queries where we have to compute the answers from scratch.

Now we flesh out the above arguments.

2.1. Problem 1. Let n and q denote the number of updates and queries, respectively, and assume that $q \leq n^2$ (otherwise a lower bound of $\Omega(q) = \Omega(q + n\sqrt{q})$ is trivial). We construct a family $\mathscr{F}$ of $\sqrt{q}$ subsets, each a subset of $\mathscr{U} = \{1, \ldots, 2n/\sqrt{q}\}$. Let $u = |\mathscr{U}|$ and $f = |\mathscr{F}|$.

We want to construct this family of sets with the following properties:

(a) $|S_i \cap S_j|$ is $\Omega(u)$ for all i, j with $1 \leq i < j \leq f$.

(b) For any pairwise distinct indices $i_1, \ldots, i_L$, $|\bigcap_{j=1}^{L} S_{i_j}| < L$, if L is sufficiently large.

We call such a family of sets *acceptable.* If an acceptable family of sets can be found, then the theorem can be proved as follows. We first build up the sets by insertions. Note that this requires at most $fu = \Theta(n)$ update operations, since the size of each set is at most u. We then query the pairwise intersections of all the sets; i.e., we query $S_i \cap S_j$ for $1 \leq i < j \leq f$. There are

$$\binom{f}{2} = \Theta(q)$$

queries in all.

Firstly, note that the sizes of all the output sets sum to $\Omega(uf^2)$ by (a) above. The output to each query is conceptually obtained by a binary *union tree* in which each internal node combines the answers from its children; the external nodes represent singleton sets. Each node can be labeled with a set in the obvious way. Consider the entire forest; we wish to count the number of distinct nodes in the forest (that is, nodes labeled with distinct sets). Since each distinct set corresponds to at least one instruction that is not counted elsewhere, counting the number of distinct sets is a lower bound on the number of instructions executed.

We consider only nodes that correspond to sets of size L or larger. Clearly, the number of such sets is $\Omega(uf^2/L)$. Furthermore, no such set can be used to answer more than $\binom{L}{2}$ different queries. To see this, suppose that any such set B is used to answer intersection queries involving L or more different sets, say $S_{i_1}, \ldots, S_{i_L}$. Then, as discussed above, it must be the case that $B \subseteq \bigcap_{j=1}^{L} S_{i_j}$. However, $|B| \geq L$, which contradicts (b) above.

Thus there can be at most $L - 1$ distinct sets such that B can answer queries posed involving these sets, from which it follows that there are less than $\binom{L}{2}$ queries that B can be used to answer. Thus, $\Omega(f^2u/L^3)$ distinct sets can be counted, giving us a lower bound of this magnitude. Now we show that sets with the required properties exist. The argument is an application of the "probabilistic method" pioneered by Erdös [S], that is, we prove the existence of such a family by a counting argument couched in probabilistic terms. We first consider a discrete sample space and postulate a distribution on it; then we show that the conjunction of the properties we are interested in occurs with nonzero probability, from which we conclude that some sample point must indeed satisfy the conjunction of the properties.

The sample space we consider is all possible families of sets that are subsets of $\mathscr{U}$, i.e., $[2^{\mathscr{U}}]^f$, and the distribution is obtained as follows: $x \in S_j$ holds with probability 1/2 independently of all other such events, for all $1 \leq x \leq u$ and $1 \leq j \leq f$. We now verify that properties (a) and (b) hold with positive probability in the above distribution.

(a) Consider sets S_i and S_j, $i \neq j$. The probability that $x \in S_i \cap S_j = 1/4$. Furthermore, $|S_i \cap S_j|$ is a binomially distributed variable with parameters $(u, 1/4)$. It follows from the Chernoff bounds (Lemma 19 in the Appendix) that

$$\Pr[|S_i \cap S_j| < u/8] < (0.962)^u \leq 1/2f^2,$$

provided u is at least $c \log f$ for some sufficiently large c. (If not, this implies that $q = \tilde{\Omega}(n^2)$ and the lower bound is trivial anyway.) Summing over the $\binom{f}{2}$ possible values for i, j we get

$$\Pr[\exists i, j\colon |S_i \cap S_j| < u/8] < 1/4.$$

(b) Fix a set of pairwise distinct indices $i_1, \dots, i_L$. For any $x \in \mathscr{U}$, the probability that x belongs to all of $S_{i_1}, \dots, S_{i_L}$ is 2^{-L}. Thus, the size of the common intersection of $S_{i_1}, \dots, S_{i_L}$ is a binomially distributed variable with parameters $(u, 2^{-L})$. Let $L = 2 \max\{\log f, \log u\}$. Then the expected size of the intersection is $\min\{uf^{-2}, u^{-1}\}$. It follows from the Chernoff bounds (Lemma 19) that

$$\Pr[|\textstyle\bigcap_{j=1}^{L} S_{i_j}| > L] < \frac{\exp(L)}{(L \max\{f^2/u, u\})^L}.$$

Summing over the

$$\binom{f}{L} < \left(\frac{f}{e}\right)^L$$

choices for the indices we get that, for all indices $i_1 < i_2 < \cdots < i_L$,

$$\Pr[|\textstyle\bigcap_{j=1}^{L} S_{i_j}| > L] < \left(\frac{f}{L \max\{f^2/u, u\}}\right)^L < 1/4.$$

Thus, we see that both properties (a) and (b) hold with probability $> 1/2$, and so there is a family of sets $\mathscr{F}$ satisfying (a) and (b).

From the discussion above, we can now infer that at least $\Omega(f^2 u / L^3)$ operations are needed to process this sequence of queries, from which the lower bound follows since $f = \sqrt{q}$, $u = \Theta(n/\sqrt{q})$, and $L = O(\log n)$. We have just proved:

Theorem 1. *In the Fredman–Yao arithmetic model of computation, any algorithm for the set intersection problem requires $\tilde{\Omega}(q + n\sqrt{q})$ time to process a sequence of n updates and q queries.*

This lower bound can be extended to the expected run-time of randomized algorithms by invoking Yao's corollary [Ya1] to von Neumann's minimax principle [N], which can be stated as follows:

Lemma 2. *Let T_r be the expected run-time of any randomized algorithm for solving a problem P. Let D be any distribution over the inputs to problem P, and Let T_d be the minimum average-case time (under distribution D) of any deterministic algorithm for P. Then $T_r \geq T_d$.*

We note that given the above class of inputs and the above distribution, any deterministic algorithm requires $\tilde{\Omega}(q + n\sqrt{q})$ time for at least half the possible inputs. This implies (since run-times are nonnegative) that the average-case running time for any deterministic algorithm on the above distribution is also $\tilde{\Omega}(q + n\sqrt{q})$. We can thus conclude:

Corollary 3. *In the Fredman–Yao arithmetic model of computation, any randomized algorithm for the set intersection problem requires* $\tilde{\Omega}(q + n\sqrt{q})$ *expected time to process a sequence of n updates and q queries.*

2.2. Problem 2. Now let $\mathscr{U} = \{1, \dots, u\}$ be the universe of elements, let Ml be the set of memory locations, and let $m = |Ml|$ be the number of variables. The number of operations (insertions, deletions, and queries) will be $\Theta(n)$.

If $x \in Ml$ is a memory location and A is a subset of $\mathscr{U}$, then x is said to be *relevant* to A if it contains the sum over some subset of the elements of A. The *size* of a memory location (or a variable) is the number of elements which have been summed to produce its contents. A memory location x is called *major* if its size is at least L.

As shown in Section 2.1 a major memory location is relevant to at most $\binom{L}{2}$ distinct intersections $S_i \cap S_j$ of an acceptable family of sets. Informally the strategy for the lower bound proof is as follows. We choose f to be $m^{1/3}L^2$ and $u = n/f$. Each set of the acceptable family has size $\Theta(u) = \Theta(nm^{-1/3}L^{-2})$.

After building the family we choose an appropriate set S of $O(f)$ elements of $\mathscr{U}$ and delete these elements from all S_i's; this amounts to $O(f^2)$ delete operations The set S has the property that it contains at least one element of each memory location of size $\geq 2mL^2/f^2$, i.e., deletion of the elements of S makes all memory locations of size $\geq 2mL^2/f^2$ irrelevant. The existence of S will be shown by a probabilistic argument. As a consequence we show that for half of the intersections the sum over the sizes of the major memory locations which are still relevant to such an intersection is at most half of its size. Hence, answering all $O(f^2)$ intersection queries will have cost $\Omega(fu^2/L^3)$ as in the previous section. We now reinsert the deleted elements and thus reestablish the original situation. Altogether, we have identified a sequence of $O(f^2)$ operations with total cost $\Omega(fu^2/L^3)$, i.e., the amortized cost per operation is $\Omega(u/L^3) = \Omega((n/m^{1/3})/L^5)$.

We now give the details. Assume that $m \leq n^{3/2}/(\log n)^7$. (The lower bound $\tilde{\Omega}(n/m^{1/3})$ for the amortized cost of an operation follows from Theorem 1 for $m \geq n^{3/2}/(\log n)^7$. Note that Theorem 1 implies an $\Omega(n^{3/2})$ lower bound for a sequence of n updates and n queries even without any restriction of the memory size.) Let $L = 2 \log 2n$, $f = \lceil m^{1/3}L^2 \rceil$, and $u = \lfloor n/f \rfloor$, and let $S_1, \dots, S_f$ be an acceptable family of subsets of $\mathscr{U} = \{1, \dots, u\}$. Let $r_{i,j}$ be the number of major memory locations that are relevant to $S_i \cap S_j$.

Lemma 4. $\sum_{1 \leq i < j \leq f} r_{i,j} \leq mL^2/2$.

PROOF. There are at most m major memory locations and none of them is relevant to more than L of the S_i, and so cannot be relevant to more than $\binom{L}{2}$ of their intersections. □

LEMMA 5. *If variables $a_1, \ldots, a_k$ are nonnegative, and $\sum a_i = B$, then, for any $z > 1$, $|\{i \mid a_i \geq zB/k\}| \leq k/z$.*

PROOF. Obvious. □

These lemmas imply the following. Let $T = 2mL^2/f^2$, and let Z be the set $\{(i, j) \mid 1 \leq i < j \leq f, r_{i,j} \leq T\}$, then

LEMMA 6. $|Z| \geq f(f-2)/4$.

Thus we have shown that for at least a constant part of the intersections the number of relevant major memory locations is limited by T.

LEMMA 7. *Let A be a nonempty subset of $\mathscr{U}$. The probability that none of $2Lu/|A|$ randomly chosen elements (with replacement) of $\mathscr{U}$ is in A is less than $1/n^2$.*

PROOF. A randomly chosen element of $\mathscr{U}$ has probability $|A|/u$ of being in A. Therefore, the probability that k independent choices from $\mathscr{U}$ will not all be in A is $(1 - |A|/u)^k$. Since $(1 - 1/x)^x \leq 1/e$, $(1 - |A|/u)^k \leq e^{-k|A|/u}$. For $k = 2Lu/|A|$, this is $e^{-4 \log 2n} < 1/n^2$. □

Call a memory location *large* if its size is at least $(u/16)/T$. We show next that all large memory locations can be made irrelevant by a small number of deletions.

LEMMA 8. *A set of $32LT$ elements of $\mathscr{U}$ exists such that each large memory location contains an element from the set.*

PROOF. Choose $32LT = 2Lu/((u/16)/T)$ elements from $\mathscr{U}$ at random (with replacement). Then the probability that a particular large memory cell contains none of the chosen elements is bounded by $1/n^2$ according to Lemma 7 and hence the probability that one of the at most m large memory cells contains none of the chosen elements is bounded by $m/n^2 < 1$. The required set therefore exists. □

LEMMA 9. *$LTf = O(f^2)$, $f^2 = o(n)$, and $LT = o(u)$.*

PROOF. We have $LT = mL^3/f^2 = mL^6/(f^2L^3) = O(f^3/(f^2L^3)) = O(f)$. Thus $LTf = O(f^2)$. Also $f^2 = o(m^{2/3}L^4) = o((mL^6)^{2/3}) = o(n)$, since $m \leq n^{3/2}/(\log n)^7$. Finally, $LT = O(f) = O(f^2/f) = o(n/f) = o(u)$. □

Let S be the set of $32LT$ elements chosen according to Lemma 8. Delete all elements of S from all sets $S_1, \ldots, S_f$ of our acceptable family. This amounts to at most $32LTf = O(f^2)$ delete operations. After these delete operations all large memory cells are irrelevant, i.e., cannot be used to answer any intersection query. Moreover, the major memory cells that are not large cover at most half of the intersections $S_i \cap S_j$ for $(i, j) \in Z$ as we show next.

Lemma 10. *Let $(i, j) \in Z$. After the deletions of the elements of S, the major memory locations relevant to $S_i \cap S_j$ have a total size of at most $u/16$. Moreover, $|S_i \cap S_j| \geq u/8 - o(u)$.*

Proof. There are at most T relevant major memory locations for $S_i \cap S_j$, each of size at most $(u/16)/T$. Their total size is therefore at most $u/16$. We have deleted at most $32LT$ elements from $S_i \cap S_j$, so $|S_i \cap S_j| \geq u/8 - 32LT = u/8 - o(u)$. □

Lemma 11. *After performing these deletions, it requires $\Omega(f^2u/L^3)$ time to perform the queries $S_i \cap S_j$, for (i, j) in Z.*

Proof. Note that $|Z| = \Theta(f^2)$. Now we must compute the sum of at least $u/16 - o(u)$ elements for each query. Conservatively, assume that all sums of at most L variables are charged no cost. Since sums of more than L variables are useful for computing fewer than L^2 of the intersections (because the original S_i were acceptable), we can conclude that we require at least $O(f^2u/L^3)$ time to perform the queries (this includes time spent during the deletions leading up to the queries). □

We can now summarize. The cost of a cycle of $O(f^2)$ delete, query, and insert operations is $\Omega(f^2uL^3)$. Moreover, the cycle reestablishes the original acceptable family $S_1, \ldots, S_f$. Consider now the following sequence of $O(n)$ operations: $O(n)$ insert operations to build up an acceptable family $S_1, \ldots, S_f$ followed by $\lfloor n/f^2 \rfloor$ repetitions of the cycle. This sequence has cost $\Omega(nu/L^3) = \Omega(n^2/(fL^3)) = \Omega(n^2/(m^{1/3}L^5)) = \tilde{\Omega}(n^2/m^{1/3})$.

Theorem 12. *There is a sequence of $O(n)$ insert, delete, and intersect operations which takes time $\tilde{\Omega}(n^2m^{-1/3})$ with m memory locations, in the monoid model.*

3. The Upper Bounds.

In this section we prove the following two theorems:

Theorem 13. *Any intermixed sequence of n updates and q queries can be performed in $\tilde{O}(n\sqrt{q} + q)$ time using $O(\min\{n\sqrt{q}, n^2\})$ space.*

Theorem 14. *Any intermixed sequence of $O(n)$ updates and queries can be performed in $\tilde{O}(n^2/m^{1/3})$ time and $O(m)$ space.*

Theorem 13 generalizes Yellin's upper bound [Ye]. Yellin showed how to process n updates and q queries in time $\tilde{O}(nn^{1/k} + qn/n^{1/k})$ for any fixed k with $k \geq 1$, i.e., the parameter k needs to be fixed when his algorithm is started. Thus k can be set to its optimal value (where $n^{1/k} = \min\{\sqrt{q}, n\}$), only when n and q are known in advance; the optimal value of k gives a time bound of $\tilde{O}(n\sqrt{q} + q)$. We show that the bound $\tilde{O}(n\sqrt{q} + q)$ can be achieved without prior knowledge of n and q. We believe that our construction is not only more general than Yellin's but also simpler. We remark that the output of an intersection query $Query(S, S')$ is not only the monoid value $\sum_{x \in S \cap S'} I(x)$ but also a persistent search tree for $S \cap S'$; this is also true for Yellin's algorithm.

In Theorem 14 we do not distinguish between updates and queries but use n to denote the total number of operations. For this situation, Theorem 13 yields an $\tilde{O}(n^{3/2})$ time and space bound. Theorem 14 deals with the situation that the available space is $o(n^{3/2})$. We remark that the algorithm underlying Theorem 14 only computes the monoid value $\sum_{x \in S \cap S'} I(x)$ as the answer of an intersection query; it does not produce a search tree representation of the intersection.

3.1. Problem 1. Let t denote the number of time steps and let $n(t)$ and $q(t)$ be the number of updates and queries, respectively, up to and including time t. Let $f(t) = 1 + \sqrt{q(t)}$ and $u(t) = n(t)/f(t)$.

A set $S \subseteq \mathscr{U}$ is called *small* (at time t) if $|S| < u(t)$, *medium* if $u(t) \leq |S| \leq 2u(t)$, and *large* if $|S| > 2u(t)$. The algorithm *marks* some subsets of $\mathscr{U}$ using the following rule:

MARKING RULE. Initially all subsets are unmarked. When a subset becomes large it is marked and stays marked until it becomes small again at which point it is unmarked.

The marking rule implies that large sets are always marked and small sets are never marked. Medium sets may be marked or not. In particular, a marked set has cardinality at least $u(t)$ and hence there can never be more than $f(t)$ marked sets.

Let $\mathscr{M}$ be the collection of marked sets. For every pair (S_i, S_j) of marked sets we maintain a persistent balanced binary tree for the intersection $S_i \cap S_j$; each internal node of this tree contains the sum of the monoid values associated with the leaves of the subtree rooted at this node. For every set S_i (marked or not) we maintain a balanced binary tree of its elements. We also maintain for each integer i a list of the sets of cardinality i (call such a list a *block*) and a sorted list of the nonempty blocks. In the sorted list of blocks we also maintain pointers to the blocks corresponding to sizes $\lfloor u(t) \rfloor$ and $\lfloor 2u(t) \rfloor$.

LEMMA 15. *The space requirement of the data structure is* $O(\min(n(t)^2, n(t)f(t)))$.

PROOF. First note that in the case that $q(t) \geq n(t)^2$ it holds that $n(t)f(t) > n(t)^2$. Then there are at most $n(t)$ different large sets and the space requirement to store all intersections is $\Theta(n(t)^2)$. So it suffices to show that $N := \sum_{S_i, S_j \in \mathscr{M}} |S_i \cap S_j| = O(n(t)f(t))$. For every element $x \in \mathscr{U}$ let $a(x)$ be the number of marked sets containing x. Then $\sum_x a(x) \leq n(t)$ since the total size of all sets is at most $n(t)$, $a(x) \leq f(t)$ for all x since there are at most $f(t)$ marked sets, and $N = \sum_x a(x)^2$. The sum $\sum_x a(x)^2$ is maximized subject to the two constraints above if exactly $n(t)/f(t)$ values $a(x)$ are equal to $f(t)$ and the remaining values are zero. Thus $N = n(t)f(t)$. □

We can now describe the various operations.

Mark(S). To mark a set S we build for each $S' \in \mathscr{M}$ the intersection tree for $S \cap S'$ by checking for each element $x \in S$ whether x belongs to S'. This takes time $O(u(t)f(t)\log n(t))$.

Unmark(S). To unmark a set S we delete all trees $S \cap S'$ for $S' \in \mathscr{M}$. This takes time $O(u(t)f(t))$.

Insert(x, S). Add x to the tree for S, increase $n(t)$, and change $u(t)$. Unmark all marked sets which became small. If S is marked, then insert x into the appropriate intersection trees $S \cap S', S' \in \mathscr{M}$, and if S is unmarked and became large, then mark S. All of this takes time $O(f(t) \log n(t))$ plus the time for the marking and unmarking.

Delete(x, S). Delete x from the tree for S and, if S is marked, also from all intersection trees. Increase $n(t)$ and change $u(t)$. Unmark all marked sets which became small. All of this takes time $O(f(t) \log n(t))$ plus the time for the unmarking.

Query(S, S'). Increase $q(t)$ and change $f(t)$ and $u(t)$. Mark all unmarked sets which became large. If S and S' are both marked, then answer the query in time $O(1)$ using the intersection tree for $S \cap S'$. If one is unmarked, say S, then check each element of S for membership in S'. All of this takes time $O(u(t) \log n(t))$ plus the time for the marking.

To complete the analysis we define a potential function Φ and show that the amortized cost of mark and unmark is nonpositive and that the amortized cost of the other operations is within a constant factor of their actual cost. Let

$$\Phi(t) = \sum_{S \text{ medium and unmarked}} (|S| - u(t))f(t) \log n(t)$$

$$+ \sum_{S \text{ medium and unmarked}} (2u(t) - |S|)f(t) \log n(t).$$

With this potential function the amortized cost of mark and unmark is clearly nonpositive. Let us turn to the operations insert, delete, and query next. Let S' be the set involved in the operation. Assume first that the operation at time t is an insert or delete. Then

$$\begin{aligned} n(t) &= n(t-1)+1,\\ q(t) &= q(t-1),\\ f(t) &= f(t-1), \end{aligned}$$

and

$$u(t) = \frac{n(t)}{f(t)} = \frac{n(t-1)+1}{f(t-1)}.$$

Now we distinguish several cases:

Case 1: *S' was unmarked at time $t-1$ and is not marked in step t.* Then

$$\begin{aligned}
\Phi(t)-\Phi(t-1) \le{} & \sum_{S\text{ medium and unmarked}} (|S| - u(t))f(t)\log n(t)\\
& + \sum_{S\text{ medium and marked}} (2u(t) - |S|)f(t)\log n(t)\\
& - \sum_{S\text{ medium and unmarked}} (|S| - u(t-1))f(t-1)\log n(t-1)\\
& - \sum_{S\text{ medium and marked}} (2u(t-1) - |S|)f(t-1)\log n(t-1) && (1)\\
\le{} & \sum_{S\text{ medium and marked}} (2u(t) - |S|)f(t)\log n(t)\\
& - \sum_{S\text{ medium and marked}} (2u(t-1) - |S|)f(t-1)\log n(t-1) && (2)\\
\le{} & \sum_{S\text{ medium and marked}} (2u(t)f(t)\log n(t) - 2u(t-1)f(t-1)\log n(t-1))\\
={} & \sum_{S\text{ medium and marked}} (2n(t))\log n(t) - 2n(t-1)\log n(t-1))\\
={} & \sum_{S\text{ medium and marked}} \left(2n(t-1)\log\frac{n(t)}{n(t-1)} + \log n(t)\right)\\
={} & O(f(t)\log n(t)), && (3)
\end{aligned}$$

where (3) follows from the fact that there are at most $f(t)$ medium sets and that $n(t-1)\log(n(t)/n(t-1)) = o(1)$.

Case 2: *S' is medium at time $t-1$ and becomes large or small at time t.* Then $|S'| < u(t)$ or $|S'| > 2u(t)$ and

$$\Phi(t)-\Phi(t-1) \le O(f(t)\log n(t)),$$

where this is computed analogously to case 1.

Case 3: *S′ becomes medium at time t.* Then $|S'| = 2u(t)$ or $|S'| = u(t)$. In each case

$$\Phi(t) - \Phi(t-1) \le O(f(t)\log n(t)).$$

Case 4: *The operation is an insert or delete and S′ is medium (and marked or unmarked).* Then

$$\Phi(t) - \Phi(t-1) \le f(t)\log n(t) + O(f(t)\log n(t)),$$

where the first term follows from the possibly new element of S' and the second term is computed analogously to case 1.

Thus, in all cases the potential increase is bounded by a constant factor of the actual cost.

Assume next that the operation at time t is a query operation. Then

$$\begin{aligned} n(t) &= n(t-1), \\ u(t)f(t) &= u(t-1)f(t-1), \qquad (4) \\ q(t) &= q(t-1) + 1, \end{aligned}$$

and

$$\begin{aligned} f(t) - f(t-1) &= \sqrt{q(t)} - \sqrt{q(t) - 1} \\ &= O\left(\frac{1}{\sqrt{q(t)}}\right) \\ &= O\left(\frac{1}{f(t)}\right). \qquad (5) \end{aligned}$$

Hence

$$\begin{aligned} \Phi(t) - \Phi(t-1) \le & \sum_{S \text{ medium and unmarked}} (|S|f(t)\log n(t) - |S|f(t-1)\log n(t-1)) \\ & + \sum_{S \text{ medium and marked}} (u(t-1)f(t-1)\log n(t-1) - u(t)f(t)\log n(t)) \\ & + \sum_{S \text{ medium and unmarked}} (|S|f(t-1)\log n(t-1) - |S|f(t)\log n(t)) \\ & + \sum_{S \text{ medium and marked}} (2u(t)f(t)\log n(t) - 2u(t-1)f(t-1)\log n(t-1)) \\ \le & \sum_{S \text{ medium and unmarked}} (|S|f(t)\log n(t) - |S|f(t-1)\log n(t-1)) \qquad (6) \\ \le & \, n(t)(f(t) - f(t-1))\log n(t) \qquad (7) \\ = & \, O\left(\left(\frac{n(t)}{f(t)}\right)\log n(t)\right) \qquad (8) \\ = & \, O(u(t)\log n(t)), \end{aligned}$$

where (6) follows since the contribution of all the other terms is nonpositive (see (4)). For (7) note that $\sum_{S_{\mathbf{medium}}} |S| \le n(t)$. Finally (8) follows from (5).

Thus, the potential increase is within a constant factor of the actual cost. We have thus shown:

THEOREM 16. *Any intermixed sequence of n updates and q queries can be performed in $O((n\sqrt{q} + q)\log n)$ time using $O(\min\{n\sqrt{q}, n^2\})$ space.*

3.2. Problem 2. We assume that an *a priori* bound $m \le n^{3/2}$ on the number of memory locations is given. We show how to process a sequence of n updates and queries in time $\tilde{O}(n/m^{1/3})$ within $O(m)$ memory space.

We first fix a few constants. Let $f = m^{1/3}$, $u = n/f$, and $c = nf/m = n/m^{2/3}$. The algorithm to be described is a modification of the algorithm of the preceding section. We describe the differences. The quantities f, u, and n are used instead of $f(t)$, $u(t)$, and $n(t)$. As before, each set S is stored in a balanced binary tree. In addition, we maintain a partition of S into blocks of between c and $2c$ contiguous elements. Each block knows the elements belonging to it and each element knows the block it belongs to.

For every pair S and S' of marked sets one of the two sets is designated as the *leader* of the pair. The cardinality of the leader must not exceed twice the cardinality of the other set in the pair. The intersection tree for $S \cap S'$ is organized as follows: It has a leaf for each block B of the leader. The leaf corresponding to block B contains $\sum_{x \in B \cap S \cap S'} I(x)$, where $I(x)$ is the monoid value associated with x. Each internal node of the intersection tree contains as before the sum of the monoid values of the leaves below it. The intersection tree for $S \cap S'$ has at most $2\min(|S|, |S'|)/c$ leaves and hence the total size of all intersection trees is at most $nf/c = O(m)$ as the next lemma shows.

LEMMA 17. *Let k be an integer and let $u_1, \dots, u_k$ be reals with $\sum_i u_i \le n$ and $u_i \ge n/f$ for all i. Then $\sum_{i<j} \min(u_i, u_j) \le nf$.*

PROOF. Let $u_1^*, \dots, u_k^*$ maximize $N(u_1, \dots, u_k) = \sum_{i<j} \min(u_i, u_j)$ subject to the constraints $\sum_i u_i \le n$ and $u_i \ge n/f$ for all i; a simple compactness argument shows that $u_1^*, \dots, u_k^*$ exists. Assume without loss of generality that $u_1^* \le u_2^* \cdots \le u_k^*$. Then $N(u_1^*, \dots, u_k^*) = \sum_i n_i(k - i)$. Assume now that $u_i^* < u_{i+1}^*$ for some i. Let $\Delta = u_{i+1}^* - u_i^*$. Then $N(u_1^*, \dots, u_{i-1}^*, u_i^* + \Delta/2, u_{i+1}^* - \Delta/2, u_{i+2}^*, \dots, u_k^*) = N(u_1^*, \dots, u_k^*) + \Delta/2$, a contradiction. Thus $u_1^* = \cdots = u_k^*$ and hence

$$N(u_i^*, \dots, u_k^*) = \binom{\frac{n}{k}}{}\binom{k}{2} \le nf. \qquad \square$$

We can now discuss the various operations. Besides the hidden operations mark and unmark, which stay unchanged, we also need a hidden operation change-leader. Assume that S and S' are marked sets with S being the leader. When

$|S| > 2|S'|$ we make S' the leader and reconstruct the intersection tree for $S \cap S'$ in time $O(|S'| \log n)$. S was the leader of the pair for the last $\Omega(|S|) = \Omega(|S'|)$ updates of one of the members of the pair and hence the amortized cost of change-leader per update is $O(\log n)$.

We turn to the insert and delete alorithms next. They are as described above with the following three changes:

—Suppose that x is inserted into (deleted from) the marked set S and that S' is another marked set. Let $S'' \in \{S, S'\}$ be the leader of the pair. Locate the block B of S'' containing x and recompute the value associated with B. Then update the intersection tree. All of this takes time $O(c \log n)$ for a single marked set S' and hence total time $O(fc \log n) = \tilde{O}(n/m^{1/3})$.

—If some block of set S becomes too large ($>2c$ elements) or too small ($<c$ elements), then balance it with the neighboring block. This can also be done in time $O(fc \log n)$.

—If the leader of a pair of marked sets needs to be changed, then use change-leader to make the change.

We conclude that an update operation takes time $\tilde{O}(n/m^{1/3})$ plus the time for the mark and unmark operations. The query operation is as described above but note that it will not do any unmark operations. It thus takes time $O(u \log n) = \tilde{O}(n/m^{1/3})$. It remains to account for the mark and unmark operations. Each such operation takes $O(uf \log n) = O(n \log n)$ time and there can be no more than $O(n/u)$ of them (since two such operations on the same set are separated by u updates of the set); the total cost of all marks and unmarks is therefore $O((n^2/u) \log n) = \tilde{O}(nf) = \tilde{O}(nfc) = O(n^2/m^{1/3})$.

We summarize in:

THEOREM 18. *Any intermixed sequence of $O(n)$ updates and queries can be performed in $\tilde{O}(n^2/m^{1/3})$ time and $O(m)$ space.*

4. Discussion. It would be very interesting to determine the complexity of these problems when the combining operation is known to be invertible. We should point out that Fredman [F2] shows tight bounds for the problem of maintaining partial sums of an array in the above framework, both when the combining operator is invertible and when it is not. In Fredman's paper allowing inverses leads to a constant factor improvement in the running time.

We also want to point out that our lower bound does not rule out a solution with query time $O(\log n + k)$, where k is the size of the answer, and logarithmic update time. It would be interesting to find out if such a solution exists. For example, for the half-space reporting problem there is a lower bound of $\Omega(\sqrt{n})$ in the arithmetic model [C] and an upper bound of $O(\log n + k)$ [CGL]. On the other hand, the best known upper bound for *counting* the number of points in the query region is $O(\sqrt{n} \log n)$ [CGL].

In addition, Yellin's algorithm also solves the related problem of answering the yes–no query $A \subseteq B$? for sets A and B. Since decision problems such as these do not fit into the arithmetic framework, our lower bound does not apply, though the upper bound does. It would be interesting if matching upper and lower bounds could be found for the subset testing problem.

Finally, we are interested in generalizing the proofs for the upper and lower bound of Problem 2 for arbitrary q.

Appendix. Chernoff Bounds. We use the form of these bounds derived by Raghavan [R]. Suppose that X is a binomial random variable with parameters n and p, i.e., it is the number of successes in n independent trials each with probability of success p. Then:

LEMMA 19. *Let $\mu = np$. Then, for any $\delta \in (0, 1]$,*

$$\Pr[X < (1-\delta)\mu] < \left[\frac{\exp(-\delta)}{(1-\delta)^{(1-\delta)}}\right]^{\mu}, \tag{9}$$

and, for any $\delta > 0$,

$$\Pr[X > (1+\delta)\mu] < \left[\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right]^{\mu}. \tag{10}$$

References

[C] B. Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2:637–666, 1989.

[CGL] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[F1] M. L. Fredman. A lower bound on the complexity of orthogonal range queries. *J. Assoc. Comput. Mach.*, 28:696–705, 1981.

[F2] M. L. Fredman. The complexity of maintaining an array and its partial sums. *J. Assoc. Comput. Mach.*, 29:250–260, 1982.

[M] K. Mehlhorn. *Data Structures and Algorithms*, Vol. III. Springer-Verlag, Berlin, 1984.

[N] J. von Neumann. Zur Theorie der Gesellschaftspiele. *Math. Ann.*, 100:295–320, 1928.

[R] P. Raghavan. Lecture Notes in Randomized Algorithms. Technical Report RC 15340, IBM, December 1989.

[S] J. Spencer. *Ten Lectures on the Probabilistic Method.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[Ya1] A. C. Yao. Probabilistic computations: towards a unified measure of complexity. *Proc. 18th IEEE FOCS*, pp. 222–227, 1977.

[Ya2] A. C. Yao. On the complexity of maintaining partial sums. *SIAM J. Comput.*, 14:277–288, 1985.

[Ye] D. Yellin. Data structures for set equality-testing. *Proc. 3rd Annual ACM–SIAM SODA*, pp. 386–392, 1992.