A general minimal perfect hash function for canonical k-mers on arbitrary alphabets with an application to DNA sequences

Roland Wittler[0000−0002−2249−9880]

Faculty of Technology, Bielefeld Institute for Bioinformatics Infrastructure (BIBI), and Center for Biotechnology (CeBiTec), Bielefeld University, Germany.

roland.wittler@uni-bielefeld.de

Abstract

To index or compare sequences efficiently, often k-mers, i.e., substrings of fixed length k, are used. In order to store them in a table, or to assign them to different tables or threads, k-mers are encoded as integers. One way to ensure an even distribution is to use minimal perfect hashing, i.e., a bijective mapping between all possible σ^k k-mers and the interval $[0, \sigma^k - 1]$, where σ is the alphabet size.

In many applications, e.g., when the reading direction of a DNA-sequence is ambiguous, *canonical* k-mers are considered, i.e., the lexicographically smaller of a given k-mer and its reverse (or reverse complement) is chosen as a representative. In naive encodings, canonical k-mers are not evenly distributed within the interval $[0, \sigma^k - 1]$ hampering an even distribution to threads or tables.

We present a minimal perfect hash function of canonical k-mers on alphabets of arbitrary size, i.e., a mapping to the interval $[0, \sigma^k/2 - 1]$. The approach is introduced for canonicalization under reversal and extended to canonicalization under reverse complementation. We further present a space and time efficient bit-based implementation for the DNA alphabet.

1 Introduction

The increasing amount of available genome sequence data enables large-scale analyses. A common way to handle long and many genomes is using k-mers, i.e., substrings of fixed length k, to efficiently index or compare sequences.

Many k-mer-based methods hash k-mers to integer values in order to store them in a table, or to assign them to different tables or threads. Here *minimal perfect hash functions* (MPHFs) are often used. They bijectively map elements of a set of size n to a continuous interval of integer values $[0, n-1]$. Such a gapless set of hash values allows for a simple and even distribution of k-mers to buckets of equal size.

The concept of minimal perfect hashing is applied in two different ways. Some tools use a combination of a data structure to store a given, fixed set S of observed k-mers and an algorithm to determine the hash value of individual k-mers from S, see e.g., [2, 4, 6, 7]. In contrast, here, we consider an MPHF purely as a function and S as a universal set of k-mers that is independent of any data set but may be restricted by certain properties as described below.

For the four letter DNA alphabet, each character can be encoded using two bits. Reading the resulting bit sequences as integer values corresponds to a MPHF for the set of all 4^k k-mers which are mapped to an interval of size 4^k.

In many situations, e.g., when genomes are not given as complete sequences but rather in form of contigs or reads, the reading direction of a given DNA sequence is unknown. To make a method independent from whether a sequence itself of its reverse (Watson Crick) complement is observed, both a k-mer and its reverse complement are assumed equivalent and one of them is chosen as a *canonical* representative of both, e.g., the lexicographically smaller one. For odd length k-mers, two k-mers pair to one canonical k-mer each. Thus, there are half as many canonical k-mers as there are k-mers. For even length k-mers, (Watson Crick) palindromes have to be considered, which are canonical by definition and do not pair, thus increasing the number of canonical k-mers. In any case, the (lexicographically) canonical k-mers are not evenly distributed – neither within the lexicographic ordering of k-mers nor in the interval of k-mer hash values spanned by a simple 2-bit encoding. See Figure 1 for an illustration. This hampers an even distribution of k-mers to buckets as exemplified in Figure 2 on page 10.

The problem can be solved by modifying the simple 2-bit encoding or replacing it by other hash functions in order to randomize the ranking of k-mers, e.g., using an *exclusive-or* operation to toggle half of the bits [8], multiplying by a randomly chosen invertible matrix on $\mathbb{Z}/2\mathbb{Z}$ [5], or applying other efficient hash functions like *MurmurHash* [1].

But still, the canonical k-mers correspond to only a subset of all hash values. Thus, when used as index for array-based data structures, a huge amount of the address space remains unused, requiring flexible-sized hash tables in combination with re-probing and re-hashing operations.

In view of the fact that (for odd k) exactly half of all k-mers are canonical, only $2k-1$ bits instead of $2k$ bits are necessary and sufficient to span the integer values from 0 to $n-1$ where n is the number of canonical k-mers – implying an MPHF.

One approach to obtain a $2k-1$-bit encoding is to define a k-mer being canonical not with respect to their lexicographic relation but rather based on their 2-bit encoding. To the best of our knowledge, such an approach has not been published yet, but has been proposed by the community[1]: Encode characters by two bits in such a way that a character and its complement have different parity each (even or odd number of ones), e.g., $A \mapsto 00$, $C \mapsto 01$,

[1] https://cs.stackexchange.com/questions/82644/compact-mapping-from-an-involuted-set (Thanks to Antoine Limasset for this pointer.)

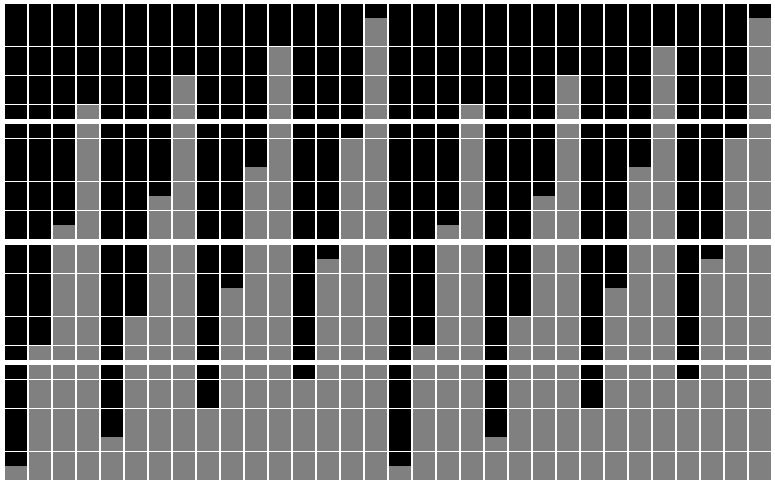

Figure 1: Distribution of (lexicographically) canonical k-mers among all k-mers. Each k-mer for $k = 5$ is represented as a block, ordered lexicographically from left to right, row by row. A block is drawn black if the k-mer is canonical, and gray otherwise. If the canonical k-mers would be assigned into four even size buckets by their lexicographic rank, as indicated by the space between the lines, the load factors would be 7:5:3:1, see also [3, endnote 3].

$G \mapsto 11$, $T \mapsto 10$. Then, the encoding of an odd length k-mer always has a different parity as its reverse complement. Hence, one parity can be chosen as canonical, and one bit, e.g. the first bit, can be omitted as it could be deduced by counting the ones in the shortened bit sequence. This approach, however, is only applicable for odd values of k, and there is no evident extension to even length k-mers.

While all above considerations are based on the DNA alphabet of size four, also alphabets of other sizes are worth to study. An alphabet size σ that is not a power of two does not allow for simple bit-based encoding. A generalization to a ranking-based approach offers an elegant integer encoding: Interpret a character as a digit between 0 and $\sigma - 1$, and a k-mer as a number in a base-σ numeral system. Then, a standard mapping to decimal values corresponds to an MPHF.

Here, we propose a general MPHF for canonical k-mers of arbitrary length (even or odd) on alphabets of arbitrary size. The approach is introduced for canonicalization under reversal and extended to canonicalization under reverse complementation. For alphabet size four, we show the relation to bit-space and provide an algorithm to encode all k-mers of a given sequence s in time $O(k + |s|)$. A simple implementation is available on GitLab: `https://gitlab.ub.uni-bielefeld.de/gi/mphfcan/`

2 Preliminaries

We consider as *alphabet* a finite, non-empty, ordered set $\{a_0, \ldots, a_{\sigma-1}\}$ with $a_0 < \cdots < a_{\sigma-1}$ and define the *rank* of character $a_i \in \mathcal{A}$ as $rank(a_i) = i$. A *string* s is a sequence of characters. Its *length* is denoted by $|s|$, the character at position i by $s[i]$, and the substring from position i through j by $s[i..j]$. Let ϵ denote the empty string. The concatenation of a string or character s and another string or character t is denoted by $s \cdot t$. The *reverse* of string s is $s^{-1} = s[|s|] \cdots s[1]$.

Strings of fixed length k are called *k-mer*. We define a k-mer x being *canonical* if it is lexicographically smaller than or equal to x^{-1}.

Let K_k^σ be the set of all k-mers over an alphabet of size σ. The number of k-mers is $|K_k^\sigma| = \sigma^k$. For the number of canonical k-mers, first note that a *palindrome*, i.e., a sequence $x = x^{-1}$, is canonical by definition. Each non-palindromic k-mer pairs with its reverse such that the number of canonical non-palindromic k-mers is half of the number of non-palindromic k-mers. The number of palindromes is $\sigma^{\lceil \frac{k}{2} \rceil}$, so the number of non-palindromic k-mers is $\sigma^k - \sigma^{\lceil \frac{k}{2} \rceil}$, half of which are canonical. In summary, the size of the set C_k^σ of canonical k-mers is:

$$|C_k^\sigma| = \frac{1}{2}\left(\sigma^k - \sigma^{\lceil k/2 \rceil}\right) + \sigma^{\lceil k/2 \rceil} = \frac{1}{2}\left(\sigma^k + \sigma^{\lceil k/2 \rceil}\right)$$

In order to efficiently handle k-mers, they can be indexed using a bijection to integers. We call a numeral representing a k-mer its *encoding*.

Definition 1. *Given a set S of n distinct elements, a function that bijectively maps elements from S into the range $[0, n-1]$ is called a* minimal perfect hash function (MPHF) *for S.*

A commonly used approach for an MPHF for k-mers is a rank based encoding. Each character is encoded by its rank, which is interpreted as a digit. The resulting sequence of digits can then be converted from the base-σ numeral system, denoted by $(\cdots)_\sigma$, to decimal values:

$$enc(x) := \big(rank\,(x[1]) \cdots rank\,(x[k])\big)_\sigma = \sum_{i=1}^{k} rank\,(x[i]) \cdot \sigma^{k-i}$$

The resulting values range from $enc(a_0 \cdots a_0) = 0$ to $enc(a_{\sigma-1} \cdots a_{\sigma-1}) = \sum_{i=1}^{k} (\sigma - 1) \cdot \sigma^{k-i} = (\sigma - 1) \cdot \sum_{j=0}^{k-1} \sigma^j = (\sigma - 1)\frac{1-\sigma^k}{1-\sigma} = \sigma^k - 1$, i.e., from 0 to $|K_k^\sigma| - 1$.

When instead encoding canonical k-mers using this approach, only part of the co-domain is actually used. In particular, the image does not correspond to an interval of integers, hampering its use for an MPHF. The following section introduces an encoding of canonical k-mers that has as co-domain the integer interval $[0, |C_k| - 1]$.

3 Encoding canonical k-mers

In the following, we will first introduce a preliminary encoding enc'_c of canonical k-mers, which is injective but not an MPFH. In a second step, gaps in the image of enc'_c will be removed to derive the MPHF enc_c. Finally, an expedient characteristic of enc_c with regard to palindromes will be highlighted.

3.1 Preliminary encoding

The encoding is based on the observation that there are fewer canonical k-mers than there are k-mers in general, i.e. $|C_k^\sigma| < |K_k^\sigma|$. A mapping is only required if k-mer x is canonical, i.e., x is lexicographically smaller than or equal to x^{-1}. This in turn can be seen when comparing its first to its last character, and if they are equal, its second to its second last etc., i.e., $x[i]$ to $x^{-1}[i] = x[k-i+1]$ for $i = 1, \ldots, \lceil k/2 \rceil$ until a pair $x[\hat{\imath}] < x[k-\hat{\imath}+1]$ is found, specifying x being canonical. If instead equality holds for all pairs, x is palindromic and thus canonical by definition.

A given k-mer will be encoded from the outside inwards, i.e. for increasing i, while different encoding schemes are used. For sake of convenience, the encoding is given as digits in base-σ space.

Unspecific pairs. If $x[i] = x[k-i+1]$, one character implies the other such that only one of the two characters needs to be encoded. Instead of actually encoding $x[i]$, digit zero is used to indicate the pair being unspecific, and $x[k-i+1]$ is encoded by its rank.

Specifying pair. If x is not palindromic, let $\hat{\imath}$ be the smallest value such that $x[\hat{\imath}] < x[k-\hat{\imath}+1]$. This pair determines x being canonical. Only a limited number of combinations of characters needs to be encoded. The left character has to be encoded with digits greater than zero to tell it apart from the unspecific case. The right character is encoded with a digit from zero to $\sigma - 1$. The combination of digits has to be continuous to avoid gaps in the co-domain. For instance, for an alphabet of size 4, six different combinations of characters are encoded by the lexicographically lowest six pairs of digits from $(1, 0)$ to $(2, 1)$. Function K is used to determine the rank of a character pair in the list of all possible combinations, see Table 1 (right) for an example. Values for $K(l, r)$ can be derived from the triangular numbers $T = [1, 3, 6, 10, 15, \ldots]$ (sequence A000217 in the On-Line Encyclopedia of Integer Sequences). In the modified matrix $T_{\sigma-1} - K(l, r)$, the values next to the diagonal correspond to the triangular numbers $T_{\sigma-1}, \ldots, T_1$. From those, the off-diagonal values can then be easily deduced. The value of K is then decomposed into the encoding of the left and right character using integer division and modulo, respectively.

$$
\begin{aligned}
K &= \tfrac{l}{2}\,(2\sigma - 3 - l) + r - 1 \\
K_l &= \lfloor K/\sigma \rfloor + 1 \\
K_r &= K \bmod \sigma
\end{aligned}
$$

specifying pair		K	encoding	
a_0	a_1	0	1	0
a_0	a_2	1	1	1
a_0	a_3	2	1	2
a_1	a_2	3	1	3
a_1	a_3	4	2	0
a_2	a_3	5	2	1

K	0	1	2	3
0		0	1	2
1			3	4
2				5
3				

Table 1: Example on the encoding of specifying pairs for alphabet size $\sigma = 4$. The six combinations of characters are listed in lexicographic order and numbered using function K shown on the right. The latter value is then decomposed to obtain the lexicographically smallest six combinations of digits that do not have a zero in its first position.

Remainder. All remaining characters for $\hat{\imath} < i < k - \hat{\imath} + 1$ (if any) are simply encoded by their rank.

Our preliminary encoding of canonical k-mers is summarized in the following function enc'_c shown below. It is formulated recursively where in the i-th step of the recursion, substring $x[i, k-i+1]$ is processed. The encoding of a palindromic k-mer solely consists of unspecific pairs until reaching the middle of the k-mer, which is either the empty string (even k) or a single character (odd k), the latter of which is encoded by its rank. For unspecific pairs and the remainder, the order of digits in the encoding corresponds to the order of characters in the given sequence. In contrast, for a reason explained later, the two digits encoding a specifying pair are placed next to each other at positions $\hat{\imath}$ and $\hat{\imath} + 1$ as shown in Example 1.

$$enc'_c(x) := \begin{cases} \text{With } l = rank(x[1]) \text{ and } r = rank(x[|x|]) : & \\ \textit{// Middle position of palindrome} & \\ \epsilon & \text{if } x = \epsilon \\ r & \text{if } |x| = 1 \\ \textit{// Unspecific pair} & \\ 0 \;\; \underbrace{enc'_c(x[2, |x| - 1])}_{\text{recurse}} \;\; r & \text{if } l = r \\ \textit{// Specifying pair} & \\ K_l \;\; K_r \;\; \underbrace{enc(x[2, |x| - 1])}_{\text{remainder}} & \text{if } l < r \end{cases}$$

Example 1. *Let $k = 6$ and $\sigma = 4$. The k-mer x shown below contains one unspecific pair. The specifying case is highlighted in bold and the remainder is underlined. $K(0,1) = 0$ such that $K_l = 1$ and $K_r = 0$ (see Table 1).*

$$x = a_0\ \boldsymbol{a_0}\ \underline{a_0\ a_0}\ \boldsymbol{a_1}\ a_0$$

$$enc'_c(x) = (\ 0\ \ \mathbf{1}\ \ \mathbf{0}\ \ \underline{0\ \ 0}\ \ 0\ )_\sigma$$

3.2 Transformation to an MPHF

In enc'_c, for a specifying pair, not all combinations of digits are used. Thus, enc'_c is non-surjective – in particular, the image of enc'_c is not an interval. For a given $enc'_c(x)$, however, knowing the position $\hat{\imath}$ of the specifying pair (if x is non-palindromic), we can calculate $missing(\hat{\imath})$, the number of all smaller elements of the co-domain that are not an encoding of any canonical k-mer. By subtracting this number from the decimal value of $enc'_c(x)$, we obtain a continuous range of encodings from 0 to $|C_k^\sigma| - 1$ and thus an MPHF.

$$enc_c(x) := (enc'_c(x))_\sigma - missing(\hat{\imath})$$

To determine the value of $missing(\hat{\imath})$, we need to count the number of digit sequences that (i) are not a valid encoding by enc'_c and (ii) have a value smaller than $enc'_c(x)$. For palindromes, no such sequences exist, i.e., $missing(x) = 0$. Otherwise, regarding (i), a sequence of digits can be identified invalid by reading from left to right and observing neither a zero that would result from encoding an unspecific pair, nor any of the valid $\binom{\sigma}{2}$ different combinations resulting from encoding a specifying pair. There are $\binom{\sigma}{2}$ such invalid combinations of digits a, b with $0 < a < b$. Regarding (ii), observe that in $enc'_c(x)$, to the left of position $\hat{\imath}$ there are only zeros. We have to consider those sequences starting with further zeros, i.e., positions of invalid pairs j, $j+1$ that are to the right of $\hat{\imath}$, $\hat{\imath}+1$. Further observe that any such pair has to be left of the middle position of the k-mer, i.e., $j \leq \lfloor \frac{k}{2} \rfloor$. For each such pair j, $j+1$, the remaining $m := k - (j+1)$ positions can take any value such that σ^m many invalid sequences have to be subtracted each:

$$\begin{aligned}
missing(\hat{\imath}) &= \sum_{m=\lceil k/2 \rceil - 1}^{k-\hat{\imath}-2} \sigma^m \cdot \binom{\sigma}{2} \\
&= \binom{\sigma}{2} \left(\sum_{m=0}^{k-\hat{\imath}-2} \sigma^m - \sum_{m=0}^{\lceil k/2 \rceil - 2} \sigma^m \right) \\
&= \frac{\sigma(\sigma-1)}{2} \left(\frac{\sigma^{k-\hat{\imath}-1} - 1}{\sigma - 1} - \frac{\sigma^{\lceil k/2 \rceil - 1} - 1}{\sigma - 1} \right) \\
&= \frac{\sigma^{k-\hat{\imath}} - \sigma^{\lceil k/2 \rceil}}{2}
\end{aligned}$$

Example 2. *Consider an alphabet of size $\sigma = 4$, k-mer length 6, and k-mer $x = a_0\, a_0\, a_0\, a_0\, a_1\, a_0$ from Example 1. In this case, $missing(\hat{\imath})$ counts all elements of the co-domain that are not in the image and have a value smaller than $enc'_c(x) = (\ 0\ 1\ 0\ 0\ 0\ 0\ )_4 = 256$. Those are all sequences which begin with two zeros, followed by a pair of digits that does not encode a specifying pair, followed by any two remaining digits:*

$$\left\{\ 0 \quad 0\ \right\} \times \left\{ \begin{array}{cc} 1 & 0 \\ 2 & 0 \\ 2 & 1 \\ 3 & 0 \\ 3 & 1 \\ 3 & 2 \end{array} \right\} \times \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\} \times \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$$

The number of these combinations is $6 \cdot 4 \cdot 4 = 96$, i.e., $enc_c(x) = enc'_c(x) - missing(2) = 256 - 96 = 160$. The next lowest valid case is:

$$enc_c(a_3\ a_3\ a_2\ a_3\ a_3\ a_3) = (\ 0\ 0\ 2\ 1\ 3\ 3\ )_4 - missing(3) = 159 - 0 = 159$$

Theorem 1. *Function enc_c is an MPHF for C_k^σ.*

Proof. First observe that $missing(\hat{\imath}) < \sigma^{k-\hat{\imath}}$, and thus position $\hat{\imath}$ in $enc'_c(x)$ will not be affected by subtracting $missing(\hat{\imath})$ and, in particular, the number of leading zeros is the same in $enc'_c(x)$ and $enc_c(x)$. Further observe that the number of leading zeros is equal to $\hat{\imath} - 1$, and thus allows us to derive $enc'_c(x)$ from $enc_c(x)$ by adding $missing(\hat{\imath})$. Hence, there is a bijection between the images of enc_c and enc'_c.

A careful inspection of enc'_c shows that it is well defined on any canonical k-mer and $enc'_c(x) \neq enc'_c(y)$ for any $x \neq y$: the number of leading zeros of $enc'_c(x)$ together with the corresponding digits at its end determine the unspecific pairs. Either the following two digits determine the specifying pair and the remaining digits determine the remaining characters, or the middle position is reached in case of a palindrome.

It remains to be shown that the $|C_k^\sigma|$ many different canonical k-mers are mapped to values not larger than $|C_k^\sigma| - 1$. To this end, consider a value larger than $|C_k^\sigma| - 1$ to be decoded. In base-σ space, this value would take at least $\lfloor \log_\sigma |C_k^\sigma| \rfloor + 1 > k$ digits. Since valid encodings are of length k (including leading zeros), this is a contradiction. □

This encoding can easily be extended to include non-canonical k-mers such that they are encoded as their canonical counterpart. To this end, the specifying cases have to be complemented with those pairs that specify a string being non-canonical, i.e., $K(a, b) := K(b, a)$ for $b < a$. Then, instead of the remainder, its reverse has to be encoded.

3.3 Palindromes

Palindromes do not pair to one canonical k-mer but are usually expected to be observed in sequence data approximately as often as non-palindromic k-mers that do pair to one canonical k-mer. For an even distribution of all observed k-mers into buckets, buckets receiving palindromes have to cover a larger range of hash values than buckets that only receive non-palindromic canonical k-mers. In order to simplify an appropriate arrangement of bucket boundaries, the above encoding is designed in such a way that palindromes are hashed to the lowest $1/2 \cdot \sigma^{\lceil k/2 \rceil}$ values, which can be seen as follows. If and only if x is a palindrome, $enc'_c(x)$ has $\lfloor k/2 \rfloor$ leading zeros. Otherwise, the number of leading zeros is smaller. As mentioned in the proof of Theorem 1, the number of leading zeros is the same for $enc_c(x)$. Thus, the encoding of palindromes has more leading zeros than any other encoding.

This characteristic of enc_c will also be exploited in the following section, where palindromes play an important role.

4 Canonicalization under reverse complementation

In certain applications, in particular when handling genomic sequences, instead of only reverting a sequence, its reverse (Watson-Crick) complement is considered. For an even-size alphabet $\{a_0, \ldots, a_{\sigma-1}\}$, the *reverse complement* of a string s is $\overline{s} := \overline{s[|s|]} \cdots \overline{s[1]}$, where $\overline{a_i} := a_{\sigma-1-i}$. For instance, for the DNA alphabet $\{A, C, G, T\}$, $\overline{A} := T, \overline{C} := G, \overline{G} := C, \overline{T} := A$. An extension to alphabets of odd size, where $\overline{a_{\lceil \sigma/2 \rceil}} := a_{\lceil \sigma/2 \rceil}$ would be possible, but those technicalities are omitted here for the sake of comprehensibility. Let R_k^σ be the set of all canonical k-mers over an alphabet of even size σ under reverse complementation.

For an odd-length k-mer, taking its reverse complement always changes the character at its middle position. Thus, x is always different from $\overline{x}$, i.e., there are no palindromes, and the number of canonical k-mers is half the number of k-mers:

$$|R_k^\sigma| = \begin{cases} \frac{1}{2}\sigma^k & \text{if } k \text{ is odd, and} \\ \frac{1}{2}\left(\sigma^k + \sigma^{k/2}\right) & \text{if } k \text{ is even.} \end{cases}$$

The encoding $enc'_c(x)$ from Section 3.1 can be applied analogously here – just $r = rank(x[|x|])$ has to be replaced by $r = rank(\overline{x[|x|]})$, which we will denote $enc_c^{r\prime}(x)$ in the following. If k is odd, there are no palindromes, but the middle position of a k-mer might specify it being canonical. So, the very first case in the definition of $enc_c^{r\prime}(x)$ will only apply to the higher half of the alphabet. Hence, the lowest $\sigma/2 \cdot \sigma^{\lfloor k/2 \rfloor} = 1/2 \cdot \sigma^{\lceil k/2 \rceil}$ values are never taken (see also Section 3.3). Their number has to be included in $missing(\hat{\imath})$ such that the image is shifted from $[1/2\,\sigma^{\lceil k/2 \rceil}, |C_k^\sigma| - 1]$ to $[0, |R_k^\sigma| - 1]$.

$$enc_c^r(x) := (enc_c^{r\prime}(x))_\sigma - missing^r(\hat{\imath})$$

$$\text{where } missing^r(\hat{\imath}) = \begin{cases} missing(\hat{\imath}) - \frac{1}{2}\sigma^{\lceil k/2 \rceil} & \text{if } k \text{ is odd, and} \\ missing(\hat{\imath}) & \text{if } k \text{ is even.} \end{cases}$$

Corollary 1. *Function enc_c^r is an MPHF for R_k^σ.*

To allow non-canonical k-mers as input to be encoded as their canonical counterpart, instead of the specifying pair and the remainder, their reverse complements have to be encoded.

4.1 Evaluation on real data

We compared the use of the standard 2-bit encoding and enc_c^r to assign k-mers to buckets. For all 15-mers from a genome sequence of *Yersinia pestis*, both encodings were determined. For non-canonical k-mers, the encoding of the canonical counterpart has been used. Figure 2 shows their distribution on 16 buckets of size $|K_{16}^4|/16$ and $|R_{16}^4|/16$, respectively. Further, *modulo* 16 has been used to assign the standard 2-bit encodings to buckets. Only enc_c^r results in an approximately balanced load.

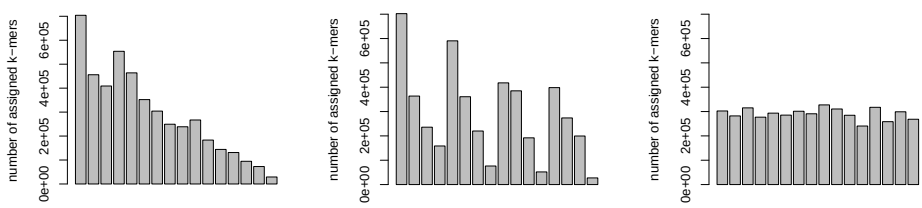

Figure 2: Distribution of all 15-mers from the genome sequence of *Yersinia pestis* CO92 (NC 003143) to 16 buckets. Left: Standard 2-bit encoding interpreted as decimal value v and assigned to bucket $\lfloor (16 \cdot v)/|K_{15}^4| \rfloor$. Center: Standard 2-bit encoding interpreted as decimal value v and assigned to bucket $v \mod 16$. Right: New encoding enc_c^r interpreted as decimal value v and assigned to bucket $\lfloor (16 \cdot v)/|R_{15}^4| \rfloor$.

5 Efficient bit-based implementation for the DNA-alphabet

Here, we consider the commonly used DNA alphabet $\{A, C, G, T\}$ with $\overline{A} := T, \overline{C} := G, \overline{G} := C, \overline{T} := A$. For the sake of convenience, we consider k being odd such that palindromes can be left aside in the following explanations and the number of canonical k-mers is $\frac{1}{2}4^k = 2^{2k-1}$.

5.1 Space efficiency

The four different rank values can be binary encoded by two bits each: $rank(A) = 0 = (00)_2, rank(C) = 1 = (01)_2, rank(G) = 2 = (10)_2$ and $rank(T) = 3 = (11)_2$. Since both the standard rank based encoding enc and the encoding for canonical k-mers enc_c^r result in k digits, any encoding is of length $2k$ bit. However, the encodings of enc_c^r only span the integer range from 0 to $|R_k^4| = 2^{2k-1}$ such that only $\log_2(2^{2k-1}) = 2k - 1$ bits are necessary. This means, the first bit of any $enc_c^r(x)$ will always be zero and can thus be omitted to store any canonical k-mer using $2k - 1$ bits.

5.2 Time efficiency

When all overlapping k-mers of a given (long) string s have to be encoded, we want to avoid processing each single character k times: for all k-mers it is contained in. Here, we provide an algorithm to compute the encoding of a k-mer $s[i..i+k-1]$ from the encoding of the previously read k-mer $s[i-1..i+k]$ in constant time, i.e., with a run time complexity independent of k, assuming k-mers fit into a constant number of computer words.

For the standard rank based encoding enc in 2-bit representation, such a rolling hash function can easily be formulated: drop the two bits corresponding to $s[i-1]$, shift $enc(s[i-1..i+k])$ by two bits to the left, and insert $enc(s[i+k-1])$ corresponding to the new last character from the right.

For the MPHF enc_c^r, a shift/update approach is not applicable, because, within a k-mer x, the encoding of a character $x[i]$ depends on its mate $x[k-i+1]$, and this pairing changes with each shift. Instead, since enc_c^r encodes many characters simply by their rank, we maintain a standard encoding (in 2-bit representation) of the current k-mer x as well as $\overline{x}$ using the constant-time shift/update function described above and modify (a copy of) $enc(x)$ to obtain $enc_c^r(x)$ using a constant number of atomic operations.

Let p denote the length of the longest common prefix of x and $\overline{x}$, i.e., the number of unspecific pairs in x. Note that $p < k/2$. This value can be computed in constant time by performing an *exclusive-or* operation on $enc(x)$ and $enc(\overline{x})$. Since the binary encoding of a character is the negation of its complement, unspecific pairs will result in a prefix of zeros, the length of which can be obtained in constant time by the operation *count leading zeros*, which (or an equivalent operation of which) belongs to the repertoire of atomic CPU instructions on modern architectures. An integer division by two yields p.

Recall that from left to right, the encoding is composed of the following four elements:

1. a prefix of $2p$ zero-bits for the characters to the left of the specifying case,
2. four bits for the specifying pair
   (or two bits for a specifying middle character),
3. the encoded remainder (characters within the specifying pair, if any),
4. the encoded characters to the right of the specifying case (if any).

In the given standard 2-bit encoding of a k-mer, element 2 can be localized with respect to p and be replaced accordingly with a constant number of operations. Elements 3 and 4 remain unchanged. It remains element 1, i.e., replacing a prefix of length $2p$ bit by a sequence of zeros. This can also be performed in constant time by using an *and* operation with a corresponding bit mask of $2p$ zeros followed by ones. These bit masks for $\lfloor k/2 \rfloor$ putative values of p can be precomputed.

The above argumentation can easily be extended to even-length k-mers and yields the following lemma.

Lemma 1. *For a k-mer x, given the encodings $enc(x)$ and $enc(\overline{x})$, the encoding $enc_c^r(x)$ can be computed in constant time, assuming $2k$ bits fit into a constant number of computer words.*

Once the first k-mer of a string and its reverse complement have been 2-bit encoded, this and all successive k-mers can be encoded by enc_c^r in time linear in the sequence length and independent from k.

Corollary 2. *All k-mers of a sequence s can be encoded by the MPHF enc_c in $O(k+|s|)$ time, assuming $2k$ bits fit into a constant number of computer words.*

6 Conclusions

We proposed a general minimal perfect hash function (MPHF) for canonical k-mers of arbitrary (even or odd) length on alphabets of arbitrary size. The approach applies to both cases: considering reverse strings or reverse complements.

Canonical k-mers are not evenly distributed within a standard rank based encoding. In particular, k-mers on the DNA alphabet are not evenly distributed within a standard 2-bit encoding. This hampers an even distribution of canonical k-mers into buckets, e.g., for an assignment to different hash tables or threads. Using an MPHF, this is easily possible. Furthermore, our MPHF has the additional property that palindromic k-mers, which are statistically observed much more often than other canonical k-mers and thus require special consideration, are mapped to a specific range of integers.

For the prevailing case of k-mers on the DNA alphabet, our approach allows to encode all k-mers of a sequence s in $O(k + |s|)$ time using a shift/update approach and efficient bit operations.

Another approach to enable an even distribution of canonical k-mers would be to stick to a 2-bit encoding but instead determine the bucket boundaries such as to ensure an even load. In current practice, random hash functions are used to achieve an approximately even load of each bucket.

In contrast to other hash functions, an MPHF renders the possibility of using array based-data structures for efficiently storing canonical k-mers. Further, since the encoding (of odd length k-mers) requires one bit fewer than a standard

2-bit encoding, it can be of use to save space when canonical k-mers are not stored in individual computer words (of even length), e.g. in [9].

An alternative MPHF for canonical k-mers proposed by the community is to define a k-mer being canonical based on the parity of its 2-bit encoding (see Section 1). This approach stands out by its simplicity, but it is highly questionable if it could be extended to k-mers of even length.

Another line of research would be efficiently encoding *homopolymer compressed* k-mers, which are k-mers in which runs of the same characters are replaced by a single copy of the characters each. This restricts the set of k-mers under consideration to $4 \cdot 3^{k-1}$ and thus, again, raising the question for an MPHF.

Acknowledgements

I thank Fabian Kolesch for pointing me to the problem, and the DSB community (in particular Antoine Limasset and Sven Rahmann) for input and motivation.

References

[1] aappleby. Smhasher. `https://github.com/aappleby/smhasher`. [Online; accessed 15-August-2022].

[2] Michael R Crusoe, Hussien F Alameldin, Sherine Awad, Elmar Boucher, Adam Caldwell, Reed Cartwright, Amanda Charbonneau, Bede Constantinides, Greg Edvenson, Scott Fay, et al. The khmer software package: enabling efficient nucleotide sequence analysis. *F1000Research*, 4, 2015. `doi:10.12688/f1000research.6924.1`.

[3] Sebastian Deorowicz, Agnieszka Debudaj-Grabysz, and Szymon Grabowski. Disk-based k-mer counting on a PC. *BMC bioinformatics*, 14(1):1–12, 2013. `doi:10.1186/1471-2105-14-160`.

[4] Antoine Limasset, Guillaume Rizk, Rayan Chikhi, and Pierre Peterlongo. Fast and scalable minimal perfect hashing for massive key sets. *arXiv preprint arXiv:1702.03154*, 2017. `doi:10.48550/arXiv.1702.03154`.

[5] Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011. `doi:10.1093/bioinformatics/btr011`.

[6] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464, 2014. `doi:10.1038/nbt.2862`.

[7] Giulio Ermanno Pibiri. Sparse and skew hashing of k-mers. *bioRxiv*, 2022. `doi:10.1093/bioinformatics/btac245`.

[8] Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3):1–12, 2014. doi:10.1186/gb-2014-15-3-r46.

[9] Jens Zentgraf and Sven Rahmann. Fast gapped k-mer counting with subdivided multi-way bucketed Cuckoo hash table. In *22nd International Workshop on Algorithms in Bioinformatics (WABI)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.WABI.2022.12.