Efficient Constructions of Prefer-Max Sequences and of Infinite de Bruijn Sequences

January 5, 2017

1 Introduction

A k-ary de Bruijn sequence of span n is a cyclic sequence of length k^n in which each k-ary string of length n appears exactly once as a substring. A well known construction of such a sequence, first proposed by Ford [1], can be described as follows:

Algorithm 1 The (k, n)-prefer-max sequence.

Repeatedly append the maximal symbol over the alphabet $0, \ldots, k-1$ such that the word formed by the n most recent symbols is new (in the first n steps, act as if thee are n invisible zeros before the sequence). Stop after appending n consecutive zeros.

For example, if we choose $k = 3$ and $n = 3$ we get the sequence:

$$\langle 2, 2, 2, 1, 2, 2, 0, 2, 1, 1, 2, 1, 0, 2, 0, 1, 2, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0 \rangle.$$

An obvious limitation of the above construction is, of course, that, when n and k grow, the amount of time and memory needed to produce the next symbol grows exponentially. Therefore, this construction is not practical for generating de Bruijn sequences of large alphabets and spans. It is used here as a mathematical definition of the sequence. An efficient algorithm for constructing this sequence is one of the contributions of this paper.

Note that the (n, k)-prefer-max sequence begins with a prefix (the first 19 symbols) that contains all the words in $\{0, 1, 2\}^n \setminus \{0, 1\}^n$ (the words that contain the symbol 2) as substrings. Note also that the rest of the sequence, after this prefix, is exactly the sequence that we would have obtained if we ran the construction with the alphabet $k = 2$ (and the same n). The first result of this paper, that we call the Onion lemma, is that this is not a coincidence:

Theorem 1 (Onion)**.** *The last* $(k-1)^n + 1$ *symbols of the* (k, n)*-prefer-max sequence form the* $(k-1, n)$*-prefer-max sequence.*

Proof. Denote the symbols of the $(n-k)$-prefer-max sequence by $s_1, s_2, \ldots, s_{k^n+1}$ and let, for notational convenience, $s_i = 0$ for all $i < 1$. Let $i_0 = \min\{i \colon (k-1) \notin \{s_{i-n+1}, \ldots, s_i\}\}$. By minimality we have that $s_{i_0-n} = k-1$. Since $s_{i_0} \neq k-1$ and because $s_{i_0-n+1} \cdots s_{i_0-1}(k-2)$ did not appear as a substring before i_0 we get, by construction, that $s_{i_0} = k-2$. This means that the exists some $i_1 < i_0$ such that $s_{i_1-n+1} \cdots s_{i_1} = s_{i_0-n+1} \cdots s_{i_0-1}(k-1)$.

Assume, towards contradiction, that $s_{i_0-n+1} \cdots s_{i_0-1} \neq 0^{n-1}$. In particular $s_{i_1-n+1} \cdots s_{i_1} \neq 0^{n-1}(k-1)$ which means that $i_1 > 1$. Therefore, $k-1 \in \{s_{i_1-n}, \ldots, s_{i_1-1}\} = \{s_{i_1-n}\} \cup \{s_{i_0-n+1}, \ldots, s_{i_0-1}\}$. Since the second set in this union does not contain $k-1$, we get that $s_{i_1-n} = k-1$. This leads to a contradiction because it means that $s_{i_1-n} \cdots s_{i_1-1} = s_{i_0-n} \cdots s_{i_0-1}$ which cannot happen in a de Bruijn sequence. This contradiction gives us that $s_{i_0-n+1} \cdots s_{i_0-1} = 0^{n-1}$.

The sequence is therefore as follows:

$$\underbrace{s_1, \ldots, s_{i_0-n-1}, k-1, 0, \ldots, 0}_{\substack{\text{all substrings of length } n \\ \text{contain } k-1}}, \underbrace{k-2}_{s_{i_0}}, s_{i_0+1}, \ldots, s_{k^n+1}$$

To complete the proof, we need to show that all substrings of length n that contain $k-1$ appear before s_{i_0}.

For $w \in \{0, \ldots, k-1\}^n$ and $1 \leq m \leq n$, let j_m be such that $s_{j_m-n+1} \cdots s_{j_m} = w[m..n]0^{m-1}$, where $w[m..n]$ is the $n-m+1$ letters suffix of w. Because 0 is the smallest symbol in our alphabet, we get that it is added only if all the other symbols cannot be added, i.e., when adding other symbols would generate a substring that has already being seen before. Since w comes right before some $w[2..m]\tau$ as a window $w[2..m]0$ comes after (or is equal to) $w[2..m]\tau$ which is after w. Similarly we can see that $w[2..m]0$ appears before $w[3..m]0^2$ and so on. This gives us that $j_1 < j_2 < \cdots < j_n$. Note that $s_{j_1-n+1} \cdots s_{j_1} = w$, i.e, that w appears as a substring at the j_1's symbol of the sequence.

We will now show that if w contains the symbol $k-1$ it must appear as a substring before the i_0's symbol in the sequence. Consider first the case where the last (right-most) symbol of w is $k-1$. In this case, $j_n = i_0 - 1$ so we have, because $j_1 < j_n$, that w appears as a substring before the i_0'th symbol in the sequence. The second case is when the last (right-most) symbol in w is not $k-1$ but there is another letter in w which is $k-1$. In this case we have that w appears less than n steps after a window that ends with $k-1$ and the window $s_{i_0-n} \cdots s_{i_0-1} = (k-1)0^{n-1}$ does not appear in the windows between them because they all contain $k-1$ at a letter that is not their first. Thus, also in this case, the window w appears before the index i_0.

To conclude the proof we note that we established that the windows up to the index i_0 contain all and only the words in $\{0, \ldots, k-1\}^n \setminus \{0, \ldots, k-1\}^n$ and that this part of the sequence ends with $n-1$ zeros and then the next part begins with the symbol $k-2$. This means that from this point on the prefer-max construction acts exactly as it does when the sequence begins with $k-2$, so it constructs the $(k-1, n)$-prefer-max sequence. □

The observation made in the last theorem directed us to examine the reverse of the prefer-max sequence. Specifically, the structure identified in Theorem 1 means that if we somehow manage to construct the sequence backwards, we do not have to specify k ahead of time. We can add symbols forever while k grows in layers: we first write 0^n, then a word whose substrings of length n are $\{0,1\}^n$, then extend with other words in $\{0,1,2\}^n$, and so on.

The second contribution of this work is then an efficient construction of this reversed sequence. This is formulated in Algorithm 2. We say that this construction is efficient because each step can be computed in $O(n^2)$ operations by enumerating all the n rotations of σw and by comparing them lexicographically (reading each word from right to left).

Algorithm 2 An infinite de Bruijn sequence.

Append n zeros. Then, repeatedly, let σw be the word formed by the n most recent symbols. Append $\sigma+1$ if $(\sigma+1)w$ is of the form $u0^l$ where $0^l u$ is maximal in right-to-left lexicographic order among all of its rotations. If $(\sigma+1)w$ is not of that form and σw is, append the symbol 0. Otherwise, if neither $(\sigma+1)w$ nor $(\sigma)w$ are of that form, append σ.

The fact that this generates the reverse of the prefer-max sequence is formulated in the following theorem:

Theorem 2. *The first k^n symbols generated by Algorithm 2 form the reverse of the sequence generated by the prefer-max construction with the parameters k and n.*

In particular, the above theorem says that the sequence generated by Algorithm 2 is a de Bruijn sequence, i.e., that each word in $\mathbb{N}^n$ appears exactly once as a substring.

While our main focus is on analyzing the reversed sequence, we also have a result about the prefer-max sequence itself. Our result gives the first (to the best of our knowledge) efficient construction of that sequence:

Algorithm 3 An efficient construction of the prefer-max sequence.

Append $n-1$ zeros and then the symbol k. Then, repeatedly, let σw be the word formed by the n most recent symbols. Append the symbol $\sigma-1$ if σw is of the form specified in Algorithm 2. Otherwise, if there is a σ' such that $\sigma' w$ is of that form, append the maximal such σ'. Otherwise, if there is no such σ', append the symbol σ.

Each step of Algorithm 3 can be computed in $O(n^2)$ time because it suffices to examine for σ' only symbols that are in σw or are smaller by one than a symbol in σw, as we will establish later in the paper. The fact that this is indeed the prefer-max sequence is stated in the following theorem:

Theorem 3. *The sequence generated by Algorithm 3 is the prefer-max sequence with parameters k and n.*

A central tool to our analysis of the sequences descibed above is an alternative formulation of the prefer-max sequence. For this, we need to introduce some language, as follows.

The n dimensional infinite de Bruijn graph is the graph $G_n = \langle V, E\rangle$ where $V = \mathbb{N}^n$ and $E = \{\langle \sigma_1 w, w\sigma_2\rangle \colon \sigma_1, \sigma_2 \in \mathbb{N}, w \in \mathbb{N}^{n-1}\}$. The de Bruijn sequences, defined in the beginning of this paper, are in one-to-one correspondence with the Hamiltonian cycles in the de Bruijn graph, as follows: (1) Concatenating the first letter of each vertex in a cycle produces a de Bruijn sequence of order n; and (2) A cycle can be constructed from a de Bruijn sequence $\langle \sigma_1, \ldots, \sigma_{2^n}\rangle$ of order n by visiting the vertex $\sigma_1 \cdots \sigma_n$, then $\sigma_2 \cdots \sigma_{n+1}$ and so on.

Let $rtl(\sigma_1\sigma_2 \ldots \sigma_n) = \sigma_2 \ldots \sigma_n\sigma_1$ denote the rotation of a word to the left and let $rtl^i(w)$ denote a rotation of i letters to the left. A *necklace* in the de Bruijn graph G_n is the set $\{rtl^i(w)\}_{i=1}^n$ where $w \in \mathbb{N}^n$, i.e., it is an equivalence class of n-character strings over $\mathbb{N}$ when strings that are rotations of each other are considered equivalent. A $w \in \mathbb{N}^n$ is called a *key word* if it is larger in right-to-left lexicographic order than all of its rotations (all the words in its necklace). Note the resemblance of this to the notion of Lyndon words [4], on which we will elaborate later.

We are now ready to introduce another construction:

Algorithm 4 A necklace joining construction.

Start with the one element sequence $\langle 0^n\rangle$. The sequence at the ith step is constructed by adding words to the previous sequence as follows: Let k_i be the ith key word in right-to-left lexicographic order. Let l be the number of leading zeros in k_i (l may be zero) and let $\sigma \in \mathbb{N}$ and $w \in \mathbb{N}^{n-l-1}$ be such that $k_i = 0^l(\sigma + 1)w$. Now, add the sub-sequence $\langle rtl^i(w0^l(\sigma + 1))\rangle_{i=0}^{n-1}$ between the word $\sigma w0^l$ and its follower in the previously constructed sequence.

To following proposition establishes that the above construction describes an infinite sequence of words:

Proposition 4. *For every $l > 0$ there is $m > 0$ such that for every $i > m$ the insertions of words at the ith step of Algorithm 4 are all after the lth element of the sequence.*

Let $S = \langle w_i\rangle_{i=0}^{\infty}$ be the infinite sequence all whose prefixes agree with infinitely many prefixes of the finite sequences generated at the steps of Algorithm 4. Since every word is a rotation of some key word, we have that this sequence contains all the words in $\mathbb{N}^n$. Since there is a directed edge in G_n from each word in this sequence to its follower, the sequence corresponds to an Hamiltonian path in G_n. The following theorem states that this path maps, by the standard correspondence described above, to the reverse of the prefer-map sequence:

Theorem 5. *If we take the first letter of each word in S we get the sequence described in Algorithm 2.*

This result allows for another construction of the reversed prefer-max sequence, as follows. By the definition of the necklace joining construction, shown in Algorithm 4, one can infer that key words appear in lexicographic order and that the distance between one key words to the next is the size of the necklace of the latter. For a word $w \in \mathbb{N}^n$, the *primitive root* of w is the unique primitive (non-periodic) word $\rho(w)$ such that $w = \rho(w)^l$ for some $l \geq 1$. Since the length of the primitive root of a ket word is the length of its cycle, we get the following construction:

Algorithm 5 Word concatenation construction.

Construct an infinite sequence of symbols by concatenating the primitive roots of the key words of length n in left-to-right lexicographic order.

As the reader may expect, our result is this sequence is also the reverse of the prefer-max sequence:

Theorem 6. *The sequence described in Algoithm 5 is the same sequence described in Algorithm 2.*

The last result is similar in nature to the theorem of Fredricksen and Maiorana in [3] (see also [5] and [6]). These papers show that the concatenation in lexicographic order of the Lyndon words (words that are minimal in lexicographic among their rotations) of length dividing n produces a de Bruijn sequence of span n, and that this word is lexicographically minimal among all de Bruijn sequences of span n. In another paper [2], Fredricksen proves that (in the binary case) the prefer-max (there called Ford) sequence is the lexicographically least sequence. With some mapping (changing the order of the letters and the direction of reading words when considering lexicographic order) our last result can be regarded as an alternative proof of the result identified by Fredricksen and Maiorana.

References

[1] Lester Randolph Ford. A cyclic arrangement of m-tuples. *Report no. P-1071, RAND Corp*, 1957.

[2] Harold Fredricksen. The lexicographically least de Bruijn cycle. *Journal of Combinatorial Theory*, 9(1):1–5, 1970.

[3] Harold Fredricksen and James Maiorana. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Mathematics*, 23(3):207–210, 1978.

[4] Roger Conant Lyndon. On Burnside's Problem. *Transactions of the American Mathematical Society*, 77(2):202, sep 1954.

[5] Eduardo Moreno. On the theorem of Fredricksen and Maiorana about de Bruijn sequences. *Advances in Applied Mathematics*, 33(2):413–415, 2004.

[6] Eduardo Moreno and Dominique Perrin. Corrigendum to On the theorem of Fredricksen and Maiorana about de Bruijn sequences [Adv. in Appl. Math. 33 (2) (2004) 413415], 2015.