Correcting Deletions in Multiple-Heads Racetrack Memories

Jin Sima and **Jehoshua Bruck**
Department of Electrical Engineering, California Institute of Technology, Pasadena 91125, CA, USA

***Abstract*—One of the main challenges in developing racetrack memory systems is the limited precision in controlling the track shifts, that in turn affects the reliability of reading and writing the data. The current proposal for combating deletions in racetrack memories is to use redundant heads per-track resulting in multiple copies (potentially erroneous) and solving a specialized version of a sequence reconstruction problem. Using this approach, k-deletion correcting codes of length n, with d heads per-track, with redundancy $\log\log n+4$ were constructed. However, the code construction requires that $k \leq d$. For $k > d$, the best known construction improves slightly over the classic one head deletion code. Here we address the question: What is the best redundancy that can be achieved for a k-deletion code (k is a constant) if the number of heads is fixed at d (due to area limitations)? Our key result is an answer to this question, namely, we construct codes that can correct k deletions, for any k beyond the known limit of d. The code has $O(k^4 d \log\log n)$ redundancy for the case when $k \leq 2d-1$. In addition, when $k \geq 2d$, the code has $2\lfloor k/d \rfloor \log n + o(\log n)$ redundancy.**

I. Introduction

Racetrack memory is a promising non-volatile memory that possesses the advantages of ultra-high storage density and low latency (comparable to SRAM latency) [8], [9]. It has a tape-like structure where the data is stored sequentially as a track of single-bit memory cells. The cells are accessed through read/write ports, called heads. When reading/writing the data, the heads stay fixed and the track is shifting.

One of the main challenges in developing racetrack memory systems is the limited precision in controlling the track shifts, that in turn affects the reliability of reading and writing the data [5], [11]. Specifically, the track may either not shift or shift more steps than expected. When the track does not shift, the same cell is read twice, causing a sticky insertion. When the track shifts more than a single step, cells are skipped, causing deletions in the reads [3].

It is natural to use deletion and sticky insertion correcting codes to deal with shift errors. Also, it is known that a code correcting k deletions is capable of correcting s deletions and r insertions when $s + r \leq k$ [6]. However, designing redundancy and complexity efficient deletion correcting codes has been an open problem for decades. In fact, no deletion correcting codes with rate approaching 1 were known until recently, when [1] proposed a code with redundancy $128k^2 \log k \log n + o(\log n)$. Evidently, for k, a constant number of deletions, the redundancy of this code is orders of magnitude away from optimal, known to be in the range $k \log n + o(\log n)$ to $2k \log n + o(\log n)$. Hence, it is natural to explore constructions of deletion correcting codes that are specialized for racetrack memories and might provide more efficient redundancy and lower complexity encoding/decoding algorithms.

There are two approaches for construction of codes for racetrack memories. The first is to leverage the fact that there are multiple parallel tracks with a single head per-track, and the second, is to add redundant heads per-track. For the multiple parallel head structure, the proposed codes in [10] can correct up to two deletions per head and the proposed codes in [2] can correct l bursts of deletions, each of length at most b. The codes in [2] are asymptotically (in the number of heads) rate-optimal. The second approach for combating deletions in racetrack memories is to use redundant heads per-track [11], [3], [4], resulting in multiple copies (potentially erroneous) of the same sequence. This can be regarded as a sequence reconstruction problem, where a sequence $\mathbf{c}$ needs to be recovered from multiple copies, each obtained after k deletions in $\mathbf{c}$. We emphasize that the general sequence reconstruction problem [7] is different from the current settings, as here the heads are at fixed and known positions, hence, the set of deletions locations in one head is a shift of that in another head [3]. Demonstrating the advantage of multiple heads, the paper [4] proposed an efficient k-deletion code of length n with redundancy $\log\log n+4$. In contrast, for general k-deletion codes the lower bound on the redundancy is $k \log n$. However, the code in [4] is required to use d heads and is limiting k to be smaller or equal to d. For k larger than d, the best construction [3] reduces the problem to designing a $k-d+1$ deletion code (e.g., [1]) for single head cases. It is known that the number of heads affects the area overhead of the racetrack memory device [3], hence, it motivates the following **natural question**: What is the best redundancy that can be achieved for a k-deletion code (k is a constant) if the number of heads is fixed at d (due to area limitations)?

Our **key result** is an answer to this question, namely, we construct codes that can correct k deletions, for any k beyond the known limit of d. Our code has $O(k^4 d \log\log n)$ redundancy for the case when $k \leq 2d-1$. In addition, when $k \geq 2d$, the code has $2\lfloor k/d \rfloor \log n + o(\log n)$ redundancy. Our key result is summarized formally by the following theorem. Notice that the theorem implies that the redundancy of our codes is asymptotically larger than optimal by a factor of at most four.

Theorem 1. *For a constant integer k, let the distance t*

The work was supported in part by NSF grants CCF-1816965 and CCF-1717884 .

between any two consecutive heads be $\max\{(3k+\lceil\log n\rceil+2)[k(k-1)/2+1]+(7k-k^3)/6, (4k+1)(5k+\lceil\log n\rceil+3)\}$. *Then for* $d \le k \le 2d-1$, *there exists a length* $N = n+4k[2(4k+1)kd+4k+5/2]\log\log n+o(\log\log n)$ d *head* k*-deletion correcting code with redundancy* $4k[2(4k+1)kd+4k+5/2]\log\log n+o(\log\log n)$. *For* $k \ge 2d$, *there exists a length* $N = n+2\lfloor k/d\rfloor\log n+o(\log n)$ d *head* k*-deletion correcting code with redundancy* $2\lfloor k/d\rfloor\log n+o(\log n)$. *The encoding and decoding functions can be computed in* $O_k(poly(n))$ *time. Moreover, for* $k \ge 2d$ *and* $t_i = n^{o(1)}$, *the amount of redundancy of a* d *head* k*-deletion correcting code is lower bounded by* $\lfloor k/2d\rfloor\log n+o(\log n)$.

Organization: Due to space limitation, we prove our main result for the case $k \le 2d-1$. The proof for the case $k \ge 2d$ is similar. The proof of the lower bound on the redundancy will be presented in a longer version. In Section II we present some basic lemmas needed in our proof. Section III presents the proof of the main result for the case $k \le 2d-1$. Section IV describes in detail how to synchronize the reads.

II. Preliminaries

A. Notations and Model

For any two numbers $i \le j$, let $[i,j] = \{i, i+1, \ldots, j-1, j\}$ be the set of consecutive numbers between i and j. Let $[i,j] = \emptyset$ for $i > j$. For a length n sequence $\mathbf{c} = (c_1, \ldots, c_n)$ and an index set $\mathcal{I} \subseteq [1,n]$, let $\mathbf{c}_{\mathcal{I}} = (c_i : i \in \mathcal{I})$ be a subsequence of $\mathbf{c}$. For a $n_1 \times n_2$ matrix D and two index sets $\mathcal{I}_1 \subseteq [1,n_1]$ and $\mathcal{I}_2 \subseteq [1,n_2]$, let $D_{\mathcal{I}_1,\mathcal{I}_2}$ denote the submatrix of D formed by choosing the intersection of the rows $i \in \mathcal{I}_1$ and the columns $j \in \mathcal{I}_2$.

For a deletion location set $\boldsymbol{\delta} \subseteq [1,n]$, denote by $\boldsymbol{\delta}^c = [1,n]\backslash\boldsymbol{\delta}$ the complement of $\boldsymbol{\delta}$. Let $\mathbf{c}(\boldsymbol{\delta}) = \mathbf{c}_{\boldsymbol{\delta}^c} = (c_i : i \notin \boldsymbol{\delta})$ be the subsequence obtained by deleting bits with locations in the deletion location set $\boldsymbol{\delta}$.

In a d head racetrack memory, the heads are placed on the same track and stay fixed. As the track moves step by step, the heads read the same copy of data d times. A deletion or insertion happens for all reads when the track moves two steps or does not move at a time. As a result, the deletion/insertion locations in one head is a shift version of that in another head. In this paper we focus on the deletion errors.

Let the distance between the i-th head and the $i+1$-th head be t_i, $i \in [1, d-1]$. Denote as $\boldsymbol{\delta}_i = \{\delta_{i,1}, \ldots, \delta_{i,k}\} \subseteq [1,n]$, $i \in [1,d]$ the location of deletions occur at the i-th head. Then we have that $\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t_i$, where for a set of numbers A and a number t, $A + t = \{x + t : x \in A\}$.

The read of the i-th head is given by $\mathbf{c}(\boldsymbol{\delta}_i)$, $i \in [1,d]$. Define the read matrix $D(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d)$ to be a $d \times (n-k)$ matrix where the i-th row of $D(\mathbf{c}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d)$ is $\mathbf{c}(\boldsymbol{\delta}_i)$. The deletion ball $D_k(\mathbf{c})$ of a sequence $\mathbf{c} \in \{0,1\}^n$ is the set of all possible read matrices in a d head racetrack memory, i.e.,

$$D_k(\mathbf{c}) = \{D(\mathbf{c},\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_d) : \boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t_i, \\ |\mathbf{c}| = n, \boldsymbol{\delta}_i \subseteq [1,n], |\boldsymbol{\delta}_i| = k, i \in [1, d-1]\}.$$

A d head k deletion code $\mathcal{C}$ is the set of all length n sequences such that the deletion balls of any two do not intersect, i.e., for any $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$, $D_k(\mathbf{c}) \cap D_k(\mathbf{c}') = \emptyset$. The redundancy of the code is $n - \log|\mathcal{C}|$.

A sequence $\mathbf{c} \in \{0,1\}^n$ is said to have period ℓ if $c_i = c_{i+\ell}$ for $i \in [1, n-\ell]$. We use $L(\mathbf{c}, i)$ to denote the length of the longest subsequence of consecutive bits in $\mathbf{c}$ that has period i. Furthermore, define

$$L(\mathbf{c}, \le k) = \max_{i \le k} L(\mathbf{c}, i).$$

Finally, the number of elements of a set A is denoted by $|A|$. For any two sets A and B, the set $A\backslash B = \{x : x \in A, x \notin B\}$ denote the difference of sets A and B.

B. Useful Lemmas

Here we present some lemmas that will be useful throughout the paper. Our construction is based upon a deletion code for short lengths, which can be computed by brute force in $O_k(poly(n))$ time when the length is of order $O_k(\log n)$. The following result appears in [1].

Lemma 1. *Let* k *be a fixed integer. There exists a hash function* $H : \{0,1\}^n \to \{0,1\}^{2k\log n+O(1)}$, *computable in* $O_k(n^{2k}2^n)$ *time, such that any sequence* $\mathbf{c} \in \{0,1\}^n$ *can be recovered from any of its length* $n-k$ *subsequences and the hash* $H(\mathbf{c})$.

With the function H, it is not hard to protect a longer length sequence from k deletions by chopping it into short subsequences and protect each short subsequence using H. The following lemma is a slight variation of the result in [1].

Lemma 2. *Let* k *be a fixed integer. For integers* B *and* n. *There exists a hash function* $Hash : \{0,1\}^B \to \{0,1\}^{\lceil(B/\lceil\log n\rceil)\rceil(2k\log\log n+O(1))}$, *computable in* $O_k(\lceil B/\log n\rceil n\log^{2k} n)$ *time, such that any sequence* $\mathbf{c} \in \{0,1\}^B$ *can be recovered from any of its length* $B-k$ *subsequences and the hash* $Hash(\mathbf{c})$.

In addition, in order to synchronize the sequence $\mathbf{c}$ in the presence of deletions, we need to transform $\mathbf{c}$ to a sequence that has a limited length constraint on its periodic subsequences. Such constraint was used in [3], where it was proved that the redundancy of the code $\{\mathbf{c} : L(\mathbf{c}, \le k) \le \lceil\log n\rceil + k + 1\}$ is at most 1 bit. In the following lemma we present a function to transform any sequence to one that satisfies this constraint. The redundancy of our construction is $k+1$ bits. However, it is small compared to the redundancy of the d head k-deletion code.

Lemma 3. *For any integers* k *and* n, *there exists an injective function* $F : \{0,1\}^n \to \{0,1\}^{n+k+1}$, *computable in* $O(kn^2\log n)$ *time, such that for any sequence* $\{0,1\}^n$, *we have that* $L(F(\mathbf{c}), \le k) \le 3k+2+\lceil\log n\rceil$.

Finally, we restate one of the main results in [3] that will be use in our construction. It is a code that can correct $d-1$ deletions in a d head racetrack memory, given that the distance between any two consecutive heads is large enough.

Lemma 4. *Let* $d \le k$ *be two integers and* $\mathcal{C}$ *be a* $k-d+1$ *deletion code, then* $\mathcal{C} \cap \{\mathbf{c} : L(\mathbf{c}, \le k) \le T\}$ *is a* d *head* k*-deletion correcting code, given that the distance between any*

two consecutive heads $t_i \geq T[k(k-1)/2+1]+(7k-k^3)/6$ *for* $i \in [1, d-1]$.

III. Correcting up to $2d-1$ deletions with d heads

In this section we construct a k-deletion d head code for the case when $k \leq 2d-1$. To this end, we first present a lemma that is the basis of our code construction. The lemma states that the deletion locations can be identified within a set of short intervals. Moreover, the number of deletions within each interval can be determined. The proof of this lemma will be given in Section IV.

Definition 1. *Let* $\boldsymbol{\delta}_i = \{\delta_{i,1}, \ldots, \delta_{i,k}\}$ *be the set of deletion locations in the* i*-th head of an* d *head racetrack memory, i.e.* $\boldsymbol{\delta}_{i+1} = \boldsymbol{\delta}_i + t_i$*, for* $i \in [1, d-1]$*. An interval* $\mathcal{I}$ *is* deletion isolated *if*

$$\boldsymbol{\delta}_{i+1} \cap \mathcal{I} = t_i + \boldsymbol{\delta}_i \cap \mathcal{I},$$

for $i \in [1, d-1]$.

Example 1. *Consider a* 3 *head racetrack memory with head distances* $t_1 = 1$ *and* $t_2 = 2$*. Let the length of the sequence* $\mathbf{c}$ *be* $n = 22$ *and the deletion positions in the heads be given by*

$$\begin{aligned}\boldsymbol{\delta}_1 &= \{1, 2, 4, 8, 14, 17\},\\ \boldsymbol{\delta}_2 &= \{2, 3, 5, 9, 15, 18\}, \text{ and}\\ \boldsymbol{\delta}_3 &= \{4, 5, 7, 11, 17, 20\},\end{aligned}$$

then intervals $[1, 7]$*,* $[8, 12]$*, and* $[12, 22]$ *are deletion isolated.*

Intuitively, an interval $\mathcal{I}$ is deletion isolated when the subsequences $\mathbf{c}_{\mathcal{I}\cap\boldsymbol{\delta}_i^c}$ for $i \in [1, d]$ can be regarded as the d reads of the sequence $\mathbf{c}_{\mathcal{I}}$ in a d head racetrack memory after $|\boldsymbol{\delta}_1 \cap \mathcal{I}|$ deletions.

Lemma 5. *For any positive integers* n *and* $R \geq k+1$*, let* $\mathbf{c} \in \{0,1\}^{n+R}$ *be a sequence such that* $L(\mathbf{c}_{[1,n+k+1]}, \leq k) \leq 3k + \lceil \log n \rceil + 2 \triangleq T$*. Let the distance* t_i *between head* i *and head* $i+1$ *satisfy* $t_i \geq (4k+1)(T+2k+1) \triangleq T_{min}$ *and* $t_{max} = \max_{i\in\{1,\ldots,d-1\}} t_i$ *be the largest distance between two consecutive heads. Then given* $D \in D_k(\mathbf{c})$*, it is possible to find a set of* $J \leq k$ *disjoint and deletion isolated intervals* $\mathcal{I}_j \subseteq [1, n+R]$*,* $j \in [i, J]$ *such that* $\boldsymbol{\delta}_w \subset \cup_{j=1}^{J}\mathcal{I}_j$ *for* $w \in [1, d]$ *and* $|\mathcal{I}_j \cap [1, n+k+1]| \leq (2\lfloor (2t_{max}+T+1)/2 \rfloor + 1)kd + \lfloor (2t_{max}+T+1)/2 \rfloor + k \triangleq B$*, for* $j = 1, \ldots, J$*. Moreover,* $|\boldsymbol{\delta}_1 \cap \mathcal{I}_j|$ *can be determined for* $j \in \{1, \ldots, J\}$.

Let $\mathbf{c}$ be a sequence satisfying $L(\mathbf{c}_{[1,n+k+1]}, \leq k) \leq T$. Then from this lemma, the bit c_i can be identified for $i \in [1, n+k+1] \backslash (\cup_{j=1}^{J}\mathcal{I}_j)$, given $D \in D_k(\mathbf{c})$. This is because

$$c_i = D_{1, i-\sum_{j:\mathcal{I}_j\in[1,i-1]}|\boldsymbol{\delta}_1\cap\mathcal{I}_j|} \tag{1}$$

for $i \in [1, n+k+1] \backslash (\cup_{j=1}^{J}\mathcal{I}_j)$. Hence we are left to recover $\mathbf{c}_{\mathcal{I}_j}$ for $j \in [1, J]$. Split $\mathbf{c}_{[1,n+k+1]}$ into blocks

$$\mathbf{a}_i = c_{[(i-1)B+1, \max\{iB, n+k+1\}]},\ i \in \{1, \ldots, \lceil (n+k+1)/B \rceil\} \tag{2}$$

of length B except that $\mathbf{a}_{\lceil (n+k+1)/B \rceil}$ may have length shorter than B. By Lemma 4 the bits $\mathbf{c}_{\mathcal{I}_j}$ with $|\boldsymbol{\delta}_1 \cap \mathcal{I}_j| < d$ can be recovered, since $\mathcal{I}_j$ is deletion isolated and the head distance satisfies $t_i \geq T[k(k-1)/2+1]+(7k-k^3)/6$ for $i \in \{1, \ldots, d-1\}$. Note that there is at most a single interval $\mathcal{I}_{j_1}$ that satisfies $|\boldsymbol{\delta}_1 \cap \mathcal{I}_{j_1}| \geq d$ when $k \leq 2d-1$. Since $|\mathcal{I}_{j_1} \cap [1, n+k+1]| \leq B$, the interval $\mathcal{I}_{j_1}$ covers at most two blocks $\mathbf{a}_{j_1}$ and $\mathbf{a}_{j_1+1}$. It follows that there are at most two consecutive blocks, covered by $\mathcal{I}_{j_1}$, that remain to be recovered. Both block contain at most k deletions.

For any integer n and sequence $\mathbf{c} \in \{0,1\}^{n+k+1}$ of length $n+k+1$, let the function $S : \{0,1\}^{n+k+1} \to \mathbb{F}_{2^{\lceil (B/\lceil \log n \rceil)\rceil (2k\log\log n + O(1))}}^{\lceil (n+k+1)/B \rceil}$ be defined by

$$S(\mathbf{c}) = (Hash(\mathbf{a}_1), Hash(\mathbf{a}_2), \ldots, Hash(\mathbf{a}_{\lceil (n+k+1)/B \rceil})), \tag{3}$$

where $\mathbf{a}_i$, $i \in \{1, \ldots, \lceil (n+k+1)/B \rceil\}$ are the blocks of $\mathbf{c}$ defined in Eq. (2). The sequence $S(\mathbf{c})$ is a concatenation of the hashes $Hash$ of blocks of $\mathbf{c}$. Note that $Hash(\mathbf{a}_i)$ is the i-th symbol of $S(\mathbf{c})$.

Lemma 6. *If* $B > k$*, there exists a function* $DecS : \{0,1\}^{n+1} \times \{0,1\}^{\lceil (n+k+1)/B \rceil \lceil B/\log n \rceil (2k\log\log n + O(1))} \to \{0,1\}^{n+k+1}$*, such that for any sequence* $\mathbf{c} \in \{0,1\}^{n+k+1}$ *and its length* $n+1$ *subsequence* $\mathbf{d} \in \{0,1\}^{n+1}$*, we have that* $DecS(\mathbf{d}, S_k(\mathbf{c})) = \mathbf{c}$*, i.e., the sequence* $\mathbf{c}$ *can be recovered from* k *deletions with the help of* $S(\mathbf{c})$.

Now we are ready to present the code construction. For any sequence $\mathbf{c} \in \{0,1\}^n$, define the following encoding function:

$$Enc(\mathbf{c}) = (F(\mathbf{c}), R'(\mathbf{c}), R''(\mathbf{c})) \tag{4}$$

where

$$\begin{aligned}R'(\mathbf{c}) &= ER(S(F(\mathbf{c}))),\\ R''(\mathbf{c}) &= Rep_{k+1}(H(R'(\mathbf{c}))).\end{aligned} \tag{5}$$

The function ER is the parity of a code that corrects two consecutive erasures, which computes the modulo sum of symbols at the even and odd positions respectively. The function Rep_{k+1} is a $k+1$-fold repetition function that repeats each bit $k+1$ times. Note that we use $F(\mathbf{c}) \in \{0,1\}^{n+k+1}$ to obtain a sequence satisfying $L(F(\mathbf{c}), \leq k) \leq T$ so that Lemma 5 can be applied. The redundancy consists of two layers. The function $R'(\mathbf{c})$ can be regarded as the first layer redundancy, with the help of which $F(\mathbf{c})$ can be recovered from k deletions. It computes the redundancy of a code that corrects two consecutive symbol erasures in $S(F(\mathbf{c}))$. Notice that $S(F(\mathbf{c}))$ is a sequence of $\lceil n/B \rceil$ symbols. The function $R''(\mathbf{c})$ can be seen as the second layer redundancy that helps recover itself and $R'(\mathbf{c})$ from k deletions. The length of the codeword $Enc(\mathbf{c})$ is given by $N = n + 4k[3k^2(k-1)d/2 + 3kd + 3/2]\log\log n + o(\log\log n)$ when $t_i = \max\{(3k + \lceil \log n \rceil + 2)[k(k-1)/2+1] + (7k-k^3)/6, (4k+1)(5k + \lceil \log n \rceil + 3)\}$ for $i \in [1, d-1]$. The next theorem proves Theorem 1 for cases when $d \leq k \leq 2d-1$.

Theorem 2. *The set* $\mathcal{C} = \{Enc(\mathbf{c}) : \mathbf{c} \in \{0,1\}^n\}$ *is a* k *deletion* d *head correcting code for* $d \leq k \leq 2d-1$*, if the distance between any two consecutive heads satisfies* $t_i = \max\{(3k + \lceil \log n \rceil + 2)[k(k-1)/2+1] + (7k-k^3)/6, (4k+$

$1)(5k+\lceil\log n\rceil+3)\}$, $i\in\{1,\ldots,d-1\}$. *The code $\mathcal{C}$ can be constructed, encoded, and decoded in $O_k(poly(n))$ time. The redundancy of $\mathcal{C}$ is* $N-n=4k[3k^2(k-1)d/2+3kd+3/2]\log\log n+o(\log\log n)$.

IV. Proof of Lemma 5

Let $D\in D_k(\mathbf{c})$ be the d reads from all heads, where $\mathbf{c}$ satisfies $L(\mathbf{c}_{[1,n+k+1],\leq k})\leq T$. Then D is a d by $n+R-k$ matrix. The proof of Lemma 5 consists of two steps. The first step is to identify a set of disjoint intervals $\mathcal{I}'_j$, $j\in[1,J]$ that satisfy $|\mathcal{I}'_j\cap[1,n+1]|\leq B-k$ and $J\leq k$. Moreover, there exist a set of disjoint and deletion isolated intervals $\mathcal{I}_j$, $j\in[1,J]$, such that $\boldsymbol{\delta}_i\subseteq\cup_{j=1}^{J}\mathcal{I}_j$ and $D_{i,\mathcal{I}'_j}=\mathbf{c}_{\mathcal{I}_j\cap\boldsymbol{\delta}_i^c}$ for $i\in[1,d]$ and $j\in[1,J]$, i.e., the subsequence $D_{i,\mathcal{I}'_j}$ of the i-th read can be obtained by deleting $\mathbf{c}_{\mathcal{I}_j\cap\boldsymbol{\delta}_i}$ in $\mathbf{c}_{\mathcal{I}_j}$. Note that $\boldsymbol{\delta}_{i+1}\cap\mathcal{I}_j=t_i+\boldsymbol{\delta}_i\cap\mathcal{I}_j$ for $i\in[1,d-1]$ and $j\in[1,J]$, since $\mathcal{I}_j$ is deletion isolated. In addition, we have that $|\mathcal{I}_j\cap[1,n+k+1]|\leq B$, since $|\mathcal{I}'_j\cap[1,n+1]|\leq B-k$ and $|\mathcal{I}_j|\leq|\mathcal{I}'_j|+k$. The second step is to determine the number of deletions $|\boldsymbol{\delta}_i\cap\mathcal{I}_j|$ for $i\in[1,d]$ and $j\in[1,J]$, based on $D_{[1,d],\mathcal{I}'_j}$. Then,

$$\mathcal{I}_j=[i_{2j-1}+\sum_{\ell=1}^{j-1}|\boldsymbol{\delta}_1\cap\mathcal{I}_\ell|,i_{2j}+\sum_{\ell=1}^{j}|\boldsymbol{\delta}_1\cap\mathcal{I}_\ell|],$$

where $\mathcal{I}'_j=[i_{2j-1},i_{2j}]$ for $j\in[1,J]$. It is assumed that $i_j>i_l$ for $l<j$. The disjointness of $\mathcal{I}_j$, $j\in[1,J]$ follows from the fact that $\mathcal{I}'_j$, $j\in[1,J]$ are disjoint. The two steps will be made explicit in the following two subsections respectively.

A. Identifying Intervals $\mathcal{I}'_j$

The procedure for identifying intervals $\mathcal{I}'_j$ is as follows.

- **Initialization:** Set all $w\in[1,n+R-k]$ unmarked. Let $i=1$. Find the largest positive integer L such that the sequences $D_{j,[i,i+L-1]}$ are equal for all $j\in[1,d]$. If such L exists and satisfies $L>t_{max}$, mark the numbers $w\in[1,L-t_{max}]$. Go to Step 1.
- **Step 1:** Find the largest positive integer L such that the sequences $D_{j,[i,i+L-1]}$ are equal for $j\in[1,d]$. Go to Step 2. If no such L exists, set $L=0$ and go to Step 2.
- **Step 2:** If $L\geq 2t_{max}+T+1$, mark the numbers $w\in[i+t_{max},\min\{i+L-1,n+1\}-t_{max}]$. Set $i=i+L+1$ and go to Step 3. Else $i=i+1$ and go to Step 3.
- **Step 3:** If $i\leq n+1$, go to Step 1. Else go to Step 4.
- **Step 4:** If the number of unmarked intervals[1] within $[1,n+1]$ is not greater than k, output all unmarked intervals. Else output the first k intervals, i.e., the intervals with the minimum k starting indices.

We prove that the output intervals satisfy the above constraints. The following lemma will be used.

Lemma 7. *Let $D\in D_k(\mathbf{c})$ for some sequence $\mathbf{c}$ satisfying $L(\mathbf{c}_{[1,n+k+1]},\leq k)\leq T$. Let $T_{min}=\min_{i\in[1,d-1]}t_i$ and $t_{max}=\max_{i\in[1,d-1]}t_i$ such that $T_{min}\geq k(T+1)+1$. If $D_{w_1,[i_1,i_2]}=D_{w_2,[i_1,i_2]}$ for some interval $[i_1,i_2]\subseteq[1,n+1]$ with length $i_2-i_1+1\geq 2t_{max}+T+1$ and for all different $w_1,w_2\in[1,d]$, then there are no deletions happen within bits $D_{w,[i_1+t_{max},i_2-t_{max}]}$ for all w, i.e., there exists integers $i'_1=i_1+t_{max}+|\boldsymbol{\delta}_w\cap[1,i'_1-1]|$ and $i'_2=i_2-t_{max}+|\boldsymbol{\delta}_w\cap[1,i'_2-1]|$, such that $\mathbf{c}_{[i'_1,i'_2]}=D_{w,[i_1+t_{max},i_2-t_{max}]}$ and $[i'_1,i'_2]\cap\boldsymbol{\delta}_w=\emptyset$ for $w\in[1,d]$. Moreover, both intervals $[1,i'_1-1]$ and $[i'_2+1,n+R]$ are deletion isolated.*

Let $[p_{2j-1},p_{2j}]$, $j\in[1,J']$ be the marked intervals in the procedure, where $p_1<\ldots<p_{2J'}$. Let $p_0=0$ and $p_{2J'+1}=n+R+1$, then the output intervals are the first up to k nonempty intervals among $\{[p_{2j}+1,p_{2j+1}-1]\}_{j=0}^{J'}$. Note that the interval $[n+1-t_{max},n+R-k]$ is not marked in the procedure. Hence, according to Lemma 7, there exist intervals $[p'_{2j-1},p'_{2j}]$, $j\in[1,J']$, where

$$\begin{aligned}&p'_j=p_j+|\boldsymbol{\delta}_w\cap[1,p'_j-1]|,\text{ and}\\&[p'_{2\ell-1},p'_{2\ell}]\cap\boldsymbol{\delta}_w=\emptyset,\end{aligned}\tag{6}$$

for all $j\in[1,2J']$, $\ell\in[1,J']$, and $w\in[1,d]$. Moreover, intervals $[1,p'_{2j-1}-1]$ and $[p'_{2j}+1,n+R]$ are deletion isolated[2] for $j\in[1,J']$. Then we have that the intervals $[p'_{2j}+1,p'_{2j+1}-1]$, $j\in[0,J']$, where $p'_0=0,p'_{2J+1}=n+R+1$, are deletion isolated. From (6) we have that $D_{w,[p_{2j}+1,p_{2j+1}-1]}=\mathbf{c}_{[p'_{2j}+1,p'_{2j+1}-1]\cap\boldsymbol{\delta}_w^c}$. In addition, the intervals $\{[p'_{2j}+1,p'_{2j+1}-1]\}_{j=0}^{J'}$ are disjoint.

Furthermore, for any output interval $[p_{2j}+1,p_{2j+1}-1]\subseteq[1,n+1-t_{max}]$, the corresponding interval $[p'_{2j}+1,p'_{2j+1}-1]$ contains at least one deletion in $\boldsymbol{\delta}_w$, i.e., $[p'_{2j}+1,p'_{2j+1}-1]\cap\boldsymbol{\delta}_w\neq\emptyset$ for some $w\in[1,d]$. Otherwise, we have that $[p'_{2j}+1,p'_{2j+1}-1]\cap\boldsymbol{\delta}_w=\emptyset$ for $w\in[1,d]$, which implies that the interval $[p_{2j}+1,p_{2j+1}-1]$ is marked during the algorithm. Therefore, there are at most k unmarked intervals that lie within the interval $[1,n+1]$. Then it can be shown that the deletions $\boldsymbol{\delta}_w$ are contained in first up to k output intervals.

Finally, we show that $|\mathcal{I}'_j\cap[1,n+1]|\leq B-k$. It can be proved that for any unmarked index $i\in[1,n+1-\lfloor t_{max}+(T+1)/2\rfloor]$, there exists some $w\in[1,d]$ and $k_1\in[1,k]$, such that a deletion δ_{w,k_1} occurs within distance $\lfloor t_{max}+(T+1)/2\rfloor$ to the corresponding bit $\mathbf{c}_{i'=i+|\boldsymbol{\delta}_w\cap[1,i'-1]|}$ of $D_{w,i}$, i.e., $\delta_{w,k_1}\in[i'-\lfloor t_{max}+(T+1)/2\rfloor,i'+\lfloor t_{max}+(T+1)/2\rfloor]$[3]. Otherwise, we have that i is marked. On the other hand, any deletion δ_{w,k_1} covers at most $2\lfloor(2t_{max}+T+1)/2\rfloor+1$ unmarked positions in $[1,n+1-\lfloor t_{max}+(T+1)/2\rfloor]$. Then the number of unmarked bits in $[1,n+1]$ is at most $(2\lfloor(2t_{max}+T+1)/2\rfloor+1)kd+\lfloor(2t_{max}+T+1)/2\rfloor=B-k$.

B. Determining the Number of Deletions

We now present the algorithm for determining the number of deletions $|\boldsymbol{\delta}_i\cap\mathcal{I}_j|$, $i\in[1,d]$, for any deletion isolated interval $\mathcal{I}_j\subseteq[1,n+k+1]$. The inputs to this algorithm are the reads $D_{[1,d],\mathcal{I}'_j}$ obtained by deleting $\mathbf{c}_{\boldsymbol{\delta}_i\cap\mathcal{I}_j}$, $i\in[1,d]$ from $\mathbf{c}_{\mathcal{I}_j}$. Note that the interval $\mathcal{I}_j$ is not known at this point. In the algorithm only the first two reads $D_{[1,2],\mathcal{I}'_j}$ are used. Let $\mathcal{I}_j=$

[1] An unmarked interval $[i,j]$ means that $w\in[i,j]$ are not marked and $i-1$ and $j+1$ are marked. It is assumed that 0 and $n+R+1-k$ are marked.

[2] If the interval $[p_1,p_2]$ is marked in the initialization step and has length less than $2T+t_{max}+1$, apply Lemma 7 by imagining an interval $[-t_{max}+T+1,0]$ where $D_{w,[-t_{max}+T+1,0]}$ are equal for $w\in[1,d]$.

[3] When $i'-\lfloor t_{max}+(T+1)/2\rfloor<0$, consider imaginary bits $D_{w,[i'-\lfloor t_{max}+(T+1)/2\rfloor,0]}$ that are equal for $w\in[1,d]$.

$[b_{min}, b_{max}]$ for some integers b_{min} and b_{max}. Consider the following intervals,

$$\begin{aligned}\mathcal{B}_{i,m} =&[b_{min} + (i-1)t_1 + (m-1)(T+2k+1),\\ &\min\{b_{min} + (i-1)t_1 + m(T+2k+1) - 1, b_{max}\}],\end{aligned}$$

for $i \in [1, \lceil (b_{max} - b_{min} + 1)/t_1 \rceil]$ and $m \in [1, \min\{4k+1, \lceil ((b_{max} - b_{min} + 1) \bmod t_1)/(T+2k+1) \rceil\}]$. The intervals $\mathcal{B}_{i,m}$ are disjoint and have length $T+2k+1$ except when $i = \lceil (b_{max} - b_{min} + 1)/t_1 \rceil$ and $m = \min\{4k+1, \lceil ((b_{max} - b_{min} + 1) \bmod t_1)/(T+2k+1) \rceil\}$ the length might be less. Let $\mathcal{U}_m = \cup_i \mathcal{B}_{i,m}$ be the union of intervals $\mathcal{B}_{i,m}$ with the same m for $m \in [1, 4k+1]$. Then the unions $\mathcal{U}_m$, $m \in [1, 4k+1]$, are disjoint. Since the deletions occur in at most $2k$ positions in the first two heads, at least $2k+1$ unions $\{\mathcal{U}_{m_1}, \ldots, \mathcal{U}_{m_{2k+1}}\}$ satisfy $\mathcal{U}_{m_l} \cap (\boldsymbol{\delta}_1 \cup \boldsymbol{\delta}_2) = \emptyset$ for $l \in [1, 2k+1]$. Similarly, let $\mathcal{I}_j = [b'_{min}, b'_{max}]$ for some integers b'_{min} and b'_{max}. Define the intervals

$$\begin{aligned}\mathcal{B}'_{i,m} =& [b'_{min} + (i-1)t_1 + (m-1)(T+2k+1),\\ &\min\{b'_{min} + (i-1)t_1 + m(T+2k+1) - k - 1, b'_{max}\}],\end{aligned}$$

for $i \in [1, \lceil (b'_{max} - b'_{min} + 1)/t_1 \rceil]$ and $m \in [1, \min\{4k+1, \lceil ((b'_{max} - b'_{min} + 1) \bmod t_1)/(T+2k+1) \rceil\}]$. Since $D_{e,\mathcal{I}'_j}$ can be obtained by deleting bits $\mathbf{c}_{\boldsymbol{\delta}_e \cap \mathcal{I}_j}$ from $\mathbf{c}_{\mathcal{I}_j}$, it can be shown that the sequence $D_{e,\mathcal{B}'_{i,j}}$ is a length $T+k+1$ subsequence of $\mathbf{c}_{\mathcal{B}_{i,m}}$ for $e \in \{1,2\}$ and for all i, j except for the last at most two pairs (i,m), the interval $\mathcal{B}'_{i,m}$ may not exist. Let $\mathcal{IM}' = \{(i,m) : \mathcal{B}'_{i,m} \neq \emptyset\}$ be the set of (i,m) pairs for which $\mathcal{B}'_{i,m}$ is defined. For $(i,m) \in \mathcal{IM}'$, let $p'_{i,m}$ and $q'_{i,m}$ be the beginning and end points of interval $\mathcal{B}'_{i,m}$, i.e., $\mathcal{B}'_{i,m} = [p'_{i,m}, q'_{i,m}]$. Similarly, let $[p_{i,m}, q_{i,m}] = \mathcal{B}_{i,m}$. The algorithm is given as follows.

- **Step 1:** For all $(i,m) \in \mathcal{IM}'$, find a unique integer $0 \leq x_{i,m} \leq k$ such that $D_{1,[p'_{i,m}, q'_{i,m} - x_{i,m}]} = D_{2,[p'_{i,m} + x_{i,m}, q'_{i,m}]}$. If no or more than one such integers exist, let $x_{i,m} = 0$. Go to Step 2.
- **Step 2:** For all $m \in [1, 4k+1]$, compute the sum $s_m = \sum_{(i,m) \in \mathcal{IM}'} x_{i,m}$. Go to step 3.
- **Step 3:** Output the majority among $\{s_m\}_{m=1}^{4k+1}$.

To prove the correctness of the algorithm, it suffices to show that $s_{m_l} = |\mathcal{I}_j \cap \boldsymbol{\delta}_1|$ for all $l \in [1, 2k+1]$. First, we show that the unique integer x_{i,m_l} satisfying $D_{1,[p'_{i,m_l}, q'_{i,m_l} - x_{i,m_l}]} = D_{2,[p'_{i,m_l} + x_{i,m_l}, q'_{i,m_l}]}$ exists. Moreover, the integer x_{i,m_l} equals $|\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]| - |\boldsymbol{\delta}_2 \cap [p_{1,1}, p_{i,m_l} - 1]|$, the difference between deletion numbers in the first two heads before the interval $\mathcal{B}_{i,m_l}$. Let $x = |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]| - |\boldsymbol{\delta}_2 \cap [p_{1,1}, p_{i,m_l} - 1]|$, it can be verified that the integer $x_{i,m_l} = x$ satisfies $D_{1,[p'_{i,m_l}, q'_{i,m_l} - x_{i,m_l}]} = D_{2,[p'_{i,m_l} + x_{i,m}, q'_{i,m_l}]}$. We show such x_{i,m_l} is unique. Suppose there exists another integer $y > x$ for which $D_{1,[p'_{i,m_l}, q'_{i,m_l} - y]} = D_{2,[p'_{i,m_l} + y, q'_{i,m_l}]}$. Then,

$$\begin{aligned}&\mathbf{c}_{[p'_{i,m_l} + |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]|, q'_{i,m_l} + |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]| - y]}\\ =&\mathbf{c}_{[p'_{i,m_l} + |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]| + y - x, q'_{i,m_l} + |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]| - x]}.\end{aligned}$$

It then follows that

$$\begin{aligned}&L(\mathbf{c}_{[p'_{i,m_l} + |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]|, q'_{i,m_l} + |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]| - x]}, y - x)\\ =&q'_{i,m_l} - x - p'_{i,m_l} + 1 \geq T + k + 1 - k + 1 > T + 1\end{aligned}$$

which is a contradiction to the fact that $L(\mathbf{c}, \leq k) \leq T$. Similarly, such contradiction occurs when $y < x$. Hence such x_{i,m_l} is unique.

Next, we show that $s_{m_l} = |\boldsymbol{\delta}_1 \cap \mathcal{I}_j|$ for $l \in [1, 2k+1]$. Since $p_{i,m_l} - p_{i-1,m_l} = t_1$ for $i \in [2, \max_{(i,m) \in \mathcal{IM}'} i]$, we have that

$$\begin{aligned}&|\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{i,m_l} - 1]|\\ =&|\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{1,m_l} - 1]| + \sum_{w=1}^{i-1} |\boldsymbol{\delta}_1 \cap [p_{w,m_l}, p_{w+1,m_l} - 1]|\\ =&|\boldsymbol{\delta}_2 \cap [p_{2,1}, p_{2,m_l} - 1]| + \sum_{w=1}^{i-2} |\boldsymbol{\delta}_2 \cap [p_{w+1,m_l}, p_{w+2,m_l} - 1]|\\ &+ |\boldsymbol{\delta}_1 \cap [p_{i-1,m_l}, p_{i,m_l} - 1]|\\ =&|\boldsymbol{\delta}_2 \cap [p_{1,1}, p_{i,m_l} - 1]| + |\boldsymbol{\delta}_1 \cap [p_{i-1,m_l}, p_{i,m_l} - 1]|,\end{aligned}$$

It then follows that $x_{i,m_l} = |\boldsymbol{\delta}_1 \cap [p_{i-1,m_l}, p_{i,m_l} - 1]|$ ($p_{0,m_l} = p_{1,1}$) and that

$$s_{m_l} = |\boldsymbol{\delta}_1 \cap [p_{1,1}, p_{\max_{(i,m) \in \mathcal{IM}'} i, m_l} - 1]|$$

Note that $\boldsymbol{\delta}_1 \cap [p_{\max_{(i,m) \in \mathcal{IM}'} i, m_l}, b_{max}] \subseteq \boldsymbol{\delta}_1 \cap [b_{max} - t_1 + 1, b_{max}] = \emptyset$. Hence, we have that $s_{m_l} = |\boldsymbol{\delta}_1 \cap \mathcal{I}_j|$. Then the majority rule works.

REFERENCES

[1] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (SODA), pp. 1884–1892. 2016

[2] Y. Chee, H. Kiah, A.Vardy, V. Vu and E. Yaakobi, "Codes Correcting limited-shift errors in racetrack memories," *Proc. IEEE Int. Symp. on Inform. Theory*, 2018, pp. 96–100.

[3] Y. Chee, H. Kiah, A.Vardy, V. Vu and E. Yaakobi "Coding for racetrack memories," *IEEE Transactions on Information Theory*, 2018.

[4] Y. Chee, R. Gabrys, A. Vardy, and V. K. Vu, and E. Yaakobi, "Reconstruction from deletions in racetrack memories," *Proc. IEEE Inform. Theory Workshop*, 2018.

[5] M. Hayashi, L. Thomas, R. Moriya, C. Rettner and S. S. Parkin, "Current-controlled magnetic domain-wall nanowire shift register," in *Science*, vol. 320, no. 5873, 2008, pp. 209–211.

[6] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[7] V. I. Levenshtein, "Reconstructing objects from a minimal number of distorted patterns," (in Russian), Dokl. Acad. Nauk 354 pp. 593-596; English translation, Doklady Mathematics, vol. 55 pp. 417420, 1997.

[8] S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," in *Science*, vol. 320, no. 5873, 2008, pp. 190–194.

[9] Z. Sun, W. Wu and H. Li, "Cross-layer racetrack memory design for ultra high density and low power consumption," in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, 2013, pp. 1–6.

[10] A. Vahid, G. Mappouras, D. J. Sorin and R. Calderbank, "Correcting Two Deletions and Insertions in Racetrack Memory." *arXiv preprint arXiv:1701.06478*, 2017.

[11] C. Zhang, G. Sun, X. Zhang, W. Zhang, W. Zhao, T. Wang, Y. Liang, Y. Liu, Y. Wang and J. Shu, "Hi-fi playback: Tolerating position errors in shift operations of racetrack memory," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3, 2015, pp. 694–706.