Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT

Theoretical Computer Science 301 (2003) 321–340

Theoretical Computer Science

www.elsevier.com/locate/tcs

Euclidean strings☆

John Ellis[a], Frank Ruskey[a,*], Joe Sawada[b], Jamie Simpson[c]

[a] *Department of Computer Science, University of Victoria, P.O. Box 3055, Victoria, British Columbia, Canada V8W 3P6*

[b] *Department of Computer Science, University of Toronto, Toronto, Canada*

[c] *Department of Mathematics and Statistics, Curtin University, Perth, Western Australia, Australia*

Received 15 December 2000; received in revised form 11 July 2002; accepted 29 July 2002
Communicated by G. Ausiello

Abstract

A string $\mathbf{p} = p_0 p_1 \cdots p_{n-1}$ of non-negative integers is a *Euclidean string* if the string $(p_0 + 1)p_1 \cdots (p_{n-1} - 1)$ is rotationally equivalent (i.e., conjugate) to $\mathbf{p}$. We show that Euclidean strings exist if and only if n and $p_0 + p_1 + \cdots + p_{n-1}$ are relatively prime and that, if they exist, they are unique. We show how to construct them using an algorithm with the same structure as the Euclidean algorithm, hence the name. We show that Euclidean strings are Lyndon words and we describe relationships between Euclidean strings and the Stern–Brocot tree, Fibonacci strings, Beatty sequences, and Sturmian sequences. We also describe an application to a graph embedding problem.

Keywords: Euclidean algorithm; Lyndon word; Stern–Brocot tree; Fibonacci string; Beatty sequence; Rational mechanical sequence; Sturmian sequence; Christoffel word; Morphism

1. Introduction

The string $010\underline{01}010$ has a curious property. The string is not equal to any of its non-trivial rotations, but if we reverse the marked 01 pair, we get the string $010\underline{10}010$, which is rotationally equivalent to the original string. In this paper, we will investigate this phenomenon in a slightly more general setting.

☆ This work was supported by the Natural Sciences and Engineering Research Council of Canada and by Czech grant GACR 201/99/0242.

* Corresponding author.
E-mail address: fruskey@cs.uvic.ca (F. Ruskey).

PII: S0304-3975(02)00589-3

Our initial interest in strings with this property stems from a certain graph embedding problem which is explained in Section 9 of this paper. Subsequent investigations revealed that these strings had many interesting properties and were related to classic concepts in the subject area of combinatorics on words.

Informally, a Euclidean string is a string of integers with the property that if the first character is increased by one and the last character is decreased by one, then the result is conjugate to the original string. We will show how to generate Euclidean strings by a variant of the Euclidean algorithm, from which they get their name. We prove various characterizations of Euclidean strings, and derive some of their properties. As first pointed out to us by the referees, Euclidean strings are closely related to Sturmian words; in fact, we will eventually show that the binary Euclidean strings are exactly the so-called *lower rational mechanical words*. The rational mechanical words are also called *Christoffel words* and *characteristic sequences* [1, 15], and arise in several different fields. All words studied in this paper are finite. The Sturmian words are infinite but their finite subwords are related to the Euclidean strings.

1.1. Definitions

Let $\mathbf{p} = p_0 p_1 \cdots p_{n-1}$ denote a string of n non-negative integers. When the length of $\mathbf{p}$ is at least 2, we let $\rho(\mathbf{p})$ denote a right rotation of $\mathbf{p}$ by one position, i.e., $\rho(\mathbf{p}) = p_{n-1} p_0 p_1 \cdots p_{n-2}$. Then let $\rho^d(\mathbf{p})$ denote a right rotation through d positions and $\rho^{-d}(\mathbf{p})$ denote a left rotation through d positions. Let $\tau(\mathbf{p})$ be the string obtained from $\mathbf{p}$ by replacing p_0 by $p_0 + 1$ and p_{n-1} by $p_{n-1} - 1$.

Definition 1. The string $\boldsymbol{p}$ is a *Euclidean* string if it is of unit length or if there exists an integer d such that $\tau(\mathbf{p}) = \rho^d(\mathbf{p})$.

For example, $\mathbf{p} = 22322322322323$ is a Euclidean string because

$$\tau(\mathbf{p}) = 32322322322322 = \rho^3(\mathbf{p}).$$

We will refer to the parameter d in the definition as a *displacement* of the Euclidean string. For consistency we define the displacement of a unit length Euclidean string to be one. Two strings $\mathbf{p}$ and $\mathbf{q}$ are said to be *conjugate* if there is an integer d such that $\mathbf{p} = \rho^d(\mathbf{q})$. So another way of defining a Euclidean string is as a string $\mathbf{p}$ which is conjugate to $\tau(\mathbf{p})$.

Definition 2. The *weight* of a Euclidean string is $\sum_{i=0}^{n-1} p_i$, i.e., the sum of all its elements.

Definition 3. The *cost* of a unit length Euclidean string is $p_0 - 1$. Otherwise, the cost, relative to a displacement d, is $\sum_{i=0}^{d-1} p_i$, i.e., the sum of the d elements to the right of and including p_0.

Throughout the paper, we use k to denote the weight and c to denote the cost of a Euclidean string. We denote a Euclidean string of length n and weight k by $E_{n,k}$. In arithmetic expressions, mod will denote the remainder after integer division. We adopt the convention that all arithmetic on the string indices is done modulo n, the length of the string, e.g., we write p_{i+j} for $p_{(i+j) \bmod n}$.

1.2. Organization

In Section 2, we show that a Euclidean string exists if and only if its length and weight are relatively prime and that, if it exists, it is unique, as are the cost and displacement. Thus there is a natural bijection between the set of Euclidean strings and the set of rational numbers. We show how to efficiently compute Euclidean strings, together with their displacements and costs, using a variant of the Euclidean algorithm.

In Section 3, we show some interesting properties relating the complement, the reversal and the "dual" of a Euclidean string. In Section 4, we show that Euclidean strings are Lyndon words and deduce some related properties. A Lyndon word is one which is lexicographically less than all of its non-trivial conjugates. The relationship between Lyndon words and Sturmian words is explained in [15].

In Section 5, we show a correspondence between the set of all binary Euclidean strings and the Stern–Brocot tree of reduced fractions. The *Stern–Brocot tree* is an infinite construction of the set of all non-negative fractions k/n between 0 and 1 where k and n are relatively prime. The tree is a rooted binary tree in which each node in the tree is defined in terms of its nearest left ancestor (L), and nearest right ancestor (R). If a node x has $L = k'/n'$ and $R = k''/n''$ then $x = (k' + k'')/(n' + n'')$. Similar results are discussed by Berstel and de Luca [3].

In Section 6, we show that every Fibonacci string is conjugate to a Euclidean string. A Fibonacci string is defined by the morphism $b \mapsto a$, $a \mapsto ab$.

In Section 7, we show that Euclidean strings are related to Beatty sequences. A Beatty sequence is a sequence of the form $a_j = \lfloor \alpha j + \beta \rfloor$ where $\alpha > 1$ and $j \in \mathbb{Z}$. We show that a Euclidean string is the reversal of the histogram of a rational Beatty sequence.

In Section 8, we show the relation of Euclidean strings to Sturmian strings; in particular that the Euclidean strings are the (lower) rational mechanical words. For real numbers α and β, the *lower mechanical word* $s_{\alpha,\beta}$ has nth character $\lfloor (n+1)\alpha + \beta \rfloor - \lfloor n\alpha + \beta \rfloor$ [15, Section 2.1.2]. If α is a positive rational number and $\beta = 0$, then we refer to these as *rational mechanical words*. A string s is Sturmian if and only if it is irrational mechanical [15, Theorem 2.1.13]. The definition of a Sturmian word is not directly related to the content of this paper; the interested reader is referred to Lothaire [15] for many fascinating results. Proposition 2.2.12 of [15] implies that the rational mechanical words with $0 < \alpha < 1$ have the form $0w1$, where w is a *central* word. The set of all central words is denoted *PER* and was studied by de Luca and Mignosi in [16].

We conclude in Section 9 with the application to a graph embedding problem that was the original motivation for the investigation. Some of the results in this paper were first presented in [8].

2. Characterization and generation

Lemma 1. *Any length n Euclidean string $p_0 p_1 \cdots p_{n-1}$ has the following properties*:
(a) *the length and the weight are relatively prime and*
(b) *it is unique, for given length and weight, as are its displacement and cost and*
(c) *for all i in $\{jd \bmod n\colon\ 0 \leqslant j \leqslant n-(k \bmod n)-1\}$, $p_i = \lfloor k/n \rfloor$ and for all i in $\{(n-1+jd) \bmod n\colon\ 0 \leqslant j \leqslant (k \bmod n)-1\}$, $p_i = \lceil k/n \rceil$ or, alternatively, $p_i = |\{j : n-1+jd \equiv i \bmod n,\ 0 \leqslant j < k\}|$ and*
(d) $dk \equiv 1 \bmod n$.

Proof. Let $\mathbf{p}$ be a Euclidean string of length n and weight k and with a displacement d so that $\rho^d(\mathbf{p}) = \tau(\mathbf{p})$. We can deduce from the definition of a Euclidean string that

$$\text{if } (0 \leqslant j \leqslant n-d-2) \text{ or } (n-d+1 \leqslant j \leqslant n-1) \quad \text{then } p_j = p_{j+d} \tag{1}$$

$$p_{n-d-1} = p_{n-1} - 1 \quad \text{and} \quad p_{n-d} = p_0 + 1. \tag{2}$$

Consider the sequence of string elements

$$S = p_0, p_d, p_{2d}, \ldots, p_{jd},$$

where all the indices are taken modulo n and where j is the smallest positive integer such that $jd \equiv 0 \bmod n$. Such a j must exist. It must be that p_{n-d} is the penultimate item in S since it is the predecessor of p_0. By (2) $p_{n-d} = p_0 + 1$. But, by Eqs. (1) and (2), every element in the sequence is equal to its predecessor in the sequence, unless the element is p_0 or p_{n-1}. Since the sequence starts and ends with p_0, it must then also contain p_{n-1}, because p_{n-d-1} is the only remaining element which can break the sequence of identical elements. So S is a sequence of string elements

$$S = p_0, p_d, p_{2d}, \ldots, p_{n-d-1}, p_{n-1}, \ldots, p_{n-d}, p_0,$$

where every element is identical to its successor except for p_{n-d-1} and p_{n-d}.

It follows that there are only two values in S, differing by one. Let them be $x = p_0$ and $x+1 = p_{n-1}$. Then S consists of a sequence of x's followed by a sequence of $(x+1)$'s. Let there be r of the x's. It must be that the first $(x+1)$ is p_{n-1} and that $n-1 \equiv rd \bmod n$, i.e., $rd \equiv -1 \bmod n$, which implies that $\gcd(r,n) = 1$ and that $\gcd(d,n) = 1$. Since $\gcd(d,n) = 1$, it must be that S contains all the elements of $\mathbf{p}$.

Then $x = \lfloor k/n \rfloor$ and $x+1 = \lceil k/n \rceil, rx + (n-r)(x+1) = k$ and hence $k + r \equiv 0 \bmod n$. We can then derive the items in the lemma as follows.

(a) From $\gcd(r,n) = 1$ and $k + r \equiv 0 \bmod n$ we derive $\gcd(n,k) = 1$.
(b) We have that $rd \equiv -1 \bmod n$, where $r = n - (k \bmod n)$. This implies that d has a unique value in the reduced residue set modulo n. Hence the sequence S is unique and the corresponding Euclidean string is unique.

(c) Since r, the number of smaller elements, is necessarily $n-(k \bmod n)$, the first two statements follow. The third statement follows from the observation that, since $\gcd(d,n)=1, |\{j : n-1+jd \equiv i \bmod n,\ 0 \leqslant j < n\}| = 1$ for all i.

(d) From $k+r \equiv 0 \bmod n$ and $rd \equiv -1 \bmod n$ we derive $dk \equiv 1 \bmod n$. □

In the proof of the following lemma we will make use of two string operations, *increment* and *expand*. Increment, denoted *inc*, adds one to every integer in the string. Expand, denoted *exp*, replaces every integer i in the string by 01^i, where 1^i denotes a string of i ones. In other words, *exp* and *inc* are the morphisms

$$i \mapsto i+1 \quad \text{and} \quad i \mapsto 01^i.$$

We have not seen these morphisms used before in the literature of combinatorics on words. However, they will be crucial to proving some of our new results and in deriving simpler proofs of some known results.

Lemma 2. *If n and k are relatively prime, positive integers, then there exists a Euclidean string of length n and weight k.*

Proof. The proof is constructive. The function E-STRING constructs a Euclidean string, of length n and weight k, when n and k are relatively prime. It mirrors the structure of the symmetric Euclidean algorithm, and is the reason that we call our strings "Euclidean strings".

```
function E-STRING (n, k : Z+) : String over N*;
    if  k < n → return(exp(E-STRING (n − k, k)))
    []  k = n → return('1');
    []  k > n → return(inc(E-STRING (n, k − n)))
    fi
```

We demonstrate the correctness of the procedure by induction on the number of invocations to the increment and expand procedures. For the base case, where $n=k=1$, the procedure returns the string "1", which conforms to Definition 1.

It can not be the case that $n=k$ and $n>1$ because n and k are relatively prime. Suppose $k>n$, in which case the increment operation is applied to the result of invoking the procedure with parameters $(n, k-n)$. Since n and k are relatively prime, so are n and $k-n$. Hence we may assume that a Euclidean string of length n and weight $k-n$, say $\mathbf{p}$, is returned. It is clear that since $\tau(\mathbf{p}) = \rho^d(\mathbf{p})$, where d is the displacement of $\mathbf{p}$, then $\tau(inc(\mathbf{p})) = \rho^d(inc(\mathbf{p}))$. Further, the incrementation increases the weight of the string from $k-n$ to k. Hence $inc(\mathbf{p})$ is a Euclidean string of length n and weight k. The displacement is unchanged by the operation. The cost (recall Definition 3) is increased by d, since the values of the leftmost d elements are each increased by one.

Now suppose $n>k$, in which case the expand operation is applied to the result of invoking the procedure with parameters $(n-k, k)$. Since n and k are relatively prime, so are $n-k$ and k. Hence we may assume that a Euclidean string of length $n-k$ and

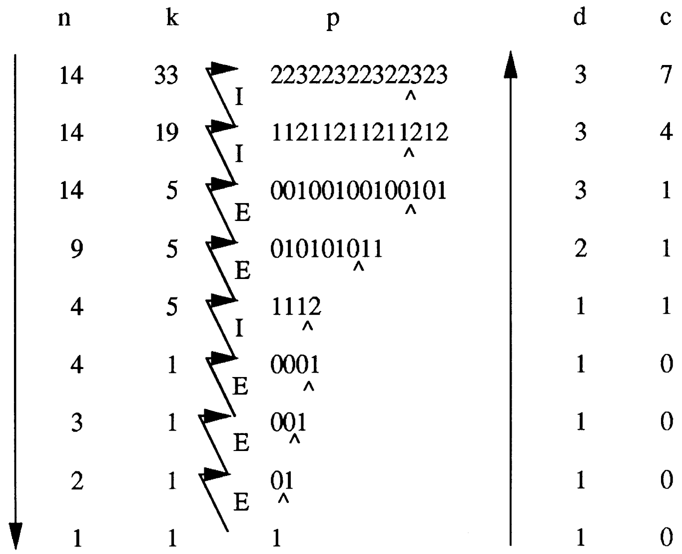

Fig. 1. Computing the Euclidean string $E_{14,33}$. The symbol I denotes an increment, E denotes an expansion and the caret denotes the exchange point (Lemma 5).

weight k, say $\mathbf{p}$, is returned and that $\tau(\mathbf{p}) = \rho^d(\mathbf{p})$, where d is the displacement of $\mathbf{p}$. Now

$$exp(\mathbf{p}) = 01^{p_0}01^{p_1}\cdots 01^{p_{n-k-1}}.$$

It follows that $\tau(exp(\mathbf{p})) = \rho^{d+c}(exp(\mathbf{p}))$, where c is the cost of $\mathbf{p}$, because the leftmost d elements in $\mathbf{p}$ are replaced by $d+c$ elements by the expansion. Further, the length of the expanded string is the number of 0's plus the number of 1's which is $(n-k)+k=n$, but the weight of the expanded string remains k. Hence $exp(\mathbf{p})$ is a Euclidean string of length n and weight k. The expansion increases the displacement by c and the cost is unchanged. □

An example of the computation of a string is given in Fig. 1. In the figure the cost and displacement are also shown. The computation of these parameters is discussed in the next section.

The unwinding of the recursion can be viewed as a walk from the root in an infinite binary tree $\mathscr{T}$. In $\mathscr{T}$ each node is labelled by an ordered pair (x, y). The left child of (x, y) is $(x, x+y)$ and the right child is $(x+y, y)$. The root is $(1, 1)$. See Fig. 2. Every reduced fraction x/y occurs once in this tree and a breadth first traversal of this tree provides a proof that the rational numbers are countable [4]. Fig. 3 shows the corresponding tree of strings.

From Lemmas 2 and 1(a), we immediately obtain Theorem 1. This theorem implies that the number of Euclidean strings of length n is the Euler totient function $\phi(n)$.

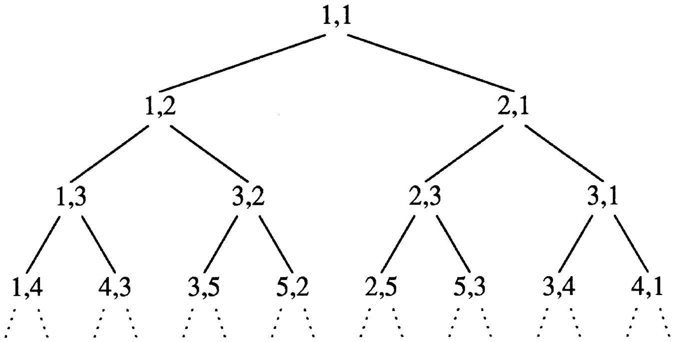

Fig. 2. The tree of reduced fractions.

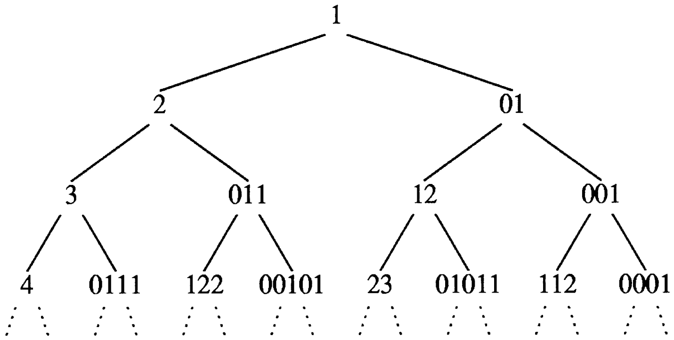

Fig. 3. The tree of strings corresponding to the tree of reduced fractions in Fig. 2.

Theorem 1. *There exists a Euclidean string of length n and weight k if and only if* $\gcd(n,k) = 1$.

If d is known, then we can use Lemma 1(c), to generate the Euclidean string in time linear in the length of the string. Below is a procedure which computes the displacement and cost, d and c. Variables c and d are global and the initial call is $\text{DISP–COST}(n,k)$. The correctness of the procedure follows immediately from the definition of each parameter, i.e., an incrementation requires that the cost be increased by d, whereas an expansion requires that the displacement be increased by c. For consistency, we have defined the displacement and cost of a unit length Euclidean string to be 1 and $k-1$, respectively. Also note that one of the terminating conditions must be reached.

procedure $\text{DISP–COST}\ (n,k : \mathbb{Z}^+)$
  if $k < n \rightarrow \text{DISP–COST}(n-k,k);\ d := d + c$
  [] $k = n \rightarrow c, d := 0, 1$
  [] $k > n \rightarrow \text{DISP–COST}(n,k-n);\ c := c + d$
  fi

The following lemma describes a relationship between cost and displacement.

Lemma 3. *If* $n+k \geqslant 2$ *then* $dk = cn + 1$.

Proof. Let $d_{n,k}$ and $c_{n,k}$ denote the displacement and cost, respectively, of a Euclidean string of length n and weight k. We argue by induction on the number of recursive invocations of the procedure. For the basis we note that if $n=1$ and $k=1$ the initialization defined by the algorithm satisfies the lemma.

For the induction suppose first that $n<k$. By the inductive hypothesis: $d_{n,k-n}(k-n) = c_{n,k-n}n+1$. Hence $d_{n,k-n}(k-n)+d_{n,k-n}n = (c_{n,k-n}+d_{n,k-n})n+1$. Hence $d_{n,k-n}k = (c_{n,k-n}+d_{n,k-n})n+1$. But, when $n<k$, the algorithm sets $d_{n,k} = d_{n,k-n}$ and $c_{n,k} = c_{n,k-n} + d_{n,k-n}$. Hence $d_{n,k}k = c_{n,k}n+1$.

Suppose $k<n$. By the inductive hypothesis: $d_{n-k,k}k = c_{n-k,k}(n-k)+1$. Hence $(d_{n-k,k}+c_{n-k,k})k = c_{n-k,k}n+1$. But, when $n<k$, the algorithm sets $d_{n,k} = d_{n-k,k}+c_{n-k,k}$ and $c_{n,k} = c_{n-k,k}$. Hence $d_{n,k}k = c_{n,k}n+1$. □

So d and $-c$ are in fact the constants computed by the standard extended Euclidean algorithm, see for example [5, p. 811]. We have already shown, in Lemma 1(d), that $dk \equiv 1 \bmod n$, i.e., d is the multiplicative inverse of k modulo n. Lemma 3 immediately yields an analogous corollary, which is crucial to the application described in the final section of this paper and taken from [7].

Corollary 1. *The number* c *is the multiplicative inverse of* $(k-n)$ *modulo* k, *i.e.*, $c(k-n) \equiv 1 \bmod k$.

3. Some properties of Euclidean strings

Let $R(\mathbf{p})$ denote the reversal (or mirror image) of $\mathbf{p}$, i.e., $R(\mathbf{p}) = p_{n-1}p_{n-2}\cdots p_1 p_0$.

Lemma 4. *If* $\mathbf{p}$ *is a Euclidean string of length at least* 2 *and with displacement* d, *then* $\mathbf{p} = R(\tau(\mathbf{p})) = R(\rho^d(\mathbf{p}))$ *and* $\tau(\mathbf{p}) = R(\mathbf{p})$.

Proof. We proceed by induction on the number of applications of the expand and increment operations in the construction of $\mathbf{p}$. The statement is true for strings of length 2 and for strings of the form 01^k. All strings are derived from some string in one of these forms, see Fig. 3.

For the induction, suppose $\mathbf{p} = R(\tau(\mathbf{p}))$. The application of the increment operation obviously preserves the truth of the statement. Let $exp(\mathbf{p})$ denote the expansion of $\mathbf{p}$. We have assumed that $p_0p_1\cdots p_{n-2}p_{n-1} = (p_{n-1}-1)p_{n-2}\cdots p_1(p_0+1)$.

But

$$exp(\mathbf{p}) = 01^{p_0}01^{p_1}\cdots 01^{p_{n-2}}01^{p_{n-1}}.$$

Hence

$$\tau(exp(\mathbf{p})) = 11^{p_0}01^{p_1}\cdots 01^{p_{n-2}}01^{p_{n-1}-1}0.$$

Hence

$$R(\tau(exp(\mathbf{p}))) = 01^{p_{n-1}-1}01^{p_{n-2}}\cdots 01^{p_1}01^{p_0+1}$$

$$= 01^{p_0}01^{p_1}\cdots 01^{p_{n-2}}01^{p_{n-1}},$$

where the last equation is obtained from the first. □

Lemma 4 implies that $w = p_1 p_2 \cdots p_{n-2}$ is a palindrome. Since we eventually show that Euclidean strings are rational mechanical, that w is a palindrome is also implied by the fact that the central words are palindromes [15, Corollary 2.2.9]. It also implies that in any Euclidean string of length at least 3 there is another adjacent pair of elements, besides those on the ends, whose exchange results in a string rotationally equivalent to the original. This is the "curious property" noted in the first paragraph of this paper. We make this explicit in the following lemma. Let $swap(\mathbf{p})$ be the string obtained from $\mathbf{p}$ by exchanging elements p_{n-d-1} and p_{n-d}.

Lemma 5. *If* $\mathbf{p}$ *is a Euclidean string with displacement* d, *then* $swap(\mathbf{p}) = \rho^{-d}(\mathbf{p})$.

Proof. From the proof of Lemma 1 we know that $p_{n-d-1} + 1 = p_{n-d}$. Hence the exchange of those two elements is equivalent to a sequence of rotations, reversals and a τ operation as expressed in the following equation:

$$swap(\mathbf{p}) = \rho^{-d}(R(\tau(R(\rho^{d}(\mathbf{p}))))).$$

Thus, by Lemma 4,

$$swap(\mathbf{p}) = \rho^{-d}(R(\tau(\mathbf{p}))).$$

By Lemma 4, $\tau(\mathbf{p}) = R(\mathbf{p})$, and thus

$$swap(\mathbf{p}) = \rho^{-d}(R(R(\mathbf{p}))) = \rho^{-d}(\mathbf{p}). \qquad \square$$

In the example illustrated in Fig. 1, the second exchangeable pair is indicated by the caret symbol.

If $\mathbf{p}$ is a binary string then the complement of $\mathbf{p}$, denoted $C(\mathbf{p})$, is the string obtained from the morphism that sends 0 to 1, and 1 to 0. The following lemma is implicit in [3, Corollary 3.1].

Lemma 6. *If* n *and* k *are relatively prime and* $n > k$, *then* $C(R(E_{n,k})) = E_{n,n-k}$.

Proof. Let $\mathbf{p}$ be $E_{n,k}$, where $n > k$. The weight of $C(\mathbf{p})$ is $n - k$. By Lemma 4, $R(\mathbf{p}) = \tau(\mathbf{p}) = \rho^{d}(\mathbf{p})$. We note that

$$\tau(C(R(\mathbf{p}))) = \tau(C(\tau(\mathbf{p}))) = C(\mathbf{p})$$

and that

$$\rho^{-d}(C(R(\mathbf{p}))) = \rho^{-d}(C(\rho^{d}(\mathbf{p}))) = C(\mathbf{p}).$$

Hence $\tau(C(R(\mathbf{p}))) = \rho^{-d}(C(R(\mathbf{p})))$; i.e., $C(R(\mathbf{p})) = E_{n,n-k}$. □

Definition 4. The morphism $\delta(\mathbf{p})$ is defined for all strings of positive integers $\mathbf{p}$. It is obtained by replacing every integer i in the string by the string $0^{i-1}1$.

The following theorem shows a relationship between $E_{n,k}$ and its "dual" $E_{k,n}$.

Theorem 2. *If n and k are relatively prime and $n<k$, then $\delta(R(E_{n,k})) = E_{k,n}$.*

Proof. We observe that for any string $\mathbf{p}$ of positive integers

$$\delta(\mathbf{p}) = \rho^{-1}(C(exp(inc^{-1}(\mathbf{p})))).$$

Hence,

$$\begin{aligned}\delta(R(\mathbf{p})) &= \rho^{-1}(C(exp(inc^{-1}(R(\mathbf{p}))))) \\ &= \rho^{-1}(C(exp(R(inc^{-1}(\mathbf{p}))))) \\ &= \rho^{-1}(C(\rho(R(exp(R(inc^{-1}(\mathbf{p}))))))) \text{ since } exp(R(\mathbf{p})) = \rho(R(exp(\mathbf{p}))) \\ &= C(R(exp(inc^{-1}(\mathbf{p})))).\end{aligned}$$

Now, suppose $\mathbf{p} = E_{n,k}$ where $n<k$ implying that $E_{n,k}$ is a string of positive integers. From the discussion in Section 2, we have that $inc(E_{n,k-n}) = E_{n,k}$ and $exp(E_{n,k-n}) = E_{k,k-n}$. Hence $exp(inc^{-1}(E_{n,k})) = E_{k,k-n}$. From Lemma, 6 $C(R(E_{k,k-n})) = E_{k,n}$. Hence $\delta(R(E_{n,k})) = E_{k,n}$. □

4. Lyndon words

A *Lyndon* word is one which is lexicographically less than all of its non-trivial rotations (conjugates). We use the symbols $\prec$ and $\succ$ to denote "is lexicographically less than" and "greater than", respectively.

We say that a length n string $\mathbf{p}$ is *rotationally monotone* if there exists an integer e such that

$$\mathbf{p} \prec \rho^{e}(\mathbf{p}) \prec \rho^{2e}(\mathbf{p}) \prec \cdots \prec \rho^{(n-1)e}(\mathbf{p}).$$

For example, 01011 is rotationally monotone with $e=3$, since $01011 \prec 01101 \prec 10101 \prec 10110 \prec 11010$.

Lemma 7. *Every Euclidean string is rotationally monotone.*

Proof. Let $\mathbf{p}$ be a Euclidean string of length n and displacement d. Note that

$$\rho^i(\mathbf{p}) = p_{n-i} \cdots p_{n-2} \underbrace{p_{n-1}\, p_0}\, p_1 \cdots p_{n-i-1}.$$

If $i \neq 0$, then

$$\begin{aligned}\rho^{i+d}(\mathbf{p}) &= \rho^i(\rho^d(\mathbf{p}))\\ &= \rho^i(\tau(\mathbf{p}))\\ &= \rho^i(p_{n-1}p_1 \cdots p_{n-2}p_0)\\ &= p_{n-i} \cdots p_{n-2} \underbrace{p_0\, p_{n-1}}\, p_1 \cdots p_{n-i-1}.\end{aligned}$$

Since $p_0 \prec p_{n-1}$ we have $\rho^{i+d}(\mathbf{p}) \prec \rho^i(\mathbf{p})$. Thus

$$\rho^d(\mathbf{p}) \succ \rho^{2d}(\mathbf{p}) \succ \cdots \succ \rho^{(n-1)d}(\mathbf{p}) \succ \rho^{nd}(\mathbf{p}) = \mathbf{p}$$

and so $\mathbf{p}$ is rotationally monotone with $e = -d$. □

A partial converse of Lemma 7 is also true: every binary string that is rotationally monotone is a Euclidean string. This can be proven by the same sort of reasoning used in the proof of Lemma 1.

It follows from Lemma 7 that every Euclidean string is a Lyndon word. This result was proven earlier in [3, Theorem 3.2] by a different technique.

Corollary 2. *Every Euclidean string is a Lyndon word.*

Definition 5. The *content* of a string is the multiset of characters that occur in the string. In other words, for the string $\mathbf{p} = p_0 p_1 \cdots p_{n-1}$, the content of $\mathbf{p}$, denoted *content*($\mathbf{p}$), is the multiset $\{p_0, \ldots, p_{n-1}\}$.

The following lemma is an immediate consequences of the definitions of the increment and expand functions, denoted *inc* and *exp*, respectively.

Lemma 8. *If* $\mathbf{p}$ *and* $\mathbf{q}$ *are strings, where* $\mathbf{p} \prec \mathbf{q}$, *then* $exp(\mathbf{p}) \prec exp(\mathbf{q})$ *and* $inc(\mathbf{p}) \prec inc(\mathbf{q})$.

Lemma 9. *Among all numeric Lyndon words with the same length n and weight k, where* $n > 1$ and $\gcd(n,k) = 1$, *the lexicographically largest has exactly two symbol types*, $\lfloor k/n \rfloor$ and $\lceil k/n \rceil$.

Proof. Every Lyndon word with $n > 1$ has at least two symbols, and must start with the smallest symbol in its content. Since $\gcd(n,k) = 1$, the values $\lfloor k/n \rfloor$ and $\lceil k/n \rceil$ are distinct. The string $\lfloor k/n \rfloor^s \lceil k/n \rceil^t$, with $s + t = n$ and $t = k \bmod n$, is a Lyndon word, because it is smaller than any of its non-trivial conjugates. Clearly, no length n Lyndon word with weight k could have a first symbol larger than $\lfloor k/n \rfloor$. □

Lemma 10. *If* $\mathbf{q}$ *is a binary Lyndon word different from* 1, *then* exp^{-1} *is well defined and* $exp^{-1}(\mathbf{q})$ *is a Lyndon word.*

Proof. It is well defined since a binary Lyndon word starts with a 0 and ends with a 1 (except for the single character strings 1 and 0; string 1 was excluded, and 0 is a fixed-point of exp^{-1}).

Now suppose that $exp^{-1}(\mathbf{q}) = \mathbf{uv}$, where $\mathbf{vu} \prec \mathbf{uv}$. By Lemma 8, we have $exp(\mathbf{vu}) \prec \mathbf{q}$. But this is a contradiction since $exp(\mathbf{vu}) = exp(\mathbf{v})\,exp(\mathbf{u})$ is a rotation of $\mathbf{q}$. □

Theorem 3. *If* $\mathbf{p}$ *is a Euclidean string and* $\mathbf{q}$ *is a different Lyndon word with the same length and weight*, *then* $\mathbf{q} \prec \mathbf{p}$.

Proof. We argue by induction on the sum of the length and weight of the string. The theorem is obviously true for strings of length one or two. By Lemma 9, we may assume that $content(\mathbf{p}) = content(\mathbf{q})$, and that content consists of two consecutive non-negative integers.

If $\mathbf{p}$ and $\mathbf{q}$ are not binary, then inductively $inc^{-1}(\mathbf{q}) \prec inc^{-1}(\mathbf{p})$, from which Lemma 8 gives $\mathbf{q} \prec \mathbf{p}$.

If $\mathbf{p}$ and $\mathbf{q}$ are both binary strings, then by Lemma 10 $exp^{-1}(\mathbf{q})$ exists and is a Lyndon word. If $content(exp^{-1}(\mathbf{q})) \neq content(exp^{-1}(\mathbf{p}))$, then by Lemma 9 $exp^{-1}(\mathbf{q}) \prec exp^{-1}(\mathbf{p})$. If $content(exp^{-1}(\mathbf{q})) = content(exp^{-1}(\mathbf{p}))$, then inductively, $exp^{-1}(\mathbf{q}) \prec exp^{-1}(\mathbf{p})$. Thus, in either case, by Lemma 8, $\mathbf{q} \prec \mathbf{p}$. □

Lyndon words are counted by length and "weight" in [17]. For example, the number of q-ary Lyndon words of length n and weight equal to $t \bmod q$ is

$$L_q(n,t) = \frac{1}{qn} \sum_{\substack{d|n \\ \gcd(d,q)|t}} \gcd(d,q)\mu(d)q^{n/d}.$$

5. Stern–Brocot strings

In this section we demonstrate an interesting correspondence between the Stern–Brocot tree of reduced fractions and the set of all binary Euclidean strings.

The *Stern–Brocot tree* [11] is an infinite construction of the set of all non-negative fractions k/n between 0 and 1 where k and n are relatively prime. The tree is a rooted binary tree in which each node in the tree is constructed by using its nearest left ancestor (L), and nearest right ancestor (R). A nearest left ancestor of a node u is the ancestor v such that the length of the path v to u is minimum and contains exactly one right child. A nearest right ancestor of a node u is the ancestor v such that the length of the path v to u is minimum and contains exactly one left child. If a node x has $L = k'/n'$ and $R = k''/n''$ then $x = (k'+k'')/(n'+n'')$. To start this recursive construction, L is initialized to 0/1 and R is initialized to 1/1. The Stern–Brocot tree is illustrated in Fig. 4. Traversing the tree "in-order" produces a Farey series [12].

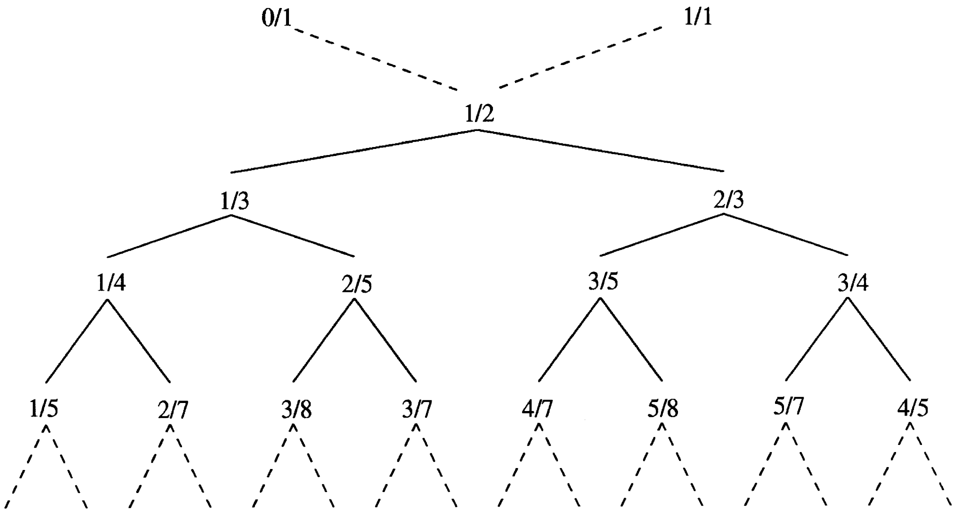

Fig. 4. The Stern–Brocot tree of reduced fractions.

We now construct an equivalent tree composed of binary strings where each fraction k/n in the Stern–Brocot corresponds to a length n binary string with k ones. In this new tree, each string α is the concatenation of its nearest left ancestor L and its nearest right ancestor R. If L and R are initially assigned the characters 0 and 1, respectively, then we obtain the corresponding Stern–Brocot fraction by computing the number of ones in the string along with the length. This tree of strings, denoted $\mathscr{S}$, is illustrated in Fig. 5. Interestingly, this tree includes all and only the binary Euclidean strings.

Theorem 4. *The length n binary string α is Euclidean if and only if α is in $\mathscr{S}$. Furthermore, if $n > 1$ where $\alpha = LR$, then $\tau(LR) = \rho^{|L|}(LR)$.*

Proof. We proceed by induction on the length of the string. Suppose the length n binary string α is in $\mathscr{S}$. In the base cases, strings 0, 1 and 01 are Euclidean and $\tau(01) = 10 = \rho(01)$.

Now consider two cases depending on whether α is a left child or a right child.

If α is a left child then by construction $R = LR'$ where R' is the nearest right ancestor of R. By induction $\tau(LR') = \rho^{|L|}(LR')$ which implies that $\tau(LLR') = \rho^{|L|}(LLR')$. Thus by definition α is Euclidean and $\tau(LR) = \rho^{|L|}(LR)$.

If α is a right child then by construction $L = L'R$ where L' is the nearest left ancestor of L. By induction $\tau(L'R) = \rho^{|L'|}(L'R)$ which implies that $\tau(L'RR) = \rho^{|L|}(L'RR)$. Thus by definition α is Euclidean and $\tau(LR) = \rho^{|L|}(LR)$.

Thus every string in $\mathscr{S}$ is Euclidean. Since this tree corresponds to the Stern–Brocot tree, there exists a length n string in $\mathscr{S}$ with k ones whenever k and n are relatively prime. By Theorem 1, this implies that if α is a binary Euclidean string, then it is in $\mathscr{S}$. $\square$

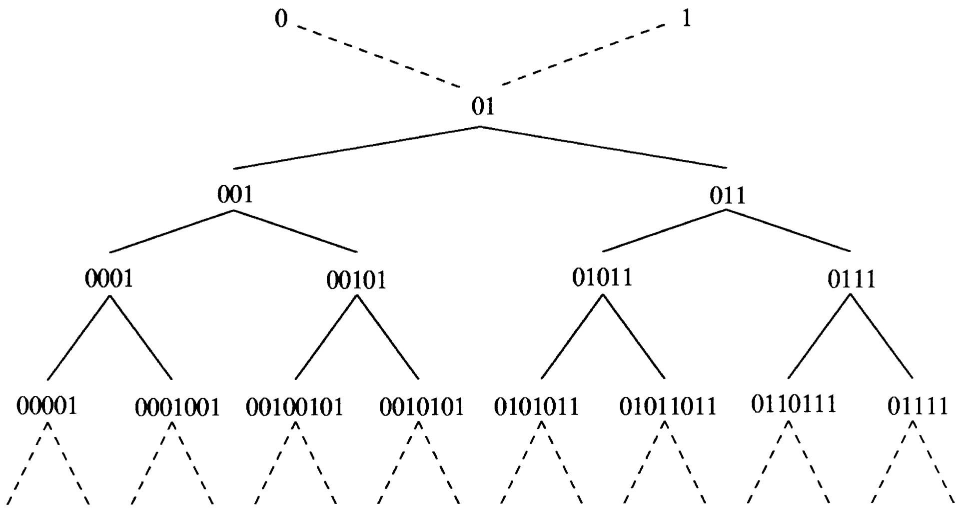

Fig. 5. The Stern–Brocot tree of Euclidean strings.

Theorem 4 implies that every Euclidean string of length greater than or equal to two is the concatenation of two shorter Euclidean strings. We note that this result is similar to Exercise 2.2.3 of [15].

Corollary 3. *Every binary Euclidean string $E_{n,k}$ where $n \geqslant 2$ and with displacement d and cost c is the concatenation of the Euclidean strings $E_{d,c}$ and $E_{n-d,k-c}$.*

Proof. By Theorem 4, the displacement of the binary Euclidean string α is $|L|$ where α is the concatenation of L and R. Since L is binary, the cost of α is the weight of L. □

The tree of binary Euclidean strings, Fig. 5, suggests yet another way of generating any such string, i.e., by way of a sequence of string concatenations, defined by a path down the tree to the node corresponding to the fraction k/n. Corollary 3 implies that the displacement and cost can be computed simultaneously.

The algorithm uses global variables for current left ancestor $L = E_{n_1,k_1}$, for the current right ancestor $R = E_{n_2,k_2}$ and for n and k. The variable L should be initialized to "0" and R to "1". The initial call to generate $E_{n,k}$ is Stern–Brocot(0,0,0,1).

procedure Stern–Brocot (n_1, k_1, n_2, k_2: $\mathbb{Z}^+$);
local $n_3, k_3 : \mathbb{Z}^+$;
    $n_3 := n_1 + n_2$; $k_3 := k_1 + k_2$;
    if $k_3/n_3 < k/n \rightarrow R := LR$;   Stern–Brocot($n_1, k_1, n_3, k_3$);
    [] $k_3/n_3 = k/n \rightarrow \text{print}(LR, n_1, k_1)$;
    [] $k_3/n_3 > k/n \rightarrow L := LR$;   Stern–Brocot($n_3, k_3, n_2, k_2$);
    fi

There are exactly $n-1$ concatenations. Hence, if the strings are represented by linked lists, the time complexity of the algorithm is linear.

All the ancestors of $E_{n,k}$ can be generated by removing the $k_3/n_3 = k/n$ clause and placing the print statement after the update to n_3 and k_3. By Lemma 1(c), the Euclidean string $E_{n,k}$ where $n<k$, can be obtained from the binary string $E_{n,k \bmod n}$ by replacing the zeros by $\lfloor k/n \rfloor$ and the ones by $\lceil k/n \rceil$. The displacement is unchanged, and if the cost of $E_{n,k \bmod n}$ is c then the cost of $E_{n,k}$ is $d\lfloor k/n \rfloor + c$.

6. Fibonacci strings

We go on to show a relationship between Fibonacci and Euclidean strings.

Definition 6. A Fibonacci string is defined by the morphism $b \mapsto a,\ a \mapsto ab$.

For example, the first seven Fibonacci strings are:

$$b, a, ab, aba, abaab, abaababa, abaababaabaab.$$

Let F_i denote the ith Fibonacci string with length f_i. It is known that, $F_i = F_{i-1}F_{i-2}$ and hence that f_i is the ith Fibonacci number. Fibonacci strings occur as the worst case inputs to certain algorithms and they possess many interesting properties [6, 10, 13, 14, 18].

Let $G = G_1, G_2, G_3, \ldots$ be an infinite sequence of Euclidean strings where each G_i is defined as follows:

$$G_i = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{if } i = 2, \\ \mathit{right}(G_{i-1}) & \text{if } i > 2 \text{ and } i \text{ odd}, \\ \mathit{left}(G_{i-1}) & \text{if } i > 2 \text{ and } i \text{ even}, \end{cases}$$

where $\mathit{right}(\alpha)$ is the right child of α in the Stern–Brocot tree of strings $\mathscr{S}$ and $\mathit{left}(\alpha)$ is the left child of α in the tree $\mathscr{S}$. This construction implies that $G_i = G_{i-2}G_{i-1}$ when $i>2$ and i is even; if $i>2$ and i odd, then $G_i = G_{i-1}G_{i-2}$.

Lemma 11. *Every Fibonacci string is conjugate to a Euclidean string. If $a = 0$ and $b = 1$, then*

$$F_i = \begin{cases} G_i & \text{if } i \leqslant 2, \\ \rho^{-(f_{i-2}+1)}(G_i) & \text{if } i > 2 \text{ and } i \text{ odd}, \\ \rho^{-}(G_i) & \text{if } i > 2 \text{ and } i \text{ even}. \end{cases}$$

Proof. This result is easy to verify for $i<5$. For $i\geqslant 5$,

$$\begin{aligned} F_i &= F_{i-1}F_{i-2} \\ &= F_{i-2}F_{i-3}F_{i-2} \\ &= F_{i-3}F_{i-4}F_{i-3}F_{i-3}F_{i-4}. \end{aligned}$$

We now consider two cases depending on the parity of i.

If i is odd then since $i-3$ is even, $F_{i-3}=xa$ for some string x. If we let $F_{i-4}=y$ then we have:

$$F_i = xayxaxay.$$

Now, by induction $F_{i-1}=xayxa=\rho^{-1}(G_{i-1})$, which implies that $G_{i-1}=axayx$. Similarly, $F_{i-2}=xay=\rho^{-(f_{i-4}+1)}(G_{i-2})$, which implies that $G_{i-2}=ayx$. Since i is odd, $G_i=G_{i-1}G_{i-2}=axayxayx$ and $\rho^{-(f_{i-2}+1)}(G_i)=xayxaxay=F_i$.

If i is even then since $i-4$ is even, $F_{i-4}=xa$ for some string x. If we let $F_{i-3}=y$ then we have:

$$F_i = yxayyxa.$$

Now, by induction $F_{i-1}=yxay=\rho^{-(f_{i-3}+1)}(G_{i-1})$, which implies that $G_{i-1}=ayyx$. Similarly, $F_{i-2}=yxa=\rho^{-1}(G_{i-2})$, which implies that $G_{i-2}=ayx$. Since i is even, $G_i=G_{i-2}G_{i-1}=ayxayyx$ and $\rho(G_i)=yxayyxa=F_i$. $\square$

From Lemmas 11 and 2, we see that we can construct the Lyndon word corresponding to any Fibonacci word. One of the referees has pointed out that the results of this section can be generalized to Sturmian words.

7. Beatty sequences

In this section, we show that Euclidean strings are related to Beatty sequences. A Beatty sequence is a sequence of the form $a_j=\lfloor \alpha j+\beta\rfloor$ where $\alpha>1$ and $j\in\mathbb{Z}$. See for example [2,9]. Generally, α and β can be rational or irrational but we will only be concerned with the case where α is rational and $\beta=0$.

Lemma 12. *Consider the Beatty sequence comprising the elements in the set* $a_j=\{\lfloor (n/k)j\rfloor : j\in\mathbb{Z}\}$, *where* $\gcd(n,k)=1$. *For each integer j there exists an i where* $0\leqslant i\leqslant k-1$ *such that*

$$id+\left\lfloor \frac{n}{k}j\right\rfloor \equiv 0 \bmod n,$$

where d is the multiplicative inverse of k, modulo n.

Proof. Since $dk\equiv 1 \bmod n$, for any z, $z\equiv zdk \bmod n$. Hence, for any j

$$-\left\lfloor \frac{n}{k}j\right\rfloor \equiv \left(nj-k\left\lfloor \frac{n}{k}j\right\rfloor\right) d \bmod n.$$

Consider the term in parentheses, which is a way of writing $nj \bmod k$. Because n and k are relatively prime, so also are $n \bmod k$ and k. Hence the term in parentheses can take on any integer value in the interval $[0, k-1]$. Thus for any j there exists $i \in [0, k-1]$ such that

$$id + \left\lfloor \frac{n}{k} j \right\rfloor \equiv 0 \bmod n.$$

Similarly for any $i \in [0, k-1]$ there exists j satisfying this congruence. □

Corollary 4. *The string* $p_0 p_1 \cdots p_{n-1}$ *is* $E_{n,k}$ *where* $p_i = |\{j : n - \lfloor (n/k)j \rfloor - 1 = i,\ 0 \leqslant j < k\}|$.

Proof. By Lemma 1(c), $p_i = |\{j : n-1+jd \equiv i \bmod n,\ 0 \leqslant j < k\}|$, where d is the displacement of the string. By Lemma 1(d), $dk \equiv 1 \bmod n$. Hence, by Lemma 12, the set on the right-hand side of the equation is identical to the set $\{j : (n - \lfloor (n/k)j \rfloor - 1) = i,\ 0 \leqslant j < k\}$. □

This gives us one more algorithm for computing a Euclidean string. First, initialize $p_0 p_1 \cdots p_{n-1}$ to be all zeros. Then execute the following line of code, which uses the C increment operator.

for $j = 0, 1, \ldots, k-1$ **do** $++p_{n - \lfloor (n/k)j \rfloor - 1}$

That is, a Euclidean string is the reversal of the histogram of a rational Beatty sequence.

As previously noted, it is faster to reduce an (n,k) instance where $k > n$ to the instance $(n, k \bmod n)$. After the application of Corollary 4, we construct the correct string by replacing the 1's by $\lceil k/n \rceil$ and the 0's by $\lfloor k/n \rfloor$.

8. Relationships with Sturmian sequences

Following [15, p. 59], we define the *rational mechanical* words, for the rational number $0 \leqslant p/q \leqslant 1$ with $\gcd(p,q) = 1$, as the finite words

$$t_{p,q} = a_0 a_1 \cdots a_{q-1}, \quad t'_{p,q} = a'_0 a'_1 \cdots a'_{q-1},$$

where

$$a_i = \left\lfloor (i+1)\frac{p}{q} \right\rfloor - \left\lfloor i\frac{p}{q} \right\rfloor, \quad a'_i = \left\lceil (i+1)\frac{p}{q} \right\rceil - \left\lceil i\frac{p}{q} \right\rceil.$$

These words are also known as *Christoffel words* and as *characteristic words* [1].

Theorem 5. *Let n and k be relatively prime with $0 < k < n$. Then $E_{n,k} = t_{k,n}$.*

Proof. Recall Corollary 4. If $n > k > 0$ then for all indices i, where $i \in \{n - \lfloor (n/k)j \rfloor - 1 : 0 \leqslant j \leqslant k-1\}$, in the Euclidean word $E_{n,k}$ we have $p_i = 1$, and $p_i = 0$ for all other indices.

Suppose that $p_i=0$. Then there exists j, $0\leqslant j<k-1$ such that

$$n-\left\lfloor\frac{n}{k}j\right\rfloor-1>i>n-\left\lfloor\frac{n}{k}(j+1)\right\rfloor-1.$$

That is,

$$\begin{aligned}
&n-\left\lfloor\frac{n}{k}j\right\rfloor-1>i \quad\text{and}\quad i>n-\left\lfloor\frac{n}{k}(j+1)\right\rfloor-1\\
&\quad\Rightarrow n-1-i>\left\lfloor\frac{n}{k}j\right\rfloor \quad\text{and}\quad \left\lfloor\frac{n}{k}(j+1)\right\rfloor\geqslant n-i\\
&\quad\Rightarrow n-1-i>\frac{n}{k}j \quad\text{and}\quad \frac{n}{k}(j+1)\geqslant n-i\\
&\quad\Rightarrow k-\frac{k}{n}-\frac{ik}{n}>j \quad\text{and}\quad j+1\geqslant k-\frac{ik}{n}\\
&\quad\Rightarrow k-j>\frac{k}{n}(1+i) \quad\text{and}\quad \frac{k}{n}i\geqslant k-j-1\\
&\quad\Rightarrow \left\lfloor\frac{k}{n}(1+i)\right\rfloor=\left\lfloor\frac{k}{n}i\right\rfloor\\
&\quad\Rightarrow \left\lfloor\frac{k}{n}(1+i)\right\rfloor-\left\lfloor\frac{k}{n}i\right\rfloor=0.
\end{aligned}$$

Thus $a_i=0$ in $t_{k,n}$. The implications may all be reversed yielding $a_i=1$ in $t_{k,n}$ if and only if $p_i=1$ is in the Euclidean word $E_{n,k}$. Thus the words are the same. □

9. An application

The original motivation for studying these strings came from a graph embedding problem, in particular, the problem of many-to-one mappings from the nodes of a two-dimensional grid onto the nodes of a torus and thence into a hypercube. Here we require that the size of the grid is maximum with respect to the size of the torus and some specified “load”, i.e., the maximum number of grid nodes that can be mapped onto a single torus node. We ask whether there exists a mapping with dilation one, i.e., in which any pair of grid nodes connected by an edge are either mapped to the same torus node, or to the ends of an edge in the torus. A solution to this problem which uses the analysis of Euclidean strings is given in [7], for loads $\geqslant 4$.

We give a very informal description of why the Euclidean strings are useful. Consider Fig. 6. This diagram defines a mapping from part of a 13×21 grid onto an 8×13 torus with load 2, dilation 1. For each element in the grid, the number at each grid position specifies the torus row onto which the grid node is to be mapped. The shading distinguishes torus columns, numbered across the top. Thus for example, the second grid element in the top row of the grid is to be mapped to row 8, column 1 of the torus.

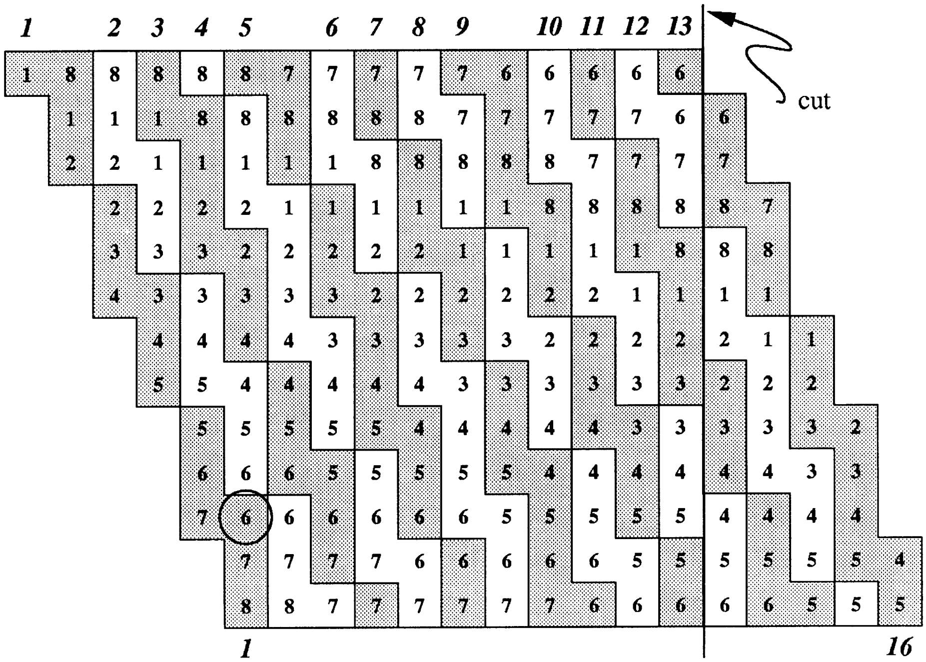

Fig. 6. Grid to torus mapping.

Note that row 1 of the torus is adjacent to row 8 and that column 1 is adjacent to column 13. In the mapping defined in Fig. 6, no torus (row, column) pair appears more than once, so the mapping is load 2. Note also that adjacent grid nodes are either mapped to the same row in adjacent torus columns, or to adjacent rows in the same torus column. Hence the mapping is a dilation 1 mapping.

Suppose we chop off the "steps" at the right-hand end, as indicated by the "cut" and insert the detached piece at the left end. Note that the shapes at the ends "match". If the numbers also matched (a separate problem) we would have a load 2, dilation 1 embedding of the 13×16 grid into the 8×13 torus with load 2, dilation 1.

We may define the left and right "profiles" of a torus column in the diagram to be the sequence of step heights on the left or right of the column taken circularly. For example, the left profile of column 2 is (3, 2, 3, 2, 3). Note that each successive profile is a rotation of its predecessor. For the cut and paste method to work we want the left profile of the leftmost column to match the right profile of the rightmost column.

The Euclidean string analysis permits us to argue that it is always possible to find a profile such that left and right end profiles of the pattern match. The profile pattern in Fig. 6 is periodic with period 13. In general, the width of the grid is not a multiple of this period. Further, it is necessary to "drop" one of more torus nodes from the pattern because the number of nodes in the grid is usually strictly less than the number of nodes in the torus times the load.

Consider the effect of dropping one torus node from the profile. For example, let us drop the circled element from column 1. One element in the profile is incremented and one is decremented by one. The resulting profile is now (3, 2, 3, 3, 2), which is a rotation of the unmodified profile and which matches the left profile of column 6.

The analysis of Euclidean strings establishes that we can always find a profile with the properties we need, namely that the result of dropping an element is a profile that is rotationally equivalent to the original.

Acknowledgements

The authors are grateful to the referees for carefully reading the paper and pointing out that Euclidean strings are related to Sturmian sequences. To our chagrin, we then discovered that there is a sizeable literature on Sturmian sequences, and on the Christoffel words in particular, and that some of what we thought we had discovered was already known, if in a slightly different form. We also thank Jeff Shallit for helpful discussion.

References

[1] J.-P. Allouche, J.O. Shallit, Automatic Sequences, Cambridge University Press, Cambridge, 2002.
[2] S. Beatty, Problem 3173, Amer. Math. Monthly 33 (1926) 159.
[3] J. Berstel, A. de Luca, Sturmian words, Lyndon words and trees, Theoret. Comput. Sci. 178 (1997) 171–203.
[4] N. Calkin, H.S. Wilf, Recounting the rationals, Amer. Math. Monthly 107 (2000) 360–363.
[5] T.H. Corman, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, McGraw-Hill, New York, 1990.
[6] X. Droubay, Palindromes in the Fibonacci word, Inform. Process. Lett. 55 (1995) 217–221.
[7] J. Ellis, S. Chow, D. Manke, Embedding grids into cylinders, tori and hypercubes, SIAM J. Comput., to appear.
[8] J. Ellis, F. Ruskey, J. Sawada, Euclidean strings, in: L. Brankovic, J. Ryan (Eds.), Proc. Eleventh Australian Workshop on Combinatorial Algorithms, University of Newcastle, Australia, 2000, pp. 87–92.
[9] A.S. Fraenkel, The bracket function and complementary sets of integers, Canad. J. Math. 21 (1967) 6–27.
[10] A.S. Fraenkel, J. Simpson, The exact number of squares in Fibonacci words, Theoret. Comput. Sci. 218 (1999) 95–106.
[11] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics, Addison-Wesley, Reading, MA, 1989.
[12] G.H. Hardy, E.M. Wright, An Introduction to the Theory of Numbers, Clarendon Press, Oxford, 1960.
[13] C.S. Iliopoulos, D. Moore, W.F. Smyth, A characterization of the squares in a Fibonacci string, Theoret. Comput. Sci. 172 (1–2) (1997) 281–291.
[14] C.S. Iliopoulos, D. Moore, W.F. Smyth, The covers of a circular Fibonacci string, J. Combin. Math. Combin. Comput. 26 (1998) 227–236.
[15] M. Lothaire, Algebraic Combinatorics on Words, Cambridge University Press, Cambridge, 2002.
[16] A. de Luca, F. Mignosi, Some combinatorial properties of Sturmian words, Theoret. Comput. Sci. 136 (1994) 361–385.
[17] F. Ruskey, C.R. Miers, J. Sawada, The number of Lyndon words and irreducible polynomials of given trace, SIAM J. Discrete Math. 14 (2001) 240–245.
[18] W. Zhi-Xiong, W. Zhi-Ying, Some properties of the singular words of the Fibonacci word, Séminaire Lotharingien de Combinatoire, 30, 1993.