Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

EFFICIENT GENERATION OF SOME GREEDY BINARY GRAY CODES

Nathanaël Hassler[1], Vincent Vajnovszki[2] and Dennis Wong[3]

Abstract. In 2013, Aaron Williams introduced the notion of a greedy Gray code algorithm and reinterpreted known Gray codes in a unified manner using greedy algorithms. Recently, this notion was further generalized and investigated by Merino, Mütze, and Williams in 2022, and by Merino and Mütze in 2024, in the context of generating the bases of a matroid or the spanning trees of a graph, among other combintorial structures.

In this article, we investigate the existence of homogeneous greedy Gray codes for Fibonacci words and generalized Dyck prefixes. We also establish useful properties and provide efficient generation algorithms for them.

2020 Mathematics Subject Classification. 05A05, 05A10, 68R15.

1. Introduction

A Gray code for a class of combinatorial objects is a list that contains each object from the class exactly once, such that any two consecutive objects in the list differ only in a predefined "small" way. In [9], Aaron Williams introduced the notion of a greedy Gray code algorithm and reinterpreted known Gray codes in a unified way using the framework of greedy algorithms. He mentions that the greedy method is not suitable for efficiently generating Gray codes, as it may require storing an exponential number of objects. This notion was subsequently extended and studied in the context of various combinatorial objects in [4, 5].

Based on Williams' original greedy algorithm, the authors of the present paper provided in [3] Gray codes and corresponding generation algorithms for several classes of constrained binary words: Fibonacci words, (generalized) Dyck words,

[1] LIB, Université de Bourgogne Europe, Dijon, France, `nathanael.hassler@ens-rennes.fr`
[2] LIB, Université de Bourgogne Europe, Dijon, France, `vvajnov@u-bourgogne.fr`
[3] Polytechnic University Macao, China, `cwong@uoguelph.ca`

and their prefixes. In the present article, depending on the choice of the first element in the code, we investigate the existence of homogeneous greedy Gray codes for these classes. Moreover, we show that the resulting codes possess useful properties, enabling the development of efficient generation algorithms.

1.1. Constrained binary words

1.1.1. *1s run constrained binary words*

Let $n, p \in \mathbb{N}$, $p \geq 2$, and $F_p(n)$ be the set of length n binary words with no p consecutive 1s. We also set $F_p = \bigcup_n F_p(n)$. $F_2(n)$ is counted by the Fibonacci numbers f_n, and in general, $F_p(n)$ is counted by the p-order Fibonacci numbers $f_n^{(p)}$. Now let $k \in \mathbb{N}$, and $F_p(n,k)$ be the subset of words in $F_p(n)$ of weight k (i.e., with exactly k 1s). $F_p(n,k)$ is counted by the univariate p-nomial coefficient:

$$|F_p(n,k)| = \binom{n-k+1}{k}_p \tag{1}$$

where $\binom{m}{j}_p$ for $0 \leq j \leq (p-1)m$ is the coefficient of x^j in the polynomial $(1+x+\ldots+x^{p-1})^m$:

$$(1+x+\ldots+x^{p-1})^m = \sum_{j=0}^{(p-1)m} \binom{m}{j}_p x^j. \tag{2}$$

Remark 1.1. $\binom{m}{j}_p$ also counts the number of compositions of j into m parts, where each part belongs to $\{0, \ldots, p-1\}$.

Note that when $p = 2$, we get the usual binomial coefficient, $\binom{m}{j}_2 = \binom{m}{j}$. Since $F_p(n,k)$ for $0 \leq k \leq n$ is a partition of $F_p(n)$, we get the following formula:

$$f_n^{(p)} = \sum_{k=0}^{n} \binom{n-k+1}{k}_p. \tag{3}$$

1.1.2. *Prefix constrained binary words*

Let $k, n, p \in \mathbb{N}$ with $(p+1)k \leq n$, and $C_p(n,k)$ be the set of length n binary words of weight k with the property that any prefix contains at least p times as many 0s as 1s. Further, let $C_p = \bigcup_{n,k} C_p(n,k)$. In particular:

- $C_0(n,k)$ is the set of length n binary words of weight k (combinations in binary word representation),
- $C_1(2n,n)$ is the set of length $2n$ Dyck words, and it is counted by the Catalan numbers $\binom{2n}{n} - \binom{2n}{n-1} = \frac{1}{n+1}\binom{2n}{n}$,
- $C_2(3n,n)$ is in bijection with size $3n$ ternary trees (see A001764 in OEIS).

More generally, $C_p((p+1)n, n)$ is counted by $\frac{1}{pn+1}\binom{(p+1)n}{n}$, known as the Pfaff–Fuss–Catalan numbers, and the cardinality of $C_p(n,k)$ is established for instance in [8, Equation (2)], using generating functions:

$$|C_p(n,k)| = \binom{n}{k} - p\binom{n}{k-1}. \tag{4}$$

1.2. Gray codes and greedy algorithms

As mentioned in Introduction, our Gray codes for binary words are based on the initial work in [9], which has been further developed recently in [4, 5]. The greedy algorithm in Definition 1 below, introduced originally in [4, 5] (where it is referred to as Algorithm G), aims to minimize the length of the changed prefix in the transition from one word to the next in a Gray code for a set of binary words.

Definition 1. Algorithm G that attempts to greedily compute $\mathcal{L}$, a Gray code for the set of binary words S.

(1) Initialize $\mathcal{L}$ with a particular word in S.
(2) Consider the set N of words in S but not in $\mathcal{L}$ that differ from the last word in $\mathcal{L}$ in the predefined "small" way that characterizes the Gray code.
(3) If the set N from step (2) is not empty, *choose* a word $y \in N$ such that the length of the changed prefix in the transition from the last word in $\mathcal{L}$ to y is minimized. Then append y to $\mathcal{L}$, and return to step (2).
Otherwise, terminate the algorithm.

Seemingly, there is a lack of accuracy in point 3 of the previous definition: '*choose* a word y' (called tie-breaking rule in [4, 5]) could mean either drawing at random or selecting according to a specified rule; we will adopt the latter interpretation in the algorithm in Definition 2 below.

A list of words is said to be *suffix-partitioned* if all words sharing the same suffix appear consecutively in the list. This property is also referred to in the literature as having the *genlex* property. In [5, Theorem 5], a powerful result is stated: if a Gray code $\mathcal{L}$ for a set of binary words is prefix-partitioned, then it can be generated by Algorithm G, equipped with a specific rule (depending on $\mathcal{L}$) for choosing $y \in N$ in step (3).

In this paper we restrict ourselves to Gray codes for restricted classes of same length and same weight binary words (the weight of a binary word is its number of 1s). The "small" changes we consider here are homogeneous transpositions: two binary words differ by a *homogeneous* transposition if one can be obtained from the other by transposing a 1 with a 0, and there are no 1s between the transposed bits. A Gray code is called homogeneous if consecutive words differ in a such a way.

Let S be a set of binary words of the same length and same weight, and let $\mathcal{L}$ be a list of words from a subset of S. Let also $H(S, \mathcal{L})$ denote the set of all pairs (u, v), with $u > v$, such that the transposition of the bits at positions u and v in the last word of $\mathcal{L}$ is homogeneous and yields a word that belongs to S but not to $\mathcal{L}$.

Defining Gray codes, it is reasonable to maximize the length of the common suffix of consecutive words, and the following algorithm is both a specialization of the one in Definition 1 to homogeneous transpositions and achieved by minimizing the largest transposed position, and a generalization of the algorithm $\mathrm{G}_{\mathrm{hom}}$ from [4] to classes of binary words beyond combinations.

Definition 2. Specialization of the algorithm in Definition 1.

(1) Initialize $\mathcal{L}$ with a particular word in S.
(2) Consider the set $H(S, \mathcal{L})$ defined above of possible homogeneous transpositions.
(3) If $H(S, \mathcal{L})$ from step (2) is not empty, consider (u, v), the lexicographically smallest pair in $H(S, \mathcal{L})$. Then append to $\mathcal{L}$ the word obtained by transposing the bits at positions u and v in the last word of $\mathcal{L}$, and return to step (2).
Otherwise, terminate the algorithm.

Note that the Gray codes defined later in this paper are purely based on this algorithm (which by a slight abuse of language, will be referred to it as the *homogeneous greedy algorithm*). Except for a few rare cases (see the next remark), alternative and more classical definitions for the lists produced by this algorithm remain to be established. In this algorithm, the choice of the initial word is critical.

Example 1.2. Let S be the set $C_0(4, 2)$ (the set of length four binary words with two 1s), then the homogeneous greedy algorithm for $C_0(4, 2)$, starting with

(i) 1001 produces the list $1001, 0101, 0011$;
(ii) 0110 produces the list 0110, 1010, 1100 1001, 0101, 0011;
(iii) 0011 produces the list $0011, 1001, 0101, 0110, 1010, 1100$.

It is worth noting that the list in (i) is not exhaustive for S, and that the first word in the list in (iii) is the same as the last word in the list in (ii); however, the two lists are not reverses of each other. See also the examples in Table 1.

Remark 1.3.

- When $p = 0$, k is odd and the initial word is 1^k0^{n-k}, the homogeneous greedy algorithm for $C_0(n, k)$ produces Eades-McKay's Gray code, see [2, 9].
- Experimental evidences suggests that when $p = 1$, $n = 2k$ and the initial word is $(01)^k$, the homogeneous greedy algorithm for the set $C_1(n, k)$ produces Bultena-Ruskey's Gray code, see [1]. See Table 1 (b) for this Gray code when $p = 1$ and $k = 4$.

The easy-to-understand Proposition 1.4 hereafter, which we will need later, requires a few technical definitions. The *tail* of a binary word is its longest suffix of the form $011\cdots1$, and the only words with no tail have the form $11\cdots1$. A list of binary words is *increasing* (resp. *decreasing*) *tail-partitioned* if words with tails of length ℓ appear before (resp. after) words with tail of length $\ell + 1$, for any $\ell \geq 1$.

Definition 3. A list $\mathcal{L}$ of same length binary words is *recursive tail-partitioned* if

- it is increasing or decreasing tail-partitioned, and
- for any tail t, the list obtained by: (i) considering the sublist of $\mathcal{L}$ of words with tail t, then (ii) erasing the tail t in each word of this sublist, is in turn recursive tail-partitioned.

In the following $\cdot$ denotes the concatenation (of two words, or of each word in a list with a word) and the comma appends lists.

With this notation, $\mathcal{L}$ is a *recursive tail-partitioned* list if it has the form

$$\mathcal{L} = \mathcal{L}_1 \cdot 01^{u_1}, \mathcal{L}_2 \cdot 01^{u_2}, \cdots, \mathcal{L}_\ell \cdot 01^{u_\ell} \tag{5}$$

for some increasing or decreasing consecutive integers $u_1, u_2, \ldots, u_\ell$, and each list $\mathcal{L}_i$ is in turn recursive tail-partitioned.

Recall that a list of words is suffix-partitioned if the words with the same suffix are consecutively in the list. Clearly, a recursive tail-partitioned list (r-t partitioned list for short) is a suffix-partitioned list.

Proposition 1.4. *If the list $\mathcal{L}$ is a homogeneous and suffix-partitioned Gray code for a set of same length and same weight binary words, then $\mathcal{L}$ is an r-t partitioned list.*

Proof. Since $\mathcal{L}$ is suffix-partitioned, all the words with same tail t form a contiguous sublist in $\mathcal{L}$. But since we allow only homogeneous transpositions, if t and s are two tails such that the t-tail sublist and the s-tail sublist are consecutive in $\mathcal{L}$, then t and s only differ by one 1. Hence $\mathcal{L}$ is increasing or decreasing tail-partitioned, so it can be written as in relation (5) for some $u_1, u_2, \ldots, u_\ell$, and lists $\mathcal{L}_i$. Now, each $\mathcal{L}_i$ satisfies the same hypotheses as $\mathcal{L}$. Indeed, since $\mathcal{L}$ is a homogeneous and suffix-partitioned Gray code for same length and same weight binary words, so are the $\mathcal{L}_i$s. Thus, recursively we get that $\mathcal{L}$ is r-t partitioned. $\square$

2. Characterization of the first and the last words in the Gray code

In this section, we show that for any $\alpha \in F_2(n,k)$ (resp. $\alpha \in C_p(n,k)$), applying the homogeneous greedy algorithm (given in Definition 2) for $F_2(n,k)$ (resp. for $C_p(n,k)$) starting from α yields a suffix-partitioned list. Moreover, for each α, we characterize the last word in the obtained list and identify those words α (called generators) for which the algorithm exhaustively produces $F_2(n,k)$ (resp. $C_p(n,k)$). In particular, Propositions 2.5 and 2.9 state that, with an appropriate choice of the initial word (a generator), this algorithm produces homogeneous Gray codes for $F_2(n,k)$ and $C_p(n,k)$.

2.1. Fibonacci words $F_2(n,k)$

For $n < 2k-1$, $F_2(n,k)$ is empty and in two particular cases $F_2(n,k)$ is a singleton set.

Fact 2.1. If $k = 0$, then $F_2(n, k) = \{0^n\}$, and if $n = 2k - 1$, then $F_2(n, k) = \{1(01)^{k-1}\}$.

For $\alpha \in F_2(n, k)$ we denote by $G(\alpha, F_2)$ the list obtained by applying the homogeneous greedy algorithm for F_2 starting with α. For instance, the list $G(0101000, F_2)$ is given in Table 1 (a).

For n, k with $n \geq 2k$, we define

- $\alpha^i_{n,k} := 0^i 1(01)^{k-1} 0^{n-2k+1-i}$, for $0 \leq i \leq n - 2k + 1$, and
- $\gamma_{n,k} := \alpha^{n-2k+1}_{n,k} = 0^{n-2k}(01)^k$.

Furthermore, let $\mathcal{F}(n, k)$ denote the set of generators, that is, words $\alpha \in F_2(n, k)$ for which $G(\alpha, F_2)$ contains every word of $F_2(n, k)$.

Theorem 2.2. *Let $n \in \mathbb{N}^\star$. Then for all k with $n \geq 2k$, we have*

(1) $\alpha^i_{n,k} \in \mathcal{F}(n, k)$ for all $0 \leq i \leq n - 2k$, and $G(\alpha, F_2)$ is a suffix-partitioned list, with $\gamma_{n,k}$ as last word.

(2) $\gamma_{n,k} \in \mathcal{F}(n, k)$, and $G(\alpha, F_2)$ is a suffix-partitioned list and its last word is
   (i) $\alpha^0_{n,k}$ if k is even,
   (ii) $\alpha^{n-2k}_{n,k}$ if k is odd.

Proof. The statement is trivially true if $n = 1, 2$ or $k = 0$, and we proceed by induction on n. For $n \in \mathbb{N}^\star$, let (P_n) be the following statement: 'for all k, $0 < k \leq \lfloor \frac{n}{2} \rfloor$, points 1 and 2 of the present theorem hold'. Let $n \geq 3$ and suppose that (P_m) holds for all $m < n$. Let $0 < k \leq \lfloor \frac{n}{2} \rfloor$.

(1) Let $0 \leq i \leq n - 2k$. We have $\alpha^i_{n,k} = \alpha^i_{n-1,k} \cdot 0$ Let us first assume that $i \neq n - 2k$, so that $\alpha^i_{n-1,k} \neq \gamma_{n-1,k}$. Then, by definition of the homogeneous greedy algorithm (it first changes the leftmost bits), $G(\alpha^i_{n,k}, F_2)$ starts with $G(\alpha^i_{n-1,k}, F_2) \cdot 0$. By the induction hypothesis, the last word of $G(\alpha^i_{n-1,k}, F_2) \cdot 0$ is $\gamma_{n-1,k}$, so the next word in $G(\alpha^i_{n,k}, F_2)$ is $\alpha^i_{n-2,k-1} \cdot 01$. Again using the definition of the homogeneous greedy algorithm, we have $G(\alpha^i_{n,k}, F_2) = G(\alpha^i_{n-1,k}, F_2) \cdot 0, G(\alpha^i_{n-2,k-1}, F_2) \cdot 01$. By the induction hypothesis, $G(\alpha^i_{n,k}, F_2)$ is exhaustive because $F_2(n, k) = F_2(n-1, k) \cdot 0 \sqcup F_2(n-2, k-1) \cdot 01$, it is suffix-partitioned and its last word is $\gamma_{n,k}$. Now if $i = n - 2k$, we have $\alpha^i_{n,k} = \gamma_{n-1,k} \cdot 0$. Similarly, $G(\alpha^i_{n,k}, F_2)$ starts with $G(\gamma_{n-1,k}, F_2) \cdot 0$. Then we have two possibilities.
   (i) If k is even, by the induction hypothesis we have $G(\alpha^i_{n,k}, F_2) = G(\gamma_{n-1,k}, F_2) \cdot 0, G(\alpha^0_{n-2,k-1}, F_2) \cdot 01$ and we can conclude as before.
   (ii) If k is odd, then we have $G(\alpha^i_{n,k}, F_2) = G(\gamma_{n-1,k}, F_2) \cdot 0, G(\alpha^{n-2k}_{n-2,k-1}, F_2) \cdot 01$ and we conclude similarly.

(2) We have $\gamma_{n,k} = \gamma_{n-2,k-1} \cdot 01$.
   (i) If $k - 1$ is even, then k is odd. By the induction hypothesis, we have $G(\gamma_{n,k}, F_2) = G(\gamma_{n-2,k-1}, F_2) \cdot 01, G(\alpha^0_{n-1,k}, F_2) \cdot 0$. The last word of $G(\gamma_{n,k}, F_2)$ is $\alpha^{n-2k}_{n,k}$, and we can conclude.
   (ii) If $k - 1$ is odd, then k is even. By the induction hypothesis, we have

$G(\gamma_{n,k}, F_2) = G(\gamma_{n-2,k-1}, F_2) \cdot 01, G(\gamma_{n-1,k}, F_2) \cdot 0$. The last word of $G(\gamma_{n,k}, F_2)$ is $\alpha^0_{n,k}$, and we can conclude.

□

In the next lemma, we extend the result of (1) in Theorem 2.2 about the last element of $G(\alpha, F_2)$ to each $\alpha \in F_2(n, k)$.

Lemma 2.3. *Let $n \geq 2k$. For any $\alpha \in F_2(n, k)$ with $\alpha \neq \gamma_{n,k}$, the last word of $G(\alpha, F_2)$ is $\gamma_{n,k}$.*

Proof. Let $\alpha \in F_2(n, k)$ with $\alpha \neq \gamma_{n,k}$. If $\gamma_{n,k}$ appears in $G(\alpha, F_2)$, then it has a predecessor. By the homogeneous greedy algorithm, this predecessor has the form $0^i10^{n-2k+1-i}(01)^{k-1}$ with $0 \leq i \leq n-2k$, and each word with the suffix $(01)^{k-1}$ has appeared before in the listing. Thus, if $\gamma_{n,k}$ is in $G(\alpha, F_2)$, then it is the last word. Now we prove that $\gamma_{n,k}$ appears in $G(\alpha, F_2)$. Let j be the greatest integer such that there exists a word with suffix $(01)^j$ in $G(\alpha, F_2)$, and let β be the last word with suffix $(01)^j$ in $G(\alpha, F_2)$, so β has suffix $0(01)^j$. Then we have either that β is the last word, or its successor has suffix $0(01)^{j-1}$. In both cases, by the homogeneous greedy algorithm, every word with suffix $(01)^{j+1}$ has appeared before in $G(\alpha, F_2)$, which is a contradiction with the definition of j, unless $j = k$. Thus, $\gamma_{n,k}$ does appear in $G(\alpha, F_2)$. □

Now we generalize the property for $G(\alpha, F_2)$ to be suffix-partitioned to each word of $F_2(n, k)$.

Lemma 2.4. *Let $n \geq 2k-1$. For any $\alpha \in F_2(n, k)$, $G(\alpha, F_2)$ is a suffix-partitioned list.*

Proof. We proceed by induction on n. This is easy to check for $n = 1, 2, 3$. Let $n \geq 4$ and assume that for $m \in \{n-1, n-2\}$ we have $G(\alpha, F_2)$ is suffix-partitioned for any $\alpha \in F_2(m, k)$ with $m \geq 2k-1$. Let k be such that $n \geq 2k-1$ and $\alpha \in F_2(n, k)$.

- If the last bit of α is 0, then $\alpha = \alpha' \cdot 0$ with $\alpha' \in F_2(n-1, k)$. If $\alpha = \alpha^i_{n,k}$ for some $0 \leq i \leq n-2k+1$, then we can conclude by Theorem 2.2. Otherwise, we can assume $\alpha' \neq \gamma_{n-1,k}$, so by the homogeneous greedy algorithm and Lemma 2.3, $G(\alpha, F_2) = G(\alpha', F_2) \cdot 0, G(\gamma_{n-2,k-1}, F_2) \cdot 01$. By the induction hypothesis, $G(\alpha, F_2)$ is suffix-partitioned.
- If the last bit of α is 1, then $\alpha = \alpha' \cdot 01$ with $\alpha' \in F_2(n-2, k-1)$. Once again if $\alpha = \gamma_{n,k}$ we can conclude by Theorem 2.2, otherwise $\alpha' \neq \gamma_{n-2,k-1}$. Thus, by the homogeneous greedy algorithm and Lemma 2.3 (applied to α and α'), $G(\alpha, F_2) = G(\alpha', F_2) \cdot 01$. So by the induction hypothesis, $G(\alpha, F_2)$ is suffix-partitioned.

□

Now we are able to describe exactly the set $\mathcal{F}(n, k)$ of generators of $F(n, k)$.

Proposition 2.5. *Let $n \geq 2k-1$. Then*

$$\mathcal{F}(n, k) = \{0^i1(01)^{k-1}0^{n-2k+1-i} \mid 0 \leq i \leq n-2k+1\}.$$

In particular, $|\mathcal{F}(n,k)| = n-2k+2$.

Proof. The inclusion from right to left has been established in Theorem 2.2. Let $\alpha \in \mathcal{F}(n,k)$. Suppose that the last bit of α is 1, then α has the suffix 01. By Lemma 2.3, if $\alpha \neq \gamma_{n,k}$, then $\gamma_{n,k}$ is the last word of $G(\alpha, F_2)$. By Lemma 2.4, this means that every word of $G(\alpha, F_2)$ has suffix 01. So the only possible element of $\mathcal{F}(n,k)$ with last bit 1 is $\gamma_{n,k}$. Now suppose that the last bit of α is 0. Then $\alpha = \alpha' \cdot 0$ and by the homogeneous greedy algorithm and Lemma 2.4, $G(\alpha', F_2) \cdot 0$ must contain every word of $F_2(n,k)$ ending with a 0. We conclude that $\alpha' \in \mathcal{F}(n-1,k)$. Then we have either the last bit of α' is 1, and $\alpha' = \gamma_{n-1,k}$, or $\alpha' = \alpha'' \cdot 0$ with $\alpha'' \in \mathcal{F}(n-2,k)$. By iterating this process we get that $\alpha = \alpha^i_{n,k}$ for some $0 \le i \le n-2k+1$, hence the result. $\square$

2.2. $C_p(n,k)$

For n,p,k such that $n \ge (p+1)k$ and $\alpha \in C_p(n,k)$, we denote by $G(\alpha, C_p)$ the list obtained by applying the homogeneous greedy algorithm for C_p starting with α. For instance, the list $G(01010101, C_1)$ is given in Table 1 (b). Let also $\mathcal{C}_p(n,k)$ be the set of words $\alpha \in C_p(n,k)$ such that $G(\alpha, C_p)$ is a homogeneous Gray code for $C_p(n,k)$. Equivalently, $\alpha \in \mathcal{C}_p(n,k)$ if and only if $G(\alpha, C_p)$ contains every word of $C_p(n,k)$. Since the situation is a bit more complicated than the one for Fibonacci words, and for clarity, we first investigate the case $p \in \mathbb{N}$, explaining all details of the proofs. Then, we explain how the results are generalized to any $p \in \mathbb{R}$ omitting the proofs, which are are similar, but more intricate than those for $p \in \mathbb{N}$.

2.2.1. $p \in \mathbb{N}$

We start by investigating the case $p \in \mathbb{N}$. We fix such a p throughout this section. For $n \ge (p+1)k$, let

- $\alpha^{i,j}_{n,k} := 0^{pj-i}1^{j-1}0^i1(0^p1)^{k-j}0^{n-(p+1)k}$ (resp. $\alpha_{n,k} = 1^k0^{n-k}$) for $i = 0, \ldots, p-1$ and $j = 1, \ldots, k$ if $p \ge 1$ (resp. if $p = 0$), and
- $\beta^i_{n,k} := 0^i1^k0^{n-i-k}$ for $i = pk+1, \ldots, n-k$.

Because it plays a special role, we will use a different notation for $\beta^{n-k}_{n,k}$, so we set

- $\gamma_{n,k} := \beta^{n-k}_{n,k} = 0^{n-k}1^k$.

Theorem 2.6. *Let $n \ge (p+1)k$, with $k \ge 1$.*

(1) For all $0 \le i \le p-1$ and $1 \le j \le k$, we have $\alpha^{i,j}_{n,k} \in \mathcal{C}_p(n,k)$ (resp. $\alpha_{n,k} \in \mathcal{C}_0(n,k)$), and $G(\alpha^{i,j}_{n,k}, C_p)$ (resp. $G(\alpha_{n,k}, C_0)$) is a suffix-partitioned list with $\gamma_{n,k}$ as last word.

(2) For all j, $pk+1 \le j \le n-k-1$, we have $\beta^j_{n,k} \in \mathcal{C}_p(n,k)$, and $G(\beta^j_{n,k}, C_p)$ is a suffix-partitioned list with $\gamma_{n,k}$ as last word.

(3) $\gamma_{n,k} \in \mathcal{C}_p(n,k)$, $G(\gamma_{n,k}, C_p)$ is a suffix-partitioned list and its last word is

(i) 0^p1 if $(n,k) = (p+1,1)$,

(ii) $\beta_{n,k}^{n-k-1} = 0^{n-k-1}1^k0$ *if* k *is odd and* $n \neq (p+1)k$*,*
(iii) $\alpha_{n,k}^{1,k-1} = 0^{(k-1)p-1}1^{k-2}010^p1$ *if* $k \geq 3$ *is odd and* $n = (p+1)k$ *(resp.* $\alpha_{n,k} = 1^k$ *if* $p = 0$*),*
(iv) $\alpha_{n,k}^{1,k} = 0^{kp-1}1^{k-1}010^{n-(p+1)k}$ *if* k *is even (resp.* $\alpha_{n,k} = 1^k0^{n-k}$ *if* $p = 0$*).*

Proof. We proceed by induction on n. It is easy to check for $n = 2$. Let $n > 2$ and assume that for all $m < n$ the theorem holds for each k such that $(p+1)k \leq m$. Let $k \geq 1$ with $(p+1)k \leq n$.

(1) Let $1 \leq j \leq k$ and $0 \leq i \leq p-1$. Suppose first that $n > (p+1)k$. Then $\alpha_{n,k}^{i,j} = \alpha_{n-1,k}^{i,j} \cdot 0$. By the homogeneous greedy algorithm, $G(\alpha_{n,k}^{i,j}, C_p)$ starts with $G(\alpha_{n-1,k}^{i,j}, C_p) \cdot 0$. By the induction hypothesis, the last word of $G(\alpha_{n-1,k}^{i,j}, C_p)$ is $\gamma_{n-1,k}$, thus $G(\alpha_{n,k}^{i,j}, C_p) = G(\alpha_{n-1,k}^{i,j}, C_p) \cdot 0, G(\beta_{n-1,k-1}^{n-k-1}, C_p) \cdot 1$. By the induction hypothesis, $G(\alpha_{n-1,k}^{i,j}, C_p)$ (resp. $G(\beta_{n-1,k-1}^{n-k-1}, C_p)$) contains every word of $C_p(n-1,k)$ (resp. $C_p(n-1,k-1)$) and is suffix-partitioned. Moreover, the last word of $G(\beta_{n-1,k-1}^{n-k-1}, C_p)$ is $\gamma_{n-1,k-1}$. Since $C_p(n,k) = C_p(n-1,k) \cdot 0 \sqcup C_p(n-1,k-1) \cdot 1$, $G(\alpha_{n,k}^{i,j}, C_p)$ contains every word in $C_p(n,k)$, it is suffix-partitioned with $\gamma_{n,k}$ as last word. Now suppose that $n = (p+1)k$. Then, using similar arguments we get that $G(\alpha_{n,k}^{i,j}, C_p) = G(\alpha_{n-2,k-1}^{i,j}, C_p) \cdot 01, G(\beta_{n-2,k-2}^{n-1-k}, C_p) \cdot 11$. Since $C_p((p+1)k,k) = C_p((p+1)k-2,k-1) \cdot 01 \sqcup C_p((p+1)k-2,k-2) \cdot 11$, we can conclude using the induction hypothesis.

(2) Let $pk+1 \leq j \leq n-k-1$. First, we suppose that $j \neq n-k-1$. Then we have $\beta_{n,k}^j = \beta_{n-1,k}^j \cdot 0$, and $G(\beta_{n,k}^j, C_p) = G(\beta_{n-1,k}^j, C_p) \cdot 0, G(\beta_{n-1,k-1}^{n-k-1}, C_p) \cdot 1$ and we can conclude by the induction hypothesis. Now suppose that $j = n-k-1$. Then $\beta_{n,k}^{n-k-1} = \gamma_{n-1,k} \cdot 0$. By the induction hypothesis, there are four possible cases.
   (i) If $(n-1,k) = (p+1,1)$, then $(n,k) = (p+2,1)$ and it is easy to check that the statement is true.
   (ii) If k is odd and $n-1 \neq (p+1)k$, then the last word of $G(\gamma_{n-1,k}, C_p)$ is $\beta_{n-1,k}^{n-k-2}$. Thus we have $G(\beta_{n,k}^{n-k-1}, C_p) = G(\gamma_{n-1,k}, C_p) \cdot 0, G(\beta_{n-1,k-1}^{n-k-2}, C_p) \cdot 1$ and we can conclude.
   (iii) If $k \geq 3$ is odd and $n = (p+1)k+1$, then the last word of $G(\gamma_{n-1,k}, C_p)$ is $\alpha_{n-1,k}^{1,k-1}$. Then $G(\gamma_{n-1,k}, C_p) \cdot 0, G(\alpha_{n-1,k-1}^{1,k-1}, C_p) \cdot 1$ and we can conclude too.
   (iv) If k is even, then the last word of $G(\gamma_{n-1,k}, C_p)$ is $\alpha_{n-1,k}^{1,k}$. In this case, $G(\beta_{n,k}^{n-k-1}, C_p) = G(\gamma_{n-1,k}, C_p) \cdot 0, G(\beta_{n-1,k-1}^{k-1}, C_p) \cdot 1$, and we can conclude similarly.

(3) Now let us consider $\gamma_{n,k}$. We have $\gamma_{n,k} = \gamma_{n-1,k-1} \cdot 1$. Again, we have four possible cases.
   (i) $(n-1,k-1) = (p+1,1)$ is impossible, because we would have $n = p+2 < (p+1)k$.

(ii) If $k-1$ is odd and $n-1 \neq (p+1)(k-1)$, then k is even. In this case, the last word of $G(\gamma_{n-1,k-1}, C_p)$ is $\beta^{n-k-1}_{n-1,k-1}$. Then $G(\gamma_{n,k}, C_p) = G(\gamma_{n-1,k-1}, C_p) \cdot 1, G(\gamma_{n-1,k}, C_p) \cdot 0$. Since k is even, the last word of $G(\gamma_{n-1,k}, C_p)$ is $\alpha^{1,k}_{n-1,k}$, so the last word of $G(\gamma_{n,k}, C_p)$ is $\alpha^{1,k}_{n,k}$.

(iii) If $k-1 \geq 3$ is odd and $n-1 = (p+1)(k-1)$, then $n = (p+1)(k-1)+1$, which is impossible because $n \geq (p+1)k$.

(iv) If $k-1$ is even, then k is odd. If $k-1 = 0$, then the result is easy to check. Thus we can assume $k-1 \geq 2$, so $k \geq 3$. The last word of $G(\gamma_{n-1,k-1}, C_p)$ is $\alpha^{1,k-1}_{n-1,k-1}$. Suppose that $n = (p+1)k$. Then $G(\gamma_{n,k}, C_p) = G(\gamma_{n-1,k-1}, C_p) \cdot 1$, and we have all the elements of $C_p((p+1)k, k)$ (they all end with 1), with $\alpha^{1,k-1}_{n,k}$ as last word. Now if $n \neq (p+1)k$, we have $G(\gamma_{n,k}, C_p) = G(\gamma_{n-1,k-1}, C_p) \cdot 1, G(\alpha^{1,k-1}_{n-1,k}, C_p) \cdot 0$. Thus by the induction hypothesis, $G(\gamma_{n,k}, C_p)$ contains every word of $C_p(n,k)$, with $\beta^{n-k-1}_{n,k}$ as last word, which concludes the proof.

□

Lemma 2.7. *Let $n \geq (p+1)k$. For all $\alpha \in C_p(n,k)$, with $\alpha \neq \gamma_{n,k}$, $G(\alpha, C_p)$ ends with $\gamma_{n,k}$.*

Proof. Even though the proof of [10, Lemma 1] can easily be generalized to each $p \geq 1$, we give an alternative argument. Let $\alpha \in C_p(n,k)$, with $\alpha \neq \gamma_{n,k}$. If $\gamma_{n,k}$ appears in $G(\alpha, C_p)$, then it has a predecessor. This predecessor has necessarily the form $0^{n-k-t}10^t1^{k-1}$ for some $t > 0$, and every word with suffix 01^{k-1} has appeared before in $G(\alpha, C_p)$. Since a successor of $\gamma_{n,k}$ would also have suffix 01^{k-1}, if $\gamma_{n,k}$ appears in $G(\alpha, C_p)$ then it is necessarily the last word. Now we prove that it does appear. Let j be the greatest integer such that there exists a word with suffix 01^j in $G(\alpha, C_p)$, and let β be the last word in $G(\alpha, C_p)$ with this suffix. Then we have either that β is the last word, or its successor has suffix 01^{j-1}. In both cases this means that a word with the suffix 01^{j+1} has appeared before. This is a contradiction to the definition of j, unless $j = k$. Thus $\gamma_{n,k}$ appears in $G(\alpha, C_p)$. □

Lemma 2.8. *Let $n \geq 1$. Then for all k with $n \geq (p+1)k$, and for each $\alpha \in C_p(n,k)$, $G(\alpha, C_p)$ is a suffix-partitioned list.*

Proof. We proceed by induction on n. This is trivial for $n = 1$. Let $n \geq 2$ and suppose that the statement is true for $n-1$. Let k be such that $n \geq (p+1)k$ and $\alpha \in C_p(n,k)$.

- If the last bit of α is 0, then $\alpha = \alpha' \cdot 0$ with $\alpha' \in C_p(n-1,k)$. Since the homogeneous greedy algorithm first changes the leftmost bits, $G(\alpha, C_p)$ starts with $G(\alpha', C_p) \cdot 0$. If $\alpha = \beta^{n-k-1}_{n,k}$, we can conclude by Theorem 2.6. Otherwise, we have $\alpha' \neq \gamma_{n-1,k}$ and by Lemma 2.7, the last word of $G(\alpha', C_p)$ is $\gamma_{n-1,k}$. Then by the homogeneous greedy algorithm, the next word in $G(\alpha, C_p)$ is $0^{n-k-1}1^{k-1}01$. Again by the homogeneous greedy algorithm, we have $G(\alpha, C_p) = G(\alpha', C_p) \cdot 0, G(\beta^{n-k-1}_{n-1,k-1}, C_p) \cdot 1, \mathcal{L}$ for some list $\mathcal{L}$. However, by Lemma 2.7 (or Theorem 2.6), the last word of $G(\beta^{n-k-1}_{n-1,k-1}, C_p)$ is $\gamma_{n-1,k-1}$. By applying Lemma 2.7 to α, we have in fact that $\mathcal{L}$ is empty. By the induction hypothesis (or Theorem 2.6 for $\beta^{n-k-1}_{n-1,k-1}$), $G(\alpha', C_p)$ and $G(\beta^{n-k-1}_{n-1,k-1}, C_p)$ are suffix-partitioned list, thus $G(\alpha, C_p)$ is too.

- If the last bit of α is 1, then $\alpha = \alpha' \cdot 1$ with $\alpha' \in C_p(n-1, k-1)$. So $G(\alpha, C_p)$ starts with $G(\alpha', C_p) \cdot 1$. If $\alpha = \gamma_{n,k}$, we can conclude by Theorem 2.6. Otherwise, $\alpha' \neq \gamma_{n-1,k-1}$ and by Lemma 2.7, the last word of $G(\alpha', C_p)$ is $\gamma_{n-1,k-1}$. Then again by Lemma 2.7 applied to α, we have $G(\alpha, C_p) = G(\alpha', C_p) \cdot 1$. By the induction hypothesis, we conclude that $G(\alpha, C_p)$ is suffix-partitioned.

□

Theorem 17 in [4] proves that if α has the form $0^i 1^k 0^{n-k-i}$, then it is a generator for $C_0(n,k)$; that is, $\alpha \in \mathcal{C}_0(n,k)$. In the next proposition, we show that these are all the generators of $C_0(n,k)$ and establish corresponding results for $p \geq 1$.

Proposition 2.9. *Let $n \geq (p+1)k$. If $p \geq 1$ then*

$$\mathcal{C}_p(n,k) = \bigcup_{j=1}^{k} \{0^{pj-i} 1^{j-1} 0^i 1 (0^p 1)^{k-j} 0^{n-(p+1)k} \mid 0 \leq i \leq p-1\}$$
$$\cup \{0^i 1^k 0^{n-i-k} \mid pk+1 \leq i \leq n-k\}.$$

If $p = 0$ then $\mathcal{C}_0(n,k) = \{0^i 1^k 0^{n-i-k} \mid 0 \leq i \leq n-k\}$. In particular,

$$|\mathcal{C}_p(n,k)| = n - k + 1 - p.$$

Proof. The inclusion from right to left has been established in Theorem 2.6. We prove the other inclusion when $p \geq 1$ (it is very similar and actually easier for $p = 0$). We start be investigating the elements of $\mathcal{C}_p((p+1)k, k)$. Let $\alpha \in \mathcal{C}_p((p+1)k, k)$. Each element of $C_p((p+1)k, k)$ has last bit 1, so $\alpha = \alpha' \cdot 1$, with $\alpha' \in \mathcal{C}_p((p+1)k-1, k-1)$. By Proposition 1.4 and Lemma 2.8, $G(\alpha, C_p)$ is r-t partitioned. Since it contains every word in $C_p((p+1)k-1, k-1)$, α' must have the longest or the shortest possible tail. The only word with the longest tail is $\gamma_{(p+1)k-1,k-1} = 0^{pk} 1^{k-1}$. Thus we have either $\alpha = 0^{pk} 1^k$ or $\alpha' = \alpha'' \cdot 0$ with $\alpha'' \in \mathcal{C}_p((p+1)k-2, k-1)$. By iterating this process, we have either $\alpha \in \{0^{pk-i} 1^{k-1} 0^i 1 \mid 0 \leq i \leq p-1\}$, or $\alpha = \beta \cdot 0^p 1$ with $\beta \in \mathcal{C}_p((p+1)(k-1), k-1)$. Again by iterating, we get that

$$\alpha \in \bigcup_{j=1}^{k} \{0^{pj-i} 1^{j-1} 0^i 1 (0^p 1)^{k-j} 0^{n-(p+1)k} \mid 0 \leq i \leq p-1\}.$$

Now let $\delta \in \mathcal{C}_p(n,k)$, with $\delta \notin \{0^i 1^k 0^{n-i-k} \mid pk+1 \leq i \leq n-k\}$. By the same argument as before, either $\delta = \delta' \cdot 0$ with $\delta' \in \mathcal{C}_p(n-1,k)$, or $\delta = \gamma_{n,k}$. By hypothesis on δ, the latter cannot happen. The same dichotomy applies to δ', but the hypothesis on δ implies $\delta' = \delta'' \cdot 0$ with $\delta'' \in \mathcal{C}_p(n-2,k)$. By iterating, we get that $\delta = \alpha \cdot 0^{n-(p+1)k}$ with $\alpha \in \mathcal{C}_p((p+1)k,k)$. We can conclude by the previous investigation. □

Remark 2.10. Propositions 2.5 and 2.9 highlight the fact that the choice of the first element for the homogeneous greedy algorithm is crucial. Indeed, only a few elements will produce a Gray code for $F_2(n,k)$ or $C_p(n,k)$ with this algorithm.

2.2.2. $p \in [1,+\infty)$

For $p \in [1,+\infty)$ and $i \in \mathbb{N}^\star$, we set $\nu_i(p) := \lceil ip \rceil - \lceil (i-1)p \rceil$. Then $\nu_i(p) \geq 1$ for all i. Note that $\nu_i(p) = p$ for all $i \in \mathbb{N}^\star$ if p is an integer, and the sequence $(\nu_i(p))_i$ is b-periodic if $p = a/b$ with a and b coprime. For $1 \leq j \leq k$ and $0 \leq i \leq \nu_j(p) - 1$, let $\alpha_{n,k}^{i,j} = 0^{\lceil jp \rceil - i} 1^{j-1} 0^i 1 (0^{\nu_{j+1}(p)} 1) \ldots (0^{\nu_k(p)} 1) 0^{n-k-\lceil kp \rceil}$. For $\lceil pk \rceil + 1 \leq i \leq n-k-1$ let $\beta_{n,k}^i = 0^i 1^k 0^{n-i-k}$, and let $\gamma_{n,k} = 0^{n-k} 1^k$. With these notations, the counterpart of Theorem 2.6 for $p \in [1,+\infty)$ is as follows.

Theorem 2.11. *Let $p \in [1,+\infty)$ and $n \geq (p+1)k$, with $k \geq 1$.*

(1) For all $1 \leq j \leq k$ and $0 \leq i \leq \nu_j(p) - 1$, we have $\alpha_{n,k}^{i,j} \in \mathcal{C}_p(n,k)$, and $G(\alpha_{n,k}^{i,j}, C_p)$ is a suffix-partitioned list with $\gamma_{n,k}$ as last word.

(2) For all $\lceil pk \rceil + 1 \leq j \leq n-k-1$, we have $\beta_{n,k}^j \in \mathcal{C}_p(n,k)$, and $G(\beta_{n,k}^j, C_p)$ is a suffix-partitioned list with $\gamma_{n,k}$ as last word.

(3) $\gamma_{n,k} \in \mathcal{C}_p(n,k)$, $G(\gamma_{n,k}, C_p)$ is a suffix-partitioned list and its last word is
  - *(i) $\beta_{n,k}^{n-k-1} = 0^{n-k-1} 1^k 0$ if k is odd and $n > k + \lceil pk \rceil$,*
  - *(ii) $\alpha_{n,k}^{1,k-1}$ if $k \geq 1$ is odd and $n = k + \lceil pk \rceil$,*
  - *(iii) $\alpha_{n,k}^{1,k}$ if k is even.*

Then Proposition 2.9 can be generalised as follows.

Proposition 2.12. *Let $p \in [1,+\infty)$ and $n \geq (p+1)k$. Then*

$$\mathcal{C}_p(n,k) = \bigcup_{j=1}^{k} \{0^{\lceil jp \rceil - i} 1^{j-1} 0^i 1 (0^{\nu_{j+1}(p)} 1) \ldots (0^{\nu_k(p)} 1) 0^{n-k-\lceil kp \rceil} \mid 0 \leq i \leq \nu_j(p) - 1\}$$
$$\cup \{0^i 1^k 0^{n-i-k} \mid \lceil pk \rceil + 1 \leq i \leq n-k\}.$$

In particular,

$$|\mathcal{C}_p(n,k)| = n - k + 1 - \lceil p \rceil.$$

2.2.3. $0 < p < 1$

Let $0 < p < 1$. For $i \in \mathbb{N}$, we set $u_i(p) := \lfloor i/p \rfloor - \lfloor (i-1)/p \rfloor$. Let $N = N_{p,k} = \max\{m \in \mathbb{N} \mid \lfloor m/p \rfloor \leq k\} = \lceil p(k+1) \rceil - 1$. For $1 \leq j \leq N$, let

$$\alpha_{n,k}^j = \begin{cases} 0^j 1^{\lfloor j/p \rfloor} (01^{u_{j+1}(p)}) \ldots (01^{u_N(p)}) (01^{k-\lfloor N/p \rfloor}) 0^{n-k-N-1} & \text{if } k > \lfloor N/p \rfloor \\ 0^j 1^{\lfloor j/p \rfloor} (01^{u_{j+1}(p)}) \ldots (01^{u_N(p)}) 0^{n-k-N} & \text{if } k = \lfloor N/p \rfloor. \end{cases}$$

For $N+1 \leq i \leq n-k$, let $\beta^i_{n,k} = 0^i 1^k 0^{n-i-k}$. Then

$$\mathcal{C}_p(n,k) = \{\alpha^j_{n,k} \mid 1 \leq j \leq N\} \cup \{\beta^i_{n,k} \mid N+1 \leq i \leq n-k\}.$$

In particular, $|\mathcal{C}_p(n,k)| = n-k = n-k+1-\lceil p \rceil$.

3. Generating algorithms

Recursive procedures FIB in Algorithm 1 and PREF in Algorithm 2 are specific implementations for F_2, and respectively, for C_p of the homogeneous greedy algorithm defined in Definition 2. In these algorithms, binary words are represented by the global array b. The global array s stores the positions of the 1s in b, with $s[i]$ giving the position of the ith 1 (reading b from left to right). In the case of the FIB procedure, we define $s[0] = -1$ for convenience.

In order to generate Fibonacci words (resp. prefixes of generalized Dyck words) of length n and weight k in homogeneous Gray code order, the word b is initialized with an element of $F_2(n,k)$ (resp. $C_p(n,k)$) and the array s is initialized accordingly. For convenience, in the Fibonacci case, we set $b[n+1] = 0$, and b is printed before the main call of FIB(n,k) (resp. PREF(n,k)). In particular, when the initial word b belongs to $\mathcal{F}(n,k)$ (resp. $\mathcal{C}_p(n,k)$) — that is, when b is a generator as defined in the previous section — the entire list for the set $F_2(n,k)$ (resp. $C_p(n,k)$) is produced.

The call of FIB(m,j) or PREF(m,j) produces at most two recursive calls and operates on the prefix of length m and weight j of the current word b.
In the case of FIB:

- When $j = 0$ or $m = 1$, or $m = 2j-1$, the length-m prefix of b is unique, and the call is terminal and a new word is generated;
- Otherwise, when $b[m+1] = 1$ (and therefore $b[m] = 0$) a single recursive call is made;
- In all other cases, two recursive calls are made.

In the case of PREF:

- When $j = 0$ or $j = m$, the length-m prefix of b is unique, and the call is terminal and a new word is generated;
- Otherwise, when $p \geq 1$ and $m = (p+1)j$ (and therefore $b[m] = 1$) a single recursive call is made;
- In all other cases, two recursive calls are made.

See Table 1 for two examples of generated lists, and Figures 1, 2, and 3 for the trees induced by the recursive calls. In these trees, nodes are labeled with the current value of the word b, they have at most two successors and words in sibling nodes at level i (the root having level 0) have different values at position $n-i+1$ (in bold), and share the same suffix of length $i-1$.

Since our generating algorithms are specific implementations of the homogeneous greedy algorithm defined in Definition 2, and by using Propositions 2.5 and 2.9, we obtain the following corollary.

Algorithm 1 Greedy Gray code algorithm for Fibonacci words of length n and weight k.

```
procedure Fib($m, j$: integer)
    global $b, s$: array
    if $j > 0$ and $m > 1$ and $m \neq 2j - 1$ then
        if $b[m + 1] = 1$ then
            Fib($m - 1, j$)
        else
            if $b[m] = 0$ then
                Fib($m - 1, j$)
                $(b[m],\, b[s[j]]) \leftarrow (1, 0)$  # transpose in the length-m prefix of b
                                                   the rightmost bit with the rightmost 1
                $s[j] \leftarrow m$  # update the position of the jth 1 in b
                print $b$
                Fib($m - 1, j - 1$)
            else  # $b[m] = 1$
                Fib($m - 1, j - 1$)
                $u \leftarrow s[j - 1] + 2$
                $(b[m],\, b[u]) \leftarrow (0, 1)$  # transpose homogeneously in the
                                              length-m prefix of b the rightmost
                                              bit with the leftmost possible 0
                $s[j] \leftarrow u$  # update the position of the jth 1 in b
                print $b$
                Fib($m - 1, j$)
            endif
        endif
    endif
end procedure
```

Corollary 3.1. *Algorithm 1, and respectively, Algorithm 2, generate a homogeneous Gray code for the set $F_2(n, k)$ and, respectively, for the set $C_p(n, k)$, provided that the initial words b belong to $\mathcal{F}(n, k)$ and, respectively, to $\mathcal{C}_p(n, k)$.*

It is easy to see that in the trees produced by the recursive calls of the generating procedures Fib and Pref, there are no two consecutive calls of degree one (where the degree of a call is defined as the number of recursive calls it produces); see again Figures 1, 2 and 3. With this remark, and by using standard techniques from [7] we have the following result.

Proposition 3.2. *Both Algorithms 1 and 2 run in constant amortized time.*

Notably, Algorithms 1 and 2 can be sped up by bypassing degree-one recursive calls. However, their definitions become more complex while preserving the same

(a)		(b)	
0101000	0100010	01010101	00101011
1001000	0001010	00110101	00110011
1010000	0001001	00101101	01010011
1000100	1000001	01001101	01000111
0100100	0100001	00011101	00100111
0010100	0010001	00011011	00010111
0010010	0000101	01001011	00001111
1000010			

TABLE 1. (a) The homogeneous greedy Gray code for $F_2(7,2)$ obtained by the call FIB$(7,2)$ when the initial word is $b = 0101000$; and (b) that is obtained for $C_1(8,4)$ by the call PREF$(8,4)$ when $p = 1$ and the initial word $b = 01010101$.

time complexity class. Both algorithms are implemented in C and appear on the second author web page `https://v.vincent.u-bourgogne.fr/0ABS/0ART/Progs.pdf`.

Acknowledgments: The authors would like to thank the anonymous referees for their comments, which have significantly improved the presentation of the paper, and Sergey Kirgizov for fruitful discussions during an early stage of the revision.

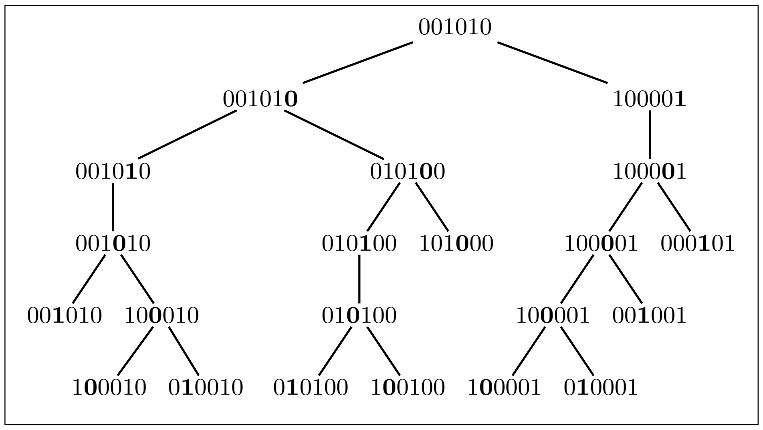

FIGURE 1. The tree of recursive calls produced by the call of FIB$(6,2)$ with initial word $b = 001010$. A homogeneous Gray code for $F_2(6,2)$ is obtained by collecting the words b at the leaves (terminal calls).

Algorithm 2 Greedy Gray code algorithm for prefixes of generalized Dyck words of length m and weight k.

```
procedure PREF($m, j$: integer)
    global $p$: integer; $b, s$: array
    if $j > 0$ and $j \neq m$ then
        if $b[m] = 0$ then
            PREF($m-1, j$)
            ($b[m]$, $b[s[j]]$) ← $(1, 0)$  # transpose in the length-m prefix of b
                                                the rightmost bit with the rightmost 1
            $s[j] \leftarrow m$  # update the position of the jth 1 in b
            print $b$
            PREF($m-1, j-1$)
        else  # $b[m] = 1$
            PREF($m-1, j-1$)
            $u \leftarrow \max(s[j-1]+1, (p+1)j)$
            if $u \neq m$
                ($b[m]$, $b[u]$) ← $(0, 1)$  # transpose homogeneously in the length-m prefix of b
                                                the rightmost bit with the leftmost possible 0
                $s[j] \leftarrow u$  # update the position of the jth 1 in b
                print $b$
                PREF($m-1, j$)
            endif
        endif
    endif
end procedure
```

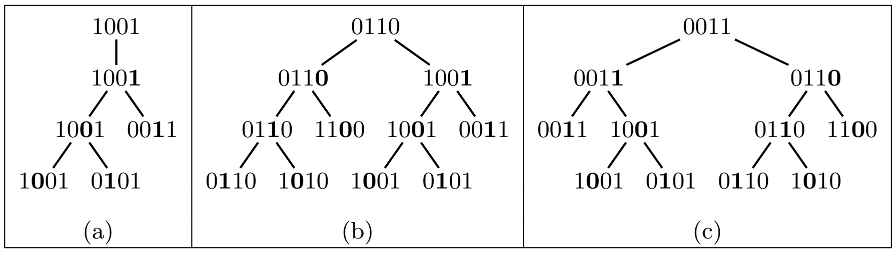

FIGURE 2. The tree of recursive calls produced by the call of PREF(4, 2) with $p = 0$, when the initial word is: (a) $b = 1001$; (b) $b = 0110$; and (c) $b = 0011$. For the last two cases, homogeneous Gray codes for $C_0(4, 2)$ are obtained by collecting the words b at the leaves. See Example 1.2.

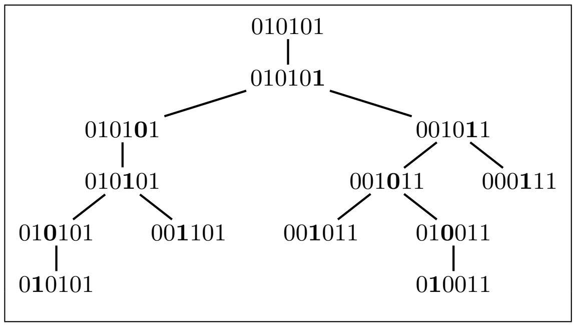

FIGURE 3. The tree of recursive calls produced by the call of PREF(6, 3) with $p = 1$, when the initial word is $b = 010101$. A homogeneous Gray code for $C_1(6, 3)$ is obtained by collecting the words b at the leaves.

References

[1] A. Bultena and F. Ruskey, An Eades-McKay algorithm for well-formed parentheses strings, *Information Processing Letters*, 68 (1998), 255-259.

[2] P. Eades and B. McKay, An algorithm for generating subsets of fixed size with a strong minimal change property, *Information Processing Letters*, 19 (1984), 131-133.

[3] N. Hassler, V. Vajnovszki, D. Wong, Greedy Gray codes for some restricted classes of binary words, *GASCom 2024*, June 24–28, Bordeaux, France.

[4] A. Merino, T. Mütze, A. Williams, All your bases are belong to us: listing all bases of a matroid by greedy exchanges, *FUN 2022*, May 30–June 3, 2022, Island of Favignana, Sicily, Italy

[5] A. Merino and T. Mütze, Traversing Combinatorial 0/1-Polytopes via Optimization, *SIAM Journal on Computing*, 53(5), 2024.

[6] T. Mütze, Combinatorial Gray codes – an updated survey, *Electronic Journal of Combinatorics*, Dynamic Survey DS26, 93 pp., 2023.

[7] F. Ruskey. *Combinatorial Generation.* 2003. Working version.

[8] V. Vajnovszki and T. Walsh, A loop-free two-close Gray-code algorithm for listing k-ary Dyck words, *Journal of Discrete Algorithms*, 4 (2006).

[9] A. Williams, The greedy Gray code algorithm, in F. Dehne, R. Solis-Oba, and J.-R. Sack, editors, *Algorithms and Data Structures*, pp. 525–536, Berlin, Heidelberg, 2013.

[10] D. Wong and V. Vajnovszki, Greedy Gray codes for Dyck words and ballot sequences, *COCOON 2023*, 15–17 December 2023, Hawaii, USA.

Communicated by (The editor will be set by the publisher).
(The dates will be set by the publisher).