1 Introduction

We consider the following document exchange problem: given an $n$-bit file $F$ and $k$, which is an upper bound on the edit distance$^1$, we want to compute a summary $S_{F,k}$ such that there is an efficient procedure to recover $F$ from any corrupted file $F'$ with $ED(F, F') \leq k$. We focus on single-round schemes and assume a good bound $k$ is known.

**Trivial lower and upper bounds.** For a fixed $F'$, interactively perform $k$ binary searches thus $O(k \log n)$; on the other hand, to distinguish $\binom{n}{k}$ possible corruptions, we need at least $\Omega(\log \binom{n}{k}) = \Omega(k \log \frac{n}{k})$ bits.

[Orl91] initiated the theoretical study of this problem where he showed the existence of $O(n \log \frac{n}{k})$ summary but left its constructive counterpart as an open problem. Building on a series of works, the optimal (randomized) constructive summary is finally shown in [Hae18] (theorem 2.1). The derandomized algorithm produces $O(k \log^2 \frac{n}{k})$-bit summary. Moreover it is shown that there is a deterministic recovery algorithm which takes $S_{F,k}$ and any $F'$ and recovers $F$ with high probability.

In this lecture, we describe a simplified version of the randomized algorithm which produces $O(k \log^2 \frac{n}{k})$ summary. The more involved result can be found in [Hae18].

2 Overview of Techniques

A natural idea is divide-and-conquer: break $F$ into several blocks, say $4k$ blocks and send hashed blocks for comparison. If we do so, all but $k$ parts are only shifted due to deletion. We can recurse on those $k$ blocks to identify smaller blocks which do not contain an edit. Furthermore, we could run an adaptive multi-level check to see which blocks are already found to be identical and no longer needed for recursion.

Let $o$ be the hash size. This approach requires $\log \frac{n}{k}$ recursion levels where in each level $k$ parts are sent. To choose hash size $o$, we note that the probability of collision is $2^{-o}$ since each bit of the inner product is uniformly distributed. To sum up the naive approach gives $O(k \log n \log \frac{n}{k})$ communication cost. To improve this result, we need to:

1. Avoid interaction: use the non-systematic part of systematic error correcting code (ECC) as in [IMS05], which allows us to send coded hashes of all levels.

2. Reduce hash size: we show that on each level, we can in fact tolerate $O(k)$ hashing collisions via the above ECC trick and a ‘helper’ from the above level.

---

$^1$The edit distance $ED(F, F')$ is the minimum number of insertions, deletions or symbol changes than transform one string into the other.
3 Summary Structure

Notation. Let $S \in \Sigma^*$ be a string. For $i, j \in [1, |S|]$, let $S[i, j]$ be the substring of $S$ including both indices $i$ and $j$. We use multi-dimensional array notation, e.g., $H[i,]$ is the $i$-th row of $H$. A $q$-ary linear code of block length $n$ and dimension $k$ with minimum distance $d$ is referred to as an $[n, k, d]_q$ code.

3.1 Inner Product Hash Function

Recall the hashing algorithm takes a block of file $F$ which starts at position $s$ in a level $l$ and outputs a string of length $o$. We think of the randomness as supplied by a 3D table $R[s, l, i]$ where $s$ is the starting position, $l$ is the level index and $i$ is the bit index of the inner product. For a string $S$, our hashing function is defined as

$$\text{hash}^o_R(S, s, l) = \sum_{j=1}^{|S|} S[j] \cdot R[s + j - 1, l, i],$$

when $|S| > o$. Otherwise we simply output $S$ padded with zeros.

3.2 Summary Construction via $\text{hash}^o_R(S, s, l)$ and ECC

Let $L = \lceil \log \frac{n}{4k} \rceil$ be the level of recursions. The construction is simple: we hash all level zero blocks and store them in the summary; for level 1, ..., we store the encoded hash values to save space. In the next section, we show that such an encoding allows us to approximately recover the summary and subsequently recover $F$. Below we formalize this construction:

- For level zero, we hash each block and add them into the summary, taking up $O(ko)$ space
- For level $l = 1, \ldots, L$, we break each block from the previous level into two equally-sized block, i.e. we get $4k \cdot 2^l$ blocks. We hash all the blocks and encode those hashes via $C_j$, which is the non-systematic part of a systematic linear $[4k \cdot 2^l + 100k, 4k \cdot 2^l, 13k]$ code. Constants before $k$ is not important. Such a code is guaranteed when $o = O(\log \frac{n}{k})$.

4 Recovery Algorithm

The idea for recovery is induction: suppose on level $l$ we can correctly recover all hashes, then we can compute an approximately correct guess of all hashes on level $l + 1$ by comparing hashes of the previous level and the corrupted file $F'$. If this guess is not too far away, we can recover level $l + 1$ due to the property of ECC. To formalize the algorithm, we define the notion of a matching. We say two index sequences $i_1, \ldots, i_t$ and $i'_1, \ldots, i'_t$ is a level-$l$ size-$t$ matching between $F$ and $F' \in \{0, 1\}^n$ if

- $i_1 \leq \ldots \leq i_t$
- all $i$-indices are starting points of blocks of $F$ on level $l$
- hashes corresponding to $i$ and $i'$ blocks are identical
Note the most general version of [Hae18] relaxes the third requirement above. Since we focus on describing the easier version, below we further assume $i'$-indices are also monotone and disjoint, i.e., the intervals matched in $F'$ do not overlap. We call a matching $t$-bad if every matched block has non-identical string values. A $t$-bad self-matching refers to a size-$t$ bad matching between $F$ and itself. Let $H[l,..]$ be the hash values of $F$ on level $l$.

Algorithm 1 Recovery with $o = \Theta(\log^\frac{n}{k})$

1: get $H[0,..] \leftarrow$ from the summary
2: for $l = 0, \ldots, L - 1$ do
3: $M_l \leftarrow$ largest level-$l$ matching from $H[l,..]$ into $F'$  
\hspace{1cm} $\triangleright$ ‘Helper’ for recovering $l + 1$ hashes
4: $\hat{H}_F[l + 1,..] \leftarrow$ guesses for level-$l + 1$ hashes using $M_l$ and $F'$
5: $\hat{H}_F[l + 1,..] \leftarrow$ decode from $\hat{H}_F[l,..]$ using ECC
\hspace{1cm} $\triangleright$ See the remark below
6: $H[l + 1,..] \leftarrow$ decode from $\hat{H}_F[l,..]$ using ECC
7: $F \leftarrow H[L,..]$  
\hspace{1cm} $\triangleright$ Block size on level-$L$ is constant therefore hash is identity

Remark. Note step 3 of the algorithm can be implemented in poly($n, k$) using standard DP technique. We compute $\hat{H}[l + 1,..]$ as follows: each level-$l$ block $B$ is split into $B_1$ and $B_2$ on level $l + 1$. If block $B$ is matched to $B'$ in $F'$ based on $M_l$, we guess the hash values of $B_1$ and $B_2$ equal to those of $B'_1$ and $B'_2$. Otherwise we fill in something arbitrarily.

Question. When does algorithm 1 fail?

Suppose our guess becomes irrecoverable on some level $l$. Since ECC has minimum distance $\Omega(k)$, this means the level-$l$ matching contains $\Omega(k)$ pairs of substrings which have identical hash values but non-identical string values. We can therefore focus on those pairs that do not contain an edit. The key observation is that those pairs in fact form a $k$-bad self-matching. To complete the proof, we need to show such a witness is unlikely to exist when using hash size $o = \Theta(\log \frac{n}{k})$. Below we formalize this intuition.

Lemma 1. Assume level-$l$ hashes are correctly recovered, then $|M_l| \geq 4k \cdot 2^l - ED(F, F')$

Proof. It suffices to show there exists a matching of the above size. This is true because we can match blocks that do not contain an edit.

Lemma 2. Assume level-$l$ hashes are correctly recovered and there is no $k$-bad self-matching on level $l$, then $M_l$ contains at most $ED(F, F') + k$ non-identical pairs of blocks.

Proof. There are two error sources in $M_l$: 1. match blocks that contain at least one edit; 2. matching blocks that do not contain any edit which correspond to bad self-matching. # first error $\leq ED(F, F')$ by definition of edit distance and # second error $\leq k$ by the bound on the cardinality of bad self-matching.

Now we can show algorithm 1 correctly recovers hashes of all levels if no $k$-bad self-matching exists on any level.

Lemma 3. Assume that no $k$-bad self-matching exists for all levels and that $ED(F, F') \leq k$, then algorithm 1 correctly recovers $F$.

Proof. We prove the algorithm correctly recovers hash values of all levels by induction on the level. This is enough to prove the lemma. Base case $l = 0$ is trivial since hash values are not encoded.
Now assume the claim holds for level \( l \), it suffices to show the hamming distance between \( H[l+1,\ldots] \) and \( H'[l+1,\ldots] \) is \( O(k) \). There are two sources of discrepancies: 1. unmatched blocks on level \( l \); 2. matched but non-identical blocks on level \( l \). Note the first \( \leq k \) (lemma 1) and second \( \leq 2k \) (lemma 2). Since each block is further split into two, we see the hamming distance is bounded by \( 6k \).

Finally we show \( o = \Theta(\log \frac{n}{k}) \) suffices to make \( k \)-bad self-matching unlikely to exist.

**Lemma 4.** Assume \( o = C \log \frac{n}{k} \) for some large enough \( C \) and \( R \) is sampled from i.i.d. uniform bits, then for every level \( l \) the probability that there exists a \( k \)-bad self-matching is at most \( 2^{-\Omega(ok)} \).

**Proof.** Since the hashing collision probability is \( 2^{-o} \), the probability of a matching being \( k \)-bad is \( 2^{-ko} \). There are at most \( \binom{n}{k}^2 = O(2^{k \log \frac{n}{k}}) \) choices for a size-\( k \) matching. Using the union bound we want \( O(2^{-k \log \frac{n}{k}} \cdot 2^{-ko}) \leq \frac{1}{n^o} \). Setting \( o = C \log \frac{n}{k} \) for large enough \( C \) works.

**Remark.** Note lemma 4 still holds for \( 2^{-co(k)} \)-biased distribution: the \( k \)-badness of a matching depends on \( ko \) linear tests and the probability of each test can deviate from the uniform distribution scenario by at most \( 2^{-co(k)} \) from the definition of \( \epsilon \)-biased distribution. Therefore the probability of a matching being \( k \)-bad is at most \( \left( \frac{1}{2} + 2^{-co(k)} \right)^{ko} = 2^{-\Omega(ko)} \) for large \( c \). This idea is used for derandomization in the next section.

## 5 Derandomization

It is known (e.g. [NN93]) that one can construct an \( \epsilon \)-biased probability space where a sample point over \( n^{O(1)} \) bits can be described by \( O(\log n + \log \frac{1}{\epsilon}) \) bits. On the other hand, recall from the above remark that the tolerance \( \epsilon \) is proportional to the probability of \( k \)-bad self-matching. As a result, if we can relax the probability of bad self-matching to be polynomially small in \( n \), then the probability space has a polynomial-sized support and can thus be efficiently explored. Below we show that this is indeed possible if we refine our union bound in lemma 4 by focusing on polynomially many bad sub-matchings.

**Lemma 5.** If \( F \) has a \( k \)-bad self-matching on some level, then for any \( 1 \leq k' \leq \frac{k}{2} \), it also has a \( k' \)-bad matching on the same level with \( i_{k'} - i_1 + i'_{k'} - i'_1 \leq \frac{4k'}{k}n \).

Its proof is based on decomposing the \( k \)-bad self-matching into \( \left\lfloor \frac{k}{k'} \right\rfloor \) many \( k' \)-bad sub-matchings, followed by an average argument. See [Hae18] for more detail.

Based on the above observation, we can now refine the union bound to show that,

**Lemma 6.** Assume \( o = c \log \frac{n}{k} \) for some large \( c \) and \( R \) is sampled from \( n^{-\Theta(c)} \)-biased distribution, then for every level the probability that there exists a \( k \)-bad self-matching under \( hash_R \) is at most \( n^{O(c)} \).

**Proof.** We bound the number of \( k' \)-bad sub-matchings: there are \( n^2 \) choices for \((i_1, i'_1)\) and there are \( \binom{\frac{k'c}{k}}{2k' - 2} = O((\frac{n}{k})^{k'}) \) choices for the remaining indices (lemma 5). Set \( k' = \frac{\log n}{\log \frac{n}{k}} \). The number of \( k' \)-bad sub-matching is at most \( n^2 \cdot O(2^{k' \log \frac{n}{k}}) = n^{O(1)} \). On the other hand, the probability of a sub-matching being \( k' \) bad is \( 2^{ok'} = n^{-c} \). Therefore the union bound works for the \( n^{-\Theta(c)} \)-biased distribution.

13-4
We can therefore construct a poly-sized support for this $n^{-\Theta(c)}$-biased distribution using [NN93]. It only remains to observe that, as remarked below algorithm 1, given $R$ and $k$ there is a simple DP to check the existence of a $k$-bad matching.

References


