The Gapped k-Deck Problem

Rebecca Golm‡
ECE Department
University of Illinois
rgolm2@illinois.edu

Mina Nahvi‡
Department of Mathematics
University of Illinois
mnahvi2@illinois.edu

Ryan Gabrys
Calit2
University of California, San Diego
ryan.gabrys@gmail.com

Olgica Milenkovic
ECE Department
University of Illinois
milenkov@illinois.edu

***Abstract*—The k-deck problem is concerned with finding the smallest positive integer $S(k)$ such that there exist at least two strings of length $S(k)$ that share the same k-deck, i.e., the multiset of subsequences of length k. We introduce the new problem of gapped k-deck reconstruction: For a given gap parameter s, we seek the smallest positive integer $G_s(k)$ such that there exist at least two distinct strings of length $G_s(k)$ that cannot be distinguished based on a "gapped" set of k-subsequences. The gap constraint requires the elements in the subsequences to be at least s positions apart within the original string. Our results are as follows. First, we show how to construct sequences sharing the same 2-gapped k-deck using a nontrivial modification of the recursive Morse-Thue string construction procedure. This establishes the first known constructive upper bound on $G_2(k)$. Second, we further improve this bound using the approach by Dudik and Schulman [6].**

***Index Terms*—Gapped subsequences, k-deck, Morse-Thue sequences, String reconstruction**

I. Introduction

The problem of reconstructing strings based on evidence sets of the form of subsequences, substrings or weights of substrings has received significant attention from the theoretical computer science, bioinformatics, and information theory communities alike [1], [3], [4], [8], [10], [11], [13], [15]. One special instance of this class of problems is the k-deck problem [4], [6], [7], [9], [10], [14], of interest due to its connection to trace reconstruction [3], [5] and its applications in DNA-based data storage [16].

For a string $\boldsymbol{x}$ of length n, the multiset of the $\binom{n}{k}$ subsequences (i.e., ordered collections of not necessarily adjacent entries) of $\boldsymbol{x}$ of length k is called the *k-deck* of $\boldsymbol{x}$. We say that $\boldsymbol{x}$ is *k-reconstructible* if it is uniquely determined by its k-deck, meaning that there exists no other string that has the same k-deck as $\boldsymbol{x}$. For example, $(1,0,0,1)$ and $(0,1,1,0)$ have the same 2-deck, and are hence not 2-reconstructible. A simple counting argument shows that if two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same k-deck, they also have the same l-deck for all $1 \leqslant l \leqslant k$.

Let $S(k)$ be the smallest positive integer n such that there exist two distinct strings of length n with the same k-deck. Kalashnik [10] raised the question of determining $S(k)$. Manvel, Meyerowitz, Schwenk, Smith and Stockmeyer [12] showed that $2k \leqslant S(k) \leqslant 2^k$. They proved the upper bound as follows. For two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ of length n, let $\boldsymbol{xy}$ be the string obtained by concatenating $\boldsymbol{x}$ and $\boldsymbol{y}$ (note that when concatenating a single bit, say 0, and a string $\boldsymbol{x}$, we also use the notation $(0,\boldsymbol{x})$). If $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same k-deck, then $\boldsymbol{xy}$ and $\boldsymbol{yx}$ have the same $(k+1)$-deck. The upper bound follows immediately when coupled with the fact that $(0,1)$ and $(1,0)$ have the same 1-deck. The construction is often referred to as the Morse-Thue construction and the resulting strings are the well-known Morse-Thue strings [2]. Furthermore, the authors of [12] also showed that in order to prove that every string of length n is k-reconstructible, it is enough to prove that every binary string of length n is k-reconstructible. Dudik and Schulman [6] improved the above upper bound on $S(k)$ to $\exp\left(\frac{3+o(1)}{2\log 3}\log^2 k\right)$. In the literature, both bounds on the smallest k and n (for a given n and k, respectively) for unique and nonunique k-deck reconstruction have been reported.

We define the *gapped k-deck* of a binary string $\boldsymbol{x}$ as the multiset of all subsequences of length $\leqslant k$ that do not include two consecutive entries in $\boldsymbol{x}$. This definition can be extended to larger gaps between entries in $\boldsymbol{x}$: The *s-gapped k-deck* of a binary string $\boldsymbol{x}$ is the multiset of all subsequences $(x_{i_1},\ldots,x_{i_\ell}), 1 \leqslant \ell \leqslant k$, such that for all $1 \leqslant j \leqslant \ell-1$, we have $i_{j+1} \geqslant i_j + s$. With this definition, the gapped k-deck reduces to the 2-gapped k-deck. The problem of interest is to bound $G_s(k)$, the smallest positive integer n for which there exist two binary strings that share the same s-gapped k-deck. For simplicity, when $s=2$, we write $G(s)$ and refer to the corresponding setting as the *gapped k-deck*. Note that unlike the case without gaps, two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ having the same multiset of gapped subsequences of length k does not imply that they also have the same multiset of gapped sequences of length l for some $l < k$. For example, the strings $(0,1,1,1,0)$ and $(1,0,0,0,1)$ have the same multiset of gapped subsequences of length 2, but they clearly have different multisets of gapped subsequences of length 1 (which by definition, is the multiset of bits (composition) of the strings). The gapped k-deck problem is of interest in molecular storage systems for which readouts are based on nanopore technologies, in which "gaps" in readouts arise due to skipping effects [16].

We initiate the study of reconstruction limits of strings given their s-gapped k-decks and present the first upper bounds on $G_s(k)$ and $G(k)$ in particular. In Section II we provide necessary preliminaries, while in Section III we describe a nontrivial extension of a Morse-Thue type construction for

The work was funded by NSF grant 2008125, Coded String Reconstruction Problems in Molecular Storage. In the author list, ‡ denotes equal contribution.

2-gapped k-decks. In Section III, we state the result for general values of s but omit the proof. Section IV presents an improvement of the upper bound for $G(k)$ from Section III, based on an adaptation of the method described in [6].

II. Preliminaries

For a string $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, let

$$\mathcal{B}^{(k)}(\boldsymbol{x}) = \{(x_{i_1}, x_{i_2}, \ldots, x_{i_\ell}) : i_j \geqslant i_{j-1} + 2, 0 \leqslant \ell \leqslant k\} \quad (1)$$

denote the multiset of all subsequences of $\boldsymbol{x}$ of length $\leqslant k$ such that the index of every entry used in a subsequence is nonadjacent in the original string. Also, let

$$\mathcal{D}^{(k)}(\boldsymbol{x}) = \{(x_{i_1}, x_{i_2}, \ldots, x_{i_k}) : i_j \geqslant i_{j-1} + 2\},$$

be the *exact* gapped k-deck of $\boldsymbol{x}$. Here, we assume that $\mathcal{B}^{(0)}(\boldsymbol{x}) = \mathcal{D}^{(0)}(\boldsymbol{x}) = \varnothing$. Clearly, $\mathcal{B}^{(k)}(\boldsymbol{x}) = \bigcup_{i=0}^{k} \mathcal{D}^{(i)}(\boldsymbol{x})$. As mentioned in the introduction, unlike the classical (ungapped) case, the problem of reconstructing $\boldsymbol{x}$ from $\mathcal{B}^{(k)}(\boldsymbol{x})$ differs from that of reconstructing $\boldsymbol{x}$ from $\mathcal{D}^{(k)}(\boldsymbol{x})$. Our focus is on finding $G(k)$, the smallest integer n such that there exist two distinct binary strings of length n with the same gapped i-deck for all $1 \leqslant i \leqslant k$. Alternatively, $G(k)$ is the smallest integer n such that there exist two distinct binary strings of length n, $\boldsymbol{x}$ and $\boldsymbol{y}$, satisfying $\mathcal{B}^{(k)}(\boldsymbol{x}) = \mathcal{B}^{(k)}(\boldsymbol{y})$. It is worth pointing out that if n is the smallest integer such that there exist two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ of length n with $\mathcal{D}^{(k)}(\boldsymbol{x}) = \mathcal{D}^{(k)}(\boldsymbol{y})$, then $n = 2k-1$. We have $n \geqslant 2k-1$ because a string of length less than $2k-1$ has no gapped subsequence of length k. On the other hand, for any string $\boldsymbol{z} = (z_1 \ldots z_k)$ of length k, all the strings of length $2k-1$ of the form $(z_1 x_1 z_2 x_2 \ldots z_{k-1} x_{k-1} z_k)$ have the same gapped k-deck because the only gapped k-subsequence of $\boldsymbol{x}$ is $\boldsymbol{z}$. This observation generalizes for s-gapped k-decks and $n = sk-1$.

In Section III, we prove that $G(k) \leqslant 4(2^k - 1) - 2$. We also provide an upper bound on $G_s(k)$, the smallest integer n such that there exist two distinct strings of length n with the same s-gapped i-deck for all $1 \leqslant i \leqslant k$, where $s \geqslant 2$. The bound reads as $G_s(k) \leqslant (5s-2)2^{k-1} - 5s + 4$, but the accompanying proof is omitted due to space limitations. The proof of our first bound on $G(k)$ builds upon the next lemma.

Lemma 1. *[12] If $\boldsymbol{x} = (x_1, x_2, \cdots, x_m)$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)$ have the same k-deck, then the two concatenation strings $\boldsymbol{xy} = (x_1, \cdots, x_m, y_1, \cdots, y_m)$ and $\boldsymbol{yx} = (y_1, \cdots, y_m, x_1, \cdots, x_m)$ have the same $(k+1)$-deck.*

Proof: The following correspondence proves the claim: Pick any subsequence $\boldsymbol{z}$ of $\boldsymbol{xy}$ of length at most $k+1$. If $\boldsymbol{z}$ is fully contained within the $\boldsymbol{x}$ (or $\boldsymbol{y}$) substring, let $\phi(\boldsymbol{z})$ be the same subsequence in the $\boldsymbol{x}$ (or $\boldsymbol{y}$) substring of $\boldsymbol{yx}$. Now, assume $\boldsymbol{z} = \boldsymbol{z}_1\boldsymbol{z}_2$, where $\boldsymbol{z}_1$ is a subsequence of $\boldsymbol{x}$ and $\boldsymbol{z}_2$ is a subsequence of $\boldsymbol{y}$. Note that $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ have length at most k, therefore there exists a subsequence $\boldsymbol{w}_1$ of $\boldsymbol{y}$ that equals $\boldsymbol{z}_1$, due to the fact that $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same i-deck for all $1 \leqslant i \leqslant k$. Similarly, $\boldsymbol{x}$ contains a subsequence $\boldsymbol{w}_2$ that equals $\boldsymbol{z}_2$. Now, let $\phi(\boldsymbol{z}) = \boldsymbol{w}_1\boldsymbol{w}_2$. Therefore, $\boldsymbol{xy}$ and $\boldsymbol{yx}$ have the same $(k+1)$-deck. ∎

Using the strings $\boldsymbol{x} = (0,1)$, $\boldsymbol{y} = (1,0)$ and $k = 1$ to initialize the recursion, we can see that $(0,1,1,0)$ and $(1,0,0,1)$ have the same 2-deck. Repeating the process, we find that $(0,1,1,0,1,0,0,1)$ and $(1,0,0,1,0,1,1,0)$ have the same 3-deck and so on. However, this construction does not work for the gapped case. For example, $(1,0)$ and $(0,1)$ have the same gapped 1-deck (i.e., composition), but $(1,0,0,1)$ and $(0,1,1,0)$ do not have the same gapped 2-deck. The reason why the construction fails is that we cannot pick both x_m and y_1 (as defined in Lemma 1) when choosing a gapped subsequence of $\boldsymbol{xy}$. Hence, we need to "pad" the boundary between the two concatenated strings in an adequate manner.

III. The Padded Morse-Thue Sequence Approach

We prove the existence of two strings $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$, where $n = 4(2^k - 1) - 2$, that satisfy $\mathcal{B}^{(k)}(\boldsymbol{x}) = \mathcal{B}^{(k)}(\boldsymbol{y})$, using induction. We start with a few definitions. For a binary string $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ let

$$\mathcal{B}_L^{(k)}(\boldsymbol{x}) := \mathcal{B}^{(k)}(x_2, x_3, \ldots, x_n), \quad (2)$$

$$\mathcal{B}_R^{(k)}(\boldsymbol{x}) := \mathcal{B}^{(k)}(x_1, x_2, \ldots, x_{n-1}), \quad (3)$$

$$\mathcal{B}_{LR}^{(k)}(\boldsymbol{x}) := \mathcal{B}^{(k)}(x_2, \ldots, x_{n-1}). \quad (4)$$

Note that (2) represents the multiset of all gapped subsequences formed by puncturing $\boldsymbol{x}$ on the left, (3) represents the multiset of all gapped subsequences formed by puncturing $\boldsymbol{x}$ on the right, while (4) represents the multiset of all gapped subsequences formed by puncturing $\boldsymbol{x}$ on both ends. We define the sets $\mathcal{D}_L^{(k)}(\boldsymbol{x})$, $\mathcal{D}_R^{(k)}(\boldsymbol{x})$, and $\mathcal{D}_{LR}^{(k)}(\boldsymbol{x})$ analogously.

We initialize two strings for the "degenerate" case of $k = 1$, corresponding to equal compositions, as follows:

$$\boldsymbol{x}^{(1)} = (0,0,1,0), \quad \boldsymbol{y}^{(1)} = (0,1,0,0). \quad (5)$$

Puncturing the first bit from both $\boldsymbol{x}^{(1)}$ and $\boldsymbol{y}^{(1)}$ produces strings that still share the same gapped 1-deck. The same claim holds for the case when one punctures the last bit from both $\boldsymbol{x}^{(1)}$ and $\boldsymbol{y}^{(1)}$. Finally, the claim is true when one punctures both the first and the last bit from both strings. Hence, for $i = 1$,

$$\begin{aligned} &\mathcal{B}^{(i)}(\boldsymbol{x}^{(i)}) = \mathcal{B}^{(i)}(\boldsymbol{y}^{(i)}), \quad \mathcal{B}_{LR}^{(i)}(\boldsymbol{x}^{(i)}) = \mathcal{B}_{LR}^{(i)}(\boldsymbol{y}^{(i)}) \\ &\mathcal{B}_L^{(i)}(\boldsymbol{x}^{(i)}) = \mathcal{B}_L^{(i)}(\boldsymbol{y}^{(i)}), \quad \mathcal{B}_R^{(i)}(\boldsymbol{x}^{(i)}) = \mathcal{B}_R^{(i)}(\boldsymbol{y}^{(i)}). \end{aligned} \quad (6)$$

Let $G^*(k)$ be the smallest integer n such that there exist two distinct binary strings of length n, $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$, for which (6) holds for the case $i = k$.

Theorem 2. *With $\boldsymbol{x}^{(1)}$ and $\boldsymbol{y}^{(1)}$ defined as in (5) and*

$$\begin{aligned} \boldsymbol{x}^{(k)} &= \left(0, \boldsymbol{x}^{(k-1)}, 0, 0, \boldsymbol{y}^{(k-1)}, 0\right), \\ \boldsymbol{y}^{(k)} &= \left(0, \boldsymbol{y}^{(k-1)}, 0, 0, \boldsymbol{x}^{(k-1)}, 0\right), \end{aligned} \quad (7)$$

defined recursively, we have that (6) holds for all i. As a result, $G^*(k) \leqslant 4(2^k - 1)$ and $G(k) \leqslant 4(2^k - 1) - 2$.

Proof: We split the proof into four subproofs, in order to show that each of the four conditions in (6) hold for

$i = k$ if they hold for $i = k-1$. We do this by partitioning each deck in (6) with respect to whether each padded 0 is included in a subsequence or not, and by showing that there exists a correspondence between each pair of decks. The bound follows since the length of $\boldsymbol{x}^{(k)}$ equals $4(2^k-1)$ and $G(k) \leqslant G^*(k) - 2$, given that one can remove the padded 0s.

Part 1: Proof that $\mathcal{B}_{LR}^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{B}_{LR}^{(k)}\left(\boldsymbol{y}^{(k)}\right)$.

By definitions (4) and (7), this is equivalent to showing that

$$\mathcal{B}^{(k)}\left(\boldsymbol{x}^{(k-1)}, 0, 0, \boldsymbol{y}^{(k-1)}\right) = \mathcal{B}^{(k)}\left(\boldsymbol{y}^{(k-1)}, 0, 0, \boldsymbol{x}^{(k-1)}\right).$$

We can partition $\mathcal{B}^{(k)}\left(\boldsymbol{x}^{(k-1)}, 0, 0, \boldsymbol{y}^{(k-1)}\right)$ depending on which of the two 0s, if any, is included in the subsequence:

1) $\{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), \mathcal{D}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right)\right)\}$, $K \leqslant k$;
2) $\{\left(\mathcal{D}_R^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right)\right)\}$, $K \leqslant k-1$;
3) $\{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}_L^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right)\right)\}$, $K \leqslant k-1$,

where k_1 varies from 0 to K, for all K. First, we consider the case where neither of the two 0s is used and show that

$$\{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{y}^{(k-1)}\right), \mathcal{D}^{(K-k_1)}\left(\boldsymbol{x}^{(k-1)}\right)\right)\} = \{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), \mathcal{D}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right)\right)\}, \tag{8}$$

for any $K \leqslant k$. In this case, each string comprises $k_1 \leqslant K$ symbols from $\boldsymbol{x}^{(k-1)}$ and $K-k_1$ symbols from $\boldsymbol{y}^{(k-1)}$, where $K \leqslant k$ denotes the length of the resulting string. When $k_1 = K$ or $k_1 = 0$, (8) holds, since the subsequences $\boldsymbol{x}^{(k-1)}, \boldsymbol{y}^{(k-1)}$ appear in both $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$. Otherwise, when $0 < k_1 < K$, since the subsequences $\boldsymbol{x}^{(k-1)}, \boldsymbol{y}^{(k-1)}$ appear (and are "nonadjacent") in both $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$, $\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right) = \mathcal{D}^{(k_1)}\left(\boldsymbol{y}^{(k-1)}\right)$, and $\mathcal{D}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right) = \mathcal{D}^{(K-k_1)}\left(\boldsymbol{x}^{(k-1)}\right)$ (since both $k_1 < k$ and $K-k_1 < k$), it follows that (8) also holds for $0 < k_1 < K$.

The multiset of subsequences covered by case 2 contains strings that are formed by concatenating k_1 bits from $\boldsymbol{x}^{(k-1)}$, the first 0 between the subsequences $\boldsymbol{x}^{(k-1)}$ and $\boldsymbol{y}^{(k-1)}$ and $K-k_1$ bits from $\boldsymbol{y}^{(k-1)}$. Next, we show that

$$\{\left(\mathcal{D}_R^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right)\right)\} = \{\left(\mathcal{D}_R^{(k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0, \mathcal{D}^{(K-k_1)}\left(\boldsymbol{x}^{(k-1)}\right)\right)\} \tag{9}$$

for $K \leqslant k-1$. Since $K \leqslant k-1$, we have $\mathcal{D}_R^{(k_1)}(\boldsymbol{x}^{(k-1)}) = \mathcal{D}_R^{(k_1)}(\boldsymbol{y}^{(k-1)})$ and $\mathcal{D}^{(K-k_1)}(\boldsymbol{y}^{(k-1)}) = \mathcal{D}^{(K-k_1)}(\boldsymbol{x}^{(k-1)})$, which implies that we can form strings by concatenating k_1 bits from $\boldsymbol{y}^{(k-1)}$, the first 0 between the substrings $\boldsymbol{y}^{(k-1)}$ and $\boldsymbol{x}^{(k-1)}$, and $K-k_1$ bits from $\boldsymbol{x}^{(k-1)}$. Thus, (9) holds.

Using the same approach, it can be shown that

$$\{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}_L^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right)\right)\} = \{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0, \mathcal{D}_L^{(K-k_1)}\left(\boldsymbol{x}^{(k-1)}\right)\right)\}, \tag{10}$$

for any $K \leqslant k-1$. From (8), (9), and (10), it then follows that $\mathcal{B}_{LR}^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{B}_{LR}^{(k)}\left(\boldsymbol{y}^{(k)}\right)$.

Part 2: Proof that $\mathcal{B}_{L}^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{B}_{L}^{(k)}\left(\boldsymbol{y}^{(k)}\right)$.

By definitions (2) and (7), this is equivalent to showing that

$$\mathcal{B}^{(k)}\left(\boldsymbol{x}^{(k-1)}, 0, 0, \boldsymbol{y}^{(k-1)}, 0\right) = \mathcal{B}^{(k)}\left(\boldsymbol{y}^{(k-1)}, 0, 0, \boldsymbol{x}^{(k-1)}, 0\right).$$

We first partition $\mathcal{B}_{L}^{(k)}\left(\boldsymbol{x}^{(k)}\right)$ into two multisets: The first contains subsequences that include the last (trailing) 0 while the second contains those which do not (equivalent to $\mathcal{B}_{LR}^{(k)}(\boldsymbol{x}^{(k)})$). The first multiset can be partitioned into three classes:

1) $\{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), \mathcal{D}_R^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0\right)\}$, $K \leqslant k-1$;
2) $\{\left(\mathcal{D}_R^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}_R^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0\right)\}$, $K \leqslant k-2$;
3) $\{\left(\mathcal{D}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}_{LR}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0\right)\}$, $K \leqslant k-2$.

Using an almost identical argument as the one described in *Part 1*, one can show that $\mathcal{B}_{L}^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{B}_{L}^{(k)}\left(\boldsymbol{y}^{(k)}\right)$.

Part 3: Proof that $\mathcal{B}_{R}^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{B}_{R}^{(k)}\left(\boldsymbol{y}^{(k)}\right)$.

The proof of this case follows by symmetry from *Part 2*.

Part 4: Proof that $\mathcal{B}^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \mathcal{B}^{(k)}\left(\boldsymbol{y}^{(k)}\right)$.

The final step in the proof is to show that

$$\mathcal{B}^{(k)}\left(0, \boldsymbol{x}^{(k-1)}, 0, 0, \boldsymbol{y}^{(k-1)}, 0\right) = \mathcal{B}^{(k)}\left(0, \boldsymbol{y}^{(k-1)}, 0, 0, \boldsymbol{x}^{(k-1)}, 0\right)$$

Using a similar approach as before, we now partition the subsequences in $\mathcal{B}^{(k)}(\boldsymbol{x}^{(k)})$ according to whether they

1) contain the leading 0, but not the trailing 0;
2) contain the trailing 0, but not the leading 0;
3) contain neither the trailing nor the leading 0;
4) contain both the leading and the trailing 0.

This is equivalent to:

1) $\mathcal{B}_{R}^{(k)}\left(\boldsymbol{x}^{(k)}\right) \setminus \mathcal{B}_{LR}^{(k)}\left(\boldsymbol{x}^{(k)}\right)$;
2) $\mathcal{B}_{L}^{(k)}\left(\boldsymbol{x}^{(k)}\right) \setminus \mathcal{B}_{LR}^{(k)}\left(\boldsymbol{x}^{(k)}\right)$;
3) $\mathcal{B}_{LR}^{(k)}\left(\boldsymbol{x}^{(k)}\right)$;
4) $\mathcal{B}^{(k)}\left(\boldsymbol{x}^{(k)}\right) \setminus (\mathcal{B}_{R}^{(k)}\left(\boldsymbol{x}^{(k)}\right) \cup (\mathcal{B}_{L}^{(k)}\left(\boldsymbol{x}^{(k)}\right))$.

From the first three parts of the proof, we know that the first three multisets are the same for $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$. We only need to prove that the fourth multiset is the same as well. Again we partition the multiset of interest into three classes:

1) $\{\left(0, \mathcal{D}_L^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), \mathcal{D}_R^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0\right)\}$;
2) $\{\left(0, \mathcal{D}_{LR}^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}_R^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0\right)\}$;
3) $\{\left(0, \mathcal{D}_L^{(k_1)}\left(\boldsymbol{x}^{(k-1)}\right), 0, \mathcal{D}_{LR}^{(K-k_1)}\left(\boldsymbol{y}^{(k-1)}\right), 0\right)\}$,

where for case 1, $K \leqslant k-2$, and for cases 2 and 3, $K \leqslant k-3$. Using similar arguments as before completes the proof. ■

Using a similar approach, we can extend the bound to the s-gapped case to get $G_s(k) \leqslant (5s-2)2^{k-1} - 5s + 4$. This is

done by adding $s-1$ 0s on the outside and s 0s between xy and yx. We remove $2s-2$ 0s for the bound since the s-gapped k-deck does not need to satisfy the extra conditions required by the recursive construction.

We numerically computed $G(k)$ for $k = 2, 3, 4$. The results are displayed below, which clearly indicate that the upper

k	$G(k)$	Confusable pairs (examples)
2	6	(0,1,0,0,1,1), (0,0,1,1,0,1)
3	13	(1,1,0,1,1,1,1,0,1,0,1,1,1), (1,1,1,0,1,0,1,1,1,1,0,1,1)
4	24	(1,1,0,0,1,1,0,1,0,1,0,1,0,0,1,1,0,0,1,1,0,1,0,0), (1,1,0,1,0,0,1,1,0,0,1,1,0,1,0,1,0,1,0,0,1,1,0,0)

bound $4(2^k-1)-2$ is loose for larger values of k: For $k=4$, the bound equals 58 while the correct value is only 24. Also, the exact values of $G(k)$ are significantly larger than those for the ungapped case, for which we know that $S(k) = 4, 7, 12$ (compared to $G(k) = 6, 13, 24$) for $k = 2, 3, 4$, respectively. We therefore turn our attention to improving the bound on $G(k)$ using more sophisticated counting arguments.

IV. Improved Upper Bounds for Gapped k-Decks

We find the following definitions and notation from [6] useful for our subsequent derivations. Let $\Gamma = \{X, Y\}$ and let J denote a "wildcard". For integers $0 \leqslant r \leqslant k$ let

$$U_r(k) = \{\boldsymbol{w} \in \bigcup_{j=r} (\Gamma \cup \{J\})^j : \boldsymbol{w} \text{ has exactly } r \text{ non-}J \text{ symbols}\}.$$

For $t \geqslant 1$ and $k_1 \geqslant \cdots \geqslant k_t \geqslant t$, let

$$U(k_1, \ldots, k_t) = U_1(k_1) \cup U_2(k_2) \cup \cdots \cup U_t(k_t). \qquad (11)$$

We restrict our attention to $U(k_1, k_2) = U_1(k_1) \cup U_2(k_2)$, the set of all strings of length at most k_1 that have exactly one non-J character and the set of strings of length at most k_2 that have exactly two non-J characters.

When we refer to the multiplicity with which a string $\boldsymbol{w}$ that contains wildcards (J's) occurs as a subsequence of a string $\boldsymbol{p}$ that contains no wildcards (denoted by $N(\boldsymbol{w}, \boldsymbol{p})$), we map each wildcard to either X or Y. For example, if $\boldsymbol{w} = (J, X)$ and $\boldsymbol{p} = (Y, X, Y, X)$, we have $N(\boldsymbol{w}, \boldsymbol{p}) = 4$ because (X, X) and (Y, X) occur as subsequences of $\boldsymbol{p}$ with multiplicity 1 and 3, respectively. Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be two binary strings. We write $\boldsymbol{p} \sim^{U_r(k)} \boldsymbol{q}$ if $N(\boldsymbol{w}, \boldsymbol{p}) = N(\boldsymbol{w}, \boldsymbol{q})$ for all $\boldsymbol{w} \in U_r(k)$. In addition, we write $\boldsymbol{p} \sim^{U(k_1,k_2)} \boldsymbol{q}$ if $N(\boldsymbol{w}, \boldsymbol{p}) = N(\boldsymbol{w}, \boldsymbol{q})$ for all $\boldsymbol{w} \in U(k_1, k_2)$.

Next, let $S_U(k_1)$ be the smallest integer m for which there exist distinct strings $\boldsymbol{p}$ and $\boldsymbol{q}$ of length m such that $\boldsymbol{p} \sim^{U_r(k_1)} \boldsymbol{q}$. Similarly, let $S_U(k_1, k_2)$ be the smallest integer m for which there exist distinct strings $\boldsymbol{p}$ and $\boldsymbol{q}$ of length m such that $\boldsymbol{p} \sim^{U(k_1,k_2)} \boldsymbol{q}$. The following lemma is used in our subsequent derivations.

Lemma 3. *[6] Let $k_1 \geqslant k_2 \geqslant 2$ and $\kappa = k_1^2 + k_2^2(k_2-1)/2$. Then $S_U(k_1, k_2) \leqslant \kappa(\lg\kappa + \lg\lg\kappa + 1) = (1 + o(1))\kappa\lg\kappa$.*

Let $N_g(\boldsymbol{w}, \boldsymbol{x})$ be the number of times a string $\boldsymbol{w}$ appears as a gapped subsequence of $\boldsymbol{x}$ (i.e., so that all indices in $\boldsymbol{x}$ are nonadjacent). When $N_g(\boldsymbol{w}, \boldsymbol{x}) = N_g(\boldsymbol{w}, \boldsymbol{y})$ for all strings $\boldsymbol{w}$ of length $\leqslant k$, then we write $\boldsymbol{x} \sim^{k_{(g)}} \boldsymbol{y}$, i.e. $\mathcal{B}^{(k)}(\boldsymbol{x}) = \mathcal{B}^{(k)}(\boldsymbol{y})$.

Let $\Gamma = \{X, Y\}$ and let Σ be an arbitrary alphabet. For a finite-length string $\boldsymbol{x}$ over Σ, define $\boldsymbol{x}_0$ to be the string obtained by padding x with one 0 at both ends. For a finite-length string $\boldsymbol{p}$ over Γ and two finite-length strings $\boldsymbol{x}$ and $\boldsymbol{y}$ over Σ, let $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ be the string obtained from $\boldsymbol{p}$ by replacing each X by the string $\boldsymbol{x}_0$ and each Y by the string $\boldsymbol{y}_0$. For example, If $\boldsymbol{p} = (X, Y)$ and $\boldsymbol{x} = (0, 1, 0, 1)$ and $\boldsymbol{y} = (1, 1, 0, 0)$, then $\boldsymbol{x}_0 = (0, 0, 1, 0, 1, 0)$, $\boldsymbol{y}_0 = (0, 1, 1, 0, 0, 0)$ and $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}) = (0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0)$. We are now ready to prove an analogue of Lemma 9 from [6] for the case of gapped k-decks.

Lemma 4. *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two distinct strings in Σ^n such that $\boldsymbol{x} \sim^{k_{(g)}} \boldsymbol{y}$, $\mathcal{B}_L^{(k)}(\boldsymbol{x}) = \mathcal{B}_L^{(k)}(\boldsymbol{y})$, $\mathcal{B}_R^{(k)}(\boldsymbol{x}) = \mathcal{B}_R^{(k)}(\boldsymbol{y})$ and $\mathcal{B}_{LR}^{(k)}(\boldsymbol{x}) = \mathcal{B}_{LR}^{(k)}(\boldsymbol{y})$. Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be two distinct binary strings in Γ^m, such that for some $\sigma \in \{0, 1, 2\}$, we have $\boldsymbol{p} \sim^{U(2k+\sigma,k+\sigma)} \boldsymbol{q}$. Then, $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ and $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{q})$ are distinct and we have $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}) \sim^{3k+\sigma_{(g)}} h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{q})$. The same result holds when puncturing $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ and $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{q})$ on the left, right, and on both sides by one bit.*

Proof: Due to space limitations, we only provide a sketch of the proof. Let $\boldsymbol{w}$ be a string of length at most $3k+\sigma$ in Σ, and $p = (p_1, \ldots, p_m)$. The idea of the original proof [6] for the ungapped case is as follows: Each mapping that takes $\boldsymbol{w}$ to $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ (as a subsequence) defines a splicing of $\boldsymbol{w}$ of the form $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_m)$, where $\boldsymbol{w}_i$ is the preimage of $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}_i)$. Note that some $\boldsymbol{w}_i$ strings may be empty. Hence, we can write the set of all mappings which take $\boldsymbol{w}$ to $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ as the union of direct products (see [6] for the specific notation) $\bigcup_{t\geqslant 1}\bigcup_l\bigcup_r \prod_{i=1}^t \mathcal{N}(\boldsymbol{w}_{l,i}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}_{r(i)}))$, where t is the number of nonempty segments, the second union is taken over all functions l which partition $\boldsymbol{w}$ into t nonempty segments for a fixed t, the third union is taken over all functions r mapping the t chosen nonempty segments in $\boldsymbol{w}$ to t of the segments in $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ (equivalently, each r corresponds to a way in which we pick t out of m segments of $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$), and finally, $\mathcal{N}(\boldsymbol{w}_{l,i}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}_{r(i)}))$ denotes the set of mappings taking the i-th nonempty segment of $\boldsymbol{w}$ into the corresponding chosen segment in $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$. Note that for every mapping f which takes $\boldsymbol{w}$ to $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$, there is a specific t, l and r that correspond to f. Furthermore, f is the direct product of t mappings, each taking one of the t nonempty segments of $\boldsymbol{w}$ to the corresponding segment of $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$ (which is all uniquely determined by fixing t, l and r). Now, by converting this expression into a corresponding sum, we get $N(\boldsymbol{w}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})) = \sum_{t\geqslant 1}\sum_l\sum_r \prod_{i=1}^t N(\boldsymbol{w}_{l,i}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}_{r(i)}))$. Since $\boldsymbol{w}$ has length at most $3k+\sigma \leqslant 3k+2$, it has at most two segments of length $> k$. Therefore, we have three types of l's (i.e., ways of partitioning $\boldsymbol{w}$ into t nonempty segments for a fixed t): The ones with no segments of length $> k$, the ones with one such segment and the ones with two such segments. This means that $N(\boldsymbol{w}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p}))$ is a triple sum. Since $\boldsymbol{p} \sim^{U(2k+\sigma,k+\sigma)} \boldsymbol{q}$, after some calculations we get $N(\boldsymbol{w}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})) = N(\boldsymbol{w}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{q}))$.

To adapt this procedure for the gapped case, we need to show that $N_g(\boldsymbol{w}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})) = N_g(\boldsymbol{w}, h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{q}))$. The first difference is that each gapped mapping that takes $\boldsymbol{w}$ to $h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{p})$

(as a gapped subsequence) defines a splicing of $\boldsymbol{w}$ of the form $\boldsymbol{w} = \boldsymbol{w}_1 z_1 \boldsymbol{w}_2 z_2 \dots z_{m-1} \boldsymbol{w}_m$, where $\boldsymbol{w}_i$ is the preimage of $h_{x,y}(\boldsymbol{p}_i)$ and z_i is the preimage of the i-th pair of 0s that we added between the $\boldsymbol{x}$ and between the $\boldsymbol{y}$ strings and between the $\boldsymbol{x}$ and $\boldsymbol{y}$ strings when constructing $h_{x,y}(\boldsymbol{p})$. Again, note that some $\boldsymbol{w}_i$ and z_i strings may be empty.

However, an important difference between the gapped and ungapped case is that we need to consider different cases based on whether z_i is empty or not, for all indices i. This is because if z_1 is nonempty, for example, then we need to make sure that we do not use the rightmost bit in $h_{x,y}(\boldsymbol{p}_1)$ (or the leftmost bit in $h_{x,y}(\boldsymbol{p}_2)$, depending on whether the gapped mapping takes z_1 to the first or second 0 from the pair of 0s in $h_{x,y}(\boldsymbol{p})$ that are positioned between $h_{x,y}(\boldsymbol{p}_1)$ and $h_{x,y}(\boldsymbol{p}_2)$). This case ($z_1$ nonempty) gives rise to several additional cases that need to be considered, depending on which of the strings $z_2, z_3, \dots, z_{m-1}$ are empty. In other words, we can write out $h_{x,y}(\boldsymbol{p}) = (0, s_1, 0, 0, s_2, 0, 0, ..., 0, 0, s_m, 0)$ where each $s_i \in \{\boldsymbol{x}, \boldsymbol{y}\}$. Any gapped subsequence will then be of the form $(J_0, \mathcal{D}^{(i_1)}_{\alpha_1}(s_1), J_1, \mathcal{D}^{(i_2)}_{\alpha_2}(s_2), J_2, ..., J_{m-1}, \mathcal{D}^{(i_m)}_{\alpha_m}(s_m), J_m)$ where $J_j \in \{0, \varnothing\}$ and $\alpha_j \in \{\varnothing, L, R, LR\}$. Here, each J_j represents a 0 in the padding and whether it is a part of the subsequence or not. Then, depending on whether or not we use the padding, we puncture s_j on the left, right, both, or neither. This is captured by the indices α_j's. We also have that $\sum_{j=1}^{m} i_j + \sum_{j=1}^{m} |J_j| \leqslant 3k + \sigma$. Since by our assumptions $\mathcal{D}^{(i)}_{\alpha}(p_j) = \mathcal{D}^{(i)}_{\alpha}(q_j)$ for all $i \leqslant k$, $1 \leqslant j \leqslant m$, $\alpha_j \in \{\varnothing, L, R, LR\}$, we have an equivalence between $h_{x,y}(\boldsymbol{p})$ and $h_{x,y}(\boldsymbol{q})$ for each $i_j \leqslant k$. By the summation constraint, there are at most two indices j such that $i_j \geqslant k+1$. Let us consider the case when there is exactly one such j, denoted by j^*. In this case we have to pick fewer than $2k + \sigma$ of the remaining characters to obtain the final subsequence. We can also divide $h_{x,y}(\boldsymbol{p})$ into subblocks of the form $(0, s_j, 0)$, i.e., we can splice $\boldsymbol{w}$ into a collection of $\boldsymbol{w}_j$'s, where $\boldsymbol{w}_j$ is the string mapped to one of the blocks and $|\boldsymbol{w}_j| < k$ for $j \neq j^*$. The multiplicity of $\boldsymbol{w}$ can be seen to be $N_g(\boldsymbol{w}_{j^*}, (s_{j^*})_{\alpha_{j^*}}) \prod_{j \neq j^*} N_g(\boldsymbol{w}_j, (\boldsymbol{x})_{\alpha_j})$, where α once again depends on whether the bit used for padding is included in the subsequence. By our assumption we have $N(J^{a_1} A J^{a_2}, \boldsymbol{p}) = N(J^{a_1} A J^{a_2}, \boldsymbol{q})$, where J^{a_1} is a sequence of a_1 concatenated wildcard characters, $A \in \{x, y\}$, $a_1 + a_2 < 2k + \sigma$. Hence, there are equally many s_{j^*}'s in $h_{x,y}(\boldsymbol{p})$ and $h_{x,y}(\boldsymbol{q})$. Using similar arguments and the fact that $N(J^{a_1} A J^{a_2} B J^{a_3}, \boldsymbol{p}) = N(J^{a_1} A J^{a_2} B J^{a_3}, \boldsymbol{q})$ we can also prove the equivalence for the case of two indices j for which $i_j \geqslant k+1$. This leads to $N_g(\boldsymbol{w}, h_{x,y}(\boldsymbol{p})) = N_g(\boldsymbol{w}, h_{x,y}(\boldsymbol{q}))$. ■

The lemma gives rise to the following important Corollary.

Corollary 5. *For every $\sigma \in \{0,1,2\}$, one has $G^*(3k+\sigma) \leqslant (G^*(k)+2)(S_U(2k+\sigma, k+\sigma))$.*

Combining the above corollary with Lemma 3 and Lemma 4 leads to an upper bound for $G(k)$ as follows. First, we set

$$\kappa = (2k+\sigma)^2 + (k+\sigma)^2(k+\sigma-1)/2,$$

which equals

$$\frac{1}{2}(\sigma^3 + (3k+1)\sigma^2 + 3\sigma(k^2+2k) + (1+7/k)k^3)$$
$$= (\frac{1}{2} + o(1))\, k^3,$$

in Lemma 3 to obtain

$$S_U(2k+\sigma, k+\sigma) \leqslant C(k) k^3 log_3 k, \tag{12}$$

where $C(k) = \frac{3 \lg 3}{2} + o(1)$. We also have that $C(k) \leqslant 10$ for $k \geqslant 9$, $C(k) \leqslant 3$ for $k \geqslant 3^5$ [6]. Using the inequality from Corollary 5 and (12), we set $k_0 = k$ and for $i > 0$, $k_i = \lfloor k_{i-1}/3 \rfloor \leqslant k/3^i$. We stop the recursion with $i = i_0$, where $k_{i_0} \leqslant 4$ (so $G^*(k_{i_0}) \leqslant 4(2^4 - 1) = 60$) and get

$$G^*(k) \leqslant G^*(k_{i_0}) \prod_{i=1}^{i_0} S_U(2k_i + \sigma_i, k_i + \sigma_i) \tag{13}$$
$$+ 2 \sum_{i=1}^{i_0} \prod_{j=1}^{i} S_U(2k_i + \sigma_i, k_i + \sigma_i).$$

Combining the above bound with that on S_U we obtain

$$\begin{aligned}
G^*(k) &\leqslant G^*(k_{i_0}) \prod_{i=1}^{i_0} C(k_i)(k_i)^3 log_3(k_i) \\
&+ 2 \sum_{i=1}^{i_0} \prod_{j=1}^{i} C(k_i)(k_i)^3 log_3(k_i) \\
&\leqslant 3^{\log_3(60) + \sum_{i=1}^{\lceil \log_3(k/4) \rceil} [O(1) + 3(\log_3 k - i) + \log_3(\log_3 k - i)]} \\
&+ \sum_{i=1}^{\lceil \log_3(k/4) \rceil} 3^{\log_3 2 + \sum_{j=1}^{i} [O(1) + 3(\log_3 k - j) + \log_3(\log_3 k - j)]} \\
&= 3^{O(1) + O(\log_3 k) + O(\log_3^2 k) + O(\log_3 k \log_3 \log_3 k)} \\
&+ O(\log_3(k)) 3^{O(1) + O(\log_3 k) + O(\log_3^2 k) + O(\log_3 k \log_3 \log_3 k)} \\
&= O(\log_3(k)) 3^{O(\log_3^2 k)}.
\end{aligned}$$

Since we have $G(k) \leqslant G^*(k) - 2$, we also have $G(k) \leqslant O(\log_3(k)) 3^{O(\log_3^2 k)}$. By bounding $G^*(k)$ we also obtain

$$\begin{aligned}
G(k) &\leqslant (4(2^{k/3} - 1) + 2) C(k/3) (k/3)^3 \log_3(k/3) - 2 \\
&\leqslant 4/27 * 2^{k/3} * k^3 \log_3(k/3) * C(k/3) - 2
\end{aligned}$$

for $k \geqslant 28$. Since in this case $C(k/3) \leqslant 10$, we arrive at

$$G(k) \leqslant 1.482 * 1.26^k * k^3 \log_3(k/3) - 2, \tag{14}$$

In comparison, the general bound for the ungapped case, derived in [6], reads as

$$S(k) \leqslant 1.2 \Gamma(\log_3 k)\, 3^{(3/2) \log_3^2 k - (1/2) \log_3 k}, \; k \geqslant 85.$$

The bounds are summarized in the tables below.

k	Bound
2-4	Exact values: 6,13,24
5-27	$4(2^k - 1)$
> 27	$1.482 * 1.26^k * k^3 \log_3(k/3) - 2$

k	28	29	30	31	32	33
$G(k) \leqslant$	42742211	60773950	86039831	121319982	170424514	238563374

REFERENCES

[1] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, String reconstruction from substring compositions, *SIAM Journal on Discrete Mathematics* 29, no. 3 (2015): 1340-1371.

[2] J-P. Allouche and J. Shallit, "The ubiquitous Prouhet-Thue-Morse Sequence," *In Sequences and their Applications*, pp. 1-16. Springer, London, 1999.

[3] T. Batu, S. Kannan, S. Khanna, and A. McGregor, Reconstructing strings from random traces, *Departmental Papers (CIS)* (2004): 173.

[4] Z. Chase, Separating words and trace reconstruction, *In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, (2021) 21–31.

[5] M. Cheraghchi, R. Gabrys, O. Milenkovic, and J. Ribeiro, "Coded trace reconstruction," *IEEE Transactions on Information Theory*, 66, no. 10, pp. 6084-6103, 2020.

[6] M. Dudik and L.J. Schulman, Reconstruction from subsequences, *Journal of Combinatorial Theory, Series A* 103(2) (2003), 337–348.

[7] J. Chrisnata, H. M. Kiah, S. Rao, A. Vardy, E. Yaakobi, A. Yao, "On the number of distinct k-decks: Enumeration and bounds," *19th International Symposium on Communications and Information Technologies (ISCIT)* (2019) 519–524.

[8] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded strings from multiset substring spectra," *IEEE Transactions on Information Theory* 65, no. 12 (2019): 7682–7696.

[9] R. Gabrys and O. Milenkovic, "The hybrid k-deck problem: Reconstructing sequences from short and long traces," *IEEE International Symposium on Information Theory (ISIT)* (2017) 1306–1310.

[10] L.O. Kalashnik, The reconstruction of a word from fragments, *Numerical mathematics and computer technology* (1973), 56–57.

[11] H. M. Kiah, G. J. Puleo, and O. Milenkovic, Codes for DNA sequence profiles, *IEEE Transactions on Information Theory 62*, no. 6 (2016): 3125-3146.

[12] B. Manvel, A. Meyerowitz, A. Schwenk, K. Smith and P. Stockmeyer, Reconstruction of sequences, *Discrete Mathematics* 94(3) (1991), 209–219.

[13] D. Margaritis and S. S. Skiena, Reconstructing strings from substrings in rounds, *In Proceedings of IEEE 36th Annual Foundations of Computer Science* (1995) 613–620.

[14] A. D. Scott, Reconstructing sequences, *Discrete Mathematics* (1997) 175 1–3 231-238.

[15] E. Ukkonen, Finding approximate patterns in strings, *Journal of Algorithms*, 6, no. 1 (1985): 132-137.

[16] S.M.H. Yazdi, R. Gabrys and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific Reports* 7, no. 1 pp. 1-6, 2016 (online) / 2017 (print).