

# Low-Contention Data Structures<sup>☆</sup>

James Aspnes<sup>a,1</sup>, David Eisenstat<sup>2</sup>, Yitong Yin<sup>b,3,\*</sup>

<sup>a</sup>*Department of Computer Science, Yale University, New Haven, CT 06511, USA.*

<sup>b</sup>*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China.*

---

## Abstract

We consider the problem of minimizing contention in static (read-only) dictionary data structures, where contention is measured with respect to a fixed query distribution by the maximum expected number of probes to any given cell. The query distribution is known by the algorithm that constructs the data structure but not by the algorithm that queries it. Assume that the dictionary has  $n$  items. When all queries in the dictionary are equiprobable, and all queries not in the dictionary are equiprobable, we show how to construct a data structure in  $O(n)$  space where queries require  $O(1)$  probes and the contention is  $O(1/n)$ . Asymptotically, all of these quantities are optimal. For *arbitrary* query distributions, we construct a data structure in  $O(n)$  space where each query requires  $O(\log n / \log \log n)$  probes and the contention is  $O(\log n / (n \log \log n))$ . The lack of knowledge of the query distribution by the query algorithm prevents perfect load leveling in this case: for a large class of algorithms, we present a lower bound, based on VC-dimension, that shows that for a wide range of data structure problems, achieving contention even within a polylogarithmic factor of optimal requires a cell-probe complexity of  $\Omega(\log \log n)$ .

*Keywords:* memory contention, data structure, cell-probe model

---

<sup>☆</sup>A preliminary version of this article appeared in the *Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures*, pp 345-354.

\*Corresponding author

*Email addresses:* [aspnes@cs.yale.edu](mailto:aspnes@cs.yale.edu) (James Aspnes), [eisenstatdavid@gmail.com](mailto:eisenstatdavid@gmail.com) (David Eisenstat), [yinyt@nju.edu.cn](mailto:yinyt@nju.edu.cn) (Yitong Yin)

<sup>1</sup>Supported in part by NSF grants CNS-0435201 and CCF-0916389.

<sup>2</sup>Part of the work was done while David Eisenstat was a graduate student at Brown University being supported in part by NSF grant CCF-0964037.

<sup>3</sup>Supported by the National Science Foundation of China under Grant No. 61003023 and No. 61021062.

---

## 1. Introduction

On shared-memory parallel computers, avoiding simultaneous accesses by multiple processors to the same memory locations—*contention*—is one of the keys to achieving high performance [1, 2]. Dwork, Herlihy, and Waarts [3] introduced the first formal measure of contention, though models of parallel computation that restrict contention had been proposed prior to that work. The original Parallel Random-Access Machine (PRAM) model of Fortune and Wyllie [4] simply disallows concurrent writes (CREW PRAM), while the LogP model of Culler *et al.* [5] is intended to capture the physical limits that make contention a practical concern.

In this paper, we study the trade-offs in designing *static* (read-only) data structures for which the memory contention caused by parallel queries is a consideration. With respect to a particular data structure, query distribution, and query algorithm, we introduce a notion of contention to the cell-probe model of Yao [6], where accesses to cells of the data structure have unit cost and all other computation is free. We assume that processors cannot communicate and thus that the expected rate at which a cell  $c$  is probed is proportional to the probability that a single random query accesses  $c$ , which we define to be the contention of  $c$ . This simple model is not intended to capture all details of real memory systems, where the caching and scheduling of accesses can have a large effect on contention, but we believe that it is a reasonable first step towards including contention effects in previous theoretical models.

As an illustration, consider using binary search to look for a random key  $k \leftarrow \{1, 2, \dots, 2^\ell - 1\}$  in a sorted table  $2, 4, \dots, 2^\ell - 2$ . Though each query requires at most  $\ell$  probes, the cell containing  $2^{\ell-1}$  is accessed with probability 1, and the cells containing  $2^{\ell-1} \pm 2^{\ell-2}$  are accessed with probability  $1/2 - 1/(2^\ell - 1)$ . This handful of cells with constant contention is typical of algorithms not specifically designed for our model. By comparing  $k$ , however, with a random element in the interval of possibilities rather than the middle one, we reduce the maximum contention to  $O(\ell/2^\ell)$ , at the cost of increasing the expected number of probes by a constant factor.

Observe that the modified search algorithm has low contention only for certain distributions of  $k$ : given that  $k < 4$ , for example, the cell containing 2 is probed with probability 1. We assume that the algorithm constructing the data structure knows the query distribution, as otherwise, each individual

query must have low contention, which seems difficult to achieve in a space-efficient manner.

We consider how to avoid this problem require *static* data structures, where the data structure is built in advance by a construction algorithm that may know the query distribution, but queries are performed by a fixed algorithm that does not (although it may use randomization itself to spread the query load more evenly).

The assumption that the query algorithm does not know the query distribution is natural. Often the query distribution will be highly correlated with the contents of the data structure, as in our simplest case where we consider a uniform distribution on successful queries to a static dictionary. Providing the query distribution to the query algorithm in such a case would, in effect, give it significant information about the contents of the data structure.

Nonetheless, the query algorithm can exploit information about the query distribution that is encoded in the data structure itself. This is in fact necessary for highly lopsided query distributions, as otherwise, popular elements of the data structures would have high contention. We show how to transform an arbitrary data structure and an arbitrary query distribution into a new structure with asymptotically optimal load balance on that distribution, at the cost of increasing the time and space complexity by a factor of  $O(\log n / \log \log n)$ .

This overhead apparently arises from a tension between the query algorithm's need to spread its probes evenly and its need to obtain useful information. For general properties of the data structure that do not depend much on the query distribution (e.g., hash function parameters), we can achieve low contention by randomly distributing probes over many replicas of the desired data. As we zoom in on the answer to a specific query, however, the location of each probe becomes correlated with a query drawn from a distribution that may be badly skewed. We can ameliorate this problem to some extent by replicating data for more popular queries, but the information about which queries are more popular (and thus more replicated) itself must be obtained from the table and will itself provide a source of high contention if we are not careful. We show that this problem is unavoidable without incurring some overhead in time; specifically, to achieve maximum contention  $O(\log^c n/s)$ , where  $n$  is the number of elements in a table and  $s$  is the space, we must perform  $\Omega(\log \log n)$  probes per query. The problem of closing the gap between this result and our algorithms remains open.

### 1.1. Model

Formally, a data structure problem is a function  $f : Q \times \mathcal{D} \rightarrow \{0, 1\}$ , such that for every query  $x \in Q$  and every data set  $S \in \mathcal{D}$ ,  $f(x, S)$  specifies the answer to the query  $x$  to data set  $S$ . A classic problem is the **membership problem**, where  $Q = [N]$  and  $\mathcal{D} = \binom{[N]}{n}$  for some  $N \gg n$ , and  $f(x, S) = 1$  if and only if  $x \in S$ .

We assume that the query  $x \in Q$  follows some probability distribution  $q$  over  $Q$ .

For any data set  $S \in \mathcal{D}$  and any query distribution  $q$  over  $Q$ , a table  $T_{S,q} : [s] \rightarrow \{0, 1\}^b$  of  $s$  cells, each of which contains  $b$  bits, is prepared. Given a query  $x$ , a probabilistic cell-probing algorithm  $\mathcal{A}$  computes the value of  $f(x, S)$  by making  $t$  randomized adaptive cell-probes  $I_x^{(1)}, I_x^{(2)}, \dots, I_x^{(t)} \in [s]$ . The algorithm  $\mathcal{A}$  may depend on  $f$ , but not on  $S$  or  $q$  (except to the extent that later probes may depend on the outcome of earlier probes, whose results might encode information about  $S$  and  $q$ ).

The **contention** of a cell is the expected number of probes to the cell during one execution of  $\mathcal{A}$ . This will be equal to the probability that the cell is probed at all, provided  $\mathcal{A}$  is sensible enough not to probe the same cell twice, but it is easier to work with expectations. In more detail:

**Definition 1.** For a fixed table  $T_{S,q}$ , for a query  $X$  chosen randomly from  $Q$  with distribution  $q$ , the sequence of cell-probes is  $I_X^{(1)}, I_X^{(2)}, \dots, I_X^{(t)}$ . Let  $Y^{(t)}(X, j)$  be the 0-1 valued random variable indicating whether  $I_X^{(t)} = j$ . The contention of cell  $j$  at step  $t$  is defined by

$$\Phi_t(j) := \mathbb{E} \left[ Y^{(t)}(X, j) \right],$$

where the expectation is taken over both  $X$  and the random  $I_X^{(t)}$ . The total contention of cell  $j$  is  $\Phi(j) := \sum_t \Phi_t(j)$ .

It is obvious that  $\sum_j \Phi_t(j) = 1$ , therefore  $\frac{1}{s} \leq \max_j \Phi_t(j) \leq 1$ . Ideally, we want  $\max_j \Phi_t(j)$  to approach  $\frac{1}{s}$ .

A **balanced cell-probing scheme** is defined as follows:

**Definition 2.** An  $(s, b, t, \phi)$ -balanced-cell-probing scheme for problem  $f : Q \times \mathcal{D} \rightarrow \{0, 1\}$  is a cell-probing scheme such that for any  $S \in \mathcal{D}$  and any probability distribution  $q$  over  $Q$ , a table  $T_{S,q} : [s] \rightarrow \{0, 1\}^b$  is constructed, such that for any query  $x \in Q$ , the algorithm returns  $f(x, S)$  by probing at most  $t$  cells, and for a query  $x \in Q$  generated according to the distribution  $q$ , the contention  $\Phi_k(j)$  is bounded by  $\phi$  for any  $1 \leq k \leq t$  and any  $j \in [s]$ .

Such schemes have the very strong property that not only is contention bounded across an execution of the query algorithm, but each individual step gives low contention.

Given a fixed table  $T_{S,q}$ , we can summarize the contention succinctly using linear algebra. Let  $P_t$  be a  $|Q| \times s$  matrix with  $P_t(x, j) := \Pr[I_X^{(t)} = j] = \mathbb{E}[Y^{(t)}(X, j)]$ . The contention on all cells can be computed by  $\Phi_t = qP_t$ , specifically,

$$\Phi_t(j) = \mathbb{E}[Y^{(t)}(X, j)] = \sum_{x \in Q} \Pr[X = x] \mathbb{E}[Y^{(t)}(x, j)] = \sum_{x \in Q} q_x P_t(x, j).$$

Finally, for our lower bound, it will be helpful to consider data structure problems from the perspective of communication complexity. In this view, a data structure is a communication protocol between an adaptive player Alice for the cell-probing algorithm and an oblivious player Bob for the table. The input to Bob is a pair  $(S, q)$ , and the input to Alice is a query  $x \in Q$ , which is generated according to the distribution  $q$ . Together they compute  $f(x, S)$  via communication. The contention then counts the probability of each type of message sent by Alice.

### 1.2. Our contributions

This paper makes the following contributions:

- We formalize a natural and interesting problem: memory contention caused by concurrent data structure queries. We introduce contention to the classic cell-probe model. In our model, contention is measured by the chance that a memory cell is probed during the execution of the cell-probe algorithm. This level of abstraction allows us to study the trade-off between the contention and the complexity of data structures without regard to specific contention resolution schemes.

- We note an especially interesting class of query distributions: distributions that are uniform over both the set of positive queries and the set of negative queries (but not necessarily uniform over all queries).

We introduce a linear-size, constant-time cell-probing scheme for the membership problem, with maximum contention  $O(1/n)$ . It is easy to see that this data structure is asymptotically optimal in all three parameters.

- For data structures with arbitrary query distribution, we introduce a general transform from cell-probing schemes with arbitrary contention

to cell-probing schemes with bounded contention. The cost of this transform is a factor of  $O(\log n / \log \log n)$  in both time and space. As a corollary, there exists an  $(O(\frac{n \log n}{\log \log n}), O(\log N), O(\frac{\log n}{\log \log n}), O(\frac{1}{n}))$ -balanced cell-probing scheme for the membership problem with arbitrary query distributions.

- Again for arbitrary query distributions, we prove a lower bound on any balanced cell-probing scheme satisfying a certain technical restriction. The lower bound is a time-contention trade-off: for any problem with a non-degenerate subproblem of size  $n$ , and the membership problem on  $n$  elements in particular, if the contention is at most  $\text{Polylog}(n)$  times optimal, then the time complexity is  $\Omega(\log \log n)$ .

### 1.3. Related work

Our first upper bound is based on the perfect hashing scheme of Fredman *et al.* [7] (FKS) and a subsequent extension by Dietzfelbinger and Meyer auf der Heyde to the dynamic case [8, 9, 10]. We refer to the latter (and [8] in particular) as DM.

FKS, which is based on a two-level tree of hash tables, yields a static data structure for the membership problem with linear space and constant lookup time.

For DM, the pairwise independent family of hash functions used in FKS is replaced with a new family that gives a more even distribution of load across the second layer of the tree and thus bounded worst-case update costs. Dietzfelbinger and Meyer auf der Heyde have considered implementations of DM in the PRAM model [9] and in the model of a complete synchronized network of processors [10]. While both implementations optimize the contention on individual processors, neither considers the contention on individual memory locations.

The membership problem can also be solved with optimal time and space complexity using cuckoo hashing [11]; as with FKS and DM, the contention of the standard implementation is high, mostly because all queries read the hash function parameters from the same locations.

For FKS, DM, and cuckoo hashing, contention can be decreased by storing the hash function redundantly. Under the assumption that the query is distributed uniformly within both the positive set and the negative set, this gives a maximum contention of  $\Theta(\sqrt{n})$  times optimal for FKS, and  $\Theta(\ln n / \ln \ln n)$  times optimal for DM and cuckoo hashing. For arbitrary query distributions, the contentions can be arbitrarily bad. Given that none of these data structures were designed with memory contention in mind,

this is perhaps unsurprising, but it is possible nonetheless to do substantially better.

## 2. Low-contention uniform membership queries

Recall that for the membership problem, we have a universe  $U$  of size  $N$ , and a data set  $S$  of size  $n$  that is a subset of  $U$ . Assume that  $N \geq n^2$  and that each cell in the table contains a  $b$ -bit word, where  $b = \log_2 N$ .

**Theorem 3.** *Consider the membership problem. For all data sets  $S \in \binom{U}{n}$  and all query distributions where all elements in  $S$  are equally likely and all elements in  $U \setminus S$  are equally likely, there exists an  $(O(n), b, O(1), O(1/n))$ -balanced-cell-probing scheme.*

*Given  $S$ , the data structure can be constructed by a unit-cost machine in expected time  $O(n)$ .*

To understand our data structure, consider the query procedure for FKS hashing. FKS resolves collisions in an  $O(n)$ -bucket primary hash table by constructing for each primary bucket  $i$  an  $O(n_i^2)$ -bucket secondary hash table, where  $n_i$  is the number of items in primary bucket  $i$ . The secondary hash table for primary bucket  $i$  is large enough that for *all* sets of  $n_i$  items, a random hash function, with constant probability, places the items in distinct secondary buckets. On the other hand, it can be shown that in expectation,  $\sum_i n_i^2 = O(n)$ .

FKS queries make exactly three probes. The first probe reads the parameters of the hash function; the second reads a pointer to the “bucket” in which the target item will be found, as well as information about the size of the bucket and the perfect hash function used within the bucket; and the third reads the actual element. With respect to contention, however, FKS does not achieve our goal of  $O(1/n)$ , as the first probe produces contention 1, and the second produces contention  $\Theta(n_i/n)$ .

By replicating the hash function parameters  $\Omega(n)$  times, we reduce the contention of the first probe to  $O(1/n)$ . The second probe requires more sophistication, as while we would like higher replication counts for buckets that are larger and thus more heavily contended, the query algorithm does not know *a priori* which buckets those are. It is too costly to replicate all of them.

We organize the buckets into  $\Theta(n/\log n)$  groups of  $\Theta(\log n)$  buckets each. While individual buckets may vary significantly in size, we show that when using the hash functions of DM [8], the total size of each group is  $O(\log n)$

with reasonably high probability. We standardize on a set of replication counts fine enough to be space-efficient but coarse enough to allow the sizes of all buckets in a single group to be encoded in a single  $b$ -bit cell, which is replicated  $O(\log n)$  times to achieve contention  $O(1/n)$ . Knowing the size of each bucket in the group, the query algorithm deduces the storage range for the replicated headers of the target bucket, reads the relevant header information (including both a pointer to the actual location of the bucket and the parameters of its secondary hash function) from a randomly-distributed probe, and uses the bucket's perfect hash function to find the target element. This four-phase procedure requires a constant number of probes and uses only  $O(n)$  space with  $O(1/n)$  contention, for both uniform positive queries and uniform negative queries.

### 2.1. Hash families

The concept of **universal hash classes** is due to Carter and Wegman [12]. For  $d \geq 2$ , a family of functions from  $U$  to  $[m]$  is  $d$ -wise independent (or  **$d$ -universal**) if for any  $d$  distinct elements  $x_1, x_2, \dots, x_d$  from  $U$ , the hash values  $h(x_1), h(x_2), \dots, h(x_d)$  are uniformly and independently distributed over  $[m]$ .

Let  $\mathcal{H}_m^d$  denote a  $d$ -wise independent family of hash functions from  $U$  to  $[m]$ . It is well known that if  $d \geq 2$  and  $m \geq n^2$ , then for all  $S \in \binom{U}{n}$ , with probability at least  $\frac{1}{2}$ , a uniform random hash function  $h \in \mathcal{H}_m^d$  maps each element in  $S$  to a distinct value, that is,  $h$  is a *perfect hash function*.

We use the following hash family, which was introduced by Dietzfelbinger and Meyer auf der Heide [8].

**Definition 4 (DM[8]).** For all  $f \in \mathcal{H}_m^d$  and  $g \in \mathcal{H}_r^d$  and  $z \in [m]^r$ , the hash function  $h_{f,g,z} : U \rightarrow [m]$  is defined by

$$h_{f,g,z}(x) := (f(x) + z_{g(x)}) \bmod m.$$

The hash family  $\mathcal{R}_{r,m}^d$  is

$$\mathcal{R}_{r,m}^d := \{h_{f,g,z} \mid f \in \mathcal{H}_m^d, g \in \mathcal{H}_r^d, z \in [m]^r\}.$$

Given a hash function and a set of elements, we define the buckets and loads as follows.

**Definition 5.** For all  $h : U \rightarrow [m]$  and  $S \subseteq U$  and  $i \in [m]$ , the  $i$ -th **bucket** is

$$B(S, h, i) := \{x \in S \mid h(x) = i\},$$

and the **load** of the  $i$ -th bucket is

$$\ell(S, h, i) := |B(S, h, i)|.$$

The following tail bound on the sum of a 0-1 valued  $d$ -wise independent sequence is due to Kruskal, Rudolph, and Snir.

**Theorem 6 (Corollary 4.20, [13]).** *Let  $X_1, \dots, X_n$  be 0-1 valued,  $d$ -wise independent, identically distributed random variables, and let  $X = \sum_i^n X_i$ . If  $d \leq 2E[X]$ , then*

$$\Pr[X - E[X] > t] \leq O\left(\frac{(E[X])^{d/2}}{t^d}\right).$$

The following is a special case of Hoeffding's theorem.

**Theorem 7 (Hoeffding, [14]).** *Let  $Y_1, \dots, Y_r \in [0, d]$  and  $Y = \sum_i^r Y_i$  and let  $c > e$  be a constant, where  $e$  denotes the base of the natural logarithm. If  $cE[Y] \leq rd$ , then*

$$\Pr[Y \geq cE[Y]] \leq \left(\frac{e}{c}\right)^{\frac{c}{d}E[Y]}.$$

The following theorem about  $d$ -universal hash families is due to Dietzfelbinger and Meyer auf der Heide.

**Theorem 8 (Fact 2.2, [8]).** *Let  $S$  be a fixed set of  $n$  elements. Let  $f$  be chosen from  $\mathcal{H}_m^d$  uniformly at random, where  $d > 2$  is a constant and  $m \leq 2n/d$ . Then*

$$\Pr[\forall i \in [m], \ell(S, f, i) \leq d] \geq 1 - n \cdot (2n/m)^d.$$

The following lemmas extend previous characterizations of the load distribution achieved by hash functions drawn a specified family.

**Lemma 9 (extending [13]).** *Let  $c, d, \delta$  be constants, where  $c > e \approx 2.718$  and  $d > 2$  and  $\frac{2}{d+2} < \delta < 1 - \frac{1}{d}$ . Let  $r = n^{1-\delta}$ . For all data sets  $S \in \binom{U}{n}$ ,*

$$\Pr\left[g \leftarrow \mathcal{H}_r^d: \forall i \in [r], \ell(S, g, i) \leq cn/r\right] \geq 1 - o(1).$$

PROOF. Let  $S = \{x_1, x_2, \dots, x_n\}$ . Fixing  $j \in [r]$ , let  $X_i$  be a 0-1 valued random variable that indicates whether  $g(x_i) = j$ . Letting  $X = \sum_i^n X_i$ ,

the fact that  $g$  is drawn from a  $d$ -wise independent hash family implies that  $E[X] = \frac{n}{r} = n^\delta$ . By Theorem 6,

$$\Pr \left[ X - n^\delta \geq (c-1)n^\delta \right] \leq O \left( n^{\delta d/2} / n^{\delta d} \right) = O \left( n^{-\delta d/2} \right).$$

Therefore,

$$\begin{aligned} \Pr \left[ \exists j \in [r], \ell(S, g, j) > \frac{cn}{r} \right] &\leq r \cdot \Pr \left[ \ell(S, g, j) > cn^\delta \right] \\ &\leq n^{1-\delta} \cdot \Pr \left[ X - n^\delta > (c-1)n^\delta \right] \\ &\leq O \left( n^{1-\delta-\delta d/2} \right) = o(1). \end{aligned}$$

**Lemma 10 (extending [8]).** *Let  $c, d, \delta, \alpha$  be constants, where  $c > e = 2.718 \dots$  and  $d > 2$  and  $\frac{2}{d+2} < \delta < 1 - \frac{1}{d}$  and  $\alpha > \frac{d}{c(\ln c - 1)}$ . Let  $r = n^{1-\delta}$  and  $m = \frac{n}{\alpha \ln n}$ . For all data sets  $S \in \binom{[n]}{m}$ ,*

$$\Pr \left[ h' \leftarrow \mathcal{R}_{r,m}^d : \forall i \in [r], \ell(S, h', i) \leq cn/r \right] \geq 1 - o(1).$$

PROOF. Let  $S = \{x_1, x_2, \dots, x_n\}$ . We assume that  $h'$  is defined by  $(f, g, z)$  where  $f \leftarrow \mathcal{H}_m^d$  and  $g \leftarrow \mathcal{H}_r^d$  and  $z \leftarrow [m]^r$ . We denote by  $\mathcal{E}_1$  the event that  $\forall i \in [r], \ell(S, g, i) \leq cn/r$ , and denote by  $\mathcal{E}_2$  the event that  $\forall i \in [r], \forall j \in [m], \ell(B(S, g, i), f, j) \leq d$ .

By Lemma 9, the event  $\mathcal{E}_1$  holds with probability  $1 - o(1)$ . Conditioning on  $\mathcal{E}_1$ , by Theorem 8, with probability  $1 - O(n^{\delta(d+1)}/m^d)$ , it holds that  $\forall j \in [m], \ell(B(S, g, i), f, j) \leq d$ . By union bound, with probability  $1 - O(n^{1-\delta} \cdot n^{\delta(d+1)}/m^d) = 1 - O(n^{1-d(1-\delta)} \ln^d n) = 1 - o(1)$ , it holds that  $\forall i \in [r], \forall j \in [m], \ell(B(S, g, i), f, j) \leq d$ , that is,  $\mathcal{E}_2$ . Therefore,  $\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = (1 - o(1))(1 - o(1)) = 1 - o(1)$ .

Conditioning on  $\mathcal{E}_1 \wedge \mathcal{E}_2$  and fixing  $j \in [m]$ , define for  $i = 1, 2, \dots, r$  a random variable  $Y_i$ , where

$$Y_i := |\{x \in S \mid g(x) = i \text{ and } f(x) + z_{g(x)} \equiv j \pmod{m}\}|.$$

Let  $Y = \sum_i Y_i$ . Note that  $Y_i = \ell(B(S, g, i), h', j) = \ell(B(S, g, i), f, (j - z_i + m) \bmod m)$  and  $Y = \ell(S, h', j)$ .

The variables  $Y_i$  are conditionally independent given  $f, g$  because the corresponding variables  $z_i$  are unconditionally independent. Given  $\mathcal{E}_2$ , it holds for all  $i \in [r]$  that  $Y_i \leq d$ .

$$\begin{aligned}
E[Y_i | f, g] &= \sum_{\omega \in [m]} \Pr[z_i = \omega] \cdot \ell(B(S, g, i), f, (j - z_i + m) \bmod m) \\
&= \frac{1}{m} \sum_{k \in [m]} \ell(B(S, g, i), f, k) \\
&= \frac{1}{m} \ell(S, g, i).
\end{aligned}$$

Therefore

$$E[Y] = E[E[Y | f, g]] = \frac{1}{m} E \left[ \sum_{i \in [r]} \ell(S, g, i) \right] = \frac{|S|}{m} = \frac{n}{m}.$$

By Theorem 7, it holds that  $\Pr[Y \geq cn/m] \leq (e/c)^{c \ln n/d} = o(n^{-1})$ . By a union bound,  $\Pr[\forall j \in [m], \ell(S, h', j) \leq cn/m] = 1 - m \cdot \Pr[Y \geq cn/m] = 1 - o(1)$ .

Recall that the above holds when conditioning on  $\mathcal{E}_1 \wedge \mathcal{E}_2$ , and since  $\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] = 1 - o(1)$ , that  $\forall j \in [m], \ell(S, h', j) \leq cn/m$  holds unconditionally with probability at least  $(1 - o(1))(1 - o(1)) = 1 - o(1)$ .

**Lemma 11 (FKS condition; extending [7]).** *Let  $c, d, \delta, \beta$  be constants, where  $c > e = 2.718 \dots$  and  $d > 2$  and  $\frac{2}{d+2} < \delta < 1 - \frac{1}{d}$  and  $\beta \geq 2$ . Let  $s = \beta n$ . For all data sets  $S \in \binom{U}{n}$ ,*

$$\Pr \left[ h \leftarrow \mathcal{R}_{r,s}^d : \sum_{i \in [s]} (\ell(S, h, i))^2 \leq s \right] \geq \frac{1}{2}.$$

PROOF. Let  $S = \{x_1, x_2, \dots, x_n\}$ . For all  $i, j \in [n]$  with  $i \neq j$ , let  $X_{ij}$  be a 0-1 random variable that indicates whether  $h(x_i) = h(x_j)$ . Let  $X = \sum_{i \neq j} X_{ij}$  be the number of ordered pairs that collide. Since  $h$  is drawn from a 2-wise independent family of hash functions,

$$X = 2 \sum_{i \in [s]} \binom{\ell(S, h, i)}{2} = \sum_{i \in [s]} (\ell(S, h, i))^2 - n.$$

For all  $i \neq j$ , moreover,  $E[X_{ij}] = \Pr[h(x_i) = h(x_j)] = 1/s$ , so  $E[X] = n(n-1)/s$ . By Markov's inequality,

$$\Pr \left[ \sum_{i \in [s]} (\ell(S, h, i))^2 > s \right] = \Pr[X > s - n] \leq \frac{E[X]}{(s - n)} \leq \frac{1}{\beta(\beta - 1)} \leq 1/2.$$

## 2.2. Data structure construction

Let  $c = 2e$ . For  $d > 2$ , choose constants  $\alpha$  and  $\beta$  large enough to satisfy the hypotheses of Lemmas 10 and 11, and let  $r = n^{1-\delta}$  and  $m = \frac{n}{\alpha \ln n}$  and  $s = \beta n$ . Without loss of generality, assume that  $m$  divides  $s$ .

Given an arbitrary data set  $S \in \binom{U}{n}$ , choose uniform random  $f \leftarrow \mathcal{H}_s^d$  and  $g \leftarrow \mathcal{H}_r^d$  and  $z \leftarrow [s]^r$ , so that the function  $h$  where  $h(x) := (f(x) + z_{g(x)}) \bmod s$  effectively is chosen uniform from  $\mathcal{R}_{r,s}^d$ . Define a hash function  $h' : U \rightarrow [m]$  by  $h'(x) = h(x) \bmod m$ . Note that  $h'$  is a uniformly random function from the family  $\mathcal{R}_{r,m}^d$  because  $m$  divides  $s$ . Formally,

$$\begin{aligned} h'(x) &= (f(x) + z_{g(x)}) \bmod s \bmod m \\ &= (f(x) \bmod m + z_{g(x)} \bmod m) \bmod m. \end{aligned}$$

For uniform  $f \leftarrow \mathcal{H}_s^d$  and uniform  $z \leftarrow [s]^r$ , the function  $(f \bmod m)$  is uniform over  $\mathcal{H}_m^d$ , and the vector  $(z \bmod m)$  is uniform over  $[m]^r$ . Therefore,  $h'$  is uniform over  $\mathcal{R}_{r,m}^d$ .

We would like our triple of hash functions to belong to the set

$$\mathcal{P}(S) := \left\{ (g, h', h) \in \mathcal{H}_r^d \times \mathcal{R}_{r,m}^d \times \mathcal{R}_{r,s}^d \mid \max_{i \in [r]} \ell(S, g, i) \leq \frac{cn}{r}, \max_{i \in [m]} \ell(S, h', i) \leq \frac{cn}{m}, \text{ and } \sum_{i \in [s]} \ell^2(S, h, i) \leq s \right\}$$

By Lemmas 10 and 11 and a union bound over all unwanted events, it holds with probability at least  $1/2 - o(1)$  that  $(g, h', h) \in \mathcal{P}(S)$ . In expectation,  $O(1)$  independent trials suffice to generate  $(g, h', h) \in \mathcal{P}(S)$ . A unit-cost machine can verify membership in  $\mathcal{P}(S)$  in time  $O(n)$ , so the same machine can find good triples in expected time  $O(n)$ .

Algorithm 1 gives the pseudocode for the construction process. The data structure is organized into rows containing  $s$  cells each. Let  $T(i, j)$  be the contents of the  $j$ -th cell in the  $i$ -th row of the data structure.  $T$  is constructed as follows.

- Let the two  $d$ -universal hash functions  $f$  and  $g$  be represented by  $2d$  words  $a_0, a_1, \dots, a_{2d-1}$ . For all  $i \in [2d]$  and all  $j \in [s]$ , let  $T(i, j) = a_i$ . For all  $j \in [s]$ , let  $T(2d, j) = z[j \bmod r]$ .
- The hash function  $h$  assigns each of the  $n$  elements of  $S$  to one of  $s$  buckets, and the hash function  $h'$  assigns each bucket to one of  $m$  groups corresponding to congruence classes modulo  $m$ . For group

---

**Algorithm 1:** Construction of a low-contention dictionary  $T$ .

---

**Input:** a set  $S \in \binom{U}{n}$ ;  
**Output:**  $T$ , a 7-row array, each row of  $2 \left( \lceil \frac{n}{\ln n} \rceil \lceil \ln n \rceil \right)$  cells;  
**begin**  
    set  $c = 2e$ ,  $d = 3$ ,  $r = \lceil \sqrt{n} \rceil$ ,  $m = \lceil \frac{n}{\ln n} \rceil$ ,  $s = 2m \lceil \ln n \rceil$ ;  
    **repeat**  
         $f = \text{u-hash}(d, s)$ ;  
         $g = \text{u-hash}(d, r)$ ;  
        pick a  $z \in [s]^r$  uniformly at random;  
        denote  $h(\cdot) = (f(\cdot) + z[g(\cdot)]) \bmod s$ ,  
             $h'(\cdot) = (f(\cdot) + z[g(\cdot)]) \bmod m$ ;  
        compute  $\ell(S, g, j), \ell(S, h, j), \ell(S, h', j)$  for all  $j$ ;  
    **until**  $\max_{j \in [r]} \ell(S, g, j) \leq \frac{cn}{r}$  **and**  $\sum_{j \in [s]} \ell^2(S, h, j) \leq s$  **and**  
         $\max_{j \in [m]} \ell(S, h', j) \leq \frac{cn}{m}$ ;  
     $GBA[0] = 0$ ;  
    **for**  $i = 1$  **to**  $m$  **do**  
         $GBA[i] = GBA[i - 1] + \sum_{k \in [s/m]} \ell^2(S, h, km + i - 1)$ ;  
    **foreach**  $i \in [m]$  **do**  
        **for**  $k \in [s/m]$  **do**  $u_k = \text{unary representation of } \ell(S, h, km + i)$ ;  
         $\text{hist}[i] = \text{concatenation of } u_0 u_1 \cdots u_{s/m-1}$ ;  
    **foreach**  $j \in [s]$  **do**  
         $T[0, j] = f$ ;  
         $T[1, j] = g$ ;  
         $T[2, j] = z[j \bmod r]$ ;  
         $T[3, j] = GBA[j \bmod m]$ ;  
         $T[4, j] = \text{hist}[j \bmod m]$ ;  
    **foreach**  $k \in [s/m], l \in [m]$  **do**  
        let  $i = km + l$ ,  $S_i = \{x \in S \mid h(x) = i\}$ , and  $s_i = \ell^2(S, h, i)$ ;  
         $BBA[i] = GBA[i] + \sum_{k' < k} \ell^2(S, h, k'm + l)$ ;  
        **if**  $s_i > 0$  **then**  
            **repeat**  $h_i^* = \text{u-hash}(d, s_i)$  **until**  $\max_{j \in [s_i]} \ell(S_i, h_i^*, j) \leq 1$ ;  
            **foreach**  $j \in [s_i]$  **do**  $T[5, BBA[i] + j] = h_i^*$ ;  
            **foreach**  $x \in S_i$  **do**  $T[6, BBA[i] + h_i^*(x)] = x$ ;

---

Figure 1: Pseudocode for the construction of a low-contention dictionary with the settings that  $c = 2e$ ,  $d = 3$ ,  $\alpha = 1$ ,  $\beta = 2$ ,  $\delta = \frac{1}{2}$ , and each cell contains  $b \geq \max(6 \log n, \log |U|)$  bits. Each calling of  $\text{u-hash}(d, m)$  returns a uniform and independent function from  $\mathcal{H}_m^d$ .

$i \in [m]$ , we define the *group-base-address* of  $i$ , or  $GBA_S(i)$ , inductively by  $GBA_S(0) = 0$  and

$$GBA_S(i) = GBA_S(i-1) + \sum_{k \in [s/m]} \ell^2(S, h, km + i - 1).$$

These group-base-addresses are valid because by property  $\mathcal{P}(S)$ , it holds for all  $i \in [m]$  that  $GBA_S(i) \leq s$ . Moreover, the vector  $GBA_S$  can be computed in  $O(n)$  time by a unit-cost machine. Let  $T(2d+1, j) = GBA_S(j \bmod m)$ , so that each bucket stores the group-base-address of the group to which that bucket belongs.

- Each group contains  $s/m = \alpha\beta \ln n$  buckets. The *group-histogram* of a group whose buckets have loads  $l_1, l_2, \dots, l_{s/m}$  is the binary string

$$1^{l_1} 0 1^{l_2} 0 \dots 1^{l_{s/m}} 0.$$

By property  $\mathcal{P}(S)$ , each group contains at most  $cn/m = c\alpha \ln n$  elements of  $S$ , so the length of this string is at most  $\alpha(\beta + c) \ln n$ . Let  $\rho := \lceil \frac{\alpha(\beta+c) \ln n}{b} \rceil$ . Observe that because  $b = \Theta(\log n)$ , we have  $\rho = O(1)$ . For all  $j \in [s]$ , let  $a'_{0j}, a'_{1j}, \dots, a'_{\rho-1,j}$  be  $\rho$  words that store the group-histogram of group  $j$ . For all  $i = 0, 1, \dots, \rho - 1$  and all  $j \in [s]$ , define  $T(2d+2+i, j) = a'_{i, (j \bmod m)}$ .

- The last two rows store perfect hashes of each bucket. Each bucket  $i \in [s]$  is assigned  $\ell^2(S, h, i)$  cells in each row. By property  $\mathcal{P}(S)$ , in each row, at most  $s$  cells are assigned.

In the  $(2d + \rho + 2)$ th row, for each individual bucket  $i$ , the parameters of perfect hash function  $h_i^*$  are replicated across the space allotted to the bucket. In the  $(2d + \rho + 3)$ th row, the members of  $S$  in each bucket  $i$  are stored according to the hash function  $h_i^*$ . A unit-cost machine can initialize this storage in time  $O(n)$ .

In all, the table  $T$  has  $(2d + \rho + 3) = O(1)$  rows each containing  $s = O(n)$  words, for a total of  $O(n)$  words. A unit-cost machine can construct  $T$  in time  $O(n)$ .

### 2.3. Queries and contention

We query whether  $x$  is in  $S$  with the following algorithm. Each random choice is assumed to be independent and uniform within its range.

1. For all  $i \in [2d]$ , choose random  $j \leftarrow [s]$  and read  $T(i, j)$ . This yields  $f$  and  $g$ . Next choose random  $k \leftarrow [s/r]$  and read  $T(2d, kr + g(x))$  to obtain  $z_{g(x)}$ . We can now compute  $h = (f + z_{g(x)}) \bmod s$  and  $h' = h \bmod m$ .
2. Choose random  $k \leftarrow [s/m]$  and read  $T(2d + 1, km + h'(x))$ , which stores  $GBA_S(h'(x))$ . For all  $i \in [\rho]$  where  $\rho = \lceil \frac{\alpha(\beta+c)\ln n}{b} \rceil$ , choose random  $j \leftarrow [s/m]$  and read  $T(2d + 2 + i, jm + h'(x))$  to obtain the group-histogram group  $h'(x)$ . The *bucket-base-address* of bucket  $h(x)$ , denoted  $BBA_S(h(x))$  is

$$BBA_S(h(x)) := GBA_S(h'(x)) + \sum_{k=0}^{\lceil h(x)/m \rceil - 1} \ell^2(S, h, km + h'(x)).$$

For all  $k \in [s/m]$ , the group-histogram of group  $h'(x)$  stores the value of  $\ell^2(S, h, km + h'(x))$ , thus  $BBA_S(h(x))$  is retrieved, so is  $\ell(S, h, h(x))$ . The range of addresses allotted to bucket  $h(x)$  is from  $BBA_S(h(x))$  to  $BBA_S(h(x)) + \ell^2(S, h, h(x)) - 1$ .

3. If  $\ell(S, h, h(x)) = 0$ , then the bucket  $h(x)$  is empty, so return 0. Otherwise, choose random  $j \leftarrow [\ell(S, h, h(x))]$  and read  $T(2d + \rho + 2, BBA_S(h(x)) + j)$  to obtain the perfect hash function  $h^*$ . If  $T(2d + \rho + 3, BBA_S(h(x)) + h^*(x)) = x$ , then return 1; otherwise, return 0.

The pseudocode for the query algorithm is given in Algorithm 2. The process of answering a membership query  $x$  is illustrated in Figure 2.3

This query algorithm probes each row of  $T$  at most once, so its cell-probe complexity is  $O(1)$ .

For contention, we first consider the contribution of the positive queries. All events below are conditioned the target element being in  $S$ . At each step before the last probe, an expected  $1$ ,  $\frac{1}{n}\ell(S, g, i_1)$ ,  $\frac{1}{n}\ell(S, h', i_2)$ , or  $\frac{1}{n}\ell(S, h, i_3)$  probes are balanced over a range of size  $s$ ,  $s/r$ ,  $s/m$ , or  $\ell^2(S, h, i_3)$  respectively, so by property  $\mathcal{P}(S)$ , the maximum contention is  $O(1/n)$ . For the last probe, the perfect hash function sends each query to a distinct cell so the contention is obviously  $O(1/n)$ . Therefore, the total contention contributed by positive queries is at most  $O(1/n)$ .

**Lemma 12.** *Let  $\bar{S}$  denote  $U \setminus S$ , and  $N = |U| = \omega(n)$ . For any hash function  $h : U \rightarrow [k]$  which is uniform over the domain, for sufficiently large  $n$ ,  $\forall i \in [k], \ell(\bar{S}, h, i) \leq 2(N - n)/k$ .*

PROOF. Because  $h$  is uniform over the domain,  $\ell(U, h, i) = N/k$  for any  $i \in [k]$ . For  $N = \omega(n)$ , it holds that  $\ell(\bar{S}, h, i) = \ell(U, h, i) - \ell(S, h, i) \leq N/k \leq 2(N - n)/k$ .

---

**Algorithm 2:** Query algorithm for answering membership queries.

---

**Input:**  $x \in U$ ; array  $T$  constructed by Algorithm 1 from set  $S \in \binom{U}{n}$ ;

**Output:** **Yes** if  $x \in S$ , **No** if otherwise;

**begin**

set  $r = \lceil \sqrt{n} \rceil$ ,  $m = \lceil \frac{n}{\ln n} \rceil$ ,  $s = 2m \lceil \ln n \rceil$ ;  
pick  $j_0, j_1, j_2 \in [s]$ ,  $j_2 \in [\lceil s/r \rceil]$ ,  $j_3, j_4 \in [s/m]$   
uniformly and independently at random;  
**read**  $f = T[0, j_0]$ ,  $g = T[1, j_1]$ ,  $z = T[2, j_2 \cdot r + g(x)]$ ;  
 $i_{\text{bucket}} = (f(x) + z) \bmod s$ ;  
 $i_{\text{group}} = i_{\text{bucket}} \bmod m$ ;  
**read**  $GBA = T[3, j_3 \cdot m + i_{\text{group}}]$ ,  $hist = T[4, j_4 \cdot m + i_{\text{group}}]$ ;  
represent  $hist$  as  $1^{\ell_0} 0 1^{\ell_1} 0 \dots 1^{\ell_{s/m-1}} 0$ ;  
 $BBA = GBA + \sum_{k: km < i_{\text{bucket}}} \ell_k^2$ ;  
**if**  $\ell_k = 0$  **then return No**;  
 $k = (i_{\text{bucket}} - i_{\text{group}}) / m$ ;  
pick  $j_5 \in [\ell_k^2]$  uniformly and independently at random;  
**read**  $h^* = T[5, BBA + j_5]$ ,  $key = T[6, BBA + h^*(x)]$ ;  
**if**  $key = x$  **then return Yes**;  
**else return No**;

---

Figure 2: Pseudocode for the query algorithm for answering membership queries. We assume that the data structure  $T$  is constructed by Algorithm 1.

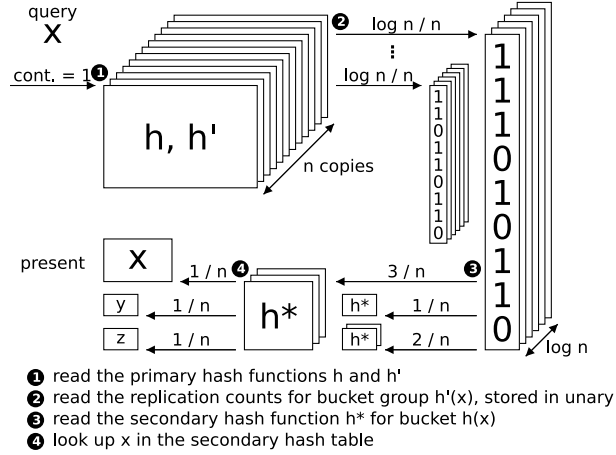


Figure 3: The process of answering a membership query.

Note that  $g$ ,  $h'$ , and  $h$  are all uniform over the domain. This is because any  $d$ -universal function must be 1-universal. Due to Lemma 12, the loads of negative queries to all types of buckets are asymptotically even. The same argument as above can be applied to bound the contention caused by negative queries to  $O(1/n)$ .

### 3. Upper bounds for arbitrary query distribution

In this section, we show that there exists a general way to transform an arbitrary cell-probing scheme to a low-contention cell-probing scheme with some cost of time and space overhead.

#### 3.1. The prefix-sum problem

Given any cell-probing scheme, the contention of each cell can be ameliorated by making each cell a number of replicas proportional to its contention and uniform probing all replicas when probing a cell. This approach asks that the addresses of these replicas must be inferred by a limited number of probes with low contention. We reduce the problem of addressing with presence of replicas to the following specialized version of the prefix-sum problem.

**Definition 13.** *The prefix-sum problem  $f : Q \times \mathcal{D} \rightarrow [n]$  is defined as follows:  $Q = \{1, 2, \dots, n\}$  and  $\mathcal{D} = \{y \in [n+1]^n \mid \sum_j y_j = n\}$ , and for every  $x \in Q$  and every  $y \in \mathcal{D}$ ,  $f(x, y)$  returns the pair such that  $f(x, y) = (\sum_{j < x} y_j, y_x)$ .*

The data  $y$  is a tuple of  $n$  numbers whose sum is  $n$  and the query  $x$  is one of the indices  $1, 2, \dots, n$ . The result of the query is the sum of first  $x - 1$  entries of  $y$  together with the  $x$ -th entry  $y_x$  itself. This problem is trivial to solve if contention is not a concern. We construct a low-contention data structure solving this problem.

**Theorem 14.** *An  $(O(\frac{n \log n}{\log \log n}), O(\log n), O(\frac{\log n}{\log \log n}), O(1/n))$ -balanced cell-probing scheme exists for the prefix-sum problem with  $n$  elements and query distribution  $q$  where  $q(x) = \frac{y_x}{n}$ .*

PROOF. Abstractly, the data structure for the prefix-sum problem is a  $(\log n)$ -ary tree. Each node  $u$  corresponds to an ordered set  $S(u)$  of consecutive integers.

- For the root  $u$ ,  $S(u) = (1, 2, \dots, n)$ .

- For any node  $u$ ,  $S(u)$  is partitioned into  $k = \min(|S(u)|, \log n)$  consecutive sub-tuples  $S_1, S_2, \dots, S_k$  in ascending order, such that the difference between the lengths of any  $S_i$  and  $S_j$  is at most 1. Each  $S_j$  is associated to a child  $u_j$  of  $u$ .

The recursion terminates when every node in the same level of the tree has that  $|S(u)| = 1$ . The depth of the tree is  $O(\frac{\log n}{\log \log n})$ .

Define the **weight** of a node  $u$  as  $w(u) := \sum_{j \in S(u)} y_j$ . For each level of the tree, by enumerating all the nodes  $u_1, u_2, \dots$  in that level in ascending order of  $S(u_i)$ , we can define the **prefix-sum**  $p(u_i)$  of a node  $u_i$  by  $p(u_i) = \sum_{j < i} w(u_j)$ . It is easy to see that for any node  $u$  and its first child, say  $u_1$ , it holds that  $p(u) = p(u_1)$ . Let every leaf node  $u$  store  $(p(u), w(u))$ . The prefix-sum problem with query  $x \in \{1, 2, \dots, n\}$  can be answered with the pair  $(p(u), w(u))$  by accessing the leaf node  $u$  with  $S(u) = (x)$ .

We then show how to route from the root to any leaf by cell-probes with low contention.

Each level of the tree is represented by two **rows** of  $n$  cells, each of  $O(\log n)$  bits. Respectively, we call these rows the **primary row** and the **secondary row** of each node in that level. Each node  $u$  owns the cells across the range from  $p(u) + 1$  to  $p(u) + w(u)$  in both rows. In total, this uses exactly  $n$  cells, because for any level of the tree, the sum of the weights of all nodes in any level is  $\sum_{j=1}^n y_j = n$ . To move from a node  $u$  to any one of its children  $u_1, u_2, \dots, u_k$ , it is sufficient to know the base address of  $u_1$ , which is  $p(u_1) + 1 = p(u) + 1$ , and the weights  $w(u_1), w(u_2), \dots, w(u_k)$  of all  $k$  children of  $u$ . The value of  $p(u)$  can be stored in one cell. Although it is impossible to represent all  $k$  weights precisely with one cell, we show that there is an efficient way to *approximately* represent and retrieve these values by one cell of  $O(\log n)$  bits.

**Lemma 15.** *Let  $2 \leq k \leq \log n$ . There exists two efficiently computable procedures **Enc** and **Dec** such that for any  $k \leq w \leq n$ , for any  $z \in [w + 1]^k$  that  $\sum_{j=1}^k z_j = w$  it holds that **Enc** $(w, z)$  is a binary string with length at most  $2 \log n$  and **Dec** $(w, \mathbf{Enc}(w, z)) = z'$  where  $z' \in [w + 1]^k$  and it holds that  $\sum_{j=1}^k z'_j \leq w$  and  $z'_j \geq \frac{1}{2} z_j$  for any  $1 \leq j \leq k$ .*

PROOF. Given any  $z \in [w + 1]^k$ , let **Enc** $(w, z)$  be the concatenation of the binary strings  $1^{\lfloor kz_j/w \rfloor} 0$  for all  $1 \leq j \leq k$ , where  $1^\ell$  denotes a string of  $\ell$  consecutive ones. The length of **Enc** $(w, z)$  is

$$\sum_{j=1}^k \left( 1 + \left\lfloor \frac{kz_j}{w} \right\rfloor \right) \leq k + \frac{k}{w} \sum_{j=1}^k z_j = 2k \leq 2 \log n.$$

Given a value  $\mathbf{Enc}(w, z)$ , one can reconstruct a  $k$ -tuple  $\langle \lfloor kz_j/w \rfloor \rangle_{j=1,2,\dots,k}$ . Let  $\mathbf{Dec}(w, \mathbf{Enc}(w, z))$  be the  $k$ -vector  $z'$  that  $z'_j := (1 + \lfloor kz_j/w \rfloor) \frac{w}{2k}$ . It holds that

$$\sum_{j=1}^k z'_j = \sum_{j=1}^k \left(1 + \left\lfloor \frac{kz_j}{w} \right\rfloor\right) \frac{w}{2k} \leq \frac{1}{2} \sum_{j=1}^k \left(1 + \frac{kz_j}{w}\right) \frac{w}{k} = \frac{1}{2} \left(w + \sum_{j=1}^k z_j\right) = w.$$

In addition, for any  $1 \leq j \leq k$ , it holds that

$$z'_j = \left(1 + \left\lfloor \frac{kz_j}{w} \right\rfloor\right) \frac{w}{2k} \geq \left(\frac{kz_j}{w}\right) \frac{w}{2k} = \frac{1}{2} z_j.$$

With this encoding scheme, routing from a node to its child can be implemented by two cell-probes, one in the primary row and the other in the secondary row.

For any  $u$  and its children  $u_1, u_2, \dots, u_k$ , let vector  $z$  defined as that  $z_j := w(u_j)$  for  $1 \leq j \leq k$ . Store  $\mathbf{Enc}(w(u), z)$  in every cell of the primary row of node  $u$ . Provided that  $(p(u), w(u))$  is known, which will be justified later, by probing one of the replicas,  $z' = \mathbf{Dec}(w(u), \mathbf{Enc}(w(u), z))$  can be decoded. For every child  $u_j$  of  $u$ , denote by  $w'(u_j) := z'_j$  the **approximate weight** of  $u_j$ .

For every node  $u$ , in its secondary row, starting from the address  $p(u) + 1$ , in the order of  $u_1, u_2, \dots, u_k$ , for every child  $u_j$  of  $u$ , represent the pair  $(p(u_j), w(u_j))$  in one cell and make  $w'(u_j)$  consecutive copies of the cell. This uses totally at most  $w(u)$  cells, because by Lemma 15,  $\sum_j w'(u_j) = \sum_j z'_j \leq w(u)$ . Node  $u$ , therefore, uses only space it owns.

Figure 5 illustrates the way to store node  $u$  and its children.

By induction, for the root  $u$ ,  $p(u) = 0$  and  $w(u) = n$  are both known. For any node  $u$ , provided that  $p(u)$  and  $w(u)$  are both known, with one uniform probe to the range from  $p(u) + 1$  to  $p(u) + w(u)$ , we can retrieve the approximate weights of all children of  $u$ ; with one uniform probe to the range from  $p(u) + \sum_{i < j} w'(u_i) + 1$  to  $p(u) + \sum_{i \leq j} w'(u_i)$ , we can retrieve  $(p(u_j), w(u_j))$  for the child  $u_j$  such that  $x \in S(u_j)$ . The procedure terminates when  $S(u_j) = (x)$  and the prefix-sum for the query  $x$  is answered, which takes  $O(\frac{\log n}{\log \log n})$  probes.

By the linearity of expectation, the probability that a node  $u$  is accessed is exactly  $\frac{w(u)}{n}$ . For the two probes to the cells corresponding to  $u$ , the probe is balanced across either  $w(u)$  cells or  $w'(u)$  cells. By Lemma 15, the approximate weight of a node has that  $w'(u) \geq \frac{1}{2}w(u)$ . Therefore, the maximum contention is  $\frac{2}{n}$ .

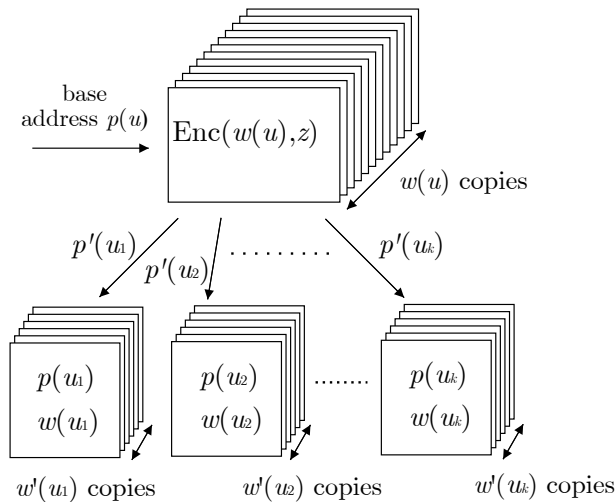


Figure 4: One node and its children in the tree

### 3.2. A general reduction

With the solution to the prefix-sum problem provided by Theorem 14, a general transformation from an arbitrary cell-probing scheme to a balanced cell-probing scheme can be built as follows.

**Theorem 16.** *For any data structure  $f$ , if there exists an  $(n, b, t)$ -cell-probing scheme for  $f$ , where  $b \geq 2 \log n$ , then there exists an  $(s_1, b, t', \phi_1)$ -balanced-cell-probing scheme for  $f$ , where  $s_1 = O(\frac{tn \log n}{\log \log n})$ ,  $t' = O(\frac{t \log n}{\log \log n})$ , and  $\phi_1 = O(1/n)$ ; and an  $(s_2, b, t', \phi_2)$ -balanced-cell-probing scheme for  $f$ , where  $s_2 = O(\frac{n \log n}{\log \log n})$ , and  $\phi_2 = O(t/n)$ .*

PROOF. For any data set  $S \in \mathcal{D}$ , let  $T_S$  be the corresponding table defined by the original cell-probing scheme. For any query distribution  $q$  over  $Q$ , let  $\Phi_t(j)$  be the expected number of probes to cell  $j$  at time  $t$  for the original cell-probing scheme.

Let  $y_j := \lfloor n \Phi_t(j) \rfloor$  for  $1 \leq j \leq n$ . It holds that  $\sum_{j=1}^n y_j \leq n + 1$  because  $\sum_j \Phi_t(j) = 1$ . Let  $y_{n+1} = n + 1 - \sum_{j=1}^n y_j$ . Construct an instance of the data structure provided in Theorem 14 with data set  $(y_1, y_2, \dots, y_{n+1})$ , and append an additional row of at most  $2n$  cells, such that the original cell  $T_S(j)$  is stored in every cell in the range from  $\sum_{i < j} (y_i + 1) + 1$  to  $\sum_{i \leq j} (y_i + 1)$  in this row. In this way the  $t$ -th probe to the original table is simulated by  $O(\frac{\log n}{\log \log n})$  probes in the new table, with max contention  $O(1/n)$  and

table size  $O(\frac{n \log n}{\log \log n})$ . Create such an instance for every  $t$ , which gives the  $(s_1, b, t', \phi_1)$ -balanced cell-probing scheme.

The second cell-probing scheme can be constructed in a similar way. This time  $y_j := \lfloor \frac{n}{t} \Phi(j) \rfloor$  where  $\Phi(j) = \sum_t \Phi_t(j)$  is the total contention of cell  $j$ , and all original probes are simulated by the same instance of prefix-sum.

The following corollary holds directly for the membership problem.

**Corollary 17.** *There exists an  $(O(\frac{n \log n}{\log \log n}), O(\log n), O(\frac{\log n}{\log \log n}), O(1/n))$ -balanced-cell-probing scheme for the membership problem with  $n$  elements.*

#### 4. A lower bound for arbitrary query distributions

In this section, we prove a cell-probe lower bound for low-contention data structures with arbitrary query distribution. The lower bound is on the cost of queries by an algorithm that knows nothing about the distribution except for what it can learn by probing the table, which was constructed by an algorithm with full knowledge of the query distribution.

The lower bound holds generally for a class of data structure problems with fixed VC-dimension [15], a measure of complexity of classification problems. By treating a data structure problem  $f : Q \times \mathcal{D} \rightarrow \{0, 1\}$  as a class consisting of  $|\mathcal{D}|$  classifications of  $Q$ , we can define the VC-dimension of a data structure problem.

**Definition 18.** *The **VC-dimension** of a data structure problem  $f : Q \times \mathcal{D} \rightarrow \{0, 1\}$ , denoted by  $VC\text{-dim}(f)$ , is the maximum  $n$  such that there exists a set  $\{x_1, x_2, \dots, x_n\} \in \binom{Q}{n}$  such that for any assignment  $y \in \{0, 1\}^n$ , there exists some  $S \in \mathcal{D}$ , with  $f(x_i, S) = y_i$  for all  $i$ .*

It is easy to see that the VC-dimension of the membership problem is  $n$ , where  $n$  is the cardinality of the data set. This allows us to translate our results for problems of arbitrary VC-dimension into specific results for membership.

We consider a special class of cell-probing schemes  $(T, \mathcal{A})$  whose cell-probing algorithm  $\mathcal{A}$  satisfies a natural restriction described as follows.

**Definition 19.** *A **table structure** is a mapping  $T : \mathcal{D} \times [0, 1]^Q \times [s] \rightarrow \{0, 1\}^b$  that for any fixed  $S \in \mathcal{D}$  and any fixed query distribution  $q$  over  $Q$ , a table  $T_{S,q} : [s] \rightarrow \{0, 1\}^b$  of  $s$  cells is constructed, where each cell contains  $b$  bits.*

Given any query  $x \in Q$ , a **cell-probing algorithm**  $\mathcal{A}$  returns  $f(x, S)$  by making at most  $t^*$  randomized adaptive probes  $I_x^{(1)}, I_x^{(2)}, \dots, I_x^{(t^*)} \in [s]$  to the table  $T_{S,q}$ , such that the maximum contention  $\Phi_t \leq \phi^*$  for any  $t \leq t^*$ , and for any fixed query  $x$  and any fixed table  $T_{S,q}$ , the random variables  $I_x^{(t)}$  for all  $t \leq t^*$  are jointly independent.

The independence of cell-probes of  $\mathcal{A}$  does not make  $\mathcal{A}$  non-adaptive, because the independence holds only when the query and the table are both fixed. Note that in this sense all deterministic cell-probing algorithms (both adaptive or non-adaptive) satisfy this property, because once the table and the query are both fixed, the sequence of the cell-probes of a deterministic cell-probing algorithm are fixed, hence they are jointly independent. Informally speaking, for  $\mathcal{A}$ , the randomness is used only for balancing the cell-probes, but is not involved in the process of decision making. The upper bound presented in the previous section and any upper bounds constructed by the technique of distributing probes across multiple copies of critical cells are all included in this definition.

We prove the following lower bound theorem.

**Theorem 20.** *For any data structure problem  $f$  with  $VC\text{-dim}(f) = n$ , if there exists a cell-probing scheme  $(T, \mathcal{A})$  as defined in Definition 19, and if  $b \leq \text{Polylog}(n)$  and  $\phi^* \leq \frac{\text{Polylog}(n)}{s}$ , then  $t^* = \Omega(\log \log n)$ .*

The lower bound itself is proved based on the following intuition: the more uniform a random probe is, the less specific information it retrieves; but non-uniform probes will only result in low contention if the query algorithm already has some knowledge about the query distribution. So to obtain information about a specific query, the query algorithm must steadily increase its knowledge of the query distribution through increasingly non-uniform probes. By tracking how much information the query algorithm has about the query distribution (and thus how evenly spread out it must keep its probes), we can bound the query time subject to bounds on the contention.

A difficulty in the above argument is formally justifying the intuition: “more uniform cell-probes are less informative,” since no matter how uniform a random cell-probe is, it returns the same amount (one cell) of information.

We overcome this difficulty by first running  $n$  instances of queries in parallel, and then coupling the parallel random cell-probes to minimize the total number of cells probed in each step. In this way, we show that indeed the more uniform the random cell-probes are, the fewer cells are probed after coupling.

#### 4.1. Parallelism and coupling

We prove the following lemma.

**Lemma 21.** *If there exists a cell-probing scheme  $(T, \mathcal{A})$  for the data structure problem  $f$  where  $(T, \mathcal{A})$  and  $f$  are as in Theorem 20, then there exists a communication protocol between an algorithm  $\mathcal{A}'$  and a black-box  $\mathcal{B}$  which is specified as follows. The input to  $\mathcal{B}$  is an arbitrary **stochastic vector**  $q \in [0, 1]^n$  that  $\sum_{i=1}^n q_i \leq 1$ , which is initially unknown to  $\mathcal{A}'$ . The communication between  $\mathcal{A}'$  and  $\mathcal{B}$  occurs in rounds.*

1. At round  $t$ ,  $\mathcal{A}'$  specifies an  $n \times s$  matrix  $P_t$ , called a **probe specification**, and sends it to  $\mathcal{B}$ , where  $P_t$  is adaptive to the information received previously by  $\mathcal{A}'$ , and for any  $1 \leq i \leq n$ , it holds that

$$\sum_{j=1}^s P_t(i, j) \leq 1; \quad (1)$$

$$\max_{1 \leq j \leq s} P_t(i, j) \leq \frac{\phi^*}{q_i}. \quad (2)$$

2. Upon receiving a  $P_t$ ,  $\mathcal{B}$  sends  $C_t$  bits to  $\mathcal{A}'$ , where  $C_t$  is a random variable satisfying

$$\mathbb{E}[C_t] \leq b \cdot \sum_{j=1}^s \max_{1 \leq i \leq n} P_t(i, j), \quad (3)$$

where the expectation is conditioned on  $P_t$ , thus conditioned on all previous communication between  $\mathcal{A}'$  and  $\mathcal{B}$ .

3. After  $t^*$  rounds, the expected number of bits received by  $\mathcal{A}'$  is at least  $n \cdot 2^{-2t^*}$  bits from  $\mathcal{B}$ .

This lemma is proved in three steps as follows:

1. Each cell-probe of the original algorithm  $\mathcal{A}$  is simulated by independently probing all cells, resulting a new algorithm  $\mathcal{A}'$ .
2. Run  $n$  instances of the algorithm  $\mathcal{A}'$  in parallel.
3. Minimize the total number of cells probed by the  $n$  instances in each step by coupling the randomized parallel cell-probes.

For the rest of the proof, we assume the assumption of Theorem 20.

**Assumption 22.** *Let  $f$  be a data structure problem with  $VC\text{-dim}(f) = n$ . There exists a cell-probing scheme  $(T, \mathcal{A})$  as described in Definition 19, such that  $(T, \mathcal{A})$  solves  $f$  with performance parameters  $(s, b, t^*, \phi^*)$ , where the size of cell  $b \leq \text{Polylog}(n)$  and the maximum contention  $\phi^* \leq \frac{\text{Polylog}(n)}{s}$ .*

First, any original cell-probe is simulated (with bounded error) by a product-space cell-probe, so that in each step, every cell is probed *independently* with some fixed probability.

**Definition 23.** A *product-space cell-probe* to a table of  $s$  cells is a random set  $J \in 2^{[s]}$  such that the probability space of  $J$  is a product probability space.

**Lemma 24 (product-space lemma).** *If Assumption 22 holds, then there exists a product-space cell-probing algorithm  $\mathcal{A}'$ , such that on any valid table  $T_{S,q}$ , for any query  $x \in Q$ , the following properties hold for the sequence of product-space cell-probes  $J_x^{(1)}, J_x^{(2)}, \dots, J_x^{(t^*)}$ .*

1. *At any step  $t \leq t^*$ ,  $\mathcal{A}'$  fails (returns a special symbol  $\perp$ ) independently with probability at most  $\frac{3}{4}$ . Conditioned on that there is no failure after  $t^*$  steps, which is an event with probability at least  $2^{-2t^*}$ , it holds that  $J_x^{(1)}, J_x^{(2)}, \dots, J_x^{(t^*)}$  are jointly independent and  $\mathcal{A}'$  returns what  $\mathcal{A}$  returns.*
2. *For any  $t \leq t^*$ , the total probability of each product-space cell-probe of  $\mathcal{A}'$*

$$\sum_{j \in [s]} \Pr [j \in J_x^{(t)}] \leq 1; \quad (4)$$

3. *For any  $t \leq t^*$ , the contention of any cell  $j \in [s]$*

$$q(x_i) \cdot \Pr [j \in J_x^{(t)}] \leq \phi^*. \quad (5)$$

PROOF. A cell-probe of  $\mathcal{A}$  can be represented as a random variable  $I \in [s]$ , where  $I$  denotes the probed cell. Let  $p_i := \Pr[I = i]$ . Let  $J \in 2^{[s]}$  represent a product-space cell-probe. Given a probability vector  $p$ , a cell-probe  $I$  is simulated by a product-space cell-probe as follows: Independently probe each cell  $i \in [s]$  with probability  $p'_i := \min\{p_i, \frac{1}{2}\}$ . The resulting set is  $J$ . If  $|J| \neq 1$ , then fails; if  $J = \{i\}$ , then fails with a probability  $\epsilon_i := \min\{p_i, 1 - p_i\}$ . Let  $I = i$  if not fail.

Case 1:  $p_i \leq \frac{1}{2}$  for all  $i \in [s]$ . Then for all  $i \in [s]$ ,  $p'_i = p_i$  and  $\epsilon_i = p_i$ . Let  $\rho = \prod_{j \in [s]} (1 - p_j)$ . Since  $p_i \leq \frac{1}{2}$  for all  $i \in [s]$ , it holds that  $\rho \geq \frac{1}{4}$ .

The probability

$$\Pr[I = i] = (1 - \epsilon_i) \cdot \Pr[J = \{i\}] = (1 - p_i) \cdot p_i \prod_{j \neq i} (1 - p_j) = p_i \rho,$$

which is proportional to  $p_i$ . The procedure succeeds with probability  $\rho \geq \frac{1}{4}$ .

Case 2: Let  $p_0 > \frac{1}{2}$  and all other  $p_i < \frac{1}{2}$ . Then  $p'_0 = \frac{1}{2}$  and  $\epsilon'_0 = 1 - p_0$ , and for all  $i > 0$ , it holds that  $p'_i = p_i$  and  $\epsilon'_i = p_i$ . Let  $\rho' = \prod_{j>0} (1 - p_j)$ . It holds that  $\rho' > \frac{1}{2}$  since  $\sum_{j>0} p_j = 1 - p_0 < \frac{1}{2}$ .

For  $i \neq 0$ , the probability  $\Pr[I = i] = (1 - p_i) \cdot \Pr[J = \{i\}] = \frac{1}{2}\rho'p_i$ ; and for cell 0,  $\Pr[I = 0] = p_0 \cdot \Pr[J = \{i\}] = \frac{1}{2}\rho'p_0$ .

The procedure succeeds with probability  $\frac{1}{2}\rho' > \frac{1}{4}$ .

In both cases, a cell-probe of  $\mathcal{A}$  is simulated by a product-space cell-probe with probability at least  $\frac{1}{4}$ . The event of a failure occurs independently with probability at most  $\frac{3}{4}$ . With probability at least  $2^{-2t^*}$ , no failure occurs at all, in which case it happens that  $\mathcal{A}'$  can simulate  $\mathcal{A}$ , and the product-space cell-probes are jointly independent since the cell-probes of  $\mathcal{A}$  are jointly independent.

The total probability of a product-space cell-probe is  $\sum_i \Pr[i \in J] = \sum_i p'_i \leq 1$ . The probability of a probe to each cell is no greater than before, therefore the maximum contention  $\phi^*$  is not increased.

Next, we run  $n$  instances of the product-space cell-probing algorithm in parallel. The behavior of each individual instance depends only on the marginal distribution of cell-probes of the instance, but so we can choose the joint distribution of cell-probes arbitrarily as long as the marginal distributions are correct.

**Lemma 25 (parallel lemma).** *Let  $\mathcal{A}''$  be an algorithm that on a valid table  $T_{S,q}$ , for a set of  $n$  queries  $\{x_1, x_2, \dots, x_n\} \in \binom{Q}{n}$ , at step  $t$ ,  $\mathcal{A}''$  randomly probes  $n$  sets of cells  $(L_{x_1}^{(t)}, L_{x_2}^{(t)}, \dots, L_{x_n}^{(t)})$ . If for every  $x_i$  where  $1 \leq i \leq n$  and every  $t \leq t^*$ , the marginal distribution of  $L_{x_i}^{(t)}$  is identical to the distribution of  $J_{x_i}^{(t)}$ , where  $J_{x_i}^{(t)}$  is the  $t$ 's product-space cell-probe of the algorithm  $\mathcal{A}'$  on the same table  $T_{S,q}$ , then  $\mathcal{A}''$  returns  $f(x_i, S)$  for expected  $n \cdot 2^{-2t^*}$  number of  $x_i$ .*

PROOF. Let  $\mathcal{A}''$  run an instance of  $\mathcal{A}'$  with input  $x_i$  in parallel for every  $x_i$ , denoted as  $\mathcal{A}'_{x_i}$ . Let the set of cells probed by each individual  $\mathcal{A}'_{x_i}$  at time  $t$  be  $L_{x_i}^{(t)}$ . On a fixed table  $T_{S,q}$  and for a fixed  $x_i$ , since  $L_{x_i}^{(t)}$  is identically distributed as  $J_{x_i}^{(t)}$ , every individual instance of  $\mathcal{A}'_{x_i}$  simulates a running instance of  $\mathcal{A}'$  with input query  $x_i$ . By Lemma 24, each  $\mathcal{A}'_{x_i}$  terminates in  $t^*$

steps without failure with probability at least  $2^{-2t^*}$ , thus by the linearity of expectation, after  $t^*$  time, the expected total number of terminated instances is at least  $n \cdot 2^{-2t^*}$ .

Then, we minimize the expected total number of probed cells by coupling the randomness of parallel cell-probes.

**Lemma 26 (coupling lemma).** *For any probability distribution of  $J_i \subseteq [s]$  where  $1 \leq i \leq n$  and each  $J_i$  is chosen from a product probability space, there exists a joint distribution  $(L_1, L_2, \dots, L_n)$ , such that for every  $1 \leq i \leq n$ ,  $L_i$  is identically distributed as  $J_i$ , and it holds that*

$$\mathbb{E} \left[ \left| \bigcup_{1 \leq i \leq n} L_i \right| \right] \leq \sum_{j \in [s]} \max_{1 \leq i \leq n} \Pr[j \in J_i].$$

PROOF. We construct the joint distribution of  $(L_i)_i \in (2^{[s]})^n$  as follow.

- Let  $\tilde{p}_j = \max_{1 \leq i \leq n} \Pr[j \in J_i]$ . Choose each  $j \in [s]$  independently with probability  $\tilde{p}_j$ . Let  $B$  denote the set of chosen elements of  $[s]$ .
- For every  $1 \leq i \leq n$ , let each  $j \in B$  join  $L_i$  independently with probability  $\frac{\Pr[j \in J_i]}{\tilde{p}_j}$ . Note that  $\tilde{p}_j \geq \Pr[j \in J_i]$ , so the probability is well-defined.

For each  $1 \leq i \leq n$ , and for every  $j \in [s]$ ,  $j$  joins  $L_i$  independently with probability  $\tilde{p}_j \cdot \Pr[j \text{ is chosen to } L_i \mid j \in B] = \Pr[j \in J_i]$ , thus each  $L_i$  is identically distributed as  $J_i$ .

Note that for every  $L_i$ , all of its elements are chosen from set  $B$ . It holds that

$$\mathbb{E} \left[ \left| \bigcup_{1 \leq i \leq n} L_i \right| \right] \leq \mathbb{E}[|B|] = \sum_{j \in [s]} \tilde{p}_j = \sum_{j \in [s]} \max_{1 \leq i \leq n} \Pr[j \in J_i].$$

Combining the above three lemmas, Lemma 21 can be proved.

PROOF OF LEMMA 21. Let  $\{x_1, x_2, \dots, x_n\} \in \binom{Q}{n}$  be a set of queries which achieves the VC-dimension  $\text{VC-dim}(f) = n$ , i.e. any Boolean assignment of  $f(x_i, S)$  is possible. Let such  $\{x_1, x_2, \dots, x_n\}$  be the input query set to  $\mathcal{A}''$ , where  $\mathcal{A}''$  is as described in Lemma 25. By an information theoretical argument, in the worst case,  $\mathcal{A}''$  has to collect expected  $n \cdot 2^{-2t^*}$  bits information after  $t^*$  steps. Let the joint distribution of  $(L_{x_1}^{(t)}, L_{x_2}^{(t)}, \dots, L_{x_n}^{(t)})$  of  $\mathcal{A}''$

be constructed as in Lemma 26, the total number of cells probed by  $\mathcal{A}''$  in step  $t$  with  $(L_{x_1}^{(t)}, L_{x_2}^{(t)}, \dots, L_{x_n}^{(t)})$  is bounded. Let the  $n \times s$  matrix  $P_t$  defined as  $P_t(i, j) := \Pr[j \in L_{x_i}^{(t)}]$ , and let  $q_i := q(x_i)$ . Since each  $L_{x_i}^{(t)}$  is identically distributed as  $J_{x_i}^{(t)}$  of  $\mathcal{A}'$  which is described in Lemma 24, due to (2) and (3), it holds for the  $P_t$  that  $\sum_{j \in [s]} P_t(i, j) \leq 1$  and  $\max_{j \in [s]} P_t(i, j) \leq \frac{\phi^*}{q_i}$ . By Lemma 26, the expected number of bits collected by  $\mathcal{A}''$  in step  $t$  is bounded by  $b \cdot \sum_{j \in [s]} \max_{1 \leq i \leq n} P_t(i, j)$ . By seeing the running instance of the algorithm  $\mathcal{A}''$  with the input  $\{x_1, x_2, \dots, x_n\}$  as the player  $\mathcal{A}''$  of the communication game, and the table  $T_{S,q}$  as the black-box  $\mathcal{B}$  with private input  $q$ , Lemma 21 is proved.

#### 4.2. Two technical lemmas

The following two combinatorial lemmas are needed for the proof of Theorem 20. The first lemma is for the strategy space of an algorithm, and the second lemma is for the probability distributions of  $n$  parallel cell-probes in each step.

**Lemma 27.** *Let  $M$  be an  $N \times n$  nonnegative matrix. Let  $r = \sqrt{5\epsilon^{-1}\delta n \ln N}$ . Assume that for every row  $1 \leq u \leq N$ , there exists a set  $R_u \in \binom{\{1,2,\dots,n\}}{r}$  of  $r$  entries such that  $\sum_{i \in R_u} M(u, i) \leq \delta$ . Then there exists  $q \in [0, 1]^n$  that  $\sum_i q_i = \epsilon$ , such that for all  $1 \leq u \leq N$ , there exists  $1 \leq i \leq n$ , such that  $M(u, i) < q_i$ .*

PROOF. For each  $1 \leq u \leq N$ , sort  $\{M(u, i) \mid i \in R_u\}$  by non-decreasing order and let  $R'_u \subseteq \{1, 2, \dots, n\}$  be the indices of the smallest  $\frac{r}{2}$  entries. It holds that  $\forall i \in R'_u, M(u, i) \leq \frac{2\delta}{r}$ , as otherwise it contradicts the assumption that  $\sum_{i \in R_u} M(u, i) \leq \delta$ .

It holds that for any choice of such  $\{R'_u\}_{1 \leq u \leq N}$ , there exists a  $T \subseteq \{1, 2, \dots, n\}$ , such that  $|T| = \frac{2n \ln N}{r}$  and  $T$  intersects all  $R'_u$ . We prove this by the probabilistic method: let  $T$  be a uniformly random subset of  $\{1, 2, \dots, n\}$  of size  $\frac{2n \ln N}{r}$ , thus each  $R'_u$  is missed by  $T$  with probability less than  $(1 - r/2n)^{2n \ln N/r} < 1/N$ , thus by the union bound,  $T$  intersects all  $R'_u$  with positive probability.

Fix such a  $T$ , define  $q \in [0, 1]^n$  as  $q_i = \epsilon|T|^{-1} = \frac{r\epsilon}{2n \ln N}$  if  $i \in T$ , and  $q_i = 0$  if otherwise. Therefore,  $\sum_i q_i = \epsilon$ , and for any  $1 \leq u \leq N$ , for such  $i \in R'_u \cap T$ , it holds that  $M(u, i) \leq \frac{2\delta}{r} < \frac{r\epsilon}{2n \ln N} = q_i$ .

**Lemma 28.** *For any nonnegative  $n \times s$  matrix  $P$  that  $\sum_j P(i, j) \leq 1$  for every  $i$ , let  $R$  be the largest subset of  $\{1, 2, \dots, n\}$  such that  $\sum_{i \in R} \frac{1}{\max_j P(i, j)} \leq$*

s. Then it holds that

$$|R| \geq \sum_{j=1}^s \max_{1 \leq i \leq n} P(i, j).$$

PROOF. The sum  $\sum_j \max_i P(i, j)$  chooses exactly  $s$  entries to sum up. Let  $A_i$  be the set of chosen columns in row  $i$ . Let  $x_i := \sum_{j \in A_i} P(i, j)$ . Note that  $x_i \leq \sum_j P(i, j) \leq 1$ . By the pigeonhole principle, for any  $1 \leq i \leq n$ ,

$$|A_i| \geq \frac{\sum_{j \in A_i} P(i, j)}{\max_j P(i, j)} = \frac{x_i}{\max_j P(i, j)}.$$

Note that  $\sum_i |A_i| = s$ , thus  $\sum_i \frac{x_i}{\max_j P(i, j)} \leq \sum_i |A_i| = s$ .

Therefore the sum  $\sum_j \max_i P(i, j)$  can be written as  $\sum_j \max_i P(i, j) = \sum_i \sum_{j \in A_i} P(i, j) = \sum_i x_i$ , subject to the constraints that  $\sum_i \frac{x_i}{\max_j P(i, j)} \leq s$  and  $x_i \leq 1$ . It is easy to see that the value of  $\sum_i x_i$  is maximized when letting  $x_i = 1$  for  $i \in R$  and  $x_i = 0$  for  $i \notin R$ , therefore  $\sum_j \max_i P(i, j) = \sum_i x_i \leq |R|$ .

### 4.3. The adversarial argument

We now prove the lower bound (Theorem 20) by applying an adversarial argument.

In Section 4.1, we show that in each step, the algorithm chooses an  $n \times s$  matrix  $P$  which specifies the distribution of  $n$  parallel cell-probes, causing maximum contention  $\max_{1 \leq i \leq n, 1 \leq j \leq s} q_i \cdot P_t(i, j)$ , and returning expected  $\sum_{j=1}^s \max_{1 \leq i \leq n} P_t(i, j)$  cells after coupling.

The adversary always chooses the potential query distribution  $q$  so that all random cell-probes with low contention (i.e.  $P$  with  $q_i \cdot \max_{1 \leq j \leq s} P_t(i, j) \leq \phi^*$  for all  $i$ ) return limited amount of information after coupling (i.e. having small  $\sum_{j=1}^s \max_{1 \leq i \leq n} P_t(i, j)$ ). However, by the assumption, after  $t^*$  steps, the algorithm must have acquired an adequate amount of information to determine the queries. This gives us the lower bound on  $t^*$ .

PROOF OF THEOREM 20. Given the algorithm  $\mathcal{A}''$  described in Lemma 21, we will bound the speed at which  $\mathcal{A}''$  gathers information. Due to Lemma 21,  $\mathcal{A}''$  is a decision tree in which the current node of depth  $(t - 1)$  has  $N_t := 2^{C_{t-1}}$  children, each of which corresponds to a next probe specification  $P_t$ . We number these  $P_t$  by  $u \in [N_t]$  and denote each as  $P_t^{(u)}$ , where  $u$  can be interpreted as the bit string received by  $\mathcal{A}''$  at round  $t - 1$ . We then inductively bound the next  $C_t$ .

Define an  $N_t \times n$  matrix  $M^{(t)}$  as that  $M^{(t)}(u, i) := \frac{\phi^*}{\max_j P_t^{(u)}(i, j)}$ . Each row of the matrix  $M^{(t)}$  corresponds to a possible next probe specification. We say that the stochastic vector  $q$  **violates** row  $u$  of  $M^{(t)}$  if there exists  $1 \leq i \leq n$ , such that  $M^{(t)}(u, i) < q_i$ . Note that if row  $u$  of  $M^{(t)}$  is violated by  $q$ , then according to (2), the next probe specification cannot be  $P_t^{(u)}$ .

Let  $r_t := \sqrt{5t^* \phi^* sn \ln N_t}$ . We say that a row  $u$  of  $M^{(t)}$  is **good** if there exists  $R \subseteq \{1, 2, \dots, n\}$  such that  $|R| = r_t$  and  $\sum_{i \in R} M^{(t)}(u, i) \leq \phi^* s$ .

We claim that if a row  $u$  is not good, then for the corresponding  $P^{(u)}$ , it holds that

$$\sum_{j=1}^s \max_{1 \leq i \leq n} P_t^{(u)}(i, j) \leq r_t. \quad (6)$$

The proof is as follows: If a row  $u$  of  $M^{(t)}$  is not good, then by definition, for any  $R$  of size  $r_t$ ,  $\sum_{i \in R} M^{(t)}(u, i) > \phi^* s$ , thus for any  $R'$  that  $\sum_{i \in R'} \frac{1}{\max_j P_t^{(u)}(i, j)} \leq s$ , it must hold that  $|R'| < r_t$ , therefore by Lemma 28, it holds that  $\sum_{j=1}^s \max_{1 \leq i \leq n} P_t^{(u)}(x_i, j) \leq r_t$ .

By (6) and (3), the amount of information brought by a set of probes  $P_t^{(u)}$  where  $u$  is a bad row in  $M^{(t)}$ , is bounded by  $br_t$  bits. We show by an adversary argument that there exists  $q$  for which  $\mathcal{A}''$  always chooses probes corresponding to bad rows. At each round  $t$ , the adversary always chooses some  $q$  that violates all of the good rows in  $M^{(t)}$ . According to Lemma 27, the adversary can do so as long as  $t \leq t^*$ . Setting  $\epsilon = \frac{1}{t^*}$  and  $\delta = \phi^* s$  in Lemma 27, in each round, the adversary can increase the value of some  $q_i$  so that  $\sum_{i=1}^n q_i$  is increased by at most  $\frac{1}{t^*}$ , thereby violating all good rows in the current  $M^{(t)}$ . Thus before round  $t^*$ , the vector  $q$  is always stochastic. Note that increasing the value of  $q_i$  will never make a violated row non-violated, so it will not make the adversary inconsistent.

Against such an adversary, at each round  $t$ ,  $\mathcal{A}''$  can only choose a probe specification  $P_t^{(u)}$  where  $u$  is a bad row in  $M^{(t)}$ , according to Claim (6), which implies that

$$\sum_{j=1}^s \max_{1 \leq i \leq n} P_t(i, j) \leq r_t = \sqrt{5t^* \phi^* sn \ln N_t} = \sqrt{5t^* \phi^* sn C_{t-1} \ln 2}.$$

Due to (3),  $\mathbb{E}[C_t \mid \dots] \leq b \cdot \sum_j \max_i P_t(i, j) \leq \sqrt{(5 \ln 2) b^2 t^* \phi^* sn C_{t-1}}$ , where the expectation is conditioned on all previous rounds of communication. Therefore the following recursion holds for the sequence of random

variables  $C_1, C_2, \dots, C_t$ :

$$\mathbb{E}[C_t | C_{t-1}] \leq \sqrt{(5 \ln 2) b^2 t^* \phi^* s n C_{t-1}}.$$

The square root function is concave, thus by Jensen's inequality, it holds for the unconditional expectation that

$$\begin{aligned} \mathbb{E}[C_t] &= \mathbb{E}[\mathbb{E}[C_t | C_{t-1}]] \\ &\leq \mathbb{E}\left[\sqrt{(5 \ln 2) b^2 t^* \phi^* s n C_{t-1}}\right] \\ &\leq \sqrt{(5 \ln 2) b^2 t^* \phi^* s n \cdot \mathbb{E}[C_{t-1}]}. \end{aligned}$$

Before the first probe,  $q$  is unknown to  $\mathcal{A}''$ , thus due to (2), for any  $i, j$ ,  $P_1(x_i, j) \leq \phi^*$ , therefore  $\mathbb{E}[C_1] \leq b \cdot \sum_j \max_i P_1(x_i, j) \leq b \phi^* s$ . Let  $a_1 := b \phi^* s$  and  $a := (5 \ln 2) b^2 t^* \phi^* s n$ . The following recursion holds for  $\mathbb{E}[C_t]$  that

$$\begin{aligned} \mathbb{E}[C_1] &\leq a_1; \\ \mathbb{E}[C_t] &\leq (a \cdot \mathbb{E}[C_{t-1}])^{\frac{1}{2}}. \end{aligned}$$

By induction,  $\mathbb{E}[C_t] \leq a_1^{2^{1-t}} a^{1-2^{1-t}}$ .

After  $t^*$  rounds, the expected total number of bits received by  $\mathcal{A}''$  is at least  $n \cdot 2^{-2t^*}$ , therefore

$$n \cdot 2^{-2t^*} \leq \sum_{t \leq t^*} \mathbb{E}[C_t] \leq \sum_{t \leq t^*} a_1^{2^{1-t}} a^{1-2^{1-t}} \leq a_1 a^{1-2^{-t^*}}.$$

With the assumption that  $b \leq \text{Polylog}(n)$  and  $\phi^* \leq \frac{\text{Polylog}(n)}{s}$ , it holds that  $a_1 \leq \text{Polylog}(n)$  and  $a \leq n \cdot \text{Polylog}(n)$ . Solving the above inequality, we have that  $t^* \geq \log \log n - o(\log \log n) = \Omega(\log \log n)$ . Theorem 20 is proved.

## 5. Conclusion

In this paper, we propose to study the memory contention caused by concurrent data structure queries. To study the problem, we introduce a measure of contention to the classic cell-probe model of static data structures. We show that if all positive queries are equally probable and similarly all negatively are equally probable, then there exists a static dictionary which answers membership queries with asymptotically optimal performance of time, space and contention. For the general case that the query distribution is arbitrary, we show that for all data structure problems with

non-degenerating VC-dimensions, if the randomness is used only for balancing the probes, then even with unbounded space, the time and contention cannot be both optimal.

A possible future direction is to remove the assumption of independent cell-probes in the lower bound. Note that we only rely on this assumption to make sure that the contention constraint of (2) holds conditioning on any particular sequence of previous cell-probes, which is required by the adversary argument. We suspect that with a more careful analysis, this assumption can be removed, which would imply that the lower bound holds not only for the randomized data structures that use the randomness only for balancing probes, but also for the true randomized data structures where the randomness is also involved in the computation of queries.

Another interesting and perhaps more realistic future direction is to study the contention caused by the *updates* in dynamic data structures.

## References

- [1] P. Tzeng, D. Lawrie, Distributing hot-spot addressing in large-scale multiprocessors, *IEEE Transactions on Computers* 100 (36) (1987) 388–395.
- [2] M. Herlihy, B. Lim, N. Shavit, Low contention load balancing on large-scale multiprocessors, in: *Proceedings of the fourth annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, ACM New York, NY, USA, 1992, pp. 219–227.
- [3] C. Dwork, M. Herlihy, O. Waarts, Contention in shared memory algorithms, *Journal of the ACM (JACM)* 44 (6) (1997) 779–805.
- [4] S. Fortune, J. Wyllie, Parallelism in random access machines, in: *Proceedings of the tenth annual ACM Symposium on Theory of Computing (STOC)*, ACM New York, NY, USA, 1978, pp. 114–118.
- [5] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramonian, T. Von Eicken, LogP: Towards a realistic model of parallel computation, *ACM SIGPLAN Notices* 28 (7) (1993) 1–12.
- [6] A. C.-C. Yao, Should tables be sorted?, *J. ACM* 28 (3) (1981) 615–628. doi:<http://doi.acm.org/10.1145/322261.322274>.
- [7] M. Fredman, J. Komlós, E. Szemerédi, Storing a Sparse Table with  $O(1)$  Worst Case Access Time, *Journal of the ACM (JACM)* 31 (3) (1984) 538–544.

- [8] M. Dietzfelbinger, F. Meyer auf der Heide, A new universal class of hash functions and dynamic hashing in real time, in: Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP), Vol. 443, Springer, 1990, pp. 6–19.
- [9] M. Dietzfelbinger, F. Meyer auf der Heide, An optimal parallel dictionary, in: Proceedings of the first annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), ACM New York, NY, USA, 1989, pp. 360–368.
- [10] M. Dietzfelbinger, F. Meyer auf der Heide, How to distribute a dictionary in a complete network, in: Proceedings of the twenty-second annual ACM Symposium on Theory of Computing (STOC), ACM New York, NY, USA, 1990, pp. 117–127.
- [11] R. Pagh, F. Rodler, Cuckoo hashing, *Journal of Algorithms* 51 (2) (2004) 122–144.
- [12] J. Carter, M. Wegman, Universal classes of hash functions, *Journal of Computer and System Sciences (JCSS)* 18 (2) (1979) 143–154.
- [13] C. P. Kruskal, L. Rudolph, M. Snir, A complexity theory of efficient parallel algorithms, *Theor. Comput. Sci.* 71 (1) (1990) 95–132. doi:[http://dx.doi.org/10.1016/0304-3975\(90\)90192-K](http://dx.doi.org/10.1016/0304-3975(90)90192-K).
- [14] W. Hoeffding, Probability inequalities for sums of bounded random variables, *Journal of the American Statistical Association* 58 (301) (1963) 13–30.
- [15] V. Vapnik, A. Y. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, *Theory of Probability and its Applications* 16 (2) (1971) 264–280.