Lower bounds for Union-Split-Find related problems on random access machines *

Peter Bro Miltersen

Aarhus University
and
University of Warwick

Abstract

We prove $\Omega(\sqrt{\log\log n})$ lower bounds on the random access machine complexity of several dynamic, partially dynamic and static data structure problems, including the union-split-find problem, dynamic prefix problems and one-dimensional range query problems. The proof techniques include a general technique using perfect hashing for reducing static data structure problems (with a restriction of the size of the structure) into partially dynamic data structure problems (with no such restriction), thus providing a way to transfer lower bounds. We use a generalization of a method due to Ajtai for proving the lower bounds on the static problems, but describe the proof in terms of communication complexity, revealing a striking similarity to the proof used by Karchmer and Wigderson for proving lower bounds on the monotone circuit depth of connectivity.

1 Introduction and summary of results

In this paper we give lower bounds for the complexity of implementing several dynamic and static data structure problems, listed below.

The Union-Split-Find problem

The *Union-Split-Find* problem (on intervals) is the task of implementing a data type
UNION-SPLIT-FIND(n) containing a set $S \subseteq [n] = \{1, \ldots, n\}$, initially empty, with the following operations:

- For each $i \in [n]$ an operation $\texttt{union}_i$. It removes i from S.

*This work was supported by a grant from the Danish Natural Science Research Council. It was partially supported by the ESPRIT II Basic Research Actions Program of the European Community under contract No. 7141 (project ALCOM II).

- For each $i \in [n]$, an operation $\texttt{split}_i$. It inserts i into S.
- For each $i \in [n]$, an operation $\texttt{find}_i$. It returns the largest element of S which is smaller than or equal to i if such an element exists, otherwise 0 is returned.

The reason for the names union, split, and find is the following: We can consider the data type as maintaining the set of intervals

$$S' = \{[x;y) | x \in S \cup \{0\},\ y \in S \cup \{n+1\},\ (x;y) \cap S = \emptyset\}$$

where $[x;y) = \{x, x+1, \ldots, y-1\}$ and $(x;y) = \{x+1, x+2, \ldots, y-1\}$. With this interpretation, a `union` operations corresponds to joining two consecutive intervals, a `split` operation corresponds to splitting an interval in two, and the $\texttt{find}_j$ operation finds the unique interval I for which $j \in I$ and returns $\min(I)$.

The *Split-Find* problem SPLIT-FIND(n) is defined in the same way, except that $\texttt{union}_i$ operations are not allowed.

In this paper, we consider implementing data types in the *cell probe* or *decision assignment tree* model, studied in several papers [2, 4, 8, 9, 10, 12, 18, 19, 23]. In this model, the complexity of a computation is the number of cells accessed in the random access memory containing the data structure during the computation, while the computation itself is for free. The number of bits b in a cell is a parameter of the model. For dynamic problem, like the Union-Split-Find problem, there is no restriction on the number of cells in the memory.

Formally, the model is as follows: In an implementation of the data type we assign to each operation a decision assignment tree, i.e. a rooted tree containing *read* nodes and *write* nodes. When performing an operation we proceed from the root of its tree to one of the leaves. The read nodes are labeled with a location of the random access memory. Each has 2^b sons, one for each possible content of the memory location. The write nodes which are unary are labeled with a memory location and a value between 0 and $2^b - 1$. When such a node is encountered, the value is written in the memory location. If the operation is to return an answer, this is found in the leaf finally encountered. The complexity of an operation in an implementation is the depth of its corresponding tree, i.e. we consider worst case, as opposed to amortized complexity.

We prove the following result:

Theorem 1 *For any $\epsilon > 0$, there is $\delta > 0$ so that the following holds for sufficiently large n: For any implementation of* SPLIT-FIND(n) *with cell size $b \leq 2^{(\log n)^{1-\epsilon}}$, if the complexities of all the $\texttt{split}_i$ operations are less than $2^{(\log n)^{1-\epsilon}}$ then the complexity of one of the $\texttt{find}_i$ operations is at least $\delta\sqrt{\log\log n}$.*

Since $O(\log n) \leq 2^{(\log n)^{1-\epsilon}}$, the lower bound is also valid as a lower bound for the complexity of implementing the data type on a random access machine where each machine word stores a polynomial sized integer (a *random access computer* [5]), no matter what instruction set is used.

Actually, $b = 2^{(\log n)^{1-\epsilon}}$ is so large (exponentially larger than polylogarithmic) that the lower bound also becomes relevant as a lower bound for the complexity of implementing the data type on *secondary storage*, where data must be transferred to and from this secondary storage in large chunks, the number of such transferals being the *I/O-complexity* of the computation [1].

Previously, the following bounds on the complexity of Union-Split-Find and Split-Find were known: UNION-SPLIT-FIND(n) can be implemented on a random access computer so that all operations have worst case complexity $O(\log\log n)$. Furthermore, if we allow the complexities of `union` and `split` operations to be n^ϵ, then `find` can be performed in constant time. This is due to Van Emde Boas, Kaas and Zijlstra [21]. Imai and Asano [13] prove that SPLIT-FIND(n) can be implemented on a random access computer so that the *amortized* complexity of the operations is $O(1)$ (i.e. for any m, m operations can be performed in time $O(m)$).

On the *pointer machine* model, some more bounds were known. The pointer machine model is in general incomparable to the decision assignment tree model, unless it is restricted in some natural way, for instance by requiring the data structure to be polynomial size, in which case it is strictly weaker than even a random access computer. The statement of the problem must be modified slightly so that it becomes meaningful for pointer machines, see Mehlhorn, Näher and Alt [17]. The original and the modified problem are equivalent for random access computers. In that case, Mehlhorn and Näher [16] show that UNION-SPLIT-FIND(n) can be implemented so that all operations have worst case complexity $O(\log\log n)$. Mehlhorn, Näher and Alt [17] show that any implementation on a pointer machine has amortized complexity $\Omega(\log\log n)$. La Poutré [15] shows that any implementation of SPLIT-FIND(n) on a pointer machine has worst case complexity $\Omega(\log\log n)$.

Dynamic prefix problems

Dynamic prefix problems in the decision assignment tree model were considered by Fredman and Saks [12] and Frandsen, Miltersen and Skyum [8]. Let M be a fixed finite monoid, i.e. a finite set equipped with an associative binary operation $\circ$ and containing an identity element 1. The dynamic prefix problem for M is the task of implementing a data type M-PREFIX(n) containing a sequence $x = (x_1, \ldots, x_n) \in M^n$, initially $(1, \ldots, 1)$, with two kinds of operations.

- For each $i \in [n]$ and $a \in M$ an operation $\texttt{change}_{i,a}$. This operation changes x_i to a.
- For each $i \in [n]$ an operation $\texttt{prefix}_i$. This operation returns the product $x_1 \circ x_2 \circ \cdots \circ x_i$.

Previously known bounds on the complexity of M-PREFIX(n) are

- If M contains a group as a subset, then the cell size $b = \log^{O(1)} n$ decision assignment tree complexity of M-PREFIX(n) is $\Theta(\log n/\log\log n)$. This result is due to Fredman and Saks [12].
- If M does *not* contain a group, then M-PREFIX(n) can be implemented on a random access computer with all operations having worst case complexity $O(\log\log n)$. This result is due to Frandsen, Miltersen and Skyum [8].

In this paper, we show:

Theorem 2 *If M is a monoid so that for all $x \in M$ there is $y \in M$ so that $x \neq x \circ y$ then the cell size $b = 2^{(\log n)^{1-\epsilon}}$ decision assignment tree complexity of M-PREFIX(n) is $\Omega(\sqrt{\log\log n})$.*

For instance, the bound holds if M is the group free *carry-look-ahead* monoid $M = \{1, S, C\}$ with $x \circ y = y$ for $y \neq 1$.

One-dimensional range query problems

Consider the following discrete version of the range query problems of computational geometry: Let $r \geq 2$ be a fixed integer. Given a data set $M \subseteq [n]$, organize a *static* data structure using at most $|M|^{O(1)}$ cells, so that for any $i, j \in [n], i \leq j$, the *counting range query* "Is $|M \cap [i;j]| \bmod r = 0$?" can be answered efficiently, where $[i;j] = \{i, i+1, \ldots, j-1, j\}$.

The cell probe model for static problems is slightly different from the one for dynamic problems. Note that a bound s on the size of the data structure is essential, in order to get a non-trivial problem, otherwise a data set could be encoded using n cells, each giving the answer to a query. Formally, an implementation of the problem with cell size b and size bound s consists of the following: To each data set we assign a memory image in $[2^b]^s$, and to each query we assign a decision assignment tree without any assignment nodes (i.e. a decision tree). In this paper we show:

Theorem 3 *In any implementation of the range query problem with cell size $2^{(\log n)^{1-\epsilon}}$ and size bound $|M|^{O(1)}$, for some data set, some query has to access at least $\Omega(\sqrt{\log\log n})$ cells.*

Previously known bounds: Willard [22] showed that there is a storage scheme using $O(|M|)$ cells of size $O(\log n)$ such than any query can be answered in time $O(\log\log n)$.

Ajtai [2] showed a non-constant lower bound for cell size $O(\log n)$ and the set of queries "What is $|M \cap [i;j]|$?". In his paper, Ajtai states that a $\Omega(\log\log n)$ lower bound is achievable using his technique, but he has later reduced his claim to only $\Omega(\sqrt{\log\log n})$ [3], so Theorem 3 is a strict generalization of his lower bound.

Outline of paper and proof techniques

In Section 2 below, we show how all the lower bounds in the Introduction follows from a lower bound for a communication game. We first show how to reduce communication problems to static data structure problems. This reduction is really a rather trivial observation, but we find it illuminating and useful. We then show how to reduce static problems to (partially) dynamic problems using constant time static dictionaries. This technique is implicitly present in the paper of Willard [22] and was used in a weaker form in [18]. In Section 3, we prove the lower bound for the communication game. The proof technique is a generalization of the one used by Ajtai for proving the lower bound on the static problem mentioned above, but in our opinion, the communication complexity framework makes its structure much clearer. Furthermore, readers familiar with the proof used by Karchmer and Wigderson for proving lower bounds on the monotone circuit depth of connectivity [14] will appreciate the striking similarity between the two proofs.

2 Reductions

In this section we show how all the results stated in the introduction follow from Theorem 4 below. We need some definitions.

Let Σ be a finite alphabet and let $L \subseteq \Sigma^*$ be a language. We say that L is *indecisive* if for all $w \in L$ there is $u \in \Sigma^*$ so that $w \cdot u \notin L$ and for all $w \notin L$ there is $u \in \Sigma^*$ so that $w \cdot u \in L$.

By a Σ-*coloured* subset of $[n]$ we mean a pair (H, ϕ) where H is a subset of $[n]$ and ϕ is a map from H to Σ. For $i \in H$, we say that $\phi(i)$ is the colour of i in H. When no confusion is possible, we will just say coloured instead of Σ-coloured. Also, we will be a bit sloppy in notation and consider H as an ordinary subset with some additional structure, e.g. we shall write H instead of (H, ϕ) and allow expressions such as $H \cap J$ where J is an ordinary subset of $[n]$. Given a coloured subset H with elements $i_1 < i_2 < \cdots < i_m$ we denote by $\phi(H)$ the string $\phi(i_1)\phi(i_2)\ldots\phi(i_m) \in \Sigma^*$.

Let $L \subseteq \Sigma^*$ be a language. Consider the following communication game.

- Alice is given $x \in [n]$
- Bob is given a Σ-coloured subset H of $[n]$ of size at most m.

Their task is to determine if $\phi([x] \cap H) \in L$ by communicating with each other. The communication is structured in the following way: Alice sends her messages in blocks containing a bits, Bob sends his messages in blocks containing b bits and the communication is strictly alternating, with Alice sending the first message. The complexity is the number of rounds required before they agree on the correct answer.

Theorem 4 *Let L is an indecisive, regular language, and let $\epsilon > 0$. For sufficiently large n, consider the above game with parameters $\log\log n \le a \le (\log n)^{1-\epsilon}$, $b \le 2^{\frac{a}{4}(\log\log n)^{-1/2}}$ and $m \ge 2^{\epsilon a}$. Then every correct protocol uses at least $\frac{\epsilon}{4}\sqrt{\log\log n}$ rounds.*

We now show how all the results mentioned in the introduction can be shown using Theorem 4. Theorem 4 itself is shown in the next section.

Theorem 5 *Let $L \subseteq \Sigma^*$ be an indecisive, regular language and let $c \ge 1$ and $\epsilon > 0$ be fixed. Consider the following static data structure problem with parameter n: Given a Σ-coloured subset H of $[n]$ with $|H| \le m = 2^{(\log n)^{1-\epsilon}}$, store H in a static data structure using m^c cells, each containing $b \le 2^{(\log n)^{1-2\epsilon}}$ bits, so that for any $j \in [n]$, the query "Is $\phi([j] \cap H)$ in L?" can be answered. Then, in any implementation, for some data set H, some query has to access at least $\Omega(\sqrt{\log\log n})$ cells in the encoding of H.*

Proof Suppose a scheme with the stated size bounds and a query time $o(\sqrt{\log\log n})$ exists. We now construct a protocol for the communication game with parameters $\epsilon' = \min(\frac{1}{2c}, \frac{\epsilon}{2})$, $n' = n$, $a' = \lceil c \log m \rceil + 1$, $b' = b$ and $m' = m$ using $o(\sqrt{\log\log n'})$ rounds, obtaining a contradiction with Theorem 4.

Suppose Alice is given $j \in [n']$ and Bob is given a coloured subset H of size at most m'. Bob computes the data structure (memory image) corresponding to H, but does not send anything yet.

Then Alice simulates the decision tree corresponding to the query "Is $\phi([j] \cap H) \in L$?" by sending Bob requests for the cells she wants to read in the data structure. A request is coded as a 0 followed by the binary notation of the address of the cell, i.e. it can be described in a' bits. Bob sends the content of the cell in question back using b' bits. This is repeated until a leaf in the decision tree is reached and Alice knows the answer. She informs Bob of this fact by sending him a 1 followed by the answer.

$\square$

Proof (of theorem 3). Put $\Sigma = \{\mathtt{a}\}$ and let $L = \{\mathtt{a}^n | n \bmod r = 0\}$. Then L is indecisive and regular, and the lower bound follows from Theorem 5.

$\square$

We now turn to dynamic problems. Given a string $x \in (\Sigma \cup \{\perp\})^*$, we denote by $x/\perp$ the string in Σ^* obtained by removing all occurrences of $\perp$ from x. For instance, $\mathtt{ab}\perp\mathtt{b}\perp\mathtt{a}/\perp = \mathtt{abba}$.

Let $L \subseteq \Sigma^*$ be a language. Let $L^\perp$-PREFIX(n) be the finite data type containing a string $x_1 x_2 \ldots x_n$ in $(\Sigma \cup \{\perp\})^n$, initially $\perp^n$, with two kinds of operations:

- For each $i \in [n]$ and $a \in \Sigma$ an operation $\mathtt{change}_{i,a}$. This operation can only be used when $x_i = \perp$. It changes x_i from $\perp$ to a.
- For each $i \in [n]$ an operation $\mathtt{prefix}_i$. This operation returns `true` if $x_1 x_2 \ldots x_i/\perp \in L$ and `false` otherwise.

We need the following fact, due to Fredman, Komlós and Szemerédi [11]. By an *n-dictionary*, we mean a subset $S \subseteq [n]$ and a map $d : S \to [n]$.

Fact 6 (Fredman, Komlós, Szemerédi) *There is a scheme for storing n-dictionaries (S, d) using $O(|S|)$ cells, each containing* $\log n$ *bits, so that for each j, the query "Is j in S, and if so, what is $d(j)$?" can be answered in $O(1)$ probes.*

Theorem 7 *Let L be an indecisive, regular language and let $\epsilon > 0$. In any cell size $b \leq 2^{(\log n)^{1-\epsilon}}$ implementation of $L^\perp$-PREFIX, if the complexities of all of the* $\mathtt{change}_{i,a}$ *operations are $2^{(\log n)^{1-\epsilon}}$, then the complexity of one of the* $\mathtt{prefix}_i$ *operations is $\Omega(\sqrt{\log\log n})$.*

Proof Suppose an implementation I with the complexity of the $\mathtt{change}_{i,a}$ operations being $t = 2^{(\log n)^{1-\epsilon}}$ and the complexity of the $\mathtt{prefix}_i$ operations being $o(\sqrt{\log\log n})$ exists. Since the depth of the deepest decision assignment tree for any operation is t and each node in the tree has at most 2^b sons, the total number of nodes in all the trees put together is at most $(|\Sigma| + 1)n2^{bt}$, which is less than $s = 2^{2^{(\log n)^{1-\epsilon/2}}}$ for sufficiently large n. This also bounds the number of cells in the data structure, i.e. we can identify the set of possible states of the random access memory (the memory images) of I with $[2^b]^s$.

We now describe scheme for constructing static data structures storing Σ-coloured subsets of $[n]$ of size less than or equal to $m = \lfloor 2^{(\log n)^{1-\epsilon/4}} \rfloor$, in order to get a contradiction with Theorem 5.

Let (H, ϕ) be such a set with $H = \{i_1, i_2, \ldots, i_m\}$. Starting from the initial memory image $M_0 \in [2^b]^s$ of I, we perform the sequence of operations

$$\mathtt{change}_{i_1,\phi(i_1)}, \mathtt{change}_{i_2,\phi(i_2)}, \ldots, \mathtt{change}_{i_m,\phi(i_m)}.$$

Let the resulting memory image be $M_H \in [2^b]^s$. By the assumption on the complexity of the change operations, M_0 and M_H differ on at most $g = tm$ indices. Let these indices be $G = \{a_1, a_2, \ldots, a_g\}$ and let the new content of a_i be d_i. The a_i's and d_i's are all smaller than s.

Using Fact 6, we store H as a dictionary containing G with $d(a_i) = d_i$. The data structure uses $O(g) \leq m^2$ cells, each containing $\log s = 2^{(\log n)^{1-\epsilon/2}}$ bits. We now show that with this data structure, any query of the form "Is $\phi([j] \cap H)$ in L ?" can be answered more efficiently than Theorem 5 permits. We would like to run the operation $\mathtt{prefix}_j$ on M_H since this would be the answer. However, instead of M_H, we only have the data structure containing the difference between M_0 and M_H to our disposal. To simulate running $\mathtt{prefix}_j$ in these circumstances we do the following: Each time we want to write the value d in cell a, we make a private note that the new content of cell a is d. If we want to read a cell a, we first see if this cell appears in our private notes. If it does not, we look it up in the dictionary. If it does not appear there, we know that its content is the same as it was in M_0. Changing or examining our private notes does not probe the data structure, it can be "hardwired" into the decision tree of the query, and so can the necessary knowledge about M_0. Each lookup requires only a constant number of probes. Thus, the number of probes required is $o((\log \log n)^{1/2})$, a contradiction.

□

Proof (of Theorem 1). Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and let $L = (\mathtt{a} \cup \mathtt{b})^* \cdot \mathtt{a}$. Then L is regular and indecisive. Given an implementation of SPLIT-FIND(n), we can implement $L^{\perp}$-PREFIX(n) with the complexity of the $\mathtt{change}_j$ operation being that of the $\mathtt{split}_j$ operation plus a constant amount of overhead and the complexity of the $\mathtt{prefix}_j$ operation being that of the $\mathtt{find}_j$ operation plus a constant amount of overhead.

□

Proof (of Theorem 2). Let $G = (M, E)$ be the Cayley graph of M, i.e. the directed graph defined by $E = \{(x, y) | \exists z : x \circ z = y\}$. Let $M_1, M_2, \ldots, M_j$ be the strongly connected components of G containing at least 2 elements (By the assumption on M, $j \geq 1$). Pick an element m_i from each M_i. Now let $L \subseteq (M - \{1\})^*$ be the language defined by $x_1 x_2 \ldots x_n \in L$ iff $x_1 \circ x_2 \circ \cdots \circ x_n \in \{m_1, \ldots, m_j\}$. Then L is an indecisive, regular language. However, if M-PREFIX(n) can be solved in time $o(\sqrt{\log \log n})$ then so can $L^{\perp}$-PREFIX(n).

□

3 Lower bound for the communication game

In this section, we prove Theorem 4.

We first make a technical point: We say that a language $L \subseteq \Sigma^*$ is *strongly indecisive* if there is a constant $q_L > 0$ so that for all $x \in \Sigma^*$, there is a string $y \in \Sigma^{q_L}$ and a string $z \in \Sigma^{q_L}$ so that $x \cdot y \in L$ but $x \cdot z \notin L$. We only have to prove Theorem 4 for regular, strongly indecisive languages: If $L \subseteq \Sigma^*$ is regular and indecisive, then $L^\perp \subseteq (\Sigma \cup \{\perp\})^*$ defined by $x \in L^\perp$ iff $x/\perp \in L$ is regular and strongly indecisive. If a protocol P with the parameters in the Theorem exists for L then a protocol P' with the same parameters exists for $L^\perp$: When Alice is given x and Bob is given H, Bob computes $\tilde{H} = \{y \in H | \phi(y) \neq \perp\}$ and they simulate P on x and $\tilde{H}$.

So in the following, we let L be a strongly indecisive, regular language with q_L as defined above. Furthermore, let M_L be the minimal, deterministic, finite automaton recognizing L and let s_L be the number of states in M_L.

Now some useful definitions: For two finite sets S and T with $S \subseteq T$ and $T \neq \emptyset$, the *density* $\mu^T(S)$ of S in T is $|S|/|T|$. We shall use the following simple fact on densities a number of times:

Fact 8 *If $S \subseteq T$ and T is the disjoint union of $T_1, T_2, \ldots, T_l$, then, for some $i \in [l]$, $\mu^{T_i}(S \cap T_i) \geq \mu^T(S)$.*

If $I = [\min_I; \max_I]$ and $J = [\min_J; \max_J]$ are two intervals of integers with $|J| = r|I|$ for some integer r, and S is a subset of I, we define the *expansion* of S into J to be the J-subset $S\langle I \to J\rangle = \{\min_J + r(i - \min_I) | i \in S\}$. If $T = S\langle J \to I\rangle$, we also say that S is the *contraction* of T into I, and write $S = T\langle J \to I\rangle$. If S is coloured, then so is $S\langle I \to J\rangle$, with the colour of $\min_J + r(i - \min_I)$ being the colour of i in S.

Given n such that $\log\log n$ is an integer, and $i \leq \log\log n$ define $[n]^{(i)}$ to be the following set of intervals:

$$[n]^{(i)} = \{[1; n^{2^{-i}}], [n^{2^{-i}} + 1; 2n^{2^{-i}}], \ldots, [n - n^{2^{-i}} + 1; n]\}$$

For an interval $I \in [n]^{(i)}$, we note that I is the disjoint union of intervals in $[n]^{(i+1)}$. We denote the set of these intervals by I'.

Assume now that Theorem 4 does *not* hold for $n = N$ for some large integer N. We can without loss of generality assume that $\log\log N$ is an integer.

Following Ajtai, we now define a family of *test sets* T_n^i for Bob. T_n^i is defined when $\log\log n$ is an integer greater than or equal to $(1 - \frac{\epsilon}{2})\log\log N + 2i(\log\log N)^{1/2}$. It consists of coloured subset of $[n]$.

- T_n^0 is the set of coloured subsets of $[n]$ containing exactly $\lfloor 2^{\epsilon a/2} \rfloor$ elements.
- T_n^i contains coloured subsets of $[n]$ constructed in the following way: Let $u = \lfloor 2(\log\log N)^{1/2} - 1 \rfloor$ and $r = \lfloor 2^{2a(\log\log N)^{-1/2}} u^{-1} \rfloor$. For each $j \in [u]$, pick a set of r intervals Int^j from $[n]^{(j)}$, such that for any $j_1 \neq j_2$ and $I \in \mathrm{Int}^{j_1}$, $J \in \mathrm{Int}^{j_2}$, I and J are disjoint. Let $\mathrm{Int} = \cup_j \mathrm{Int}^j$. Now, for each $I \in \mathrm{Int}$ pick a test set $\mathrm{Set}_I \in T^{i-1}_{|I|^{1/2}}$ and let

$$H = \bigcup_{I \in \mathrm{Int}} \mathrm{Set}_I \langle [|I|^{1/2}] \to I \rangle$$

Note that we might conceivably get the same test set by two different constructions. However, for counting purposes, we shall consider test sets constructed in different ways as different, i.e. T_n^i really consists of test sets labeled with their construction. With this in mind, we can also, for a test set H, define $\mathrm{Int}^j(H)$ to be the value of Int^j used in the construction of H, and define $\mathrm{Int}(H)$ and $\mathrm{Set}_I(H)$ for $I \in \mathrm{Int}(H)$ similarly.

Let $S(n, i)$ be the following statement:

- There is $A \subseteq [n]$ and $B \subseteq T_n^i$, with $\mu^{[n]}(A) \geq \rho = 2^{-a(\log\log N)^{-1/2}}$ and $\mu^{T_n^i}(B) \geq \delta = 2^{-b-1}q_L^{-1}$ and a protocol P with i rounds and a string α so that when Alice is given $x \in A$ and Bob is given $H \in B$, the protocol correctly decides if $\alpha \cdot \phi([x] \cap H) \in L$.

Note that the sets in in T_n^i have size less than $2^{a(\epsilon/2+2i(\log\log N)^{-1/2})}$, so, assuming that Theorem 4 does not hold for N, we would have $S(N, \lfloor\frac{\epsilon}{4}(\log\log N)^{1/2}\rfloor)$. With the help of the following two main lemmas, we arrive at a contradiction, and are done: Applying Lemma 9 repeatedly, we get $S(n, 0)$ for $n \geq 2^{2^{(1-\epsilon/2)\log\log N}}$, in contradiction with Lemma 10.

Lemma 9 *$S(n, i)$ implies* $\exists n' : (\log\log n' \geq \log\log n - u - 1) \wedge S(n', i-1)$

Lemma 10 *$S(n, 0)$ does not hold for n larger than c, where c is a constant dependent on L and ϵ only.*

The following lemma, which can be proved using standard tail estimate techniques (Chernoff bounds) is useful in the proof of both main lemmas.

Lemma 11 *Let $S \subseteq [n]$ with $\mu^{[n]}(S) = s$. Let R be a* random *subset of $[n]$ of size l. Then*

$$\Pr(|S \cap R| \leq \frac{sl}{10}) \leq 2^{-sl/2}$$

Proof (of Lemma 10). Consider the following statement:

- Let R be a random, randomly coloured subset of $[n]$ of size $k = \lfloor 2^{\epsilon a/2} \rfloor$. Let $\alpha \in \Sigma^*$. Let Z_0 be the event that $\forall x \in A : \alpha \cdot \phi([x] \cap R) \in L$ and Let Z_1 be the event that $\forall x \in A : \alpha \cdot \phi([x] \cap R) \notin L$. Then $\Pr(Z_0)$ and $\Pr(Z_1)$ are both $2^{-O(\rho k/\log k)}$, the constant in the O depending on L and ϵ only.

Since $\delta = 2^{-b-1}/q_L$ is not $2^{-O(\rho k/\log k)}$, it immediately follows from the statement that zero rounds are not sufficient for the communication game, so we only have to prove the statement.

We show the bound on $\Pr(Z_0)$, Z_1 is dealt with in the same way. Select a subset $A' = \{x_1, x_2, \ldots, x_s\}$ of A with $x_1 < x_2 < \cdots < x_s$ where $x_{j+1} - x_j \geq 10nq_L/k$ and $s \geq \lfloor \frac{\rho k}{10 q_L} \rfloor$.

Now, for $j \in [s-1]$, let E_j be the event that $|R \cap [x_j + 1; x_{j+1}]| < q_L$. According to lemma 11, $\Pr(E_j) \leq 2^{-10q_L/2} \leq 2^{-5}$. Furthermore, the events E_j are negatively correlated, i.e. for any family $i_1, i_2, \ldots, i_l$ and $j \notin \{i_1, i_2, \ldots, i_l\}$, $\Pr(E_j | E_{i_1} \cap E_{i_2} \cap \ldots \cap E_{i_l}) \leq \Pr(E_j)$.

Let $h = \lfloor \frac{s}{\log s} \rfloor$ and let C be the event that $|\{j \in [s-1] | E_j\}| > s - 1 - h$. Then $\Pr(C) \leq \sum_{i=0}^{h-1} \binom{s-1}{i} 2^{-5(s-1-i)} \leq \binom{s-1}{h} 2^{-5(s-1-h)} \leq (s-1)^h 2^{-5(s-1-h)} \leq 2^{-s} = 2^{-O(k)}$.

We now estimate $\Pr(Z_0|\neg C)$. If C does not hold, we can find $\{i_1,\ldots,i_h\}$ so that $R\cap[x_{i_j}+1,\ldots,x_{i_j+1}]\geq q_L$. Since L is strongly indecisive, we then have that $\Pr(Z_0|\neg C)\leq(1-|\Sigma|^{-q_L})^h=2^{-O(\rho\frac{k}{\log k})}$. Since $\Pr(Z_0)\leq\Pr(Z_0|\neg C)+\Pr(C)$, we are done.

□

We need the following lemma by Ajtai [2] for the proof of Lemma 9.

Lemma 12 (Ajtai) *Let $A\subseteq[n]$ have density $\mu^{[n]}(A)\geq\rho$ and let Q be any equivalence relation on A with d equivalence classes. Let $I\in[n]^{(k)}$ for some $k\geq 1$. If there is a class C in Q so that $\mu^{I'}(\{J\in I'|J\cap C\neq\emptyset\})\geq(\frac{\rho}{2d})^{\frac{1}{u-1}}$, we say that I is* dense. *Let $D_k\subseteq[n]^{(k)}$ be the set of dense intervals in $[n]^{(k)}$. Then there is $k\leq u$ for which $\mu^{[n]^{(k)}}(D_k)\geq\frac{\rho}{2u}$.*

Proof (of Lemma 9). Let Q be the equivalence relation on A determined by Alice's first message to Bob. Q has at most 2^a equivalence classes. According to the density lemma, pick an integer $k\leq u$, so that the density of dense intervals in $[n]^{(k)}$ is at least $\frac{\rho}{2u}$. Let $D=D_k$ be the set of dense intervals in $[n]^{(k)}$. Let $n'=n^{2^{-k-1}}$.

According to Fact 8, we can find $K_1,K_2,\ldots,K_{k-1},K_{k+1},\ldots,K_u$ with $K_j\subseteq[n]^{(j)}$, so that if $T_1=\{H\in T_n^i|\forall j\neq k:\mathrm{Int}^j(H)=K_j\}$ and $B_1=B\cap T_1$ then $\mu^{T_1}(B_1)\geq\mu^{T_n^i}(B)\geq\delta$.

Let $T_2=\{H\in T_1|\mathrm{Int}^k(H)\cap D\geq 1-\log\delta\}$ and let $B_2=B_1\cap T_2$. Note that since Int^j for $j\neq k$ is fixed, the first step in finding a random element in T_2 is letting Int^k be a random subset of size r of those intervals disjoint from the intervals in $\cap_{j\neq k}\mathrm{Int}^j$. The dense intervals have density at least $\frac{\rho}{2u}-\frac{ru}{|[n]^{(k)}|}\geq\frac{\rho}{4u}$ in those intervals. By lemma 11, we have that $\mu^{T_1}(T_2)\geq 1-\frac{\delta}{2}$, so $\mu^{T_2}(B_2)\geq\mu^{T_1}(B_2)\geq\mu^{T_1}(B_1)-(1-\mu^{T_1}(T_2))\geq\frac{\delta}{2}$.

According to Fact 8, there is a set E with $|E\cap D|\geq 1-\log\delta$ so that if $T_3=\{H\in T_2|\mathrm{Int}^k(H)=E\}$ and $B_3=B_2\cap T_2$ then $\mu^{T_3}(B_3)\geq\mu^{T_2}(B_2)\geq\frac{\delta}{2}$

According to Fact 8, we can find a set $H_I\subseteq T_{n'}^{i-1}$ for each $I\in E-D$ so that if $T_4=\{H\in T_3|\forall I\in E-D:\mathrm{Set}_I(H)=H_I\}$ and $B_4=B_3\cap T_4$ then $\mu^{T_4}(B_4)\geq\mu^{T_3}(B_3)\geq\frac{\delta}{2}$

Note that, by construction, T_4 is in one-to-one correspondence with $\prod_{I\in E\cap D}T_{n'}^{i-1}$ while B_4 is in one-to-one correspondence with a *subset* of $\prod_{I\in E\cap D}B_I$ where $B_I=\{\mathrm{Set}_I(H)|H\in B_4\}$. Thus $\mu^{T_4}(B_4)\leq\prod_{I\in E\cap D}\mu^{T_{n'}^{i-1}}(B_I)$ so we can find an interval $J\in E\cap D$, so that $\mu^{T_{n'}^{i-1}}(B_J)\geq(\frac{\delta}{2})^{1/|E\cap D|}\geq(\frac{\delta}{2})^{1/(1-\log\delta)}=\frac{1}{2}$.

For each $H\in B_J$, fix a set $\tilde{H}$ in B_4 so that $\mathrm{Set}_J(\tilde{H})=H$. Let $\alpha_H=\alpha\cdot\phi(\tilde{H}\cap[\min(J)-1])$. Let q_H be the state in the finite automaton M_L one reaches if one starts in the initial state of the automaton and is given α_H as input. We can find q so that if $B_J'=\{H\in B_J|q_H=q\}$ then $\mu^{T_{n'}^{i-1}}(B_J')\geq\frac{1}{2s_L}$. Let α' be a string so that one reaches q from the initial state when given α' as input.

Since $J\in D$ there is a class C in Q so that $\mu^{J'}(\{I\in J'|I\cap C\neq\emptyset\})\geq(\frac{\rho}{2^{a+1}})^{\frac{1}{u-1}}\geq\rho$. To the class C corresponds a first message $\gamma\in[2^a]$ of Alice.

For a possible message $\beta \in [2^b]$ of Bob, let B_β be those H in B'_J for which Bob, when given $\tilde{H}$ as input, replies β when Alice's first message is γ. We can fix β, so that $\mu^{T^{i-1}_{n'}}(B_\beta) \geq \frac{1}{2s_L 2^b} = \delta$. Put $B' = B_\beta$.

Put $A' = \{\min(I) | I \in J', I \cap C \neq \emptyset\}\langle J \to [n']\rangle$. For $x \in A$, fix $\tilde{x}$ in C so that $\{x\}\langle [n'] \to J\rangle = \{\min(I)\}$ where $\tilde{x} \in I \in J'$.

Let the protocol P' do the following: If Alice is given $x \in A'$ and Bob is given $H \in B'$, then Alice finds $\tilde{x}$, Bob finds $\tilde{H}$ and they simulate P from the second round on, pretending that in the first round, Alice sent γ to Bob and Bob sent β to Alice.

Since $\alpha' \cdot \phi([x] \cap H) \in L$ if and only if $\alpha \cdot \phi([\tilde{x}] \cap \tilde{H}) \in L$, P' correctly decides if $\alpha' \cdot \phi([x] \cap H) \in L$.

$\square$

4 Conclusion and open problems

We have given lower bounds for several static, partially dynamic and dynamic data structure problems. All lower bounds are of the form $\Omega(\sqrt{\log\log n})$ while the corresponding best known upper bounds are all $O(\log\log n)$. An obvious open problem is to close the gaps.

Here are some nice problems, for which we have no non-constant lower bounds.

- The *prefix OR* problem M-PREFIX(n) where $M = (\{0,1\}, \vee)$, or, equivalently, the "Union-Split-First" problem, where one, in addition to the $\texttt{union}_i$ and $\texttt{split}_i$ operations have an operation $\texttt{first}_i$ which reveals if i is smaller than the smallest element in S.
- The dynamic *word* problem [8] for the carry-look-ahead monoid. In the word problem, the $\texttt{prefix}_i$ operations of the prefix problem is replaced by a single operation `product` returning $x_1 \circ x_2 \circ \cdots \circ x_n$.
- The *integer priority queue* problem of maintaining a set $S \subseteq [n]$ under $\texttt{insert}_i$ operations which inserts i into S and `deletemin` operations which removes the smallest element m from S and returns m.
- the *insert-delete-min* problem of maintaining a set $S \subseteq [n]$ under insertions, deletions and `min` queries, returning the smallest element of S. The three first problems reduce to this one, so this should be the easiest to show a lower bound for.

Note that the *partially* dynamic versions of these problems, where a change is never undone, are all easy, so we can not use the techniques of this paper to show lower bounds for these problems.

A static problem, for which we might get a lower bound using the techniques of this paper is the *existential* range query problem, where the query "Is $|M \cap [i;j]| \bmod r = 0$?" is replaced with "Is $M \cap [i;j] = \emptyset$?". As yet, we know no non-constant lower bound for this problem.

It would also be interesting to try using the communication complexity approach to get lower bounds larger than $\sqrt{\log\log n}$ for harder problems, for instance the static *two-dimensional* orthogonal range query problem of storing a set $M \subseteq [n] \times [n]$ so that counting range queries of the form "What is $|M \cap [i;j] \times [k;l]|$?" can be answered. Note that the communication approach can only give lower bounds smaller than $\log n$ so it can not be used to show good lower for bounds problems much harder than this, like *multi-dimensional* range query problems (unless there are surprising upper bounds. One should note that good lower bounds for similar problems are available in natural, weaker, structured, algebraic models of computation [6, 20, 24]).

References

[1] A. Aggarwal, J. Scott Vitter, The I/O complexity of sorting and related problems, in: *Proc. 14th International Colloquium on Automata, Languages and Programming* (1987) 467-478.

[2] M. Ajtai, A lower bound for finding predecessors in Yao's cell probe model, *Combinatorica* **8** (1988) 235-247.

[3] M. Ajtai, Personal communication.

[4] M. Ajtai, M. Fredman, J. Komlós, Hash functions for priority queues, in: *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science* (1983) 299-303.

[5] D. Angluin, L.G. Valiant, Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings, *J. Comput. System Sci.* **18** (1979) 155-193.

[6] B. Chazelle, Lower bounds for orthogonal range searching, II: The arithmetic model, *J. Assoc. Comp. Mach.* **37** (1990) 439-463.

[7] P. Elias, R.A. Flower, The complexity of some simple retrieval problems, *J. Assoc. Comp. Mach.* **22** (1975) 367-379.

[8] G.S. Frandsen, P.B. Miltersen, S. Skyum, Dynamic word problems, to appear in the proceedings of FOCS'93.

[9] M.L. Fredman, Observations on the complexity of generating quasi-Gray codes, *SIAM J. Comput.* **7** (1978) 134-146.

[10] M.L. Fredman, The complexity of maintaining an array and computing its partial sums, *J. Assoc. Comput. Mach.* **29** (1982) 250-260.

[11] M.L. Fredman, J. Komlós, E. Szemerédi, Storing a sparse table with $O(1)$ worst case access time, *J. Assoc. Comput. Mach.* **31** (1984) 538-544.

[12] M.L. Fredman, M.E. Saks, The cell probe complexity of dynamic data structures, in: *Proc. 21st Ann. ACM Symp. on Theory of Computing* (1989) 345-354.

[13] T. Imai, T. Asano, Dynamic segment intersection with applications, in: *Proc. 25th Ann. IEEE Symp. on Foundations of Computer Science* (1984) 393-402.

[14] M. Karchmer, A. Wigderson, Monotone circuits for connectivity require super-logarithmic depth, in: *Proc. 18th Ann. ACM Symp. on Theory of Computing* (1988) 539-550.

[15] J.A. La Poutré, Lower bounds for the Union-Find and the Split-Find problem on pointer machines, in: *Proc. 20th Ann. ACM Symp. on Theory of Computing* (1990) 34-44.

[16] K. Mehlhorn, S. Näher, Dynamic fractional cascading, *Algorithmica* **5** (1990).

[17] K. Mehlhorn, S. Näher, H. Alt, A lower bound on the complexity of the union-split-find problem, *SIAM J. Comput.* **17**(1988) 1093-1102.

[18] P.B. Miltersen, The bit probe complexity measure revisited, in: *Proc. 10th Symp. on Theoretical Aspects of Computer Science* (1993) 662-671.

[19] P.B. Miltersen, S. Sairam, J. Scott Vitter, R. Tamassia, Complexity models for incremental computation, submitted (a joint version of S. Sairam, J. Scott Vitter, R. Tamassia, A complexity theoretic approach to incremental computation, in: *Proc. 10th Ann. Symp. Theoretical Aspects of Computer Science* (1993) 640-649 and P.B. Miltersen, On-line reevaluation of functions, Aarhus University tech report PB-380).

[20] P.M. Vaidya, Space-time tradeoffs for orthogonal range queries, *SIAM J. Comput.* **18** (1989) 748-458.

[21] P. Van Emde Boas, R. Kaas, E. Zijlstra, Design and implementation of an efficient priority queue, *Math. Systems Theory* **10** (1977) 99-127.

[22] D.E. Willard, Log-logarithmic worst case range queries are possible in space $\Theta(n)$, *Inform. Process. Lett.* **17** (1983) 81-84.

[23] A.C. Yao, Should tables be sorted?, *J. Assoc. Comput. Mach.* **28** (1981) 615-628.

[24] A.C. Yao, On the complexity of maintaining partial sums, *SIAM J. Comput.* **14** (1985) 277-288.