Stronger 3SUM-Indexing Lower Bounds

Eldon Chung* Kasper Green Larsen†

Abstract

The 3SUM-Indexing problem was introduced as a data structure version of the 3SUM problem, with the goal of proving strong conditional lower bounds for static data structures via reductions. Ideally, the conjectured hardness of 3SUM-Indexing should be replaced by an unconditional lower bound. Unfortunately, we are far from proving this, with the strongest current lower bound being a logarithmic query time lower bound by Golovnev et al. from STOC'20. Moreover, their lower bound holds only for non-adaptive data structures and they explicitly asked for a lower bound for adaptive data structures. Our main contribution is precisely such a lower bound against adaptive data structures. As a secondary result, we also strengthen the non-adaptive lower bound of Golovnev et al. and prove strong lower bounds for 2-bit-probe non-adaptive 3SUM-Indexing data structures via a completely new approach that we find interesting in its own right.

1 Introduction

In the 3SUM Problem, we are given a set S of n group elements from an abelian group $(G, +)$ and the goal is to determine whether there is a triple $a, b, c \in S$ such that $a + b = c$. The 3SUM Problem was originally introduced by Gajentaan and Overmars [13] as a means of establishing hardness of geometric problems. Concretely, it was conjectured that 3SUM requires $\Omega(n^2)$ time when the underlying group is the set of reals and we use the Real-RAM computational model. By reductions, this conjecture implies similar lower bounds for a wealth of geometric problems, see e.g. [5, 27].

While originally being restricted mostly to geometric problems, the seminal work by Pătraşcu [23] showed that a suitable integer version of 3SUM (e.g. G is the integers modulo n^3), may be used to prove hardness of numerous fundamental algorithmic problems (see e.g. [18, 4, 1, 23]) in the more realistic word-RAM model. These lower bounds are based on the so-called 3SUM Conjecture, asserting that no $n^{2-\delta}$ time 3SUM algorithm exists for any constant $\delta > 0$. To date, the fastest 3SUM algorithm runs in time $O(n^2(\lg\lg n)^{O(1)}/\lg^2 n)$ [9], which is far from refuting the conjecture. The 3SUM Conjecture is now one of the pillars in fine-grained complexity and much effort has gone into understanding its implications for algorithm lower bounds.

Highly related to algorithm lower bounds, is lower bounds for data structures. While more progress has been made on proving unconditional lower bounds for data structures compared to algorithms, current state-of-the-art lower bounds are still only polylogarithmic [19, 22, 20]. This lack of progress motivates fine-grained conditional lower bounds also for data structures. The first approach in this direction, is via the Online Matrix-Vector Problem by Henzinger et al. [17]. Their framework yields polynomial conditional lower bounds for dynamic data structures via reductions from multiplication of a boolean matrix and a boolean vector, with addition replaced by OR and multiplication replaced by AND. However, their framework is inherently tied to *dynamic* data structure problems, where a data set is to be maintained under update operations. As a means to addressing *static* data structure problems, Goldstein, Kopelowitz, Lewenstein, and Porat in [14] introduced the 3SUM-Indexing Problem.

3SUM-Indexing. The 3SUM-Indexing problem was first defined by Demaine and Vadhan in an unpublished manuscript [10] and then by Goldstein, Kopelowitz, Lewenstein, and Porat in [14] and is as follows:

Definition 1 (3SUM-Indexing)**.** *Let $(G, +)$ be a finite abelian group. Preprocess two sets of group elements $A_1, A_2 \subseteq G$ each of size n into a data structure of S memory cells of w bits so that given any query group element z, deciding whether there exists $a_1 \in A_1$ and $a_2 \in A_2$ such that $a_1 + a_2 = z$ is done by accessing at most T memory cells.*

*echung.math@gmail.com. Center for Quantum Technologies, National University of Singapore.

†larsen@cs.au.dk. Aarhus University. Supported by Independent Research Fund Denmark (DFF) Sapere Aude Research Leader grant No 9064-00068B.

A number of hardness conjectures was provided together with the definition of the 3SUM-Indexing Problem. Combined with reductions, these conjectures allow establishment of conditional lower bounds for static data structures. To be consistent with the terminology used for unconditional data structure lower bounds, which are typically proved in the cell probe model [29], we refer to accessing a memory cell as *probing* the cell. The following conjectures were made regarding the hardness of 3SUM-Indexing:

Conjecture 1 ([14]). *Any data structure for 3SUM-Indexing with space S and $T = O(1)$ probes must have $S = \tilde{\Omega}(n^2)$.*

Conjecture 2 ([10]). *Any data structure for 3SUM-Indexing with space S and T probes must have $ST = \tilde{\Omega}(n^2)$.*

Conjecture 3 ([14]). *Any data structure for 3SUM-Indexing with space S and $T = O(n^{1-\delta})$ probes must have $S = \tilde{\Omega}(n^2)$.*

Clearly the last conjecture is the strongest, and in general, we have the following implications:

$$\text{Conjecture 3} \Rightarrow \text{Conjecture 2} \Rightarrow \text{Conjecture 1}$$

These conjectures have been successfully used to prove fine-grained hardness of several natural static data structure problems ranging from Set Disjointness, Set Intersection, Histogram Indexing to Forbidden Pattern Document Retrieval [14].

Very surprisingly, Golovnev et al. [15] showed that the strongest of these conjectures, Conjecture 3, is false. Concretely, they gave a data structure for 3SUM-Indexing with $T = \tilde{O}(n^{3\delta})$ and $S = \tilde{O}(n^{2-\delta})$ for any constant $\delta > 0$. This refutes Conjecture 3, but not the remaining two conjectures. Their data structure is based on an elegant use of Fiat and Naor's [12] general time-space tradeoff for function inversion.

The refutation of Conjecture 3 only makes it more urgent that we replace these conjectured lower bounds by unconditional ones. However, depressingly little is still known in terms of unconditional hardness of 3SUM-Indexing. First, [10] proved Conjecture 1 in the special case of $T = 1$. Secondly, in the recent work by Golovnev et al. [15], the following was proved for *non-adaptive* data structures:

Theorem 1 ([15]). *Any* non-adaptive *cell probe data structure answering 3SUM-Indexing queries for input sets of size n from an abelian group G of size $\Omega(n^2)$ using S words of w bits must have query time $T = \Omega(\lg n / \lg(Sw/n))$.*

A non-adaptive data structure is one in which the cells to probe is chosen beforehand as a function *only* of the query element z. That is, the data structure is not allowed to choose which memory cells to probe based on the contents of previously probed cells. Proving lower bounds for non-adaptive data structures is often easier than allowing adaptivity, see e.g. [8, 6, 25], and Golovnev et al. remark: "*It is crucial for our proof that the input is chosen at random after the subset of data structure cells, yielding a lower bound only for non-adaptive algorithms.*" [15]. Golovnev et al. explicitly raised it as an interesting open problem (Open Question 3 in [15]) whether a similar lower bound can be proved also for adaptive data structures.

1.1 Our Contributions

Our main contribution is a lower bound for 3SUM-Indexing that holds also for adaptive data structures:

Theorem 2. *Any cell probe data structure answering 3SUM-Indexing queries for input sets of size n for abelian groups $([O(n^2)], +)$ and $(\{0,1\}^{2\lg(n)+O(1)}, \oplus)$ using S words of $w = \Omega(\lg n)$ bits must have query time $T = \Omega(\lg n / \lg(Sw/n))$.*

Our lower bound matches the previous bound from [15], this time however allowing adaptivity. Moreover, it (essentially) matches the strongest known lower bounds for static data structures (the strongest lower bounds peak at $T = \Omega(\lg n / \lg(S/n))$ [20]), thus ruling out further progress without a major breakthrough (also in circuit complexity [28, 11]).

Our proof is based on a novel reduction from Pătraşcu's Reachability Oracles in the Butterfly graph problem [24]. This problem, while rather abstract, has been shown to capture the hardness of a wealth of static data structure problems such as 2D Range Counting, 2D Rectangle Stabbing, 2D Skyline Counting and Range Mode Queries, see e.g. [26, 7, 16] as well as for dynamic data structure problems, including Range Selection and Median [22] and recently also all dynamic problems that the Marked Ancestor Problem reduces to [21, 3], which includes 2d Range Emptiness, Partial Sums and Worst-Case Union-Find. Our work adds 3SUM-Indexing and all problems it reduces to, to the list.

Even Smaller Universes. The reduction from Reachability Oracles in the Butterfly gives lower bounds for abelian groups of size $\Omega(n^2)$, leaving open the possibility of more efficient data structures for smaller groups. Indeed, $\Omega(n^2)$ cardinality of the groups seems like a natural requirement for hardness, as there are n^2 pairs of elements $a_1 \in A_1$ and $a_2 \in A_2$ and thus for smaller groups, one might start to exploit structures in the sumset $A_1 + A_2$ to obtain more efficient data structures. We therefore investigate whether the lower bound in Theorem 2 can be generalized to smaller groups. Quite surprisingly, we show that:

Theorem 3. *Any cell probe data structure answering 3SUM-Indexing queries for input sets of size n for abelian groups $([O(n^{1+\delta})], +)$ and $(\{0,1\}^{(1+\delta)\lg(n)+O(1)}, \oplus)$ for a constant $\delta > 0$, using S words of $w = \Omega(\lg n)$ bits must have query time $T = \Omega(\lg n / \lg(Sw/n))$.*

Thus we get logarithmic lower bounds for linear space data structures, even when the group has size only $n^{1+\delta}$.

To prove Theorem 3, we revisit Pătraşcu's Lopsided Set Disjointness (LSD) communication game, which he also used to prove his lower bound for Reachability Oracles in the Butterfly. We give a careful reduction from LSD to 3SUM-Indexing on small universes, thereby establishing Theorem 3.

Non-Adaptive Data Structures. As another contribution, we revisit the non-adaptive setting considered by Golovnev et al. [15]. Here we present a significantly shorter proof of their lower bound and also improve it from $T = \Omega(\lg n / \lg(Sw/n))$ to $T = \Omega(\lg |G| / \lg(Sw/n))$. Concretely, we prove the following theorem:

Theorem 4. *Any non-adaptive cell probe data structure answering 3SUM-Indexing queries for input sets of size n for an abelian group G of size $\omega(n^2)$, using S words of $w = \Omega(\lg n)$ bits must have query time $T = \Omega(\min\{\lg |G| / \lg(Sw/n), n/w\})$.*

We remark that the proof of Golovnev et al. [15] cannot be extended to a $\lg |G|$ (technically, they require $|G|/n$ queries to survive a cell sampling, whereas we only require n queries to survive).

Our improvement has a subtle, but interesting consequence. Concretely, if the size of the group grows to sub-exponential in n, say $|G| = 2^{\sqrt{n}}$, then the lower bound becomes $T = \Omega(\min\{\sqrt{n}/\lg(\frac{Sw}{n}), n/w\})$. Since it is most natural to assume the cell size is large enough to store a group element, i.e. $w = \Omega(\lg |G|) = \Omega(\sqrt{n})$, the lower bound is still at least $T = \Omega(\sqrt{n}/\lg S)$. While such large groups is perhaps unrealistic, one can also interpret the result as saying that if we are non-adaptive and attempt to design a data structure that does not exploit the size of the underlying group, then we are doomed to have a slow query time.

Non-Adaptive 2-Bit-Probe Data Structures. Finally, we consider non-adaptive data structures restricted to $T = 2$ probes in the *bit probe model*, meaning that each memory cell has $w = 1$ bits. The lower bound from Theorem 1 by [15] in this case is $S = \tilde{\Omega}(n^{3/2})$ (see the paper [15] for the general formulation $S = \tilde{\Omega}(n^{1+1/T})$) and our lower bound from Theorem 4 is $S = \tilde{\Omega}(n(|G|/n)^{1/T}) = \tilde{\Omega}(\sqrt{n|G|})$. We significantly strengthen this result by proving an $S = \Omega(|G|)$ lower bound for an abelian group $(G, +)$, completely ruling out any non-trivial data structure with 2 non-adaptive bit probes (with $|G|$ space, we can trivially store a bit vector representing the sumset $A_1 + A_2$ and have $T = 1$ while being non-adaptive):

Theorem 5. *Any non-adaptive data structure for 3SUM-Indexing such that $T = 2$ and $w = 1$ requires $S = \Omega(|G|)$ for an abelian group $(G, +)$.*

Our proof takes an interesting new approach to data structure lower bounds and we find that the proof itself is a valuable contribution to data structure lower bounds. The basic idea is to view the memory cells of the data structure as a graph with one node per cell. The queries then become edges corresponding to the $T = 2$ memory cells probed. If the number of memory cells is $o(|G|)$, then the graph has a super-linear number of edges. This implies that its girth is at most logarithmic and hence we can find a short cycle in the graph. A cycle is a set of m queries being answered by m memory cells. The standard cell sampling lower bounds (often used in data structure lower bounds) cannot derive a contradiction from this, as the m memory bits intuitively are sufficient to encode the m query answers. However, our novel new contribution is to examine the different types of possible query algorithms (i.e. which function of the two bits probed does it compute) and argue that in all cases, such a short cycle is impossible. Directly examining the types of query algorithms has not been done before in data structure lower bounds and we find this a valuable contribution that we hope may prove useful in future work.

2 Reduction from Reachability Oracles in the Butterfly Graph

In this section, we give a reduction from the problem of *Reachability Oracles in the Butterfly Graph* to 3SUM-Indexing with the modulo group and the XOR group, proving Theorem 2. In both cases, the size of the group is at most quadratic with respect to the input set sizes.

Definition 2 (Butterfly Graphs)**.** *A Butterfly graph of degree B and depth d is a directed graph with $d+1$ layers, each comprising of B^d nodes. For each layer, the i^{th} node can be associated with a d-digit number in base B which we will refer to as its* label v_i *where $v_i[0]$ denotes the least significant digit. Then there is an edge from node i on the k^{th} layer to node j on the $(k+1)^{th}$ layer if and only if $v_i[h] = v_j[h]$ for all $h \neq k$. That is to say, that there is an edge if and only if i and j may differ only on the k^{th} digit of their labels. We will denote such an edge by $e_k(i,j)$.*

Nodes in the layer 0 of the graph are called source nodes*, whereas nodes in layer d of the graph are called* sink nodes.

Definition 3 (Reachability Oracles in the Butterfly Graph)**.** *The problem of Reachability Oracles in the Butterfly Graph is that one has to pre-process into a data structure a subset of the edges E of the butterfly graph of degree B and depth d. Queries come in the form of (s,t) and the goal is decide if there exists a path from source node s to sink node t using the subset of edges E.*

Pătraşcu proved the following lower bound for the problem in the cell probe model:

Lemma 1 (Section 5 of [24])**.** *Any cell probe data structure answering reachability queries in subgraphs of the butterfly graph with degree B and depth d, using S words of w bits must have query time $t = \Omega(d)$, assuming that $B = \Omega(w^2)$ and $\lg(B) = \Omega(\lg(Sd/N))$ where $N = dB^d$.*

A few remarks about reachability in the Butterfly graph are in order. Firstly, note that for any source-sink pair (s,t), there exists a unique path from source s to sink t in the Butterfly graph. Namely, the path uses exactly edges of the form $e_k(i,j)$ such that for $k \in [d]$, $e_k(i,j)$ is the edge from node i on layer k to node j on layer $k+1$ such that:

1. $v_s[h] = v_i[h]$ for all $h \geq k$. That is to say that the $d-k$ most significant digits of the labels of nodes s and i are the same.
2. $v_t[h] = v_j[h]$ for all $h \geq k+1$. That is to say that the $k+1$ least significant digits of the labels of nodes t and j are the same.

Conversely, we can also say that the edge $e_k(i,j)$ connects all pairs of nodes s,t such that the label for s shares the most significant $d-k$ digits with i and the label for t shares the least significant $k+1$ digits with t.

Intuitively, this is because the traversing from node i in the k^{th} layer to node j in the $(k+1)^{th}$ layer can be seen as "setting" the k^{th} digit of the label for node i into the k^{th} digit of the label for node j while leaving the rest of the digits unaltered.

The general idea of the reduction to 3SUM-Indexing is to test whether all the required edges are present when querying for s and t. This should be done by asking one 3SUM-Indexing query. We will design it such that a sum $z = a_1 + a_2$ exists for our query z if and only if there is at least one edge missing on the path from s to t.

Constructing A_1. Our basic idea is to take every edge $e_k(i,j)$ in the Butterfly graph and encode it into a group element g in A_1. We construct g such that its digits can be broken up into 5 blocks so that conceptually the:

1. first block encodes the layer the edge is from;
2. second block encodes the presence of edge $e_k(i,j)$ in E;
3. third block encodes the $d-k$ most significant bits of i followed by k zeroes;
4. fourth block holds $d-k-1$ zeroes followed by the $k+1$ least significant digits of j;
5. fifth block holds 2 zeroes.

In short, for every edge $e_k(i,j)$, we add group element to A_1 whose digits are in the following form:

$$(k, \mathbb{1}\{e_k(i,j) \in E\}, v_i[d-1], \ldots, v_i[k], \underbrace{0, \ldots, 0}_{\text{d - 1}}, v_j[k], \ldots, v_j[0], 0, 0)$$

where $\mathbb{1}\{e_k(i,j) \in E\}$ is 1 if $e_k(i,j) \in E$ and 0 otherwise. Note that the Butterfly graph has dB^d nodes with degree B, hence a total of $n = dB^{d+1}$ edges. Since A_1 has one element for each such edge, we have $|A_1| = n$.

Constructing A_2. Next, we construct the set A_2 of group elements such that for every k, it can "helps" any group element g in A_1, originating from an $e_k(i,j)$, to sum to any value where the third block shares the $d-k$ most significant bits with i and the fourth block shares the $k+1$ least significant digits of j. This can be done by adding into set A_2 every group element such that the:

1. first block holds some value $-k$;
2. second block is zero;
3. third block is $d-k$ zeroes followed by any possible k digit value;
4. fourth block holds any possible $d-k-1$ digit value followed by $k+1$ zeroes;
5. fifth block holds any possible digit value from $[0, B-1]$.

Thus for $k \in [0, d-1]$, we add any number of the following form into A_2:

$$(-k, 0, \underbrace{0, \ldots, 0}_{d-k}, \underbrace{\star, \ldots, \star}_{d-1}, \underbrace{0, \ldots, 0}_{k+1}, \star, \star)$$

where $\star$ denotes wildcard. Note that the least significant digits is not strictly necessary but is included to enforce that the size of the sets A_1 and A_2 are the same. Observe that $|A_2| = dB^{d+1} = n = |A_1|$.

Different Groups. For the reduction to 3SUM-indexing in the modulo group, we will consider the set of integers in $[(dB^{d+1})^2]$. To that end, the encoding works by understanding the $2(d+2)$ digits as specifying a mixed-radix number, where the most significant digit is in base $4d$, the second most significant digit is in base 3 and the remaining digits are in base B. In which case, we can take $-k$ to be $4d-k$.

On the other hand, for the XOR group, assuming that d and B are powers of 2, we can then also naturally transform each digit into their binary representation with the exception of the most significant digit whose bit representation should be based on the number's complement and the second most significant digit may be in base 2.

Translating a Query. What remains is to explain how we answer a reachability query (s,t). We will first consider the reduction for the group $([(dB^{d+1})^2], +)$ and subsequently argue that the same reduction basically holds for the XOR group assuming that d and B are powers of 2. We claim that there exists $a_1 \in A_1$ and $a_2 \in A_2$ whose sum is

$$z_{(s,t)} = (0, 0, v_s[d-1], \ldots, v_s[0], v_t[d-1], \ldots, v_t[0], 0, 0)$$

if and only if there does not exist a path from s to t in the Butterfly graph.

To see this, we first argue that for a pair $a_1 + a_2$ that could potentially sum to $z_{(s,t)}$, we need not worry about carries amongst the digits of the numbers. To see this, we start by observing that $a_1 + a_2$ must have its most significant digit equal to 0. We claim this is only possible if a_1's most significant digit is k and a_2's is $4d-k = -k$. To see this, observe that the second most significant digit of a_1 is at most 1 and the second most significant of a_2 is always 0. Since the second most significant digit is in base 3, this means that we cannot get a carry from these digits. Now that we have established this, we observe that for all remaining digits of any valid pair a_1 and a_2 (pairs where the most significant digit in the sum is 0), there is at most one of the elements that have a non-zero digit, hence we will not see any carries.

Now assume there does not exists a path from some source node s to some sink node t. This must mean that there exists a $k \in [0, d-1]$ and an edge $e_k(i,j)$ not in E where:

$$v_i = (v_s[d-1], \ldots, v_s[k], v_t[k-1], \ldots, v_t[0])$$
$$v_j = (v_s[d-1], \ldots, v_s[k+1], v_t[k], \ldots, v_t[0])$$

By construction, this implies that the following group element exists in the set A_1:

$$(k, 0, v_s[d-1], \ldots, v_s[k], \underbrace{0, \ldots, 0}_{\text{d - 1}}, v_t[k], \ldots, v_t[0], 0)$$

(2) Furthermore, the following group element always exists in A_2:

$$(-k, 0, 0, \ldots, 0, v_t[k-1], \ldots, v_t[0], v_s[d-1], \ldots, v_s[k+1], 0, \ldots, 0, 0)$$

This means that the value $(0, 0, v_s[d-1], \ldots, v_s[0], v_t[d-1], \ldots, v_t[0], 0)$ is obtainable as a sum $a_1 + a_2$. If on the other hand there is a path between s and t, then all elements in A_1 of the form

$$(k, \star, v_s[d-1], \ldots, v_s[k], \underbrace{0, \ldots, 0}_{\text{d - 1}}, v_t[k], \ldots, v_t[0], 0)$$

must have $\star = 1$ and thus it is not possible to write $z_{(s,t)}$ as $a_1 + a_2$.

The XOR Group. For a reduction to the XOR group setting, we consider each element coordinate-wise using their binary representations with the exception that in the first coordinate the value is represented using the number's complement representation. Using the previous remark we also assert that for any pair $a_1 \in A_1$, $a_2 \in A_2$, the only common digit that is both non-zero is the most significant digit and thus the addition being done digit-wise. For that reason, the sum behaves exactly the same way over the XOR group as it does over the modulo group that we have defined. Thus the size of the universe and input sets A_1, A_2 remain unchanged and the reduction holds in the XOR group as well.

Analysis. Now by setting $B = \frac{Sw^2}{n}$, note that $B = \Omega(w^2)$ and:

$$\lg(Sd/N) \leq \lg\left(\frac{SB\lg(n)}{n}\right) \leq \lg(SBw/n) \leq \lg(B^2) = O(\lg B)$$

Furthermore, it holds that

$$\lg(Sw/n) = \frac{1}{2}\lg((Sw/n)^2) \geq \frac{1}{2}\lg(Sw^2/n) \geq \frac{1}{2}\lg B$$

Using Lemma 1, it then follows that for any cell-probe solution for 3SUM-Indexing for the modulo group $([n^2], +)$ and XOR group $(\{0,1\}^{2\lg(n)+O(1)}, \oplus)$ any static data structure that uses $S \geq n$ cells of $w \geq \lg(n)$ bits has query time $T = \Omega(d) = \Omega(\lg_B n) = \Omega(\lg n / \lg(Sw/n))$.

3 Reduction from Lopsided Set Disjointness

In this section, we prove Theorem 3, establishing hardness of 3SUM-Indexing also for abelian groups of size $\Delta = n^{1+\delta}$. For the proof, we focus on the integers modulo Δ, but remark that the proof readily adapts to the XOR group as well.

For the proof, we use Pătraşcu's Blocked Lopsided Set Disjointness (Blocked LSD) problem. In this problem, there are two players, Alice and Bob. Bob receives as input a set Y, which is an arbitrary subset of a universe $[N] \times [B]$. Alice receives a set $X \subset [N] \times [B]$ with the restriction that X contains exactly one element (i, b_i) for every $i = 0, \ldots, N-1$. The goal for Alice and Bob is to determine whether $X \cap Y = \emptyset$ while minimizing communication. The following is known regarding the communication complexity of Blocked LSD:

Lemma 2 (Theorem 4 of [24])**.** *Fix $\delta > 0$. Any communication protocol for Blocked LSD requires either Alice sending at least $\delta N \lg B$ bits, or Bob sending at least $NB^{1-O(\delta)}$ bits.*

The basic idea in the reduction, is to have Bob interpret his set Y as two input sets A_1, A_2 of $n = NB$ group elements to 3SUM-Indexing (we may have $|A_1|$ and $|A_2|$ smaller than NB, but we can always pad with dummy elements, so we assume $n = NB$). Given a data structure D for 3SUM-Indexing, Bob then builds D on this input. Alice on the other hand interprets her set X (which has cardinality N) as a set of N/ℓ queries to 3SUM-Indexing, where ℓ is a parameter to be determined. The key property of the reduction, is that the answers to all N/ℓ queries of Alice on D, determines whether $X \cap Y = \emptyset$.

Communication Protocol. Assume for now that we can give such a reduction. Alice and Bob then obtains a communication protocol for Blocked LSD as follows: Let T be the query time of D. For $i = 1, \ldots, T$, Alice simulates the i'th step of the query algorithm for each of her N/ℓ queries, *in parallel*. This is done by asking Bob for the set of at most N/ℓ cells that they probe in the i'th step. This costs $O(\lg \binom{S}{N/\ell}) = O((N/\ell)\lg(S\ell/N))$ bits of communication by specifying the required cells as a subset of the S memory cells of D. Bob replies with the contents of the cells, costing $((N/\ell)w)$ bits. This is done for T rounds, resulting in a communication protocol where Alice sends $O((N/\ell)T\lg(S\ell/N))$ bits and Bob sends $O((N/\ell)Tw)$ bits. If we fix $B = w^4$ and δ as a small enough

constant, then Lemma 2 says that either Alice sends $\Omega(N \lg w)$ bits or Bob sends $\Omega(N\sqrt{B}) = \Omega(Nw^2)$ bits. In our protocol, Bob's communication is $O((N/\ell)Tw) = O(NTw)$ bits. We assume $w = \Omega(\lg n)$, thus we conclude that either $NTw = \Omega(Nw^2) \Rightarrow T = \Omega(\lg n)$, or Alice's communication must be $\Omega(N \lg B) = \Omega(N \lg w)$ bits. In the first case, we are done with the proof, hence we examine the latter case. Alice's communication is $O((N/\ell)T \lg(S\ell/N))$ bits, which implies $T = \Omega(\ell \lg w / \lg(S\ell/N))$. Thus to derive our lower bound, we have to argue that it suffices for Alice to answer N/ℓ queries for a large enough ℓ.

Asking Few Queries. We will show that it suffices for Alice to ask N/ℓ queries with $\ell = \varepsilon \lg n / \lg w$. Here $\varepsilon > 0$ is a small constant depending on δ in the group size $\Delta = n^{1+\delta}$. Thus we get a lower bound of $T = \Omega(\lg n / \lg((S \lg n)/(N \lg w)))$. Since $N = n/B = n/w^4$, this simplifies to $T = \Omega(\lg n / \lg(Sw/n))$ as claimed in Theorem 3.

Thus what remains is to show how Alice and Bob computes the input and queries. For this, they conceptually partition the universe $[N] \times [B]$ into groups $\{i\ell, \ldots, (i+1)\ell - 1\} \times [B]$ for $i = 0, \ldots, N/\ell$. Alice will ask precisely one query for each such group. Denote by Y_i the subset of Y that falls in the i'th group and denote by X_i the subset of X that falls in the i'th group. Clearly $X \cap Y = \emptyset$ if and only if $X_i \cap Y_i = \emptyset$ for all i. Thus Alice will use her i'th query to determine whether $X_i \cap Y_i = \emptyset$.

Constructing A_1 and A_2. To support this, Bob first constructs the set A_1 based on his elements X. He examines each group X_i, and for every $(j, b) \in X_i$, he adds the integer $i(B+1)^{\ell+1} + (b+1)(B+1)^{j-i\ell}$ to A_1. Next, he constructs the set A_2. For this, he considers all vectors $Z = (b_0, \ldots, b_{\ell-1})$ for which the numbers are all between 0 and B and precisely one of them is 0. He adds the integer $\sum_{j \in [\ell]} b_j (B+1)^j$ to A_2. This completes Bob's construction of the input sets A_1 and A_2. We have $|A_1| \leq NB = n$ and $|A_2| \leq (B+1)^\ell$.

Asking the Queries. We next describe how Alice translates her set Y into queries. For each Y_i, she needs to construct one query z_i whose answer determines whether $X_i \cap Y_i = \emptyset$. Recall that Y_i is of the form $\{(i\ell, b_0), (i\ell + 1, b_1), \ldots, ((i+1)\ell - 1, b_{\ell-1})\}$. She starts by subtracting off $i\ell$ from the first index in each pair, obtaining the set $\{(0, b_0), (1, b_1), \ldots, (\ell - 1, b_{\ell-1})\}$. Alice now asks the query $z_i = i(B+1)^{\ell+1} + \sum_{j=0}^{\ell-1} (b_j + 1)(B+1)^j$.

Correctness. We claim that z_i is part of a 3SUM if and only if $X_i \cap Y_i \neq \emptyset$. To see this, observe first that to write z_i as $a_1 + a_2$, it must be the case that a_1 was constructed from X_i as otherwise we cannot obtain the $i(B+1)^{\ell+1}$ parts of z_i. Next, observe that if we write the integers in base $B+1$, then A_2 contains precisely every integer of the form where there is a single digit $j \in [\ell]$ that is zero and all remaining are non-zero. Also, the numbers obtained from $(j, b) \in X_i$ are of the form $i(B+1)^{\ell+1} + (b+1)B^{j-i\ell}$ and thus have exactly one non-zero digit among the first ℓ. Since z_i has exclusive non-zero digits in the first ℓ, it follows that $z_i = i(B+1)^{\ell+1} + \sum_{j=0}^{\ell-1} (b_j + 1)(B+1)^j$ can be written as $a_1 + a_2$ if and only if a_1 was obtained from a $(j, b) \in X_i$ for which b is equal to $b_{j-i\ell}$. This is the case if and only if X_i and Y_i intersect in (j, b).

Analysis. We now determine ℓ. Recall that $B = w^4$ and observe that all integers are bounded by $N(B+1)^{\ell+2} \leq n(B+1)^{\ell+2}$. If we insist on a group of size $\Delta = n^{1+\delta}$, this means we can set $\ell = \delta \lg n / \lg(B+1) - 2 \geq \varepsilon \lg n / \lg w$ for a sufficiently small constant $\varepsilon > 0$. This also implies that $|A_2| \leq n^\delta \leq n$ and thus completes the proof of Theorem 3.

4 Lower Bound for Non-Adaptive Data Structures

In this section, we prove an $\Omega(\min\{\lg |G| / \lg(Sw/n), n/w\})$ lower bound for non-adaptive 3SUM-Indexing data structures when $|G| = \omega(n^2)$. Similarly to the previous approach by Golovnev et al. [15], we use a cell sampling approach.

Consider a data structure using S memory cells of w bits and answering queries non-adaptively in T probes. Consider all subsets of $\Delta = n/(2w)$ memory cells. There are $\binom{S}{\Delta}$ such subsets. We say that a query z is answered by a set of cells C, if all the (non-adaptively chosen) cells it probes are contained in C. Any query z is answered by at least $\binom{S-T}{\Delta-T}$ sets of Δ cells, namely all those containing the T cells probed on z. It follows by averaging over the $|G|$ queries that there is a set of cells C^* answering at least

$$|G| \binom{S-T}{\Delta-T} / \binom{S}{\Delta} = |G| \frac{\Delta(\Delta-1)\cdots(\Delta-T+1)}{S(S-1)\cdots(S-T+1)} \geq |G| \left(\frac{\Delta - T + 1}{S}\right)^T.$$

If $T \geq \Delta/2$, we are already done as we have proved $T = \Omega(n/w)$. Otherwise, $T \leq \Delta/2$ and thus the above is at least $|G|(\Delta/(2S))^T = |G|(n/(4Sw))^T$. If we assume for contradiction that $T = o(\lg|G|/\lg(Sw/n))$, this is at least $|G|^{1-o(1)} > n$. Let Q be the group elements corresponding to an arbitrary subset of n of those queries. We argue that we can construct a distribution over inputs A_1, A_2 such that the queries Q cannot be answered from few cells, contradicting that we have answered them from C^*. More precisely, we show:

Lemma 3. *Let $(G,+)$ be an abelian group with $\omega(n^2)$ elements. Given any subset $Q \subseteq G$ of at most n elements, there exists an input distribution D over A_1, A_2 for which the event of any element in Q being included in the sum set $A_1 + A_2$ (defined as $\{a_1 + a_2 : a_1 \in A_1, a_2 \in A_2\}$) is fully independent. That is, $\Pr_{(A_1,A_2)\sim D}[\bigwedge_{q\in Q} q \in (A_1 + A_2)] = \prod_{q\in Q} \Pr_{(A_1,A_2)\sim D}[q \in (A_1 + A_2)]$. Furthermore, for any $q \in Q$, it is the case that $\Pr_{(A_1,A_2)\sim D}[q \in (A_1 + A_2)] = \frac{1}{2}$.*

The proof is deferred to the end of the section.

We now use Lemma 3 to derive a contradiction to the assumption that $T = o(\lg|G|/\lg(Sw/n))$. Concretely, we invoke the lemma with the Q defined above. This implies that the answers to the queries in Q has entropy n bits. However, they are being answered from a fixed set of $n/2w$ cells. These cells together have $n/2$ bits. Since their addresses are fixed, their contents must uniquely determine the n query answers, yielding the contradiction and hence $T = \Omega(\min\{\lg|G|/\lg(Sw/n), n/w\})$. This completes the proof of Theorem 4. What remains is to prove Lemma 3:

Proof of Lemma 3. We prove the lemma by showing that given $Q \subseteq G$ of n elements, for any $P \subseteq Q$ there exists an input pair A_1^P and A_2^P such that $P \subseteq (A_1^P + A_2^P)$ and $(Q \setminus P) \cap (A_1^P + A_2^P) = \emptyset$. Then D is the distribution that is uniform over all possible pairs of sets (A_1^P, A_2^P) with P ranging over all subsets of Q.

Given any P, we build the sets A_1^P and A_2^P iteratively, where they are both initially empty. Let $p_1, p_2, \ldots$ enumerate the elements of P. At each iteration, let p_i be the first value not in $(A_1^P + A_2^P)$. There are $\frac{|G|}{2}$ pairs of elements (a_1, a_2) such that $a_1 + a_2 = p_i$. Given that $|Q \setminus P| \leq n$, there are $O(n^2)$ possible values c from G such that $\{c\} + (A_1^P \cup A_2^P) \subseteq (Q \setminus P)$. Therefore there must still exist a pair (a_1, a_2) such that $a_1 + a_2 = p_i$ and $(\{a_1\} \cup A_1^P + \{a_2\} \cup A_2^P) \cap (Q \setminus E) = \emptyset$, assuming that $|G| = \omega(n^2)$.

5 Bit Probe Lower Bound for 3SUM-Indexing

In this section we give the bit probe lower bound for 3SUM-Indexing stated in Theorem 5.

The proof idea is based on an incompressibility argument. We will inspect the way the queries are structured and construct a specific input distribution that the data structure algorithm end up using too few bits for and therefore derive a contradiction. For this, we will again use Lemma 3 from the previous section. The key difference between this proof and the proof in the previous section, lies in how we find a set of queries answered by too few cells. Moreover, in this proof, we will derive a contradiction even with m queries being answered by m cells, and thus intuitively the cells actually have enough information, but yet cannot answer the queries. We start by introducing some graph theory that we need:

Lemma 4. *[Theorem 1 of [2]] Let (V, E) be a graph with n nodes, average degree $d > 2$ and girth r. Then $n \geq 2(d-2)^{r/2-2}$.*

From Lemma 4 we conclude that for a graph with $o(|G|)$ nodes and $|G|$ edges, it is the case that the graph has a girth of $O(\lg(n))$. To see this, note that the average degree d of such a graph is $\omega(1)$ and thus it follows that for some constant $c > 1$:

$$\begin{aligned} o(|G|) \geq 2(d-2)^{r/2-2} &\Rightarrow \\ o(|G|) \gg 2(c)^{r/2} &\Rightarrow \\ r \in O(\lg_c(n)) & \end{aligned}$$

Given any non-adaptive pre-processing algorithm with $T = 2$, $S = o(|G|)$, and $w = 1$, define V to be the set of S nodes each representing a memory cell and let E be the set of edges such that an edge $e_g = (u, v)$ is in the edge set if and only there exists some group element $g \in G$ such that the querying algorithm on input g accesses both memory cell u and v. Furthermore, associate with each edge e_g a function $f_g : \{0,1\}^2 \to \{0,1\}$ that defines the output behaviour of the querying algorithm upon reading the bits at node u and v. We broadly categorise the possible functions f_g into 4 types:

1. **Copy type functions**. The type of functions f_g that depend only one of its two inputs. There are 4 of such functions.

2. **Constant type functions**. The type of functions f_g that are completely independent of its two inputs. There are 2 such functions.

3. **AND type functions**. The type of functions f_g whose truth table is such that exactly 3 of the 4 possible inputs leads to the same output where the last input differs. There are 8 such functions.

4. **XOR type functions**. The type of functions f_g that are either the XOR of its 2 inputs or the negation of the XOR of its 2 inputs. There are 2 such functions.

Note that none of the edges can be the constant type, since this means that the querying algorithm's answer is independent of the input set. Also, by an averaging argument there is at least one type of function that $\Omega(|G|)$ edges are associated with. Furthermore, Lemma 3 asserts that there can be at most 2 edges that are parallel to each other, otherwise we can construct an input distribution D such that the data structure manages to use 2 bits to encode the outcome of a random variable that has Shannon entropy at least 3, which is a contradiction. Thus there are $\Omega(|G|)$ many edges that are not parallel to each other and are all of the same type. We analyse the different types separately. We start with the simplest COPY type:

(COPY type) Assuming that there are $\Omega(|G|)$ edges that are associated with the copy type function, there must exist at least one node u such that $\omega(1)$ edges e_g are such that the associated function f_g depend only on the bit at this node. Letting Q contain two such group elements, this yields a contradiction using Lemma 3 to construct a distribution over A_1 and A_2 such that the entropy of the two query answers in Q is 2 bits.

For the remaining types, we look for a short cycle. Using Lemma 4, we get that there is a cycle of $O(\lg n)$ length using only edges associated with functions of the same type. Denote by Y the set of group elements g such that e_g is in the cycle and $y_1, \ldots, y_t$ enumerates the elements of Y based on a traversal of the cycle. That is, the edge corresponding to y_i shares endpoints with edges corresponding to y_{i-1} and y_{i+1}, where $y_{t+1} = y_1$ and $y_0 = y_t$. We use Lemma 3 with $Q = Y$ to get a distribution D over (A_1, A_2) such that the answers to queries in Y are independent and they are all uniform random. We now handle the two remaining types separately.

(AND type) Let b be the output of f_{y_1} that is only obtainable by exactly 1 of the 4 possible inputs. Consider the distribution D conditioned on the event that $\mathbb{1}\{y_1 \in (A_1 + A_2)\} = b$. Since only 1 of the 4 inputs to f_{y_1} is consistent with this output, this fixes the two input bits to f_{y_1}. Therefore, there are $t-2$ bits left to encode $t-1$ independent and fully random outputs (namely, whether $y_2, \ldots, y_t$ are in $A_1 + A_2$), which yields us the desired contradiction.

(XOR type) Let $(A_1, A_2) \sim D$ be drawn from the input distribution constructed using Lemma 3 with $Q = Y$. Let the endpoints of y_i be u_i, v_i, such that $v_t = u_1$. Note for all i, it is necessarily the case that $\mathbb{1}\{y_i \in (A_1 + A_2)\} = u_i \oplus v_i$ or $\mathbb{1}\{y_i \in (A_1 + A_2)\} = 1 \oplus u_i \oplus v_i$. Then $\bigoplus_2^t \mathbb{1}\{y_i \in (A_1 + A_2)\}$ is either $u_1 \oplus v_1$ or $1 \oplus u_1 \oplus v_1$ which means that $\mathbb{1}\{y_1 \in (A_1 + A_2)\}$ is either $\bigoplus_2^t \mathbb{1}\{y_i \in (A_1 + A_2)\}$ or its negation. Then Lemma 3 yields the desired contradiction.

References

[1] A. Abboud and K. Lewi. Exact weight subgraphs and the k-sum conjecture. In F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.

[2] N. Alon, S. Hoory, and N. Linial. The moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002.

[3] S. Alstrup, T. Husfeldt, and T. Rauhe. Marked ancestor problems. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 534–544. IEEE Computer Society, 1998.

[4] A. Amir, T. M. Chan, M. Lewenstein, and N. Lewenstein. On hardness of jumbled indexing. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014.

[5] G. Barequet and S. Har-Peled. Polygon containment and translational min-hausdorff-distance between segment sets are 3sum-hard. *Int. J. Comput. Geom. Appl.*, 11(4):465–474, 2001.

[6] J. Boninger, J. Brody, and O. Kephart. Non-adaptive data structure bounds for dynamic predecessor. In S. V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 20:1–20:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[7] G. S. Brodal and K. G. Larsen. Optimal planar orthogonal skyline counting queries. In R. Ravi and I. L. Gørtz, editors, *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, volume 8503 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 2014.

[8] J. Brody and K. G. Larsen. Adapt or die: Polynomial lower bounds for non-adaptive dynamic data structures. *Theory Comput.*, 11:471–489, 2015.

[9] T. M. Chan. More logarithmic-factor speedups for 3sum, (median, +)-convolution, and some geometric 3sum-hard problems. In A. Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 881–897. SIAM, 2018.

[10] E. D. Demaine and S. P. Vadhan. Some notes on 3sum. Unpublished manuscript, December 2001.

[11] Z. Dvir, A. Golovnev, and O. Weinstein. Static data structure lower bounds imply rigidity. In M. Charikar and E. Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 967–978. ACM, 2019.

[12] A. Fiat and M. Naor. Rigorous time/space tradeoffs for inverting functions. In C. Koutsougeras and J. S. Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 534–541. ACM, 1991.

[13] A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom.*, 45(4):140–152, 2012.

[14] I. Goldstein, T. Kopelowitz, M. Lewenstein, and E. Porat. Conditional lower bounds for space/time tradeoffs. In F. Ellen, A. Kolokolova, and J. Sack, editors, *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, volume 10389 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2017.

[15] A. Golovnev, S. Guo, T. Horel, S. Park, and V. Vaikuntanathan. Data structures meet cryptography: 3sum with preprocessing. In K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 294–307. ACM, 2020.

[16] M. Greve, A. G. Jørgensen, K. D. Larsen, and J. Truelsen. Cell probe lower bounds and approximations for range mode. In S. Abramsky, C. Gavoille, C. Kirchner, F. M. auf der Heide, and P. G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 605–616. Springer, 2010.

[17] M. Henzinger, S. Krinninger, D. Nanongkai, and T. Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In R. A. Servedio and R. Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30. ACM, 2015.

[18] T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3sum conjecture. In R. Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287. SIAM, 2016.

[19] K. G. Larsen. The cell probe complexity of dynamic range counting. In H. J. Karloff and T. Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 85–94. ACM, 2012.

[20] K. G. Larsen. Higher cell probe lower bounds for evaluating polynomials. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 293–301. IEEE Computer Society, 2012.

[21] K. G. Larsen, J. L. Starup, and J. Steensgaard. Further unifying the landscape of cell probe lower bounds. In H. V. Le and V. King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 224–231. SIAM, 2021.

[22] K. G. Larsen, O. Weinstein, and H. Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. *SIAM J. Comput.*, 49(5), 2020.

[23] M. Patrascu. Towards polynomial lower bounds for dynamic problems. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010.

[24] M. Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011.

[25] S. N. Ramamoorthy and A. Rao. Lower Bounds on Non-Adaptive Data Structures Maintaining Sets of Numbers, from Sunflowers. In R. A. Servedio, editor, *33rd Computational Complexity Conference (CCC 2018)*, volume 102 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[26] C. Sommer, E. Verbin, and W. Yu. Distance oracles for sparse graphs. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 703–712. IEEE Computer Society, 2009.

[27] M. A. Soss, J. Erickson, and M. H. Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Comput. Geom.*, 26(3):235–246, 2003.

[28] E. Viola. Lower bounds for data structures with space close to maximum imply circuit lower bounds. *Theory Comput.*, 15:1–9, 2019.

[29] A. C. Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.