On The I/O Complexity of Dynamic Distinct Counting*

Xiaocheng Hu[1], Yufei Tao[1], Yi Yang[2], Shengyu Zhang[1], and Shuigeng Zhou[2]

1 **Chinese University of Hong Kong**
Hong Kong, China
`{xchu, taoyf, syzhang}@cse.cuhk.edu.hk`
2 **Fudan University**
Shanghai, China
`{yyang1, sgzhou}@fudan.edu.cn`

Abstract

In *dynamic distinct counting*, we want to maintain a multi-set $\mathcal{S}$ of integers under insertions to answer efficiently the query: how many distinct elements are there in $\mathcal{S}$? In external memory, the problem admits two standard solutions. The first one maintains $\mathcal{S}$ in a hash structure, so that the distinct count can be incrementally updated after each insertion using $O(1)$ expected I/Os. A query is answered for free. The second one stores $\mathcal{S}$ in a linked list, and thus supports an insertion in $O(1/B)$ amortized I/Os. A query can be answered in $O(\frac{N}{B}\log_{M/B}\frac{N}{B})$ I/Os by sorting, where $N = |\mathcal{S}|$, B is the block size, and M is the memory size.

In this paper, we show that the above two naive solutions are already optimal within a polylog factor. Specifically, for any Las Vegas structure using $N^{O(1)}$ blocks, if its expected amortized insertion cost is $o(\frac{1}{\log B})$, then it must incur $\Omega(\frac{N}{B\log B})$ expected I/Os answering a query in the worst case, under the (realistic) condition that N is a polynomial of B. This means that the problem is repugnant to update buffering: the query cost jumps from 0 dramatically to almost linearity as soon as the insertion cost drops slightly below $\Omega(1)$.

1998 ACM Subject Classification F.2.2. [Analysis of algorithms and problem complexity]: Non-numerical algorithms and problems

Keywords and phrases Distinct counting, lower bound, external memory

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.1

1 Introduction

This paper studies the *dynamic distinct counting problem* defined as follows. Let $[2^w]$ represent the set of integers $\{0, 1, ..., 2^w - 1\}$, where w is the number of bits in a machine word. We want to support two operations on an initially empty multi-set $\mathcal{S}$:

- Insert(e): add an integer $e \in [2^w]$ to $\mathcal{S}$.
- Query: report the number of distinct elements in $\mathcal{S}$.[1]

* Xiaocheng Hu and Yufei Tao were supported in part by Projects GRF 4168/13 and GRF 142072/14 from HKRGC. Yi Yang and Shuigeng Zhou were supported in part by the Research Innovation Program of Shanghai Municipal Education Commission under grant No. 13ZZ003. Shengyu Zhang was supported in part by Project GRF 4194/13 from HKRGC.

[1] This problem should not be confused with ϵ-*approximate distinct counting* [10], where a query is allowed to return only an approximate answer.

This is a classic problem in computer science. Indeed, distinct queries are useful in such a large variety of contexts that database systems have made them a first-class citizen with direct SQL support: *select distinct count(...)*.

We consider the problem in the standard *external memory* (EM) model of computation (a.k.a. the *I/O model*). In this model, a machine has M words of memory, and a disk of an unbounded size. The disk has been formatted into disjoint *blocks* of size B words. It holds that $M \geq 2B$, i.e., the memory can accommodate at least two blocks. An I/O either reads a block from the disk into memory, or conversely writes B words from memory to a disk block. The *cost* of an algorithm is measured as the number of I/Os performed. The *space* of a structure is measured as the number of blocks occupied. CPU computation is free, but can take place only on memory data. We use N to denote the problem size (e.g., for the dynamic distinct counting problem, N equals the number of insertions). A structure is said to consume *polynomial space* if its space consumption is bounded by $N^{O(1)}$ in the worst case.

Dynamic distinct counting admits two standard solutions:

- The first one is to maintain $\mathcal{S}$ in a hash structure (e.g., [6]) of linear space $O(N/B)$. Given an $\textsc{Insert}(e)$, by probing the bucket of e, one can incrementally maintain the distinct count in $O(1)$ expected I/Os. A query can be answered by simply returning this count for free.
- The second solution organizes $\mathcal{S}$ in a linked list with the last block pinned in memory. The block is flushed to the disk after it has accumulated $\Omega(B)$ elements. This achieves the lowest amortized cost of $O(1/B)$ I/Os per insertion. A query can be answered by sorting $\mathcal{S}$ from scratch using $O(\frac{N}{B} \log_{M/B} \frac{N}{B})$ I/Os [1].

Rather naive as these solutions may appear, they still represent the best update-query tradeoffs to this date.

1.1 Our Results

In this paper, we show that both of the aforementioned naive solutions—exactly how a DBMS supports dynamic distinct counting—are already optimal up to a small factor. Specifically, no Las Vegas structure of polynomial space can do much better than $\Omega(N/B)$ in query cost if it must support fast updates:

▶ **Theorem 1.** *Let t_u be the expected amortized insertion cost of a polynomial-space Las Vegas structure for dynamic distinct counting, and t_q be its expected query cost, where both expectations are taken over the random choices made by the structure. In the scenario where $N = B^c, M = B^{c'}$ (for any integer constants $c' \geq 1$ and $c \geq c' + 1$), and $3 \log N \leq w = O(\log N)$, if $t_u = o(\frac{1}{\log B})$, then $t_q = \Omega(\frac{N}{B \log B})$.*

The theorem holds even for structures that defy the *indivisibility assumption*[2]. Furthermore, by fitting in some typical values for N, M, and B, one would quickly realize that N and M are almost always polynomials of B in practice.

Theorem 2 indicates that dynamic element counting is "repugnant" to update buffering. When there is no buffering (e.g., hashing), one can achieve $t_u = O(1)$ and $t_q = 0$ (free queries). On the other hand, if t_u needs to be improved by just an $\omega(\log B)$ factor, t_q surges

[2] This assumption says that every data element must be stored as an atom occupying a word. Thus, one cannot, for example, compress the bits of an element to save space.

dramatically to almost $\Omega(N/B)$, that is, any query algorithm must spend nearly the same cost as reading the entire dataset $\mathcal{S}$.

Technical Overview. We will consider instead the *dynamic element distinctness problem*, where we want to support two operations on an initial empty multi-set $\mathcal{S}$:

- INSERT(e): add an integer $e \in [2^w]$ to $\mathcal{S}$.
- QUERY: report *whether* all the elements in $\mathcal{S}$ are distinct.

A structure solving dynamic distinct counting also solves dynamic element distinctness with exactly the same space, query, and update cost: first obtain the number x of distinct elements in $\mathcal{S}$, and declares that all elements in $\mathcal{S}$ are distinct if and only if x equals the number of insertions in history. Therefore, a lower bound on the latter problem also carries over to the former. Indeed, the main result of this paper is:

▶ **Theorem 2.** *Let t_u be the expected amortized insertion cost of a polynomial-space Las Vegas structure for the dynamic element distinctness problem, and t_q be its expected query cost, where both expectations are taken over the random choices made by the structure. In the scenario where $N = B^c, M = B^{c'}$ (for any integer constants $c' \geq 1$ and $c \geq c' + 1$), and $3 \log N \leq w = O(\log N)$, if $t_u = o(\frac{1}{\log B})$, then $t_q = \Omega(\frac{N}{B \log B})$.*

We establish Theorem 2 by working under the cell-probe model. An immediate obstacle is that, every insertion obviously must probe at least one cell (recall that the cell-probe model is *stateless*, i.e., no information is passed between two operations), which is at odds with our goal of having an $o(1)$ bound on t_u. A main idea behind our techniques is to prove a tradeoff between the query cost and the cost of a *group* of $N/B = \Omega(M)$ insertions. Then, by requiring each group to probe $o(N/B)$ cells, we get essentially an $o(1)$ amortized bound on t_u, thus overcoming the obstacle. The tradeoff (between the query and group update costs) is obtained by a novel reduction from set disjointness.

1.2 Previous Work: Lower Bounds in EM with $o(1)$ Update Cost

In the EM model, an important line of research is to understand the limitation of buffering, or more specifically: what is the best query time achievable if the amortized update cost needs to be $o(1)$? Our work belongs to this category of work. In this subsection, we review the existing results under the category to the best of our knowledge.

The *offline* version of the dynamic element distinctness problem, where the goal is to determine if a static set $\mathcal{S}$ has duplicate elements, is known to require at least $c\frac{N}{B}\log_{M/B}\frac{N}{B}$ I/Os, for some constant c, in EM under the indivisibility assumption [2, 3]. This implies the following *dynamic* lower bound: if $t_u \leq \frac{c}{2B}\log_{M/B}\frac{N}{B}$, then $t_q = \Omega(\frac{N}{B}\log_{M/B}\frac{N}{B})$. In turn, this tradeoff implies that no structure (obeying the indivisibility assumption) with $o(1)$ update cost can answer a query faster than sorting when $\log_{M/B}(N/B) = \Omega(B)$, that is, N is exponential in B. In the more realistic settings where N and M are polynomials of B, however, the tradeoff loses its significance because it requires the impossible that $t_u = o(1/B)$. The above discussion also applies to dynamic distinct counting.

There is considerable work [4, 9, 12, 13, 15] in understanding the I/O complexity of the *dynamic membership problem*, where the goal is to maintain a set S of elements under insertions, such that queries of the following form "does element e belong to S?" can be answered efficiently. The lower bounds by Brodal and Fagerberg [4] and Wei et al. [13] were proved under the indivisibility assumption, while the others hold without the assumption.

More specifically, the techniques of Yi and Zhang [15] are geared to establish a tradeoff of the following form (abusing notations slightly, let t_u, t_q be the expected amortized insertion cost and expected query cost respectively also for dynamic membership): if $t_q \leq 1 + \delta$ where δ is a sufficiently small constant, then $t_u = \Omega(1)$. Verbin and Zhang [12] showed a different tradeoff: if $t_u \leq 1 - \epsilon$ where $0 < \epsilon < 1$ is any constant, then $t_q = \Omega(\log_B N)$. Iacono and Patrascu [9] presented an alternative tradeoff between t_u and the worst-case query cost t_q^{worst}: if $t_u \leq 1 - \epsilon$, then $t_q^{worst} = \Omega(\frac{\log N}{\log(B \cdot t_u)})$.

The results in both [9] and [15] were derived from the *chronogram technique* [7, 11]. The hard input consists of an insertion sequence followed by a single query. The sequence is then divided into subsequences, called *epochs*, whose lengths increase geometrically when they are ordered reverse chronologically. The crux of a lower-bound argument is to show that when t_u is small, for every epoch, the query must read at least a block "exclusively belonging to" that epoch with constant probability. Thus, the expected query cost is at least the number of epochs. Unfortunately, the chronogram technique does not appear to be the right tool for dynamic element distinctness. This is because the number of epochs is logarithmic to the number N of insertions, thus making it difficult to prove a super-logarithmic lower bound on the query cost. Our goal, as shown in Theorem 2, is to establish an almost linear lower bound.

Finally, Yi [14] studied *dynamic 1d range reporting*, where the goal is to maintain a set S of elements from an ordered domain under insertions so that the following queries can be answered efficiently: given a range $[e_1, e_2]$ where e_1, e_2 are elements from the domain, report $S \cap [e_1, e_2]$. He presented lower bound tradeoffs between the amortized insertion cost and the query cost of deterministic structures, based on a dynamic version of the *indexability model* [8], and thus, still inheriting the indivisibility assumption.

2 An I/O Lower Bound of Dynamic Element Distinctness

2.1 Cell-Probe Model

The core of Theorem 2 is a lower bound in a cell-probe model of computation defined as follows. The machine is equipped with a CPU and an array of memory *cells* of size wB bits. The CPU has a register of an unbounded size. A *cell probe* either reads or writes a cell. The *cost* of an algorithm is measured as the number of cells probed (CPU calculation is free). The *space* of a structure is measured as the number of cells occupied.

For a deterministic structure, an operation can be modeled as follows. At the beginning, the register contains nothing but the operation's input parameter. At each step, the operation probes a cell c, such that the address of c and whether the probe is a read or a write are both functions of the register. If the probe is a read, the register is updated as a function of the register's current form and the contents of c. If it is a write, then the value written to c is a function of the register, after which the register is updated as a function of its current form. We model a randomized structure by assuming that it has free access to a random bit sequence, and that it behaves as a deterministic structure after the sequence has been fixed.

2.2 Hard Input

Set $N = B^c$ and $M = B^{c'}$ where c and c' can be any integer constants such that $c' \geq 1$ and $c \geq c' + 1$. Fix w to be any integer such that $3 \log N \leq w = O(\log N)$. We use the term (N/B)-*subset* to refer to a set of N/B (distinct) integers from $[2^w]$. We consider a variant of the element distinctness problem where a structure maintains an initially empty multi-set $\mathcal{S}$ under two operations:

- G-INSERT(G): add an (N/B)-subset G to $\mathcal{S}$.
- QUERY: decide whether the elements of $\mathcal{S}$ are distinct.

We refer to the above as the *element distinctness with group insertions* (EDGI) *problem.*

Our hard input consists of a sequence Σ of B *batches*, where each batch has two operations: a G-INSERT(G) followed by a single QUERY, where G is taken uniformly at random from all the $\binom{2^w}{N/B}$ possible (N/B)-subsets. At the end of Σ, $\mathcal{S}$ has exactly N elements, but they are not necessarily distinct. We denote by $\boldsymbol{\Sigma}$ the set of all possible sequences as generated above. Note that Σ follows the uniform distribution over $\boldsymbol{\Sigma}$. We say that a structure uses *polynomial space* if it occupies at most $N^{O(1)}$ cells when it processes any Σ of $\boldsymbol{\Sigma}$. Our discussion will focus only on such structures.

2.3 Set Disjointness

This is a communication complexity problem—denoted as Disj henceforth—defined as follows. Alice is given a subset X of $[2^w]$, and Bob is given a subset Y of $[2^w]$. Their goal is to determine whether $X \cap Y = \emptyset$ by sending each other messages, each being a sequence of bits, according to a pre-agreed protocol Π. For our purpose, it suffices to consider that Π is *deterministic* defined as follows. The person sending the first message is always fixed (regardless of X and Y). Then, Alice and Bob take turns to send messages, such that every message is a function of the previous messages and the sender's input. We denote by $\Pi_A(X, Y)$ the bit sequence concatenating chronologically all the messages sent by Alice on input (X, Y), and by $|\Pi_A(X, Y)|$ the number of bits in $\Pi_A(X, Y)$. Let $\Pi_B(X, Y)$ and $|\Pi_B(X, Y)|$ be defined similarly for Bob.

We denote by $\boldsymbol{\mathcal{I}}$ the set of all possible inputs (X, Y) to Disj. Let μ be a probability density function (pdf) over $\boldsymbol{\mathcal{I}}$, namely, $\mu(X, Y)$ gives the probability that Alice's and Bob's subsets are X and Y, respectively. Define:

$$\alpha_\mu(\Pi) = \sum_{(X,Y)\in\boldsymbol{\mathcal{I}}} |\Pi_A(X, Y)| \cdot \mu(X, Y)$$
$$\beta_\mu(\Pi) = \sum_{(X,Y)\in\boldsymbol{\mathcal{I}}} |\Pi_B(X, Y)| \cdot \mu(X, Y).$$

We call $\alpha_\mu(\Pi)$ the *μ-average Alice cost* of Π, and $\beta_\mu(\Pi)$ the *μ-average Bob cost* of Π.

We will be particularly interested in the scenario where X and Y have specific sizes n and m, respectively, where n and m are integers in $[1, 2^w]$. Let $\boldsymbol{\mathcal{I}}^{n,m}$ represent the set of all possible inputs (X, Y) to Disj such that $|X| = n$ and $|Y| = m$. When the input to Disj is drawn *only* from $\boldsymbol{\mathcal{I}}^{n,m}$, we will refer to Disj as (n, m)-Disj.

We use $\mathsf{unif}^{n,m}$ to denote the uniform distribution μ over $\boldsymbol{\mathcal{I}}^{n,m}$, namely, $\mu(X, Y) = 1/|\boldsymbol{\mathcal{I}}^{n,m}|$ if $(X, Y) \in \boldsymbol{\mathcal{I}}^{n,m}$, while $\mu(X, Y) = 0$ for any $(X, Y) \in \boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{I}}^{n,m}$. The appendix contains a proof for the following tradeoff between $\alpha_{\mathsf{unif}^{n,m}}(\Pi)$ and $\beta_{\mathsf{unif}^{n,m}}(\Pi)$:

▶ **Theorem 3.** *When $3 \cdot \max\{\log n, \log m\} \le w = o(\min\{n, m\})$, for any deterministic protocol Π solving (n, m)-Disj, it must hold that either $\alpha_{\mathsf{unif}^{n,m}}(\Pi) = \Omega(n)$ or $\beta_{\mathsf{unif}^{n,m}}(\Pi) = \Omega(m)$.*

2.4 Cell-Probe Lower Bound for EDGI

Let us fix a *deterministic* polynomial-space cell-probe structure Υ on the EDGI problem. For each sequence $\Sigma \in \mathbf{\Sigma}$, denote by $T(\Sigma)$ the cost of Υ (in the number of cells probed) in processing Σ. Note that $T(\Sigma)$ is a random variable because Σ is uniformly distributed in $\mathbf{\Sigma}$. Next, we analyze the expectation of $T(\Sigma)$.

Recall that Σ consists of B batches, each of which has a G-Insert operation followed by a query. Let G_i ($1 \le i \le B$) be the (N/B)-subset added to $\mathcal{S}$ by the G-Insert operation in the i-th batch of Σ. We say that Σ is *i-distinct* if $G_j \cap G_{j'} = \emptyset$ for each pair of j, j' satisfying $1 \le j < j' \le i$. Denote by $T_i(\Sigma)$ the number of cells probed by Υ in handling the i-th batch. Thus, $T(\Sigma) = \sum_i T_i(\Sigma)$. By the linearity of expectation, we know: $\mathbf{E}[T(\Sigma)] = \sum_i \mathbf{E}[T_i(\Sigma)]$.

We prove in Section 3 the following relationship between set disjointness and EDGI:

▶ **Lemma 4.** *For each $i \in [1, B-1]$, there exists a deterministic protocol Π solving the $(N/B, iN/B)$-*Disj *problem with $\alpha_{\mathsf{unif}^{N/B,iN/B}}(\Pi) \le \lambda \cdot O(\log N)$ and $\beta_{\mathsf{unif}^{N/B,iN/B}}(\Pi) \le \lambda \cdot wB$, where $\lambda = \mathbf{E}[T_{i+1}(\Sigma) \mid \Sigma$ is i-distinct$]$.*

Combining Lemma 4 and Theorem 3, we have the following when $w \ge 3\log N$:

$$\mathbf{E}[T_{i+1}(\Sigma) \mid \Sigma \text{ is } i\text{-distinct}] \quad = \quad \Omega\left(\min\left\{\frac{N}{B\log N}, \frac{iN}{wB^2}\right\}\right). \tag{1}$$

▶ Observation 5. When $w \ge 3\log N$, it must hold that $\mathbf{Pr}[\Sigma \text{ is } B\text{-distinct}] \ge 1/2$.

Proof.

$$\begin{aligned}
\mathbf{Pr}[\Sigma \text{ is } B\text{-distinct}] &\ge 1 - \sum_{1 \le i < j \le B} \sum_{x \in [2^w]} \mathbf{Pr}[x \in G_i \ \wedge \ x \in G_j] \\
&= 1 - \binom{B}{2} \cdot 2^w \cdot \left(\frac{N/B}{2^w}\right)^2 \\
&\ge 1 - \frac{1}{2}B^2 \cdot 2^w \cdot \frac{N^2}{B^2 \cdot 2^{2w}} \\
&\ge 1/2. \qquad \text{(by } w \ge 3\log N\text{)}
\end{aligned}$$

◀

When $3\log N \le w = \Theta(\log N)$, we have from the above:

$$\begin{aligned}
\mathbf{E}[T(\Sigma)] &\ge \sum_{i=1+B/2}^{B} \mathbf{E}[T_i(\Sigma)] \\
&\ge \sum_{i=B/2}^{B-1} \mathbf{E}[T_{i+1}(\Sigma) \mid \Sigma \text{ is } i\text{-distinct}] \cdot \mathbf{Pr}[\Sigma \text{ is } i\text{-distinct}] \\
&\ge (1/2) \sum_{i=B/2}^{B-1} \mathbf{E}[T_{i+1}(\Sigma) \mid \Sigma \text{ is } i\text{-distinct}] \\
&\quad \text{(as } \mathbf{Pr}[\Sigma \text{ is } i\text{-distinct}] \ge \mathbf{Pr}[\Sigma \text{ is } B\text{-distinct}]\text{)} \\
&\ge (1/2) \sum_{i=B/2}^{B-1} \Omega\left(\frac{N}{B\log N}\right) \qquad \text{(by (1))} \\
&= \Omega\left(\frac{N}{\log N}\right)
\end{aligned} \tag{2}$$

where the last inequality used the fact that $\frac{N}{B \log N} = \Theta(\frac{iN}{wB^2})$ for $i \in [B/2, B]$ and $w = \Theta(\log N)$. Now it is easy to obtain the following lower bound:

▶ **Lemma 6.** *Given $N = B^c$ for some integer constant $c \geq 2$, and w such that $3 \log N \leq w = O(\log N)$, for any Las Vegas polynomial-space structure, there is at least a sequence $\Sigma_0 \in \mathbf{\Sigma}$ such that the structure requires $\Omega(N/\log B)$ expected cost processing Σ_0.*

Proof. We have proved that when Σ is uniformly distributed in $\mathbf{\Sigma}$, *every* deterministic structure must incur $\Omega(N/\log N) = \Omega(N/\log B)$ expected cost in processing Σ (applying (2) and $\log N = \Theta(\log B)$). The lemma then follows from Yao's minimax principle. ◀

2.5 From Cell-Probe to EM

Still set $N = B^c$ and $M = B^{c'}$ (for integer constants $c' \geq 1$ and $c \geq c' + 1$), and $3 \log N \leq w = O(\log N)$. In this scenario, next we show that if an EM structure Υ' solves the dynamic element distinctness problem with expected amortized insertion cost t_u and expected query cost t_q, then there is a cell-probe structure Υ for the EDGI problem, such that for every sequence $\Sigma \in \mathbf{\Sigma}$, Υ processes Σ in $t_u N + t_q B + 4M$ expected cost.

The main idea is to simulate Υ' in the cell-probe model by setting aside M/B fixed cells to preserve the memory of Υ' across two operations—refer to those cells as the *state cells.* In addition, at any moment, every disk block occupied by Υ' corresponds to a unique cell with the same contents. At the beginning, Υ (in the cell-probe model) and Υ' (in the EM model) are empty, and so are the state cells. Υ behaves according to how Υ' works. Consider, in general, the processing of the i-th ($1 \leq i \leq B$) batch (G-INSERT(G), QUERY) of Σ. Υ first reads the M/B state cells so that the (cell-probe) CPU register contains all the information in the memory of Υ' (in EM). Then, we invoke the insertion algorithm of Υ' to insert the N/B elements of G to Υ' (the ordering does not matter). In this process, (i) whenever Υ' reads (writes) a block, Υ reads (writes) the block's corresponding cell; (ii) whenever Υ' performs CPU computation, Υ performs the same computation in the register. After Υ' has finished inserting the elements of G, Υ writes the memory of Υ' to the state cells with cost M/B. At this point, Υ has completed the operation G-INSERT(G). In the same manner, Υ handles a QUERY by first reading the state cells, simulating Υ', and then writing the state cells. By definitions of t_u and t_q, Υ' performs at most $t_u N + t_q B$ expected I/Os in doing N insertions and B queries. Hence, Υ probes at most $t_u N + t_q B + 4M$ cells in expectation, noticing that $4M/B$ probes are needed for reading and writing the state cells in each batch of Σ.

We thus know from Lemma 6 that $t_u N + t_q B + 4M = \Omega(N/\log B)$, which gives $t_u + \frac{t_q B}{N} + 4/B = \Omega(1/\log B)$ as $N \geq MB$. Thus, if $t_u = o(1/\log B)$, then t_q must be $\Omega(\frac{N}{B \log B})$, as claimed in Theorem 2.

Remark. Echoing the high level description in Section 1.1, it is worth mentioning that Lemma 6 implies an update-query tradeoff for EDGI. Let τ_u be the expected amortized G-INSERT cost of an EDGI structure, and τ_q be its expected query cost. If the structure uses polynomial space, the lemma shows that $\tau_u B + \tau_q B = \Omega(N/\log B)$.

3 Reduction from Set Disjointness to EDGI

In this section, which serves as a proof of Lemma 4, let us set $n = N/B$ and $m = iN/B$. As before, denote by Υ the (deterministic) EDGI structure in the context of the lemma. Consider the $\mathcal{S}$ after having processed the first i batches of Σ. Note that $\mathcal{S}$ is a set (as opposed to a multi-set) because Σ is i-distinct. Let G be the (N/B)-subset to be inserted in the $(i+1)$-th batch. Clearly, $\mathcal{S} \cap G = \emptyset$ if and only if the query of the $(i+1)$-th batch returns "distinct". Based on this observation, next we design a protocol for the $(N/B, iN/B)$-Disj problem that works by asking Alice and Bob to simulate the execution of Υ.

Let us first introduce some notions useful in the subsequent discussion. Let $G_1, ..., G_i$ be (N/B)-subsets of $[2^w]$ that are mutually disjoint. These i subsets make an *i-distinct sequence* $\mathcal{G} = (G_1, G_2, ..., G_i)$. In general, given $\mathcal{G}$, we define $S(\mathcal{G}) = \cup_{j=1}^{i} G_j$. Let $\boldsymbol{\mathcal{G}}$ be the set of all different i-distinct sequences. Define $g = |\boldsymbol{\mathcal{G}}|$, which equals $\frac{(2^w)!}{(2^w-m)!((N/B)!)^i}$. *Conditioned on* the fact that Σ is i-distinct, every i-distinct sequence has $1/g$ probability of being the sequence of (N/B)-subsets that are added to $\mathcal{S}$ by the first i batches of Σ. Furthermore, since Υ is deterministic, every $\mathcal{G}$ defines an instance of Υ—denoted as $\Upsilon(\mathcal{G})$—which is exactly the cell contents after Υ has processed i batches: $(\text{G-}\textsc{Insert}(G_1), \textsc{Query})$, ..., $(\text{G-}\textsc{Insert}(G_i), \textsc{Query})$.

Fix a $\mathcal{G}$ and therefore $\Upsilon(\mathcal{G})$. Given an (N/B)-subset X, let $C(\Upsilon(\mathcal{G}), X)$ be the cost of processing batch $(\text{G-}\textsc{Insert}(X), \textsc{Query})$ on $\Upsilon(\mathcal{G})$. Define:

$$C(\mathcal{G}) \quad = \quad \frac{1}{\binom{2^w}{n}} \sum_{X} C(\Upsilon(\mathcal{G}), X). \tag{3}$$

In other words, $C(\mathcal{G})$ is the average $C(\Upsilon(\mathcal{G}), X)$ over all the possible X.

Now, fix Y as an (iN/B)-subset of $[2^w]$. Consider the set $Z(Y)$ of i-distinct sequences $\mathcal{G}$ such that $S(\mathcal{G}) = Y$, that is, each $\mathcal{G} \in Z(Y)$ is a possible way to break Y into a sequence of (N/B)-subsets. Let $z = \min_{\mathcal{G} \in Z(Y)} C(\mathcal{G})$. Define $\mathcal{G}(Y)$ to be the $\mathcal{G} \in Z(Y)$ with the smallest $C(\mathcal{G})$; if multiple i-distinct sequences $\mathcal{G}$ of $Z(Y)$ satisfy $C(\mathcal{G}) = z$, define $\mathcal{G}(Y)$ to be the first one among them in lexicographical order. Note that $\mathcal{G}(Y)$ is indeed a function of Y by the determinism of Υ.

We are now ready to describe a protocol for (n, m)-Disj. Let X and Y be the inputs of Alice and Bob, respectively. Bob first identifies $\mathcal{G}(Y)$ and builds a structure $\Upsilon = \Upsilon(\mathcal{G}(Y))$ locally with no communication. Then, Alice simulates what the cell-probe CPU does to perform a batch $(\text{G-}\textsc{Insert}(X), \textsc{Query})$ on Υ. Specifically, she maintains locally a set R of cells to simulate the CPU register. R is initially empty. Whenever G-$\textsc{Insert}(X)$ writes a cell c, she does not communicate any bit, but simply adds c to R, i.e., R remembers the address and contents of c. Whenever G-$\textsc{Insert}(X)$ reads a cell c, Alice first checks whether $c \in R$. If so, she takes the contents of c directly from R. Otherwise, she requests c from Bob by sending $O(\log N)$ bits to address c—note that $O(\log N)$ suffices because Υ uses polynomial space—and then, Bob sends back the contents of c in wB bits. *Without* clearing R, Alice proceeds to simulate $\textsc{Query}$ in the same manner. At the end, if $\textsc{Query}$ reports "disjoint", Alice notifies Bob in one bit that $X \cap Y = \emptyset$; otherwise, her notification bit indicates $X \cap Y \neq \emptyset$.

We argue that the protocol, denoted as Π, has low $\alpha_{\mathsf{unif}^{n,m}}(\Pi)$ and $\beta_{\mathsf{unif}^{n,m}}(\Pi)$. Note that the protocol executes in *rounds*, in each of which Alice and Bob communicate $O(\log N)$ and wB bits, respectively. Let $r(X, Y)$ be the number of rounds in the protocol on input (X, Y). Define $\bar{r} = \frac{1}{\binom{2^w}{n}\binom{2^w}{m}} \sum_{X,Y} r(X, Y)$, namely, $\bar{r}$ is the average number of rounds over all the inputs to (n, m)-Disj. Hence, $\alpha_{\mathsf{unif}^{n,m}}(\Pi) \leq \bar{r} \cdot O(\log N)$ and $\beta_{\mathsf{unif}^{n,m}}(\Pi) \leq \bar{r} \cdot wB$.

Since our protocol ensures $r(X, Y) \leq C(\Upsilon(\mathcal{G}(Y)), X)$, the following relationship becomes obvious:

$$\bar{r} \quad \leq \quad \frac{1}{\binom{2^w}{n}\binom{2^w}{m}} \sum_{X,Y} C(\Upsilon(\mathcal{G}(Y)), X). \tag{4}$$

Next, we complete the proof of Lemma 4 by showing that the right hand side of (4) is at most $\mathbf{E}[T_{i+1}(\Sigma) \mid \Sigma \text{ is } i\text{-distinct}]$ through the following derivation:

$$\begin{aligned}
\frac{1}{\binom{2^w}{n}\binom{2^w}{m}} \sum_{X,Y} C(\Upsilon(\mathcal{G}(Y)), X) &\leq \frac{1}{\binom{2^w}{n}\binom{2^w}{m}} \sum_{X,Y} \left(\frac{((N/B)!)^i}{m!} \sum_{\mathcal{G} \in Z(Y)} C(\Upsilon(\mathcal{G}), X) \right) \\
&\quad \text{(Note: } |Z(Y)| = m!/((N/B)!)^i \text{, and} \\
&\quad C(\Upsilon(\mathcal{G}(Y)), X) \leq C(\Upsilon(\mathcal{G}), X) \text{ for every } \mathcal{G} \in Z(Y)) \\
&= \frac{1}{\binom{2^w}{n}\binom{2^w}{m}} \frac{((N/B)!)^i}{m!} \sum_{X, \mathcal{G} \in \boldsymbol{\mathcal{G}}} C(\Upsilon(\mathcal{G}), X) \\
&\quad \text{(Note: every } \mathcal{G} \in \boldsymbol{\mathcal{G}} \text{ belongs to the } Z(Y) \text{ of a unique } Y) \\
&= \frac{1}{\binom{2^w}{n} \cdot g} \sum_{X, \mathcal{G} \in \boldsymbol{\mathcal{G}}} C(\Upsilon(\mathcal{G}), X) \\
&= \mathbf{E}[T_{i+1}(\Sigma) \mid \Sigma \text{ is } i\text{-distinct}].
\end{aligned}$$

References

1 Alok Aggarwal and Jeffrey Scott Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM (CACM)*, 31(9):1116–1127, 1988.

2 Lars Arge, Mikael Knudsen, and Kirsten Larsen. A general lower bound on the I/O-complexity of comparison-based algorithms. In *Algorithms and Data Structures Workshop (WADS)*, pages 83–94, 1993.

3 Lars Arge and Peter Bro Miltersen. On showing lower bounds for external-memory computational geometry problems. *DIMACS Series in Discrete Mathematics*, pages 139–159.

4 Gerth Stølting Brodal and Rolf Fagerberg. Lower bounds for external memory dictionaries. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 546–554, 2003.

5 Anirban Dasgupta, Ravi Kumar, and D. Sivakumar. Sparse and lopsided set disjointness via information theory. In *APPROX-RANDOM*, pages 517–528, 2012.

6 Erik D. Demaine, Friedhelm Meyer auf der Heide, Rasmus Pagh, and Mihai Patrascu. De dictionariis dynamicis pauco spatio utentibus (*lat.* on dynamic dictionaries using little space). In *Latin American Symposium on Theoretical Informatics (LATIN)*, pages 349–361, 2006.

7 Michael L. Fredman and Michael E. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 345–354, 1989.

8 Joseph M. Hellerstein, Elias Koutsoupias, Daniel P. Miranker, Christos H. Papadimitriou, and Vasilis Samoladas. On a model of indexability and its bounds for range queries. *Journal of the ACM (JACM)*, 49(1):35–55, 2002.

9 John Iacono and Mihai Patrascu. Using hashing to solve the dictionary problem. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 570–582, 2012.

10 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2010.

11 Mihai Patrascu and Erik D. Demaine. Lower bounds for dynamic connectivity. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 546–553, 2004.

12 Elad Verbin and Qin Zhang. The limits of buffering: a tight lower bound for dynamic membership in the external memory model. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 447–456, 2010.
13 Zhewei Wei, Ke Yi, and Qin Zhang. Dynamic external hashing: the limit of buffering. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 253–259, 2009.
14 Ke Yi. Dynamic indexability and the optimality of B-trees. *Journal of the ACM (JACM)*, 59(4), 2012.
15 Ke Yi and Qin Zhang. On the cell probe complexity of dynamic membership. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 123–133, 2010.

Appendix: Proof of Theorem 3

Set $W = 2^w$. Recall that the set disjointness problem Disj was defined by having (X, Y) drawn from the domain of $[W]$. We will from now on denote the problem as Disj_W, namely, by indicating the domain size explicitly. Also, it will be convenient to regard an input (X, Y) to Disj_W as a pair of W-dimensional vectors $\boldsymbol{x} = (x_0, x_1, ..., x_{W-1})$ and $\boldsymbol{y} = (y_0, y_1, ..., y_{W-1})$ such that for each $i \in [W]$:

$$x_i = 1 \text{ if } i \in X, \text{ or } = 0 \text{ otherwise;}$$
$$y_i = 1 \text{ if } i \in Y, \text{ or } = 0 \text{ otherwise.}$$

We will use $|\boldsymbol{x}|$ to denote the number of 1's in $\boldsymbol{x}$ (similarly, $|\boldsymbol{y}|$).

We will build our proof on a result of [5] of Dasgupta, Kumar, and Sivakumar. They considered Disj_W under a distribution of $(\boldsymbol{x}, \boldsymbol{y})$ parameterized by real values $p, q \in [0, 1]$—denoted as $\mathsf{rand}_W^{p,q}$—where, independently for each $i \in [W]$: (i) x_i equals 1 with probability p, and (ii) independently, $y_i = 1$ with probability q. They proved[3]:

▶ **Lemma 7** ([5]). *For any deterministic protocol Π solving Disj_W, it holds that either $\alpha_{\mathsf{rand}_W^{p,q}}(\Pi) = \Omega(pW)$ or $\beta_{\mathsf{rand}_W^{p,q}}(\Pi) = \Omega(qW)$.*

Theorem 3, on the other hand, is concerned with distribution $\mathsf{unif}_W^{n,m}$, that is, the uniform distribution over the set of all inputs $(\boldsymbol{x}, \boldsymbol{y}) \in \{0, 1\}^W \times \{0, 1\}^W$ satisfying $|\boldsymbol{x}| = n$ and $|\boldsymbol{y}| = m$. We will defer the proof of the next lemma till later:

▶ **Lemma 8.** *Suppose that $\max\{n, m\} \le \sqrt{W}/2$, and that we are given a deterministic protocol Π for (n, m)-Disj_W. Then, for any $n' \in [n/2, n]$ and $m' \in [m/2, m]$, there is a deterministic protocol $\Pi^{n',m'}$ for (n', m')-$\mathsf{Disj}_{W/2}$ such that*

$$\alpha_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{n',m'}) < 3 \cdot \alpha_{\mathsf{unif}_W^{n,m}}(\Pi), \text{ and } \beta_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{n',m'}) < 3 \cdot \beta_{\mathsf{unif}_W^{n,m}}(\Pi).$$

Now, given a deterministic protocol Π solving (n, m)-Disj_W, we design a protocol Π' for $\mathsf{Disj}_{W/2}$. Given an input $(\boldsymbol{x}', \boldsymbol{y}')$ to $\mathsf{Disj}_{W/2}$, Π' works as follows:

- If $|\boldsymbol{x}'| \notin [n/2, n]$, Alice sends $\boldsymbol{x}$ to Bob, who then sends back the result with one more bit. Conversely, if $|\boldsymbol{y}'| \notin [m/2, m]$, Bob sends $\boldsymbol{y}$ to Alice, who then sends back the result.
- In all other cases, Alice and Bob send $n' = |\boldsymbol{x}'|$ and $m' = |\boldsymbol{y}'|$, respectively, after which they run the protocol $\Pi^{n',m'}$ of Lemma 8 that is obtained from Π for (n', m')-$\mathsf{Disj}_{W/2}$.

[3] By combining Lemmas 5-8 of [5] with parameters $\theta = 0$, $\alpha = p$ and $\beta = q$.

Next we argue that Π is efficient under the distribution $\mathsf{rand}_{W/2}^{p,q}$ with

$$p = (3n/4)/(W/2), \quad \text{and} \quad q = (3m/4)/(W/2).$$

Define $\mathtt{out}$ to be the event where either $|\boldsymbol{x}'| \notin [n/2, n]$ or $|\boldsymbol{y}'| \notin [m/2, m]$ holds. Clearly:

$$\alpha_{\mathsf{rand}_{W/2}^{p,q}}(\Pi') \quad \leq \quad \mathbf{Pr}[\mathtt{out}](O(1) + W/2) + \mathbf{Pr}[\neg\mathtt{out}](2\log W + \alpha'),$$

where

$$\alpha' = \sum_{\substack{n' \in [\frac{n}{2}, n], \\ m' \in [\frac{m}{2}, m]}} \left(\mathbf{Pr}\left[|\boldsymbol{x}'| = n', |\boldsymbol{y}'| = m' \mid \neg\mathtt{out}\right] \cdot \alpha_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{n',m'}) \right) \leq 3 \cdot \alpha_{\mathsf{unif}_{w}^{n,m}}(\Pi). \quad \text{(by Lemma 8)}$$

By Chernoff's bounds, $\mathbf{Pr}[\mathtt{out}] \leq e^{-\Omega(n)}$ which, together with $w = o(n)$, gives $\mathbf{Pr}[\mathtt{out}]W/2 = o(n)$. Therefore:

$$\alpha_{\mathsf{rand}_{W/2}^{p,q}}(\Pi') = o(n) + 2\log W + 3 \cdot \alpha_{\mathsf{unif}_{w}^{n,m}}(\Pi) = o(n) + 3 \cdot \alpha_{\mathsf{unif}_{w}^{n,m}}(\Pi).$$

Similar analysis shows that

$$\beta_{\mathsf{rand}_{W/2}^{p,q}}(\Pi') = o(m) + 3\beta_{\mathsf{unif}_{W}^{n,m}}(\Pi).$$

On the other hand, Lemma 7 states that either $\alpha_{\mathsf{rand}_{W/2}^{p,q}}(\Pi') = \Omega(pW/2) = \Omega(n)$ or $\beta_{\mathsf{rand}_{W/2}^{p,q}}(\Pi') = \Omega(qW/2) = \Omega(m)$. Therefore, either $\alpha_{\mathsf{unif}_{W}^{n,m}}(\Pi) = \Omega(n)$ or $\beta_{\mathsf{unif}_{W}^{n,m}}(\Pi) = \Omega(m)$, as claimed.

Proof of Lemma 8

It suffices to consider that Π never incurs more than W bits of communication. Let us define T to be the set consisting of all triplets (S, U, V) satisfying both conditions below:

- S, U, and V are pairwise disjoint subsets of $[W]$;
- $|S| = W/2$, $|U| = n - n'$, and $|V| = m - m'$.

Given a triplet (S, U, V), we design a protocol Π^{SUV} for $\mathsf{Disj}_{W/2}$ as follows. Upon an input $(\boldsymbol{x}', \boldsymbol{y}') \in \{0,1\}^{W/2} \times \{0,1\}^{W/2}$, Π^{SUV} runs Π on $(\boldsymbol{x}, \boldsymbol{y})$ where

- $\boldsymbol{x}$ is the W-dimensional vector such that
  - the projection of $\boldsymbol{x}$ onto the subspace defined by S gives $\boldsymbol{x}'$;
  - $x_i = 1$ for all $i \in U$; $x_i = 0$ for all $i \in [W] - S - U$.
- $\boldsymbol{y}$ is defined in the same way after replacing $\boldsymbol{x}'$ with $\boldsymbol{y}'$, and U with V.

It is easy to verify that Π^{SUV} is correct on all $(\boldsymbol{x}', \boldsymbol{y}')$.

For each $b \in \{0,1\}$ and any $\boldsymbol{x} \in \{0,1\}^W$, define $\boldsymbol{x}^b = \{i \in [W] \mid x_i = b\}$, namely, the set of dimensions on which $\boldsymbol{x}$ takes the value b. Now, we induce from $\mathsf{unif}_W^{n,m}$ a distribution ν on T, by using the following 5-step process to generate a triplet (S, U, V):

1. Pick $(\boldsymbol{x}, \boldsymbol{y})$ according to $\mathsf{unif}_W^{n,m}$, subject to the constraint that $|\boldsymbol{x}^1 \cap \boldsymbol{y}^1| \leq \min\{n', m'\}$.
2. Pick uniformly at random $U \subseteq \boldsymbol{x}^1 \cap \boldsymbol{y}^0$ with $|U| = n - n'$ (this is always possible because $|\boldsymbol{x}^1 \cap \boldsymbol{y}^0| = n - |\boldsymbol{x}^1 \cap \boldsymbol{y}^1| \geq n - n'$).

3. Pick uniformly at random $V \subseteq \boldsymbol{x}^0 \cap \boldsymbol{y}^1$ with $|V| = m - m'$.
4. Pick uniformly at random $S' \subseteq \boldsymbol{x}^0 \cap \boldsymbol{y}^0$ with $|S'| = W/2 - |U| - |V|$ (this is always possible because $|\boldsymbol{x}^0 \cap \boldsymbol{y}^0| = W - |\boldsymbol{x}^1 \cup \boldsymbol{y}^1| \geq W - (n+m) > W/2$).
5. Finally, let $S = [W] - U - V - S'$ (note that $|S|$ is always $W/2$).

Denote by δ the probability that an input $(\boldsymbol{x}, \boldsymbol{y})$ drawn from $\mathsf{unif}_W^{n,w}$ satisfies $|\boldsymbol{x}^1 \cap \boldsymbol{y}^1| > \min\{n', m'\}$. Applying the fact $n' \geq n/2$, we know

$$\mathbf{Pr}[|\boldsymbol{x}^1 \cap \boldsymbol{y}^1| > n'] \leq \binom{n}{n/2}\binom{W - n/2}{m - n/2} \Big/ \binom{W}{m} \leq \binom{n}{n/2}\left(\frac{m - n/2}{W}\right)^{n/2} \leq (4/\sqrt{W})^{n/2},$$

and similarly $\mathbf{Pr}[|\boldsymbol{x}^1 \cap \boldsymbol{y}^1| > m'] \leq (4/\sqrt{W})^{m/2}$. Therefore,

$$\delta \leq \mathbf{Pr}[|\boldsymbol{x}^1 \cap \boldsymbol{y}^1| > n'] + \mathbf{Pr}[|\boldsymbol{x}^1 \cap \boldsymbol{y}^1| > m'] = o(1/W).$$

Observe that

$$\begin{aligned}
\alpha_{\mathsf{unif}_W^{n,m}}(\Pi) &= \mathbf{E}_{(\boldsymbol{x},\boldsymbol{y}) \leftarrow \mathsf{unif}_W^{n,m}}\left[\left|\Pi_A(\boldsymbol{x},\boldsymbol{y})\right|\right] \\
&\geq (1-\delta) \cdot \mathbf{E}_{(S,U,V) \leftarrow \nu} \mathbf{E}_{(\boldsymbol{x}',\boldsymbol{y}') \leftarrow \mathsf{unif}_{W/2}^{n',m'}}\left[\left|\Pi_A^{SUV}(\boldsymbol{x}',\boldsymbol{y}')\right|\right] \\
&= (1-\delta) \cdot \mathbf{E}_{(S,U,V) \leftarrow \nu}\left[\alpha_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{SUV})\right].
\end{aligned}$$

Turning this around gives

$$\mathbf{E}_{(S,U,V) \leftarrow \nu}\left[\alpha_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{SUV})\right] \leq \frac{\alpha_{\mathsf{unif}_W^{n,m}}(\Pi)}{1-\delta} = (1+o(1)) \cdot \alpha_{\mathsf{unif}_W^{n,m}}(\Pi).$$

By Markov's inequality, if we draw (S,U,V) from ν, $\alpha_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{SUV}) \leq 2.5(1+o(1)) \cdot \alpha_{\mathsf{unif}_W^{n,m}}(\Pi) < 3\alpha_{\mathsf{unif}_W^{n,m}}(\Pi)$ holds with probability at least probability $1 - 1/2.5 = 0.6$. Similarly, with probability at least 0.6 we have $\beta_{\mathsf{unif}_{W/2}^{n',m'}}(\Pi^{SUV}) < 3\beta_{\mathsf{unif}_W^{n,m}}(\Pi)$. Therefore, both are true simultaneously with a positive probability, thus completing the proof.