Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 395 (2008) 220–234
Theoretical Computer Science
www.elsevier.com/locate/tcs

Counting suffix arrays and strings

Klaus-Bernd Schürmann[1], Jens Stoye*

AG Genominformatik, Technische Fakultät, Universität Bielefeld, Germany

Abstract

Suffix arrays are used in various applications and research areas like data compression or computational biology. In this work, our goal is to characterise the combinatorial properties of suffix arrays and their enumeration. For a fixed alphabet size and string length, we divide the set of all strings into equivalence classes of strings that share the same suffix array. For each such equivalence class, we count the number of strings contained in it. We also give exact formulas for computing the number of equivalence classes. Our methods yield a lower bound for the compressibility of suffix arrays and build the foundation for the efficient generation of appropriate test data sets for suffix array based algorithms. We also show that summing up the elements of all equivalence classes forms a particular instance for some summation identities of Eulerian numbers.

Keywords: Strings; Suffix arrays; Permutations

1. Introduction

In the early 1990s, Manber and Myers [1] and Gonnet *et al.* [2] introduced the suffix array as an alternative data structure to suffix trees. Since then the application of and the research on suffix arrays has advanced over the years [3–6].

In bioinformatics and text mining applications, suffix arrays with some further annotations [7] are used as an indexing structure for fast string querying [8]. Also in the data compression community, suffix arrays have received more and more attention over the last decade. This interest was initially attracted by the close relation to the Burrows–Wheeler-Transform [9], which mainly has to do with the fact that computing the Burrows–Wheeler-Transform by block-sorting the input string is equivalent to suffix array construction.

Moreover, in the last years, the task of full-text index compression emerged after Grossi and Vitter introduced the compressed suffix array [10] that reduces the space requirements to a linear number of bits. Other compressed indices of that type are Ferragina and Manzini's FM-index [11] based on the Burrows–Wheeler-Transform, a compressed suffix array based index by Sadakane [12] that does not use the text itself, and Mäkinen's compact suffix array [13], just to mention a few. Lower bounds for the size of such indices are known. Demaine and López-Ortiz [14] proved a

* Corresponding author.
E-mail addresses: Klaus-Bernd.Schuermann@CeBiTec.Uni-Bielefeld.de (K.-B. Schürmann), stoye@TechFak.Uni-Bielefeld.de (J. Stoye).
[1] AG Genominformatik, Technische Fakultät, Universität Bielefeld, Postfach 10 01 31, 33501 Bielefeld, Germany.

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2008.01.011

lower bound for indices providing substring search, and Miltersen [15] showed lower bounds for selection and rank indices. For an in-depth study of compressed full-text indices and their space requirements, we refer to the survey of Navarro and Mäkinen [16].

However, all these developments on compressed indices trade space occupancy for querying time. Redundant information, which would have been necessary for more efficient querying, is lost when compressing an original base index, as it is with the suffix array. We believe that a profound knowledge of the algebraic and combinatorial properties of suffix arrays is essential to develop suffix array based, succinct indices that allow efficient querying.

In 2002, Duval and Lefebvre [17] characterised the set of strings that share the same suffix array. Furthermore, Crochemore *et al.* [18] recently presented combinatorial properties of the related Burrows–Wheeler transformation, but these properties are unassignable to suffix arrays. They rely on the fact that the Burrows–Wheeler transform is based on the order of cyclic shifts of the input sequence, whereas the suffix array is based on the order of suffixes cut at the end of the string, which destroys that nice group structure. A combinatorial approach that partly includes suffix arrays was presented by Hohlweg and Reutenauer [19]. They study connections between binary planary trees, Lyndon words, and suffix arrays.

Most suffix array applications face strings with a small, fixed alphabet like the DNA, amino acid, or ASCII alphabet. The possible suffix arrays for such strings are just a small fraction of all possible permutations. Therefore, besides discovering their combinatorial structure, our goal is to enumerate the different suffix arrays for strings over a fixed size alphabet. Our work is based upon results by Burkhardt and Kärkkäinen [20] and Bannai *et al.* [21]: The former gave a characterisation of strings and their suffix array. The latter presented an algorithm that outputs a string with minimal alphabet size with a known input suffix array; and moreover, they stated a formula for the number of certain suffix arrays.

This paper is an extended version of [22]. In Section 2 we give the basic definitions and notations regarding alphabets, strings, permutations, and suffix arrays. In Section 3 we prove an important characterisation of suffix arrays, which is the foundation for the subsequent counting schemes. Strings that are equivalent in the sense that they share the same suffix array are counted in Section 4. The number of such equivalence classes is computed in Section 5. Applications of the suffix array counting and results for the compressibility of suffix arrays are presented in Section 6. Section 7 combines the counting schemes of Sections 4 and 5 to prove identities by summing up over suffix arrays and their strings, and Section 8 concludes.

2. Strings, permutations, and suffix arrays — definitions and terminology

The interval $[g, h] = \{z \in \mathbb{Z} : g \leq z \leq h\}$ denotes the set of all integers greater than or equal to g and less than or equal to h.

Alphabet and strings. Let Σ be a finite set of size $|\Sigma|$, the *alphabet*, and $t \in \Sigma^n$ a string over Σ of length n, the *text*. For $i \in [1, n]$, $t[i]$ denotes the ith character of t, and for all pairs of indices (i, j), $1 \leq i \leq j \leq n$, $t[i, j] = t[i], t[i + 1], \ldots, t[j]$ denotes the substring of t starting at position i and ending at position j. Substrings $t[i, n]$ ending at position n are *suffixes* of t. The starting position i of a suffix $t[i, n]$ is called its *suffix number*. $\Sigma(t) = \{t[i] : 1 \leq i \leq n\} \subseteq \Sigma$ is the subset of characters actually occurring in t and is called the *character set* of t. We usually use σ for the alphabet size, but if the strings are required to use all characters such that their character set equals the alphabet, we use κ.

We deal with different kinds of equivalences of strings. The natural definition is that strings are *(string-)equivalent* if they are equal, and *(string-)distinct* otherwise. In order to define the other two equivalences, we first introduce a bijective mapping m of the characters of a string t to the first $|\Sigma(t)|$ integers, $m : \Sigma(t) \longrightarrow [1, |\Sigma(t)|]$. We call m *order-preserving* if $c_1 < c_2 \Leftrightarrow m(c_1) < m(c_2)$ for all pairs of characters $(c_1, c_2) \in \Sigma(t) \times \Sigma(t)$. The mapped string $m(t)$ is then defined by $m(t) := m(t[1])m(t[2]) \ldots m(t[n])$. We call two strings t_1 and t_2 *order-equivalent*, if there exists an order-preserving bijection m_1 for t_1 and another such bijection m_2 for t_2 such that $m_1(t_1) = m_2(t_2)$; otherwise the strings are *order-distinct*. If mappings m_1 and m_2 exist such that $m_1(t_1) = m_2(t_2)$ (not necessarily order-preserving), we call t_1 and t_2 *pattern-equivalent*; otherwise the strings are *pattern-distinct*. String-equivalent strings are also order-equivalent and order-equivalence implies pattern-equivalence. The strings `AT` and `AG`, for example, are string-distinct but order-equivalent, and the strings `AG` and `GA` are order-distinct but pattern-equivalent.

Permutations and suffix arrays. Let P be a permutation of $[1, n]$. Then $i \in [1, n-1]$ is a *permutation descent* if $P[i] > P[i+1]$. Conversely, a non-extendable ascending segment $P[i] < P[i+1] < \cdots < P[j]$ of P is called a *permutation run*, denoted by the index pair (i, j). Each permutation run of P is bordered by permutation descents or the permutation boundaries 1 or n. Hence, the permutation runs define the permutation descents and vice versa.

The *suffix array* $sa(t)$ of t is a permutation of the suffix numbers $\{1, \ldots, n\}$ according to the lexicographic ordering of the n suffixes of t. More precisely, a permutation P of $[1, n]$ is the suffix array for a string t of length n if for all pairs of indices (i, j), $1 \le i < j \le n$, the suffix with suffix number $P[i]$ is lexicographically smaller than the suffix with suffix number $P[j]$.

The *rank array* R_P for the permutation P (further on simply denoted by R), sometimes called the inverse suffix array, is defined as follows: For all indices $i \in [1, n]$ the rank of i is j, $R[i] = j$, if i occurs at position j in the permutation, $P[j] = i$. We extend the rank array by $R[n+1] = 0$, indicating that the empty suffix, not contained in the suffix array, is always the lexicographically smallest.

Further on, we define the R_+-array to be $R_+[i] = R[P[i]+1]$ for all $i \in [1, n]$. In the compressed indexing literature the R_+-array is usually called Ψ-array, or alternatively, Ψ-function. We define the R_+-descents and the R_+-runs of P similar to the permutation descents and the permutation runs respectively: A position $i \in [1, n-1]$ is called an *R_+-descent* if $R_+[i] > R_+[i+1]$. A non-extendable ascending segment $R[P[i]+1] < R[P[i+1]+1] < \cdots < R[P[j]+1]$, denoted by the index pair (i, j), $i \le j$, is called an *R_+-run*. The set of R_+-descents $\{i \in [1, n-1] : R[P[i]+1] > R[P[i+1]+1]\}$ is denoted by $R_+\text{-}desc(P)$. If the ordered set of R_+-descents of P equals $\{i_1, i_2, \ldots, i_d\}$, $i_j < i_{j+1}$ for all $j \in [1, d-1]$, then the set of R_+-runs $\{(1, i_1), (i_1+1, i_2), \ldots, (i_{d-1}+1, i_d), (i_d+1, n)\}$ is denoted by $R_+\text{-}runs(P)$. Note that R_+-runs can be of length 1.

Further definitions. Besides the binomial coefficient $\binom{x}{y} = \frac{x!}{y!(x-y)!}$, combinatorial objects related to permutations that are important for this work are the Stirling numbers and the Eulerian numbers. Although these numbers have a venerable history, their notation is less standard. We will follow the notation of Graham *et al.* [23, chap. 6] where the Stirling number of the second kind $\left\{ {n \atop k} \right\}$ stands for the number of ways to partition a set of n elements into k non-empty subsets, and the Eulerian number $\left\langle {n \atop d} \right\rangle$ gives the number of permutations of $[1, n]$ that have exactly d permutation descents, also defined through the recursion (i) $\left\langle {n \atop 0} \right\rangle = 1$, (ii) $\left\langle {n \atop d} \right\rangle = 0$ for $d \ge n$, and (iii) $\left\langle {n \atop d} \right\rangle = (d+1)\left\langle {n-1 \atop d} \right\rangle + (n-d)\left\langle {n-1 \atop d-1} \right\rangle$ for $0 < d < n$.

3. Characterising strings sharing the same suffix array

We repeat a characterisation of the set of strings sharing the same suffix array P, which states that the order of consecutive suffixes in the suffix array is determined by their first character and by the order of suffixes without their first character (Theorem 1). This result was previously given, without proof, by Burkhardt and Kärkkäinen [20] and equivalent characterisations were proved by Duval and Lefebvre [17].

Theorem 1. *Let P be a permutation of $[1, n]$ and t a string of length n. P is the suffix array of t if and only if for all $i \in [1, n]$ the following two conditions hold:*

(a) $t[P[i]] \le t[P[i+1]]$ *and*
(b) $R[P[i]+1] > R[P[i+1]+1] \Rightarrow t[P[i]] < t[P[i+1]]$.

Theorem 1 suggests criteria to divide the strings into equivalence classes according to their suffix array. These will be counted in Section 5.

3.1. Proof of Theorem 1

We first generalise a proposition about consecutive elements in a permutation to arbitrary pairs of elements.

Lemma 2. *Let P be a permutation of $[1, n]$ and t a string of length n.*
If for all $i \in [1, n-1]$ we have that

(a) $t[P[i]] \le t[P[i+1]]$ *and*
(b) $t[P[i]] = t[P[i+1]] \Rightarrow R[P[i]+1] < R[P[i+1]+1]$,

then we also have that for all pairs (i, j), $1 \leq i < j \leq n$,

$$t[P[i]] = t[P[j]] \Rightarrow R[P[i]+1] < R[P[j]+1].$$

Proof. Due to (a), the sequence of characters $t[P[i]], t[P[i+1]], \ldots, t[P[j]]$ is non-decreasing. Combining this property with $t[P[i]] = t[P[j]]$ implies that $t[P[i']] = t[P[i'+1]]$ for all $i' \in [i, j-1]$. Then (b) applied to $t[P[i']] = t[P[i'+1]]$ leads us to $R[P[i']+1] < R[P[i'+1]+1]$ for all $i' \in [i, j-1]$ and finally by transitivity we obtain $R[P[i]+1] < R[P[j]+1]$. □

Before we can prove the main result of this section, we continue with a further generalisation. We extend our proposition from elements of the permutation referring to equal characters in the string to elements referring to starting positions of equal substrings.

Lemma 3. *Let* P *be a permutation of* $[1, n]$ *and* t *a string of length* n. *If for all* $i,\ j \in [1, n]$ *with* $i < j$ *we have that*

$$t[P[i]] = t[P[j]] \quad \Rightarrow \quad R[P[i]+1] < R[P[j]+1], \tag{1}$$

then we also have that for all $i,\ j \in [1, n]$ *with* $i < j$ *and for all* $k > 0$

$$t[P[i], P[i]+k-1] = t[P[j], P[j]+k-1] \Rightarrow R[P[i]+k] < R[P[j]+k]. \tag{2}$$

Proof (Induction over *k*). For $k = 1$ the equation $t[P[i], P[i]+1-1] = t[P[j], P[j]+1-1]$ accords to $t[P[i]] = t[P[j]]$; and hence, implication (2) accords to implication (1).

We now perform the induction step starting with

$$t[P[i], P[i]+k] = t[P[j], P[j]+k],$$

which obviously implies

$$t[P[i], P[i]+k-1] = t[P[j], P[j]+k-1] \tag{3}$$

and $$t[P[i]+k] = t[P[j]+k]. \tag{4}$$

Applying the induction hypothesis (2) and (3) gives $R[P[i]+k] < R[P[j]+k]$. Then we choose i' and j' such that $P[i'] = P[i]+k$ and $P[j'] = P[j]+k$. Since R is the inverse of P, we obtain

$$i' = R[P[i']] = R[P[i]+k] < R[P[j]+k] = R[P[j']] = j'. \tag{5}$$

Combining Eq. (4) with $P[i'] = P[i]+k$ and $P[j'] = P[j]+k$ implies

$$t[P[i']] = t[P[i]+k] = t[P[j]+k] = t[P[j']].$$

By (5) i' is smaller than j', so implication (1) is applicable and leads to

$$R[P[i']+1] < R[P[j']+1].$$

Substituting $P[i']$ by $P[i]+k$ and $P[j']$ by $P[j]+k$ results in $R[P[i]+k+1] < R[P[j]+k+1]$, completing the proof. □

We are now ready for proving Theorem 1.

Proof of Theorem 1. If the permutation P is the suffix array for the string t, then the conditions (a) and (b) of the theorem clearly hold.

The opposite direction is more intricate. If P is not the suffix array for t, then there must exist two wrongly ordered suffixes in P. Assume the positions of these two suffixes are i and j such that $i < j$ and $t[P[i], n] > t[P[j], n]$.

Negating (b) produces for all $i \in [1, n-1]$

$$t[P[i]] \geq t[P[i+1]] \Rightarrow R[P[i]+1] \leq R[P[i+1]+1],$$

and by (a) and by the fact that R as well as P are different at unequal positions, we obtain for all $i \in [1, n-1]$

$$t[P[i]] = t[P[i+1]] \Rightarrow R[P[i]+1] < R[P[i+1]+1].$$

Table 1
Summary of the previous and new results on the number of string-distinct, order-distinct and pattern-distinct strings of length n

Number of	string-distinct	order-distinct	pattern-distinct
Strings with exactly κ letters	$\left\{{n \atop \kappa}\right\} \cdot \kappa!$	$\left\{{n \atop \kappa}\right\} \cdot \kappa!$	$\left\{{n \atop \kappa}\right\}$ [24]
Strings for alphabet size σ	σ^n	$\sum_{\kappa=1}^{\sigma} \left\{{n \atop \kappa}\right\} \cdot \kappa!$	$\sum_{\kappa=1}^{\sigma} \left\{{n \atop \kappa}\right\}$
String with exactly κ letters sharing the same suffix array	$\binom{n-d-1}{\kappa-d-1}$ [Theorem. 5]	$\binom{n-d-1}{\kappa-d-1}$	–
Strings for alphabet size σ sharing the same suffix array	$\binom{n+\sigma-d-1}{\sigma-d-1}$ [Theorem 4]	$\sum_{\kappa=d+1}^{\sigma} \binom{n-d-1}{\kappa-d-1}$	–

In the analyses d is always the number of R_+-descents for the respective suffix array.

We apply Lemma 2 and Lemma 3 to obtain for all $i, j \in [1, n], i < j$,

$$t[P[i], P[i]+k-1] = t[P[j], P[j]+k-1] \Rightarrow R[P[i]+k] < R[P[j]+k]. \tag{6}$$

Now let l be the longest common prefix of $t[P[i], n]$ and $t[P[j], n]$, then we distinguish between two cases.

(i) If $l = 0$, the suffixes differ in their first position. Since $t[P[i], n] > t[P[j], n]$, the first character $t[P[i]]$ of $t[P[i], n]$ must be greater than the first character $t[P[j]]$ of $t[P[j], n]$, which contradicts (a).

(ii) If $l > 0$, the suffixes $t[P[i], n]$ and $t[P[j], n]$ share a longest common prefix of length l, that is, $t[P[i], P[i]+l-1] = t[P[j], P[j]+l-1]$. Then, implication (6) leads to $R[P[i]+l] < R[P[j]+l]$. We choose i' and j' such that $P[i'] = P[i]+l$ and $P[j'] = P[j]+l$. Since R is the inverse of P, we have $i' = R[P[i']] = R[P[i]+l] < R[P[j]+l] = R[P[j']] = j'$. Therefore, using (a) we obtain

$$t[P[i]+l] = t[P[i']] \leq t[P[j']] = t[P[j]+l]. \tag{7}$$

But the assumption was that $t[P[i], n] > t[P[j], n]$ with longest common prefix l such that $t[P[i]+l] > t[P[j]+l]$, which contradicts inequality (7).

Since both cases lead to contradictions, all suffixes represented in P must be in the correct order; hence P is the suffix array for t. □

4. Counting the strings per suffix array

In this section, we count the strings over a fixed sized alphabet all sharing the same suffix array P considering particular subsets of strings: string-distinct strings composed of up to $|\Sigma|$ distinct characters and string-distinct strings where each character of the alphabet must appear.

Theorem 4. *Let P be a permutation of length n with d R_+-descents and Σ an alphabet of $\sigma = |\Sigma|$ ordered symbols. The number of strings over Σ with suffix array P is $\binom{n+\sigma-d-1}{\sigma-d-1}$.*

Theorem 5. *Let P be a permutation of length n with d R_+-descents. The number of strings with suffix array P composed of exactly κ different characters is $\binom{n-d-1}{\kappa-d-1}$.*

Theorem 4 is proved in Section 4.2 and Theorem 5 in Section 4.3.

For the various settings, Table 1 summarises the number of string-distinct, order-distinct, and pattern-distinct strings of length n. Some of the numbers were proven by other authors or given by Theorems 4 and 5. We start with the first row. Moore *et al.* [24] showed that the number of pattern-distinct strings composed of exactly κ different characters is $\left\{{n \atop \kappa}\right\}$. For each pattern-distinct string, we permute the alphabet in $\kappa!$ different ways to get a total of $\left\{{n \atop \kappa}\right\}\kappa!$ order-distinct strings. These are already all the string-distinct strings since we have no flexibility to choose different characters to produce string-distinct strings that are yet order-equivalent.

The numbers of strings over a given alphabet of size σ are shown in the second row. Needless to say, we have σ^n string-distinct strings. For the order- and pattern-distinct strings, we just sum up the number of strings for all possible κ.

The number of string-distinct strings composed of exactly κ different characters sharing a suffix array P with d R_+-descents was given in Theorem 5. All these strings are again order-distinct. For pattern-distinct strings, we cannot necessarily determine a unique suffix array. This is indicated by a dash in the table. AG and GA, for example, are pattern-equivalent, but have different suffix arrays.

The number of string-distinct and order-distinct strings over an alphabet of size σ sharing the same suffix array are given in the fourth row. Theorem 4 gave the number of string-distinct strings and to count the order-distinct strings we just sum up over all possible κ.

4.1. Foundations for the subsequent string counting

Before we prove Theorem 4 in Section 4.2 and Theorem 5 in Section 4.3, we first repeat an observation of Bannai *et al.* [21] that links the minimal alphabet size of the strings with suffix array P to the number of R_+-descents of P. For a permutation P with d R_+-descents the number of different characters in a string t with suffix array P is at least the number of R_+-descents plus one, $|\Sigma(t)| \geq d + 1$. They also presented an algorithm to construct a unique string b_P consisting of exactly $d + 1$ different characters, $|\Sigma(b_P)| = d + 1$.

W.l.o.g., we assume that the character set of b_P contains the smallest natural numbers, $\Sigma(b_P) = [1, d + 1]$, and call b_P the *base string* of the suffix array P. The algorithm suggested in [21] works as follows. It starts with the initial character $c = 1$. For each index position $i \in [1, n]$ in ascending order, the algorithm proceeds through all suffix numbers from $P[1]$ to $P[n]$ by assigning c to $b_P[P[i]]$. If i is an R_+-descent, c is incremented by one to satisfy condition (2) of Theorem 1, such that $b_P[P[i]] = d_i + 1$ where d_i is the number of R_+-descents in the prefix $P[1, \ldots, i]$ of the suffix array P.

Remark 6. Let P be a permutation with d R_+-descents, then the base string b_P has the properties

(a) $b_P[P[1]] = 1$ and $b_P[P[n]] = d + 1$,
(b) $b_P[P[i]] = b_P[P[i + 1]]$ if $i \in [1, n - 1]$ is not an R_+-descent of P,
(c) $b_P[P[i]] + 1 = b_P[P[i + 1]]$ if $i \in [1, n - 1]$ is an R_+-descent of P.

4.2. Counting strings composed of up to σ distinct characters

Strings sharing the same suffix array P of length n can be derived from the base string for the suffix array by applying a certain sequence of rewrite-operations to the base string, after which the order of suffixes remains untouched. The sequence of rewrite-operations starts with the largest suffix. Increasing the first character of the largest suffix by r does not change the order of suffixes. Then, the first character of the second largest suffix can be increased by at most r without changing the order of suffixes, and so on.

Definition 7. Let P be a permutation of $[1, n]$ with base string b_P. Moreover, let m be a sequence of length n of numbers from $[0, \psi]$, for some $\psi \in \mathbb{N}$. The *m-incremented sequence* $s_{P,m}$ of P is defined as

$$s_{P,m}[P[i]] = b_P[P[i]] + m[i] \quad \text{for all } i \in [1, n].$$

We reveal a relationship between the sequences sharing the same suffix array and non-decreasing sequences.

Lemma 8. *Let P be a permutation of $[1, n]$ with d R_+-descents and $\mathcal{S}_{P,\Sigma}$ the set of sequences over the ordered alphabet Σ, $\sigma = |\Sigma|$, with suffix array P. Moreover, let $\mathcal{M}$ be the set of non-decreasing sequences of length n over the ordered alphabet $[0, \sigma - d - 1]$.*

There exists an isomorphism between $\mathcal{S}_{P,\Sigma}$ and $\mathcal{M}$.

Proof. Let b_P be the base string for permutation P. W.l.o.g., we assume $\Sigma = [1, \sigma]$. We show: (i) for each non-decreasing sequence $m \in \mathcal{M}$, the corresponding m-incremented string $s_{P,m}$ has the suffix array P and $s_{P,m} \in \Sigma^n$ and (ii) each other sequence o of length n, $o \notin \mathcal{M}$, produces a string $s_{P,o}$ for which P is not the suffix array or $s_{P,m} \notin \Sigma^n$.

(i) Let $m \in \mathcal{M}$, such that $m[i] \leq m[i + 1]$ for all $i \in [1, n - 1]$. We verify the conditions of Theorem 1 for $s_{P,m}$:
  (ia) For all $i \in [1, n - 1]$, $b_P[P[i]] \leq b_P[P[i + 1]]$. That implies
  $$s_{P,m}[P[i]] = b_P[P[i]] + m[i] \leq b_P[P[i + 1]] + m[i + 1] = s_{P,m}[P[i + 1]],$$
  verifying Theorem 1(a).

(ib) If $R_+[i] > R_+[i+1]$ then $i \in R_+\text{-desc}(P)$. Hence, $b_P[P[i]]+1 = b_P[P[i+1]]$, which leads to

$$\begin{aligned} s_{P,m}[P[i]] &= b_P[P[i]] + m[i] \\ &< (b_P[P[i]]+1) + m[i] \\ &\leq b_P[P[i+1]] + m[i+1] = s_{P,m}[P[i+1]], \end{aligned}$$

verifying Theorem 1(b).

Therefore, P is the suffix array for $s_{P,m}$.

Moreover, for each position j of $s_{P,m}$ with $j = P[i]$ for some $i \in [1,n]$,

$$s_{P,m}[j] = s_{P,m}[P[i]] = b_P[P[i]] + m[i] \leq (d+1) + (\sigma - d - 1) = \sigma$$

and analogously $1 \leq s_{P,m}[j]$. Hence, each $m \in \mathcal{M}$ produces a sequence $s_{P,m} \in \Sigma^n$ with suffix array P.

(ii) For $o \notin \mathcal{M}$ containing a descending adjacent index pair such that $o[i] > o[i+1]$ for some $i \in [1, n-1]$, we concern ourselves with two cases:

(iia) If i is not an R_+-descent in P then $b_P[P[i]] = b_P[P[i+1]]$. Hence,

$$s_{P,o}[P[i]] = b_P[P[i]] + o[i] > b_P[P[i+1]] + o[i+1] = s_{P,o}[P[i+1]],$$

which contradicts Theorem 1(a).

(iib) If $R_+[i] > R_+[i+1]$ then $i \in R_+\text{-desc}(P)$. Thus, $b_P[P[i]] = b_P[P[i+1]]-1$ and, because of $o[i] > o[i+1]$, also $o[i] \geq o[i+1]+1$ is true. This results in

$$\begin{aligned} s_{P,o}[P[i]] &= b_P[P[i]] + o[i] \\ &\geq (b_P[P[i+1]] - 1) + (o[i+1] + 1) \\ &= b_P[P[i+1]] + o[i+1] \\ &= s_{P,o}[P[i+1]], \end{aligned}$$

which contradicts Theorem 1(b).

Therefore, only the non-decreasing sequences m produce a string $s_{P,m}$ with suffix array P.

The non-decreasing sequences $o \notin \mathcal{M}$, for which $\Sigma(o) \not\subseteq [0, \sigma - d - 1]$, remain. For all these strings we show that $s_{P,o} \notin \Sigma^n$. If o is non-decreasing but not in $\mathcal{M}$, it must contain, at some position i, a character greater than $\sigma - d - 1$ or smaller than 0. Since o is non-decreasing, this character appears at position n or 1. That is, $o[n] > \sigma - d - 1$ or $o[1] < 0$. Combining $o[n] > \sigma - d - 1$ with the fact from Remark 6(a) that $b_P[P[n]] = d+1$ implies

$$s_{P,o}[P[n]] = b_P[P[n]] + o[n] > (d+1) + (\sigma - d - 1) = \sigma.$$

Using $b_P[P[1]] = 0$ for $o[1] < 0$ analogously implies $s_{P,o}[P[1]] < 0$. Thus, $s_{P,o} \notin \Sigma^n$, completing the proof. □

Finally, the number of sequences over σ characters with the same suffix array P is the same as the number of non-decreasing sequences over $\sigma - d$ characters.

To count the number of non-decreasing sequences of length n composed of a elements, we observe the following:

Lemma 9. *Let $M(n,a)$ be the number of non-decreasing sequences of length n of elements in $[0, a-1]$. For any positive integers n and a*

$$M(n,a) = \binom{n+a-1}{a-1}.$$

Proof. The non-decreasing sequences of length n composed of a symbols can be modeled as a sequence of two different operations. Initially, the current symbol is set to 0. Then, we apply a sequence of operations to generate non-decreasing sequences of length n. One possible operation is to write the current symbol behind the so far written symbols and the other one is to increment the symbol by 1. To generate a non-decreasing sequence, we apply $n+a-1$ operations, n to write down the non-decreasing sequence and $a-1$ to increment the current symbol until $a-1$ is reached. For this sequence of length $n+a-1$, we have $\binom{n+a-1}{a-1}$ possibilities to choose the $a-1$ positions of the increment operations. □

When applying this observation to Lemma 8 we get the number of strings sharing the same suffix array.

Proof of Theorem 4. The claim follows directly from the bijection shown in Lemma 8 and the equality $M(n, \sigma-d) = \binom{n+\sigma-d-1}{\sigma-d-1}$ from Lemma 9. □

The non-decreasing sequences of length n over $[0, \sigma - d - 1]$ can simply be enumerated in-place by applying one change operation at a time, beginning with the sequence 0^n. The bijection described by Definition 7 suggests to apply these enumeration steps directly to the base string of a certain suffix array. In this way, we can enumerate all $|\mathcal{S}_{P,\Sigma}|$ strings over a given alphabet Σ for a certain suffix array P in optimal $O(n + |\mathcal{S}_{P,\Sigma}|)$ time, where n steps are used to construct the base string.

4.3. Counting strings composed of exactly κ distinct characters.

So far, we have counted all the strings over a fixed alphabet that share the same suffix array. Now, we count the subset of such strings composed of exactly κ different characters.

Proof of Theorem 5. The proof works similar as for Theorem 4. Obviously, we have to count each non-decreasing sequence m in $\mathcal{M}$ for which $s_{P,m}$ consists of exactly κ letters. To assure that none of the κ characters $[1, \kappa]$ is left out, it is sufficient to count all $m \in \mathcal{M}$ such that $s_{P,m}[P[1]] = 0$, $s_{P,m}[P[n]] = \kappa$, and consecutive characters in the resulting sequence $s_{P,m}$ are not differing by more than one. This property is realised by a sequence m, if and only if,

(a) $m[1] = 0$ and $m[n] = \kappa - d - 1$,
(b) $m[i] = m[i+1]$ or $m[i] + 1 = m[i+1]$ if $i \notin R_+\text{-desc}(P)$, and
(c) $m[i] = m[i+1]$ if $i \in R_+\text{-desc}(P)$.

We again represent these kind of non-decreasing sequences as n write operations and $a - 1$ increment operations, as it has been modelled above. Here, for the placement of the $\kappa - d - 1$ increment operations, we are restricted by the mentioned conditions. In order not to break these conditions, (a) an increment operation must not appear before the first or after the last write operation, (b) at most one increment operation must appear between two write operations, and (c) the d R_+-descent positions are blocked for the increments. We are thus left with $n - d - 1$ mutually exclusive positions from which we choose $\kappa - d - 1$ increment operations. □

5. Counting suffix arrays for strings with a fixed alphabet

In this section, the distinct suffix arrays for strings over a fixed size alphabet are counted. We first confine ourselves to the equivalent problem of counting the number of suffix arrays with a fixed number of R_+-descents.

Bannai *et al.* [21] stated that the number of suffix arrays of length n with exactly d R_+-descents is equal to the Eulerian number $\left\langle {n \atop d} \right\rangle$. In their explanation, they interpret Eulerian numbers as the number of permutations of length n with d permutation descents and explain how their algorithm checks for these permutation descents. In fact, their algorithm counts the number of R_+-descents, but the R_+-array is not a permutation. Nevertheless, as we show in this section, their proposition is true.

Theorem 10. *Let $A(n, d)$ be the number of permutations of length n with d R_+-descents, then*

$$A(n, d) = \left\langle {n \atop d} \right\rangle.$$

Bannai *et al.* [21] also showed that each suffix array with d R_+-descents can be associated with a string of at least $d + 1$ different characters. Therefore, for strings over an alphabet of size σ, we sum up the suffix arrays with up to $\sigma - 1$ R_+-descents to obtain the overall number of suffix arrays.

Corollary 11. *Let Σ be a fixed size alphabet, $\sigma = |\Sigma|$. The number of distinct suffix arrays of length n for strings over Σ is $\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle$.*

5.1. Proof of Theorem 10

We first define a mapping of a permutation P of length $n-1$ to a set $\mathcal{P}'$ of successor permutations, each of length n. This definition is the key for the further successive reasoning throughout Lemmas 13–17 leading to Theorem 10.

The mapping has the following property. Let b_P the base string of P. If b_P is extended by one character $\mathtt{C}$ to the left, then the suffix array $sa(\mathtt{C}b_P)$ of the extended string $\mathtt{C}b_P$ is a successor permutation of P, $sa(\mathtt{C}b_P) \in \mathcal{P}'$.

Definition 12. Let P be a permutation of length $n-1$. A set of successor permutations $\mathcal{P}'$ of P is defined as $\mathcal{P}' = \{P'_i : i \in [1,n]\}$ where P'_i evolves from P by incrementing each element of P by one and inserting the missing 1 at position i, such that each position j in P corresponds to a position j' in P'_i:

$$j' = j, \qquad \text{if } j < i.$$
$$\text{and} \qquad j' = j+1, \qquad \text{if } j \geq i,$$

and

$$P'_i[j'] = P[j]+1, \qquad \text{if } j' \neq i$$
$$\text{and} \qquad P'_i[j'] = 1, \qquad \text{if } j' = i.$$

The insertion at position i shifts the elements at positions j with $j \geq i$ to the right, resulting in an increased rank for the respective elements of P'_i.

Lemma 13. *Let P be a permutation of length $n-1$ and $P' = P'_i$ a successor of P with insertion position i, then we have for all $e \in [1, n-1]$ that*

(a) $R'[e+1] = R[e]$ *if* $R[e] < i$,
(b) $R'[e+1] = R[e]+1$ *if* $R[e] \geq i$, *and*
(c) $R'[1] = i$.

Proof. Let e be an arbitrary element of the permutation P occurring at position j, $e = P[j]$ and $R[e] = j$.

(a) If $R[e] < i$ then $j = R[e] < i$. Therefore, according to Definition 12, j' equals j and hence $P'[j'] = P[j]+1 = e+1$. Altogether, this implies $R'[e+1] = R'[P'[j']] = j' = j = R[e]$.
(b) If $R[e] \geq i$ then $j = R[e] \geq i$. Therefore, $j' = j+1$ and $P'[j'] = P[j]+1 = e+1$. This implies $R'[e+1] = R'[P'[j']] = j' = j+1 = R[e]+1$.
(c) $R'[1] = i$ holds because 1 is inserted at position i, $P'[i] = 1$.

In this way, the insertion position i determines the rank array R' of the successor permutation. □

Furthermore, mapping P to P' basically preserves the R_+-order:

Lemma 14. *Let P be a permutation of length $n-1$ with successor P'.*
For all indices g and h, $g, h \in [1, n-1]$,

$$R_+[g] < R_+[h] \Longrightarrow R'_+[g'] < R'_+[h'].$$

Proof. Let g and h be some positions of P such that $R_+[g] < R_+[h]$. Then, according to the definition of R_+, $R[P[g]+1] < R[P[h]+1]$. We distinguish two cases.

(i) If $R[P[g]+1] < i$ then Lemma 13(a) and (b) gives

$$R'[P[g]+1+1] = R[P[g]+1] < R[P[h]+1] \leq R'[P[h]+1+1].$$

Combining this with Definition 12 and the definition of R'_+ yields

$$R'_+[g'] = R'[P'[g']+1] < R'[P'[h']+1] = R'_+[h'].$$

(ii) If $R[P[g]+1] \geq i$, then the proof works analogously using the fact that $R[P[h]+1] > R[P[g]+1] \geq i$. Hence, Lemma 13(b) has to be used for $R[P[g]+1]$ as well as for $R[P[h]+1]$ and the rest of the proof proceeds as before.

Thus, except for the insertion position i, the R_+-order of P determines the R_+-order of P'. □

Lemma 14 considers the R_+-order of P', but leaves out the insertion position i. The next lemma states that the R_+-order at position i just depends on the position $R[1]$ of element 1 in the permutation P.

Lemma 15. *Let P' be a successor of P with insertion position i and g an index of P, then*

$$R_+[g] < R[1] \Longleftrightarrow R'_+[g'] < R'_+[i] \qquad \text{for all } g \in [1, n-1].$$

Proof. We first show that $R_+[g] < R[1] \Longrightarrow R'_+[g'] < R'_+[i]$.

If $R_+[g] < R[1]$ then using the definition of R_+ leads to $R[P[g]+1] < R[1]$. We consider two cases.

(i) If $R[P[g]+1] < i$ then by Lemma 13(a) we have $R'[P[g]+1+1] = R[P[g]+1]$. Moreover, Lemma 13(a) and (b) implies $R[1] \leq R'[1+1]$. This together leads to

$$R'[(P[g]+1)+1] < R'[1+1]. \tag{8}$$

According to Definition 12, $P'[g'] = P[g]+1$ and $P'[i] = 1$. Combining this and inequality (8) leads to

$$R'_+[g'] = R'[P'[g']+1] = R'[(P[g]+1)+1] < R'[1+1] = R'[P'[i]+1] = R'_+[i].$$

(ii) If $R[P[g]+1] \geq i$ then the proof proceeds analogously by considering $R[1] > R[P[g]+1] \geq i$.

In order to show that $R_+[g] < R[1] \Longleftarrow R'_+[g'] < R'_+[i]$, we observe that $R_+[g] > R[1] \Longrightarrow R'_+[g'] > R'_+[i]$. Since, for all $g \in [1, n-1]$, $R_+[g] \neq R[1]$ and $R'_+[g'] \neq R'_+[i]$, we obtain the stated equivalence. □

After characterising the R_+-order of successor permutations, we now prove that through the mapping from P to an arbitrary successor permutation the number of R_+-descents is preserved or increased by one.

Lemma 16. *Let P be a permutation of length $n-1$ with d R_+-descents and $\mathcal{P}'$ the set of successor permutations for P, then for all successor permutations $P'_i \in \mathcal{P}'$ we have*

$$|\mathrm{desc}(P)| \leq |\mathrm{desc}(P'_i)| \leq |\mathrm{desc}(P)| + 1.$$

Proof. According to Lemma 14, the mapping with respect to the insertion position i does not touch the R_+-order of consecutive positions not adjacent to i. More precisely, for all $j \in [2, n-1]$ with $j \neq i$

$$R_+[j-1] > R_+[j] \Longleftrightarrow R'_+[(j-1)'] > R'_+[j']. \tag{9}$$

That means, each R_+-descent at position $j-1$ with $j \neq i$ corresponds to an R_+-descent at position $(j-1)'$ in P'_i and vice versa. Therefore we just have to examine the R_+-order of the remaining pair of positions $(i-1, i)$ in P and the respective interval $[(i-1)', i']$ of P'_i. Note that $[(i-1)', i'] = \{i-1, i, i+1\}$. We distinguish the two cases that either position $i-1$ of P is an R_+-descent, or not.

(i) If $i-1$ is an R_+-descent of P so that $R_+[i-1] > R_+[i]$, then applying Lemma 14 leads to

$$R'_+[(i-1)'] > R'_+[i']. \tag{10}$$

Since $R[1] \neq R_+[f]$ for all $f \in [1, n-1]$, we consider three subcases:

(i.1) If $R[1] > R_+[i-1]$ then Lemma 15 implies $R'_+[i] > R'_+[(i-1)']$ and together with inequality (10) $R'_+[i] > R'_+[(i-1)'] > R'_+[i']$ follows. That is, $R'_+[(i-1)'] < R'_+[i]$ and $R'_+[i] > R'_+[i']$. Hence, i is an R_+-descent of P'_i and the number of R_+-descents of P'_i equals the number of R_+-descents of P.

(i.2) If $R_+[i-1] > R[1] > R_+[i]$ then Lemma 15 implies $R'_+[(i-1)'] > R_+[i] > R_+[i']$. Hence, $(i-1)'$ and i are R_+-descents of P'_i. The number of R_+-descents in P'_i is thus one more than in P.

(i.3) If $R_+[i] > R[1]$ then $R'_+[(i-1)'] > R_+[i] < R_+[i']$. Hence, the number of R_+-descents in P'_i equals the number of R_+-descents in P.

(ii) If i is not an R_+-descent of P then three cases analogous to the above show that the number of R_+-descents is retained or increases by one.

Combining all these cases shows, for each i, that the number of R_+-descents is preserved by the mapping of P to P'_i or is increased by one. □

Lemma 17. *Let P be a permutation with d R_+-descents and $\mathcal{P}'$ the set of successor permutations for P, then the number of successor permutations with d R_+-descents is $d+1$,*

$$d+1 = |\{P' \in \mathcal{P}' \,:\, |\mathrm{desc}(P')| = d\}|.$$

Proof. We assign to each R_+-run $[g,h]$ of P a *proper insertion position* $i \in [g, h+1]$ that preserves the number of R_+-descents through the mapping from P to P'_i and show that the number of R_+-descents increases for the other, non-proper insertion positions.

Let (g,h) be an R_+-run defined by a pair of consecutive R_+-descents, $(g-1,h)$, such that $R_+[g-1] > R_+[g] < R_+[g+1] < \cdots < R_+[h] > R_+[h+1]$. Remember, according to Lemma 14, that the R_+-descents not adjacent to the insertion position are preserved through the mapping to P'_i. Therefore it suffices to investigate the R_+-order of positions touched by the insertion. Since $R[1] \neq R_+[f]$ for all $f \in [1,n]$, we consider three mutually exclusive cases.

(i) If $R[1] < R_+[g]$ then the proper insertion position is i, $i = g$, such that

$$R_+[g-1] > R[1] < R_+[g] < \cdots < R_+[h] > R_+[h+1].$$

According to Lemmas 14 and 15, we obtain the series of inequalities

$$R'_+[(g-1)'] > R'_+[i] < R'_+[g'] < \cdots < R'_+[h'] > R'_+[(h+1)'].$$

Hence, for the insertion position g, there exist exactly as many R_+-descents in the interval $[g-1, h+1]$ of P as in the interval $[(g-1)', (h+1)']$ of P' and according to Lemma 14 the other R_+-descents are not affected through the mapping. Thus, $|R_+\text{-desc}(P)| = |R_+\text{-desc}(P'_i)|$.

For the insertion positions $i \in [g+1, h]$,

$$R_+[g] > R_+[g+1] < \cdots < R_+[i-1] > R[1] < R_+[i] < \cdots < R_+[h] > R_+[h+1]$$

holds. Then applying Lemmas 14 and 15 leads to

$$R'_+[g'] > R'_+[(g+1)'] < \cdots < R'_+[(i-1)'] > R'_+[i] < R'_+[i'] < \cdots < R'_+[h'] > R'_+[(h+1)'].$$

Therefore the number of R_+-descents increases through the mapping.

The bordering insertion position $h+1$ remains, for which we consider two special cases.

(i.1) If $R[1] < R_+[h+1]$ then $h+1$ would be the proper insertion position for the next R_+-run $(h+1, l)$ for some l, like in case (i).

(i.2) If $R[1] > R_+[h+1]$ then the insertion position $h+1$ increases the number of R_+-descents through the mapping from P to P'_i.

(ii) If $R_+[g] < R[1] < R_+[h]$ then analogously i, $i \in [g+1, h]$, with $R_+[i-1] < R[1] < R_+[i]$ is the proper insertion position. The other insertion positions j, $j \in [g+1, h]$ with $j \neq i$, increase the number of R_+-descents. The bordering insertion positions g or $h+1$ either increase the number of R_+-descents analogously to (i.2) or they are proper insertion positions for the adjacent R_+-runs.

(iii) If $R_+[h] < R[1]$ then the proof works analogously to (i) by handling the bordering insertion position g like (i.2).

So far, we concentrated on the inner R_+-runs (g,h) with $g \neq 1$ and $h \neq n$. For the bordering R_+-runs (g,h) with $g = 1$ or $h = n$, the proper insertion positions are defined in the same way. Just the proof is a bit simpler, because the insertion positions at the borders 1 and $n+1$ are both not affected by adjacent R_+-runs.

Finally, for each of the $d+1$ R_+-runs in P, there exists a unique insertion position i that preserves the number of R_+-descents through the mapping from P to P'_i. All other insertion positions increase the number of R_+-descents. □

Proof of Theorem 10. For $A(n,d)$, this being the number of permutations of length n with d R_+-descents, we achieve the following recursive definition with the two base cases (i) and (ii) and the recursion step (iii).

(i) Since the permutation $(n, n-1, \ldots, 1)$ is the only one without any R_+-descent, $A(n,0) = 1$.

(ii) Obviously, the number of potential R_+-descents is limited by $n-1$. Hence, there is no permutation of length n with more than $n-1$ R_+-descents, and thus $A(n,d) = 0$ for $d \geq n$.

(iii) As mentioned before, mapping each permutation P of length n to P'_i with all possible insertion positions i leads to n successor permutations, each of length n. If P contains d R_+-descents, then Lemma 17 implies: there exist exactly $d+1$ successor permutations with d R_+-descents and, according to Lemma 16, the other $n-d$ successors permutations contain $d+1$ R_+-descents. Combining these observations leads to the recursion $A(n,d) = (d+1)A(n-1,d) + (n-d)A(n-1,d-1)$ for $0 < d < n$.

The propositions (i),(ii), and (iii) yield the same recursion as for the Eulerian numbers. Hence $A(n,d) = \left\langle {n \atop d} \right\rangle$. □

6. Applications to compressed suffix arrays

The suffix array counting has applications to suffix array compression. Corollary 19 gives a tight lower bound for the compressibility of suffix arrays and Theorem 20 shows for a fixed sized alphabet and a large string length that we can essentially not achieve a better compression ratio by only compressing the suffix arrays instead of the strings.

Before formally stating and proving the mentioned proposition, we first perform some preliminary work. At first sight, the counting formula for the number of suffix arrays of Corollary 11 looks quite compact. The Eulerian numbers, however, are recursively defined, which is unfavourable in consideration of the subsequent reasoning. We rather convert the formula into a closed form.

Lemma 18. *Let σ and n be fixed positive integers, then*

$$\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle = \sum_{k=0}^{\sigma-1} \binom{n}{k} (-1)^k (\sigma-k)^n.$$

Proof. An equality rule for the Eulerian numbers [23, Eq. 6.38], equality rules for binomial coefficients, and some arithmetics lead to

$$\begin{aligned}
\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle &= \sum_{d=0}^{\sigma-1} \sum_{k=0}^{d} \binom{n+1}{k} (-1)^k (d+1-k)^n && (11)\\
&= \sum_{d=0}^{\sigma-1} \sum_{k=0}^{d} \left(\binom{n}{k} + \binom{n}{k-1} \right) (-1)^k (d+1-k)^n && (12)\\
&= \sum_{d=0}^{\sigma-1} \sum_{k=0}^{d} \binom{n}{k} (-1)^k (d+1-k)^n + \sum_{d=0}^{\sigma-1} \sum_{k=0}^{d} \binom{n}{k-1} (-1)^k (d+1-k)^n && (13)\\
&= \sum_{d=1}^{\sigma} \sum_{k=1}^{d} \binom{n}{k-1} (-1)^{k-1} (d+1-k)^n - \sum_{d=1}^{\sigma-1} \sum_{k=1}^{d} \binom{n}{k-1} (-1)^{k-1} (d+1-k)^n && (14)\\
&= \sum_{k=1}^{\sigma} \binom{n}{k-1} (-1)^{k-1} (\sigma+1-k)^n && (15)\\
&= \sum_{k=0}^{\sigma-1} \binom{n}{k} (-1)^k (\sigma-k)^n, && (16)
\end{aligned}$$

where equality (11) follows from $\left\langle {n \atop d} \right\rangle = \sum_{k=0}^{d} \binom{n+1}{k} (-1)^k (d+1-k)^n$ [23, Eq. 6.38], equality (12) from $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$, equality (13) from the distributivity, equality (14) from shifting d and k with respect to the first sum and from $\binom{n}{k-1} = 0$ for $k \leq 0$, equality (15) from subtracting both sums, and finally equality (16) from shifting k again. □

Many application areas for suffix arrays handle small alphabets like the DNA, amino acid, or ASCII alphabet. Corollary 11 thus limits the number of distinct suffix arrays for such applications. For example, for a DNA alphabet of size 4 the number of distinct suffix arrays of length 16 is $3,614,083,520 = \sum_{d=0}^{3} \left\langle {16 \atop d} \right\rangle$; whereas $20,922,789,888,000 = 16!$ is the number of possible permutations of length 16, which is about $5,789$ times larger. This difference rapidly increases for larger n. We achieve a lower bound on the compressibility of the whole information content of suffix arrays.

Corollary 19. *For strings of length n over an alphabet of size σ, the lower bound for the compressibility of their suffix arrays in the Kolmogorov sense is* $\log \sum_{k=0}^{\sigma-1} \binom{n}{k}(-1)^k(\sigma-k)^n$.

Proof. There are $\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle$ distinct suffix arrays. Among them there exists at least one binary representation with Kolmogorov complexity not less than $\log \sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle$. Due to Lemma 18 this equals to $\log \sum_{k=0}^{\sigma-1} \binom{n}{k}(-1)^k(\sigma-k)^n$. □

For a fixed size alphabet, we show that the ratio of the number of suffix arrays over the number of strings converges to 1 for an increasing string length.

Theorem 20. *Let σ be fixed, then*

$$\lim_{n\to\infty} \frac{\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle}{\sigma^n} = 1.$$

Proof. We use some equality rules to obtain

$$\lim_{n\to\infty} \frac{\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle}{\sigma^n} = \lim_{n\to\infty} \frac{\sum_{k=0}^{\sigma-1} \binom{n}{k}(-1)^k(\sigma-k)^n}{\sigma^n} \tag{17}$$

$$= \lim_{n\to\infty} \left(\frac{\sigma^n}{\sigma^n} + \sum_{k=1}^{\sigma-1} \binom{n}{k}(-1)^k \frac{(\sigma-k)^n}{\sigma^n} \right) \tag{18}$$

$$= 1 + \sum_{k=1}^{\sigma-1} (-1)^k \lim_{n\to\infty} \left(\binom{n}{k} \left(1 - \frac{k}{\sigma}\right)^n \right) \tag{19}$$

$$= 1 \tag{20}$$

where Eq. (17) follows from Lemma 18, Eqs. (18) and (19) from basic arithmetics, and Eq. (20) from the fact that $\lim_{n\to\infty} \left(\binom{n}{k} \left(1 - \frac{k}{\sigma}\right)^n \right) = 0$ for $0 < \frac{k}{\sigma} < 1$: The exponential term $\left(1 - \frac{k}{\sigma}\right)^n$ converges to 0 and dominates the polynomial term $\binom{n}{k}$. □

Note that Theorem 20 only holds if the alphabet is of a constant size. If the alphabet size grows proportionally to the string length, it is not true anymore. For $\sigma = n$, for example, $\lim_{n\to\infty} \frac{\sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle}{\sigma^n} = \lim_{n\to\infty} \frac{n!}{n^n} = 0$.

7. Summation identities

We present constructive proofs for two long known summation identities of Eulerian numbers deduced by summing up the number of different suffix arrays for a fixed alphabet size and string length. We believe that our constructive proofs are simpler than previous ones.

The identity $\sigma^n = \sum_i \left\langle {n \atop i} \right\rangle \binom{\sigma+i}{n}$, as given in [23, Eq. 6.37], was proven by J. Worpitzki, already in 1883. We prove it by summing up the number of string-distinct strings of length n over a given alphabet of size σ for each suffix array:

$$\sigma^n = \sum_{d=0}^{\sigma-1} \left\langle {n \atop d} \right\rangle \binom{n+\sigma-d-1}{\sigma-d-1} \tag{21}$$

$$= \sum_{d=0}^{\sigma-1} \left\langle {n \atop n-1-d} \right\rangle \binom{n+\sigma-d-1}{n} \tag{22}$$

$$= \sum_{i=n-\sigma}^{n-1} \left\langle {n \atop i} \right\rangle \binom{\sigma+i}{n} \tag{23}$$

$$= \sum_{i} \left\langle {n \atop i} \right\rangle \binom{\sigma+i}{n}. \tag{24}$$

Equality (22) follows from the symmetry rule for Eulerian and binomial numbers, equality (23) from substituting $i = n - d - 1$, and equality (24) from $\left\langle {n \atop i} \right\rangle = 0$ for all $i \geq n$ and $\binom{\sigma+i}{n} = 0$ for all $i < n - \sigma$.

The second summation identity is the summation rule for Eulerian numbers to generate the Stirling numbers of the second kind [23, Eq. 6.39]: $\kappa!\left\{{n \atop \kappa}\right\} = \sum_i \left\langle{n \atop i}\right\rangle \binom{i}{n-\kappa}$. To prove this identity, we count the $\kappa!\left\{{n \atop \kappa}\right\}$ strings composed of exactly κ different characters. Summing up these strings for each suffix array gives

$$\kappa!\left\{{n \atop \kappa}\right\} = \sum_{d=0}^{\kappa-1} \left\langle{n \atop d}\right\rangle \binom{n-d-1}{\kappa-d-1} \tag{25}$$

$$= \sum_{d} \left\langle{n \atop d}\right\rangle \binom{(n-\kappa)+(\kappa-d-1)}{\kappa-d-1} \tag{26}$$

$$= \sum_{d} \left\langle{n \atop n-1-d}\right\rangle \binom{n-d-1}{n-\kappa} \tag{27}$$

$$= \sum_{i} \left\langle{n \atop i}\right\rangle \binom{i}{n-\kappa}. \tag{28}$$

Equality (26) holds since $\left\langle{n \atop d}\right\rangle = 0$ for $d \geq \kappa$, equality (27) follows from the symmetry rule for Eulerian and binomial numbers, and equality (28) from substituting $i = n - d - 1$.

8. Conclusion

We have presented constructive proofs to count the strings sharing the same suffix array as well as the distinct suffix arrays for fixed size alphabets. For alphabets of size σ, $\binom{n+\sigma-d-1}{\sigma-d-1}$ strings share the same suffix array (with d R_+-descents) among which $\binom{n-d-1}{\sigma-d-1}$ are composed of exactly σ distinct characters. For these strings we have given a bijection into the set of non-decreasing sequences over $\sigma - d$ integers. The number of distinct suffix arrays is $\sum_{d=0}^{\sigma-1} \left\langle{n \atop d}\right\rangle = \sum_{k=0}^{\sigma-1} \binom{n}{k}(-1)^k(\sigma-k)^n$. This has yielded a lower bound for the compressibility of such suffix arrays.

Moreover, summing up the number of strings for each suffix array yields constructive proofs for Worpitzki's identity and for the summation rule of Eulerian numbers to generate the Stirling numbers of the second kind. One could also say that the number of suffix arrays and their strings form a particular instance of these identities.

Of further interest will be the development of efficient enumeration algorithms, for which our constructive proofs have already suggested suitable methods. For the enumeration of strings sharing the same suffix array, we have proven the equivalence to the enumeration of non-decreasing sequences, which can easily be performed in optimal time; whereas the enumeration of distinct suffix arrays in optimal time requires further development.

Acknowledgements

We thank Veli Mäkinen, Hans-Michael Kaltenbach, Constantin Bannert, and Moshe Lewenstein for the helpful discussions, Zsuzsanna Lipták and Ferdinando Cicalese for the advice on discovering and proving Lemma 18, and Sita Lange for carefully proofreading this manuscript.

References

[1] U. Manber, E.W. Myers, Suffix arrays: A new method for on-line string searches, SIAM Journal on Computing 22 (5) (1993) 935–948.
[2] G.H. Gonnet, R.A. Baeza-Yates, T. Snider, New indices for text: Pat trees and pat arrays, in: W.B. Frakes, R.A. Baeza-Yates (Eds.), Information retrieval: data structures and algorithms, Prentice-Hall, Upper Saddle River, NJ, USA, 1992, pp. 66–82.
[3] J. Kärkkäinen, P. Sanders, Simple linear work suffix array construction, in: Proceedings of the 30th International Colloquium on Automata, Languages and Programming, ICALP 2003, in: LNCS, vol. 2719, Springer Verlag, 2003, pp. 943–955.
[4] T. Kasai, G. Lee, H. Arimura, S. Arikawa, K. Park, Linear-time longest-common-prefix computation in suffix arrays and its applications, in: Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching, CPM 2003, in: LNCS, vol. 2089, Springer Verlag, 2001, pp. 181–192.
[5] D.K. Kim, J.S. Sim, H. Park, K. Park, Constructing suffix arrays in linear time, Journal of Discrete Algorithms 3 (2–4) (2005) 126–142.
[6] P. Ko, S. Aluru, Space efficient linear time construction of suffix arrays, Journal of Discrete Algorithms 3 (2–4) (2005) 143–156.
[7] M.I. Abouelhoda, S. Kurtz, E. Ohlebusch, Replacing suffix trees with enhanced suffix arrays, Journal of Discrete Algorithms 2 (1) (2004) 53–86.
[8] S. Kurtz, The vmatch homepage, http://www.vmatch.de (last visited: February 08, 2008).

[9] M. Burrows, D.J. Wheeler, A block-sorting lossless data compression algorithm, Tech. Rep. Research Report 124, Digital System Research Center, (May 1994).
[10] R. Grossi, J.S. Vitter, Compressed suffix arrays and suffix trees with applications to text indexing and string matching, in: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, STOC 2000, 2000, pp. 397–406.
[11] P. Ferragina, G. Manzini, Opportunistic data structures with applications, in: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, IEEE Computer Society, 2000, pp. 390–398.
[12] K. Sadakane, Compressed text databases with efficient query algorithms based on the compressed suffix array, in: Proceedings of the 11th International Symposium on Algorithms and Computation, ISAAC 2000, in: LNCS, vol. 1969, Springer-Verlag, London, UK, 2000, pp. 410–421.
[13] V. Mäkinen, Compact suffix array – a space-efficient full-text index, Fundamenta Informaticae 56 (1–2) (2003) 191–210.
[14] E.D. Demaine, A. López-Ortiz, A linear lower bound on index size for text retrieval, Journal of Algorithms 48 (1) (2003) 2–15.
[15] P.B. Miltersen, Lower bounds on the size of selection and rank indexes, in: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, SIAM, Philadelphia, PA, USA, 2005, pp. 11–12.
[16] G. Navarro, V. Mäkinen, Compressed full-text indexes, ACM Computing Surveys 39 (1) (2007) Article 2.
[17] J.-P. Duval, A. Lefebvre, Words over an ordered alphabet and suffix permutations, RAIRO – Theoretical Informatics and Applications 36 (3) (2002) 249–259.
[18] M. Crochemore, J. Désarménien, D. Perrin, A note on the Burrows-Wheeler transformation, Theoretical Computer Science 332 (1–3) (2005) 567–572.
[19] C. Hohlweg, C. Reutenauer, Lyndon words, permutations and trees, Theoretical Computer Science 307 (1) (2003) 173–178.
[20] S. Burkhardt, J. Kärkkäinen, Fast lightweight suffix array construction and checking, in: Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching, CPM 2003, in: LNCS, vol. 2676, Springer Verlag, Berlin, Germany, 2003, pp. 55–69.
[21] H. Bannai, S. Inenaga, A. Shinohara, M. Takeda, Inferring strings from graphs and arrays, in: Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, MFCS 2003, in: LNCS, vol. 2747, Springer Verlag, 2003, pp. 208–217.
[22] K.-B. Schürmann, J. Stoye, Counting suffix arrays and strings, in: M. Consens, G. Navarro (Eds.), Proceedings of the 12th International Symposium on String Processing and Information Retrieval, SPIRE 2005, in: LNCS, vol. 3772, Springer Verlag, 2005, pp. 55–66.
[23] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics, 2nd Edition, Addison-Wesley, 1994.
[24] D. Moore, W.F. Smyth, D. Miller, Counting distinct strings, Algorithmica 23 (1) (1999) 1–13.