Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 17–51

GRAY CODES FOR SIGNED INVOLUTIONS

GONÇALO GUTIERRES, RICARDO MAMEDE AND JOSÉ LUIS SANTOS

Abstract: In this paper we present two cyclic Gray codes for signed involutions. The first one has a natural construction, implemented by a CAT algorithm, based in the recursive formula for the number of sign involutions. The second code, although with a higher computational cost, has the smallest possible Hamming distance for this family of objects.

1. Introduction

The exhaustive generation and listing of a combinatorial class of objects according to a fixed parameter is one of the most important aims in combinatorics, with applications in a vast range of areas ranging from computer science and hardware or software testing, thermodynamic, biology and biochemistry [8]. A common approach is to list the objects such that successive objects differ by a well-defined closeness condition. These lists are usually called Gray codes, a term taken from Frank Gray [5], who patented the Binary Reflected Gray Code, $BRGC_n$, a list of all 2^n binary strings of length n in which successive strings differ by a single bit. We adopt the point of view of Walsh [12], defining a Gray code for a class of objects as an infinite set of word-lists with unbounded word-length such that the Hamming distance between any two adjacent words in any list, *i.e.* the number of positions in which these two words differ, is bounded independently of the word-length. When this distance is preserved between the last and the first words, the code is said to be cyclic.

A great interest was been shown in the generation of Gray codes for permutations [3, 9] and their restrictions or generalizations, such as permutations with a fixed number of cycles [1], derangement [2], involutions and fixed-point free involutions [11], multiset permutations [13], and sign permutations [6]. We present here two cyclic Gray codes for the set of n-length signed involutions.

The first code is constructed inductively using a natural construction derived from a formula for the number of signed involutions. In particular,

Received November 28, 2017.
This work was partially supported by the Centre for Mathematics of the University of Coimbra – UID/MAT/00324/2013, funded by the Portuguese Government through FCT/MCTES and co-funded by the European Regional Development Fund through the Partnership Agreement PT2020.

our implementation of this code is able to generate all Gray codes for k-length signed involutions, for $k \leq n$. The second code is constructed by levels, each corresponding to involutions having a fixed number of disjoint transpositions, applying a variation of the Binary Reflected Gray Code to generate the elements of each level.

In the first code, each involution is transformed into its successor by a transposition, a rotation of three elements and/or one sign change, while in the second one this difference is reduced to a transposition or at most a pair of sign changes. This implies that the Hamming distance for this second code is 2, which is proven to be minimal (see Corollary 4.9). Although optimal in terms of distance between consecutive words, this second code has the disadvantage of being more elaborated and with a higher computational cost in comparison with the previous one, which has a more natural construction and is generated by a constant amortized time (CAT) algorithm. This computational comparison was numerically confirmed for n up to 15.

2. Notation and definitions

A *signed permutation* σ of length n is a permutation on the set $[\pm n] = \{\pm 1, \ldots, \pm n\}$ satisfying

$$\sigma(-i) = -\sigma(i). \tag{2.1}$$

Under the ordinary composition of mappings, the signed permutations of $[\pm n]$ form a group, called the hyperoctahedral group of rank n, denoted by $\mathfrak{S}_n^B$. The group of ordinary permutations is denoted by $\mathfrak{S}_n$. Equation (2.1) indicates that the signed permutation σ is entirely defined by its values on $[n] = \{1, \ldots, n\}$. Therefore, we shall represent the elements $\sigma \in \mathfrak{S}_n^B$ is one line notation $\sigma = \sigma_1 \cdots \sigma_n$, where $\sigma_i = \sigma(i)$ for $i \in [n]$. It also follows that $\mathfrak{S}_n^B$ has order $2^n n!$.

Given $i \in [\pm n]$, we define the signal function $sgn(i) = 0$ if $i > 0$, and $sgn(i) = 1$ otherwise. A signed permutation σ may also be represented by a pair (p, g), where $p \in \mathfrak{S}_n$ is the list of the numbers in σ without signs, and $g \in B_n$, the set of binary words of length n, is the list of the signs in σ, that is $p_i = |\sigma_i|$ and $g_i = sgn(\sigma_i)$ for every $i \in [n]$. For example, $\sigma = \overline{4}\,3\,2\,\overline{1}\,\overline{5} \in \mathfrak{S}_5^B$ corresponds to the pair $(43215, 10011) \in \mathfrak{S}_5 \times B_5$, where for ease of notation, we denote by $\bar{i}$ the negative element $-i$. Depending on the context, it will be clear which representation of signed permutations we are using.

A sign permutation $\sigma \in \mathfrak{S}_n^B$ is an *involution* if $\sigma^2 = id$, the identity element of $\mathfrak{S}_n^B$, and we denote by $\mathfrak{I}_n$ the set of all involutions in $\mathfrak{S}_n^B$. Note that if $\sigma \equiv (p, g)$ is an involution in $\mathfrak{S}_n^B$, then p is also an involution in

References

[1] J.-L. Baril, Gray code for permutations with a fixed number of cycles, Disc. Math. 307:13 (2007) 1559–1571.

[2] J.-L. Baril and V. Vajnovszki, Gray code for derangements, Disc. App. Math. 140 (2004) 207–221.

[3] S.M. Johson, Generating of permutations by adjacent transposition, Math. Comput. 17 (1963) 282–285.

[4] Chow, Chak-On. Counting involutory, unimodal, and alternating signed permutations. Discrete Math. 306 (2006), no. 18, 2222–2228.

[5] F. Gray. Pulse code communication, March 17, 1953 (filed Nov. 1947). U.S. Patent 2,632,058.

[6] J. Korsh, P. LaFollette and S. Lipschutz. A loopless implementation of a gray code for signed permutations, Publications de l'Institut Mathématique, volume 89(103), Issue 109 (2011), 37–47.

[7] F. Ruskey. Combinatorial generation, Book in preparation.

[8] C. Savage. A survey of combinatorial Gray codes, SIAM Review 39:4 (1997), 605–629.

[9] H.F. Trotter, Algorithm 115, permutations, Comm. ACM 5 (1962), 434–435.

[10] T. Mansour. Combinatorics of Set Partitions. Discrete Mathematics and Its Applications, Taylor & Francis, 2012.

[11] T. Walsh. Gray codes for involutions, J. Combin. Math. Combin. Comput., 36 (2001), 95–118.

[12] T. Walsh. Generating Gray Codes in $O(1)$ Worst-Case Time per Word. In *Discrete mathematics and theoretical computer science*, volume 2731 of *Lecture Notes in Comput. Sci.*, pp. 77–88. Springer, Berlin, 2003.

[13] V. Vajnovszki, A loopless algorithm for generating the permutations of a multiset, Theoretical Computer Science Volume 307, Issue 2, 7 (2003), 415–431.

Gonçalo Gutierres
CMUC, Department of Mathematics, University of Coimbra, Apartado 3008, 3001–454 Coimbra, Portugal
E-mail address: ggutc@mat.uc.pt

Ricardo Mamede
CMUC, Department of Mathematics, University of Coimbra, Apartado 3008, 3001–454 Coimbra, Portugal
E-mail address: mamede@mat.uc.pt

José Luis Santos
CMUC, Department of Mathematics, University of Coimbra, Apartado 3008, 3001–454 Coimbra, Portugal
E-mail address: zeluis@mat.uc.pt

n	$\#\mathfrak{I}_n$	Alg.2	Alg.3
3	20	0.001563	0.004687
4	76	0.003125	0.013021
5	312	0.007813	0.022396
6	1384	0.022396	0.075521
7	6512	0.084375	0.351042
8	32400	0.376563	1.058854
9	168992	1.853646	4.542708
10	921184	9.975000	21.527083
11	5222208	58.415625	109.463021
12	30710464	329.468229	597.846354
13	186753920	2015.429167	3524.954688
14	1171979904	12760.761458	21857.397569
15	7573069568	81972.468750	135773.593800

TABLE 5.1. CPU time (in seconds) for Algorithms 2 and 3.

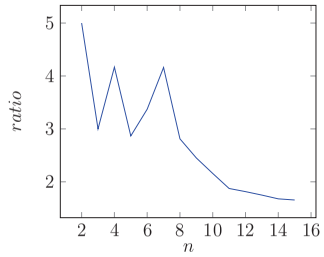

FIGURE 5.2. Ration between CPU times for Algorithms 3 and 2.

a CAT algorithm, Figure 5.1 suggest that a CAT algorithm implementation is possible for Algorithm 3.

$\mathfrak{S}_n$. The cycle decomposition of $\sigma \equiv (p, g) \in \mathfrak{I}_n$ is obtained by writing p as the disjoint union of transpositions and position fixed points, *i.e.* integers i for which $p_i = i$, and then associating the respective signs. For instance, for $\sigma = \overline{4}\,3\,2\,\overline{1}\,\overline{5} \equiv (43215, 10011)$, we have $43215 = (14)(23)(5) \in \mathfrak{S}_5$, and thus $\sigma = (\overline{1}\,\overline{4})(23)(\overline{5}) \in \mathfrak{S}_5^B$. When writing a transposition $(a\,b)$ we adopt the convention that $a < b$. The integers a and b are called, respectively, the *opener* and *closure* of $(a\,b)$ and, as part of an involution, must have the same sign. The common sign of both elements in a transposition is called a *paired sign*. Define the map

$$m : \mathfrak{I}_n \to [n] \cup \{0\},$$

where $m(\sigma)$ is the largest opener amongst all transpositions in the cycle decomposition of σ. When $\sigma = id$ we assume $m(id) = 0$. Using our running example, we find that $m(\overline{4}\,3\,2\,\overline{1}\,\overline{5}) = 2$.

3. A CAT Gray code

We start by showing how the set $\mathfrak{I}_n$ can be recursively constructed from the sets $\mathfrak{I}_{n-1}$ and $\mathfrak{I}_{n-2}$, and then use this construction to derive a Gray code for the involutions in $\mathfrak{I}_n$.

Definition 3.1. An involution $\sigma \in \mathfrak{I}_n$ is said to be position fixed on the letter $j \in [n]$ if $|\sigma_j| = j$. Let F_n be the set of all involutions having at least one position fixed letter, and let F_n^i be the set of all position fixed involutions on the letter i, that is,

$$F_n^i = \{\sigma \in \mathfrak{I}_n : |\sigma_i| = i\} \text{ and } F_n = \bigcup_{i=1}^{n} F_n^i.$$

The functions ϕ_n and ψ_n, defined below, establish bijections between $\{\pm n\} \times \mathfrak{I}_{n-1}$ and F_n^n on one hand, and between $[\pm(n-1)] \times \mathfrak{I}_{n-2}$ and $\mathfrak{I}'_n := \mathfrak{I}_n \setminus F_n^n$, the set of the involutions which are not position fixed on the letter n, on the other.

Definition 3.2. (*a*) For $n \geq 2$, an integer $s \in \{\pm n\}$, and an involution $\sigma \in \mathfrak{I}_{n-1}$, define the involution

$$\phi_n(s, \sigma) = \pi,$$

where $\pi_k = \sigma_k$ for $k \in [n-1]$ and $\pi_n = s$.

(*b*) For $n \geq 3$, an integer $j \in [\pm(n-1)]$, and an involution $\sigma \equiv (p, g) \in \mathfrak{I}_{n-2}$, define the involution

$$\psi_n(j, \sigma) = \pi,$$

where $\pi \equiv (p', g')$ is given by $p'_{|j|} = n$, $p'_n = |j|$, and

$$p'_k = \begin{cases} p_k + \mathbb{1}_A(p_k), & \text{if } 1 \leq k < |j| \\ p_{k-1} + \mathbb{1}_A(p_k), & \text{if } |j| < k \leq n-1 \end{cases},$$

with $A = \{i \in [n] : i \geq |j|\}$ and $\mathbb{1}_A$ the characteristic function of A; and the binary word $g' = g'_1 \cdots g'_n$ is given by $g'_{|j|} = g'_n = sgn(j)$, and $g'_k = g_{k-\mathbb{1}_A(k)}$ for $k \in [n-1] \setminus \{|j|\}$. Informally, the letters of $\psi_n(j, \sigma)$ are obtained from σ inserting $(-1)^{sgn(j)} \times n$ between the letters $\sigma_{|j|-1}$ and $\sigma_{|j|}$, and j at the rightmost position, and then incrementing (resp. decreasing) all letters σ_k for which $|\sigma_k| \geq |j|$ by one unit if $\sigma_k > 0$ (resp $\sigma_k < 0$).

For instance, $\phi_5(\bar{5}, \bar{4}32\bar{1}) = \bar{4}32\bar{1}\bar{5}$ and $\psi_5(2, \bar{3}2\bar{1}) = \bar{4}5\bar{3}\bar{1}2$.

Lemma 3.3. *For any integer $n \geq 2$, the maps ϕ_n and ψ_n are bijections.*

Proof: It is easy to see that the map $\phi_n^{-1} : F_n^n \to \{\pm n\} \times \mathfrak{I}_{n-1}$ given by $\phi_n^{-1}(\pi) = (\pi_n, \sigma)$, with $\sigma_i = \pi_i$ for $i \in [n-1]$, is the inverse function of ϕ_n.

To prove the bijectivity of ψ_n, we also construct its inverse function. Let

$$\psi_n^{-1} : \mathfrak{I}'_n \to [\pm(n-1)] \times \mathfrak{I}_{n-2}$$

such that if $\pi \equiv (p, g) \in \mathfrak{I}'_n$ then $\psi_n^{-1}(\pi) = (\pi_n, \sigma)$, with $\sigma \equiv (p', g')$, where g' is obtained from g by removing the letters g_{p_n} and g_n, and

$$p'_k = \begin{cases} p_k - \mathbb{1}_A(p_k), & \text{if } 1 \leq k < p_n \\ p_{k+1} - \mathbb{1}_A(p_k), & \text{if } p_n \leq k \leq n-2 \end{cases},$$

with $A = \{i \in [n] : i \geq p_n\}$. Now, it is not difficult to see that the function we just construct is the inverse function of ψ_n. ∎

Since $\mathfrak{I}_n$ is the disjoint union of F_n^n and $\mathfrak{I}'_n$, we may use bijections ϕ_n and ψ_n to derive the following recursive formula for the cardinality of $\mathfrak{I}_n$ (see [4]).

Corollary 3.4. *Let t_n be the cardinal of $\mathfrak{I}_n$. Then $t_0 = 1$, $t_1 = 2$ and for $n \geq 2$,*

$$t_n = 2t_{n-1} + 2(n-1)t_{n-2}.$$

Next result shows that the cardinal of $\mathfrak{I}'_n$ can also be obtained using only the involutions in $\mathfrak{I}_{n-1}$.

Lemma 3.5. *The cardinal of $\mathfrak{I}'_n$, $n \geq 2$, is equal to the number of position fixed letters counted over all involutions of $\mathfrak{I}_{n-1}$, i.e $|\mathfrak{I}'_n| = \Sigma_{i=1}^{n-1} |F_{n-1}^i|$.*


```
1  function BRGC(p, g, δ, π)
2  begin
3      if δ ≤ length(π) then
4          BRGC(p, g, δ + 1, π);
5          if cont == min{λ(ij) : (ij) ∈ T} then
6              (i*j*) ← argmin{λ(ij) : (ij) ∈ T};
7              T ← T\{(i*j*)};
               // Performs Ω_{p,δ}
8              π'_1 ← π_δ;   x ← 1;
9              for each y ∈ {1, ..., length(π)}\{δ, i*, j*} do
10                 x ← x + 1; π'_x ← π_y ;
11             end
12             x ← x + 1; π'_x ← π_{i*};
13             σ' ← (p · (i*j*), g);
14             SIGC(σ', π');
15         else
16             g_{π_δ} ← g_{π_δ} ⊕ 1;
17             Write(p, g);
18         end
19         BRGC(p, g, δ + 1, π);
20     end
21 end
```

Algorithm 5: Auxiliar function $BRGC$.

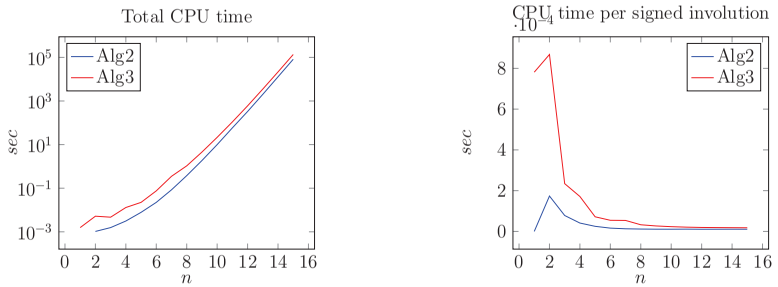

FIGURE 5.1. Comparison between Algorithm 2 and 3.

They also support the claim that Algorithm 2 is a CAT algorithm, since the average CPU time per signed involution is almost constant. Although our implementation of Algorithm 3 does not satisfy the stated conditions to be

```
1  function SIGC(σ, π)
2  begin
3      (p, g) ← σ;
4      T ← {(i, j) : p_i = i ∧ p_j = j ∧ m(p) < i < j};
5      Write(p, g);
       // Build L_k^B(p), where k = #openers
6      δ ← 1;   cont ← 1;
7      BRGC(p, g, δ, π);
       // Insert transposition (n − 1  n) in L_0
8      if cont == min{λ(ij) : (ij) ∈ T} then
9          (i*j*) ← argmin{λ(ij) : (ij) ∈ T};
10         T ← T\{(i*j*)};
           // Performs Ω_{p,δ}
11         π'_1 ← π_δ;   x ← 1;
12         for each y ∈ {1, ..., length(π)}\{δ, i*, j*} do
13             x ← x + 1; π'_x ← π_y ;
14         end
15         x ← x + 1; π'_x ← π_{i*};
16         σ' ← (p · (i*j*), g);
17         SIGC(σ', π');
18     end
19 end
```

Algorithm 4: Recursive function *SIGC* for signed involutions.

5. Numerical Experiments

The algorithms where written in MatLab on a PC with Intel(R) Core(M) i7-3770 CPU 3.4Ghz, RAM 16GB 64-bit Operating System, under Windows 8.1 Pro.

The running CPU time of algorithms 2 and 3 where compared for values of n up to 15. In order to obtain more accurate results, the CPU times reported in Table 5.1 correspond to the average of 30 executions of the algorithms. To better compare the running CPU time of both algorithms, Figure 5.1 shows the total CPU time and the CPU time per signed involution for each algorithm, and Figure 5.2 displays how many times Algorithm 3 is slower than Algorithm 2.

These graphics show that the CPU time of both algorithms grows at least exponentially with n, with Algorithm 3 beeing the slower one, with a ratio, relative to Algorithm 2, that seams to stabilize around 1.5 folds.

Proof: Let $F = \bigcup_{i=1}^{n-1} F_{n-1}^i \times \{i\}$ be the disjoint union of all the sets F_{n-1}^i, so that $|F|$ counts the number of fixed letters in all involutions of F_{n-1}.

Consider the map $\theta_n : \mathfrak{I}'_n \to F$ defined by

$$\theta_n(\pi) = (\sigma, j),$$

where $\sigma_k = \pi_k$, for $k \in [n-1] \setminus \{j\}$, $j = |\pi_n|$ and $\sigma_j = \pi_n$.

The function θ_n is a bijection and its inverse function is $\theta_n^{-1} : F \to \mathfrak{I}'_n$ defined by

$$\theta_n^{-1}(\sigma, j) = \pi,$$

where $\pi_k = \sigma_k$, for $k \in [n-1] \setminus \{j\}$, $\sigma_j = (-1)^{sgn(\pi_j)} \times n$ and $\sigma_n = (-1)^{sgn(\pi_j)} \times j$.

■

To construct our first Gray code for $\mathfrak{I}_n$, we compose the function θ_n, defined in the lemma above, with the projection over the first component of F, $p : F \to \mathfrak{I}_{n-1}$, with $p(\sigma, j) = \sigma$, to obtain

$$\varphi_n = p \circ \theta_n : \mathfrak{I}'_n \to \mathfrak{I}_{n-1}.$$

Notice that the image of φ_n is F_{n-1}, the set of all involutions in $\mathfrak{I}_{n-1}$ having at least one position fixed letter. Since φ_n is not injective, as usual, $\varphi_n^{-1}(\sigma)$ is the inverse image set. Whenever $\sigma \notin F_{n-1}$, $\varphi_n^{-1}(\sigma)$ is the empty set.

We can now construct a Gray code for the involutions in $\mathfrak{I}_n$, where the difference between any two consecutive involutions is a transposition, a rotation of 3 elements and/or a sign. For $n = 1$ the list $(1, \overline{1})$ is such a code. Consider now $n \geq 2$ and let $w^1, \ldots, w^k$ be a Gray code for $\mathfrak{I}_{n-1}$ satisfying the above conditions. The following algorithm constructs a Gray code for $\mathfrak{I}_n$.

```
1  for i ← 1 to k do
       // Construct the string B_i as follows:
2      Write φ_n((−1)^{i−1} n, w^i);
3      Write all elements in φ_n^{−1}(w^i);
4      Write φ_n((−1)^i n, w^i);
5  end
```

Algorithm 1: Algorithm for the first Gray code.

Figure 3.1 and Table 3.1 show the Gray codes for $\mathfrak{I}_n$, $n \leq 4$, obtained by this algorithm. We shall see that the ordered list $B_1 B_2 \cdots B_k$ obtained from Algorithm 1 is a Gray code for the involutions in $\mathfrak{I}_n$ in the stated

$1\ 2\ 3\ 4$	$4\ 2\ 3\ 1$	$1\ 4\ 3\ 2$	$1\ 2\ 4\ 3$	$1\ 2\ 3\ \bar{4}$	$3\ 2\ 1\ \bar{4}$	$3\ 4\ 1\ 2$
$3\ 2\ 1\ 4$	$1\ 3\ 2\ 4$	$4\ 3\ 2\ 1$	$1\ 3\ 2\ \bar{4}$	$1\ 2\ \bar{3}\ \bar{4}$	$4\ 2\ \bar{3}\ 1$	$1\ 4\ \bar{3}\ 2$
$1\ 2\ \bar{4}\ \bar{3}$	$1\ 2\ \bar{3}\ 4$	$2\ 1\ \bar{3}\ 4$	$2\ 1\ \bar{4}\ \bar{3}$	$2\ 1\ \bar{3}\ \bar{4}$	$2\ 1\ 3\ \bar{4}$	$2\ 1\ 4\ 3$
$2\ 1\ 3\ 4$	$1\ \bar{2}\ 3\ 4$	$4\ \bar{2}\ 3\ 1$	$1\ \bar{4}\ 3\ \bar{2}$	$1\ \bar{2}\ 4\ 3$	$1\ \bar{2}\ 3\ \bar{4}$	$3\ \bar{2}\ 1\ \bar{4}$
$3\ \bar{4}\ 1\ \bar{2}$	$3\ \bar{2}\ 1\ 4$	$1\ \bar{3}\ \bar{2}\ 4$	$4\ \bar{3}\ \bar{2}\ 1$	$1\ \bar{3}\ \bar{2}\ \bar{4}$	$1\ \bar{2}\ \bar{3}\ \bar{4}$	$4\ \bar{2}\ \bar{3}\ 1$
$1\ \bar{4}\ \bar{3}\ \bar{2}$	$1\ \bar{2}\ \bar{4}\ \bar{3}$	$1\ \bar{2}\ \bar{3}\ 4$	$\bar{1}\ \bar{2}\ \bar{3}\ 4$	$\bar{4}\ \bar{2}\ \bar{3}\ \bar{1}$	$\bar{1}\ \bar{4}\ \bar{3}\ \bar{2}$	$\bar{1}\ \bar{2}\ \bar{4}\ \bar{3}$
$\bar{1}\ \bar{2}\ \bar{3}\ \bar{4}$	$\bar{3}\ \bar{2}\ \bar{1}\ \bar{4}$	$\bar{3}\ \bar{4}\ \bar{1}\ \bar{2}$	$\bar{3}\ \bar{2}\ \bar{1}\ 4$	$\bar{1}\ \bar{3}\ \bar{2}\ 4$	$\bar{4}\ \bar{3}\ \bar{2}\ \bar{1}$	$\bar{1}\ \bar{3}\ \bar{2}\ \bar{4}$
$\bar{1}\ \bar{2}\ 3\ \bar{4}$	$\bar{4}\ \bar{2}\ 3\ \bar{1}$	$\bar{1}\ \bar{4}\ 3\ \bar{2}$	$\bar{1}\ \bar{2}\ 4\ 3$	$\bar{1}\ \bar{2}\ 3\ 4$	$\bar{2}\ \bar{1}\ 3\ 4$	$\bar{2}\ \bar{1}\ 4\ 3$
$\bar{2}\ \bar{1}\ 3\ \bar{4}$	$\bar{2}\ \bar{1}\ \bar{3}\ \bar{4}$	$\bar{2}\ \bar{1}\ \bar{4}\ \bar{3}$	$\bar{2}\ \bar{1}\ \bar{3}\ 4$	$\bar{1}\ 2\ \bar{3}\ 4$	$\bar{4}\ 2\ \bar{3}\ \bar{1}$	$\bar{1}\ 4\ \bar{3}\ 2$
$\bar{1}\ 2\ \bar{4}\ \bar{3}$	$\bar{1}\ 2\ \bar{3}\ \bar{4}$	$\bar{3}\ 2\ \bar{1}\ \bar{4}$	$\bar{3}\ 4\ \bar{1}\ 2$	$\bar{3}\ 2\ \bar{1}\ 4$	$\bar{1}\ 3\ 2\ 4$	$\bar{4}\ 3\ 2\ \bar{1}$
$\bar{1}\ 3\ 2\ \bar{4}$	$\bar{1}\ 2\ 3\ \bar{4}$	$\bar{4}\ 2\ 3\ \bar{1}$	$\bar{1}\ 4\ 3\ 2$	$\bar{1}\ 2\ 4\ 3$	$\bar{1}\ 2\ 3\ 4$	

TABLE 3.1. Gray code for $\mathfrak{I}_4$

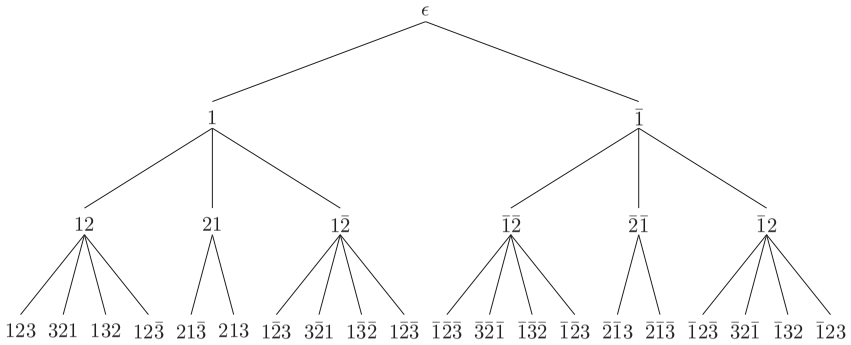

FIGURE 3.1. Gray code for $\mathfrak{I}_1$, $\mathfrak{I}_2$ and $\mathfrak{I}_3$ generated by Algorithm 2.

conditions. In the next result we analyze the difference between any two consecutive elements of the list $B_1 \cdots B_k$.

Lemma 3.6. *Let $w^1, \ldots, w^k$ be a Gray code for $\mathfrak{I}_{n-1}$ as above. Then,*

1. *The difference between $\phi_n((-1)^i n, w^i)$ and $\phi_n((-1)^i n, w^{i+1})$ is equal to the difference between w^i and w^{i+1}.*
2. *The involutions $\phi_n((-1)^{i-1} n, w^i)$ and $\phi_n((-1)^i, w^i)$ differ only by a sign.*
3. *If $\varphi^{-1}(w^i) \neq \emptyset$, then $\phi_n((-1)^i n, w^i)$ differ from $\pi \in \varphi^{-1}(w^i)$ by one transposition, or one transposition and one sign.*
4. *If $\varphi^{-1}(w^i) \neq \emptyset$, then any two involutions $\pi, \rho \in \varphi^{-1}(w^i)$ differ by a rotation of three elements, or a rotation of three elements and one sign.*

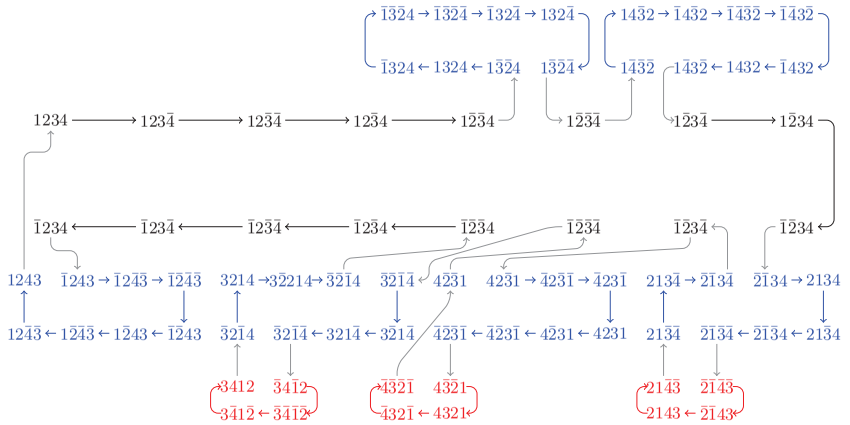

FIGURE 4.2. Gray code scheme for $\mathfrak{I}_4$ generated by algorithm 3.

A recursive implementation of Algorithm 3 is schematized below. It initially sets $\sigma \equiv (id, 0^n)$ and $\pi = id$, and the recursive calls are triggered by the initial call $SIGC((id, 0^n), id)$. For the current value of $\sigma \equiv (p, g)$ and π, in step 4 all available transpositions in the present call are generated. After the involution σ is written, the function $BRGC(p, g, \delta, \pi)$ is called in order to recursively generate all involutions in L^B_{n-k}, where k is the number of closers in p. In this function, the integer δ in $BRGC(p, g, \delta, \pi)$ represents the index of a letter in g whose sign changes in this call. This algorithm gives the elements of the $BRGC_n$ [10] while the condition in step 5 is not satisfied. Otherwise, the next transposition $(i^* j^*)$ to be inserted is identified, the set of transpositions T is updated, and new σ' and π' are obtained from σ and π, and $SIGC(\sigma', \pi')$ is called to generate $L^B_{n-k-1}(p \cdot (i^* j^*))$.

construct a Gray code for $\mathfrak{I}_3$. It is an open question whether this result is valid for $n \geq 4$.

Corollary 4.9. *The Hamming distance of the Gray code for signed involutions in $\mathfrak{I}_n$, $n \geq 3$, given by Algorithm 3, is two. This is the minimal Hamming distance of any Gray code for $\mathfrak{I}_n$.*

Proof: The Hamming distance for the code given by Algorithm 3 follows from Theorem 4.8. Note also that in a code with Hamming distance one, only one sign can be changed between two consecutive words, making it impossible to run over all $\mathfrak{I}_n$. ∎

Example 4.2. We give below a brief description of the execution of Algorithm 3 when $n = 4$. The algorithm starts with the construction of $L_0^B = ((id, 0000), (id, 0001), \ldots, (id, 1000))$. In the next step, subsequences of L_1^B are introduced to obtain $L_{\leq 1}^B$, using the transpositions of $L_1(id) = \{(12), (13), (14), (23), (24), (34)\}$. Since $\lambda_0^4(12) = 9$, then $\delta = 4$ is the only position that changes between the 9-th and 10-th elements of the $BRGC_4$, that is, between 1100 and 1101, respectively. Thus, the sequence (see Example 4.1)

$$BRGC(2134, 4) = ((2134, 1100), (2134, 0000), \ldots, (2134, 1101))$$

is introduced between the elements $(id, 1100)$ and $(id, 1101)$ of L_0^B. Repeating this process for the remaining transpositions in $L_1(id)$, we obtain a Gray code for $L_{\leq 1}^B$.

In the next step, $L_{\leq 1}^B$ is extended to a Gray code for $L_{\leq 2}^B = \mathfrak{I}_4$ by inserting subsequences of L_2^B between consecutive words of L_1^B. First, we notice that $L_2(12) = \{(34)\}$, $L_2(13) = \{(24)\}$, $L_2(14) = \{(23)\}$, $L_2(23) = L_2(24) = L_2(34) = \emptyset$. Taking $p = 2143 = q \cdot (st)$, with $q = 2134$ and $(st) = (34)$, we have $\chi_{q,4} = 431$ and thus the positions corresponding to s and t are, respectively, $i = 2$ and $j = 1$. Since the signs of s and t in the involution $(2134, 1100)$ are equal, we compute $\lambda_0^3(12) = 2^{3-1} + 1 = 5$. Then, $\delta' = 1$ is the smallest position that changes between the 5-th and 6-th elements of $BRGC(2134, 4)$. Thus, the sequence

$$BRGC(2143, 1) = ((2143, 1111), (2143, 1100), (2143, 0000), (2143, 0011))$$

is introduced between the elements $(2134, 1111)$ and $(2134, 0011)$ of $BRGC(2134, 4)$. Repeating this process for the remaining sets $L_2(ij)$, we obtain the Gray code for $\mathfrak{I}_4$ displayed in Figure 4.2, where we use the colors black, blue and red to represent the words in L_0^B, L_1^B and L_2^B, respectively.

Proof: Properties (1) and (2) follow from the definition of map ϕ_n. Let $\phi_n((-1)^i n, w^i) = \sigma_1 \cdots \sigma_n$, with $\sigma_n = n$ or $\sigma_n = \overline{n}$. This means that $w^i = \sigma_1 \cdots \sigma_{n-1}$ and if π is an involution in $\varphi^{-1}(w^i)$, then there is a position fixed point in w^i, say j, such that

$$\pi = \sigma_1 \cdots \sigma_{j-1}((-1)^{sgn(\sigma_j)} \times n)\sigma_{j+1} \cdots \sigma_{n-1}\sigma_j.$$

Property (3) now follows. Also, if ρ is another involution in $\varphi^{-1}(w^i)$, then

$$\rho = \sigma_1 \cdots \sigma_{k-1}((-1)^{sgn(\sigma_k)} \times n)\sigma_{k+1} \cdots \sigma_{n-1}\sigma_k$$

for some position fixed point of w^i, say k, from which we obtain (4). ∎

Theorem 3.7. *The list $B_1 B_2 \cdots B_k$, obtained by Algorithm 1, is a cyclic Gray code for the signed involutions in $\mathfrak{I}_n$ where the difference between any two consecutive involutions is a transposition, a rotation of 3 elements and/or a sign.*

Proof: By construction, the first and last elements in each string B_i are the only two involutions of F_n^n in that string and they are two by two distinct, so they add up to $2t_{n-1}$. Moreover, by Lemma 3.5, the number of elements in the sets $\varphi^{-1}(w^i)$, $i = 1, \ldots, k$, which are two by two disjoint, is equal to $2(n-1)t_{n-2}$, the number of fixed points in all sign involutions of $\mathfrak{I}_{n-1}$. Thus, the list $B_1 \cdots B_k$ exhaust all involutions in $\mathfrak{I}_n$.

Consider now the difference between any two adjacent involutions π and ρ in $B_1 \cdots B_k$. If π and ρ are, respectively, the last and the first elements of strings B_i and B_{i+1}, then by (1) of Lemma 3.6 we find that their difference is given by the difference between two adjacent involutions in the Gray code $w^1, \ldots, w^k$ for $\mathfrak{I}_{n-1}$, that is it is a transposition, a rotation of 3 elements and/or a sign. If π and ρ are elements of the set $\varphi^{-1}(w^i)$, then by (3) of the previous lemma, we find that their difference is rotation of three elements and possibly a sign. If π is the first, or last, element of B_i and ρ is in the set $\varphi^{-1}(w^i)$, then again by the previous lemma, their difference is one transposition and possibly a sign. Finally, if π and ρ are, respectively, the first and last element of B_i and $\varphi^{-1}(w^i)$ is the empty word, then by the previous lemma their difference is just a sign.

Since the number of elements of $\mathfrak{I}_n$ is even, (1) of Lemma 3.6 is also true for $i = k$, considering $i + 1 = 1$, and therefore the code is cyclic. ∎

Corollary 3.8. *The Hamming distance of the Gray code generated by Algorithm 1 is 3.*

The previous algorithm needs to store the Gray code for $\mathfrak{I}_{n-1}$ in order to compute the Gray code for $\mathfrak{I}_n$. However, it is possible to implement a

recursive algorithm that generates the Gray code for $\mathfrak{I}_n$ without explicitly store the Gray code for $\mathfrak{I}_{n-1}$. An implementation of such algorithm is presented in Algorithm 2. This algorithm initially sets π as the empty word ϵ, the signal $s_i = 1$ for all $i \in [n]$, and the recursive calls are triggered by the initial call *expand(ϵ,1)*. For the current value of $k \leq n$, the word π is in $\mathfrak{I}_{k-1}$ and will be expanded to a set of words in $\mathfrak{I}_k$. The first word generated in this process is $\phi_k(s_k \times k, \pi)$ in line 4, where s_k is the last signal used in the Gray code for $\mathfrak{I}_k$ in previous calls, and then the algorithm generates recursively all the words expanded from $\phi_k(s_k \times k, \pi)$ through the call *expand(π,$k+1$)* in line 5. This set of words corresponds to the subtree with root $\phi_k(s_k \times k, \pi)$ in Figure 3.1. Next, in lines 6–16 the algorithm computes the set $\varphi_k^{-1}(\pi)$. For each position fixed letter $j \in [\pm(k-1)]$, the algorithm assigns to π the word obtained by transposing the letters in positions $|j|$ and k of $\phi_k(s_k \times k, \pi)$, and generates recursively all the words expanded from π through the call *expand(π,$k+1$)* in line 12. Lines 13 and 14 undo the transposition $(|j|, k)$ in the word π, ending this cycle. Finally, the word $\phi_k(-s_k \times k, \pi)$ is computed in line 17, all the words expanded from π are generated in line 18, and the signal s_k is updated.

An algorithm for the exhaustive generating of a class of combinatorial objects is said to run in constant amortized time (CAT) if it generates each object in $O(1)$ time, in amortized sense. A recursive generating algorithm is said to be a CAT algorithm [7] if the following properties are satisfied:

- Each recursive call either generates an object or produces at least two recursive calls;
- The amount of computation in each recursive call is proportional to the number of subsequent recursive calls produced by current call.

Algorithm 2 generates a Gray code for $\mathfrak{I}_n$ and, by construction, satisfies the previous CAT conditions, and so it is an efficient exhaustive generating algorithm.

Running over all involutions of L_k, this construction produces a sequence with all signed involutions in $L^B_{\leq k}$.

The sequence of $L^B_{\leq k}$, $k \geq 1$, obtained by the inductive step, is a concatenation of subsequences of $L^B_{\leq k-1}$ and sequences of the form $BRGC((p,g),\delta)$. In each subsequence $BRGC((p,g),\delta)$, Lemma 4.7 guarantees that the difference between consecutive elements of this sequence is a single sign or a paired sign. By induction, each consecutive elements in $L^B_{\leq k-1}$ is either a transposition, or a single sign, or a paired sign. Finally, note that the difference between the words connecting $L^B_{\leq k-1}$ to $BRGC((p,g),\delta)$, *i.e.* between (q, g^ℓ) and (p, h^1), as well as between $(p, h^{2^{n-k}})$ and $(q, g^{\ell+1})$, is a single transposition. ∎

Remarks.

(1) In the proof above, for the case $k = 1$, for each $p \in L_1$ we have always $q = id$ and $g^1 = 0^n$. Consequently, the map λ_1^n is never used to construct the sequence $L^B_{\leq 1}$.

(2) In the construction of $BRGC((p,g),\delta)$, only the first and last letters of $\chi_{p,\delta}$ are relevant for the proof of Theorem 4.8. The proof is still valid if any reordering of the letters in $\chi_{p,\delta}$, keeping the first and last ones invariant, is used. This fact will be used in the recursive implementation of Algorithm 3 to reduced its computational cost.

(3) There is no cyclic Gray code for $\mathfrak{I}_2$ in the conditions of Theorem 4.8, as its clear from Figure 4.1, where an edge is displayed between two words that differs by a transposition, a single sign or a paired sign. Nevertheless, there are (non cyclic) Gray codes for $\mathfrak{I}_2$ satisfying these conditions. For example,

$$\left(1\,\overline{2},\ 1\,2,\ \overline{1}\,2,\ \overline{1}\,\overline{2},\ \overline{2}\,\overline{1},\ 2\,1\right)$$

is such a code.

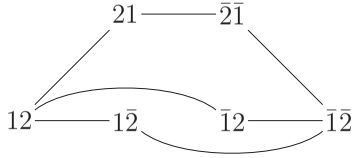

FIGURE 4.1

(4) It is easy to check that in any Gray code for $\mathfrak{I}_n$, two successive signed involutions differ by at least a transposition or a single sign. Nevertheless, with only these two operations it is not possible to

Proof: We will prove by induction that Algorithm 3 produces a Gray code for $L^B_{\leq k}$, where each element differs from its successor by a transposition, a single sign or a paired sign, and for each $p \in L_k$, there are $\delta \in [n]$ and $g \in B_n$ such that $BRGC((p,g),\delta)$ is a subsequence of the Gray code. The desired cyclic Gray code for the sign involutions is, therefore, the code for $L_{\leq \lfloor n/2 \rfloor}$.

As the code for L^B_0 is the sequence $BRGC(id,1) = \big((id,g^1),\ldots,(id,g^{2^n})\big)$, the difference between two consecutive elements of L^B_0 is just a sign.

For each transposition $p = (st)$ in L_1, insert the sequence

$$BRGC((p,g^\ell),\delta') = \Big((p,h^1),\ldots,(p,h^{2^{n-1}})\Big)$$

after position $\ell := \lambda^n_0(st)$ in L^B_0, where δ' is the only letter in g^ℓ such that $g^\ell_{\delta'} \neq g^{\ell+1}_{\delta'}$. In the case $\ell = 2^n$ we consider $\ell + 1 \equiv 1$ since the code $BRGC_n$ is cyclic. This construction produces a sequence with all signed involutions in $L^B_{\leq 1}$ in the required conditions.

Assume the algorithm produces a Gray code for $L^B_{\leq k-1}$, $k \geq 2$, in the stated conditions. Given $p \in L_k$, by Lemma 4.5, there is a unique involution $q \in L_{k-1}$ such that $p \in L_k(q)$, with $p = q \cdot (st)$. By induction hypothesis, $BRGC((q,g^1),\delta) = \Big((q,g^1),\ldots,(q,g^{2^{n-k+1}})\Big)$ is a subsequence of the Gray code for $L^B_{\leq k-1}$. Let $\chi_{q,\delta} = \widehat{q}_1 \cdots \widehat{q}_{n-k+1}$. By the definition of $\chi_{q,\delta}$, there are two integers i, j, distinct from $n-k+1$, such that

$$\widehat{q}_i = s \text{ and } \widehat{q}_j = t.$$

By Lemma 4.3 and the construction of $BRGC((q,g^1),\delta)$, there is a position $\ell < 2^{n-k+1}$ such that in the sequence $BRGC((q,g^1),\delta)$ we have $g^\ell_s = g^\ell_t = g^{\ell+1}_s = g^{\ell+1}_t$. This position is $\ell = \lambda^{n-k+1}_0(ij)$ if $g^1_s = g^1_t$, or $\ell = \lambda^{n-k+1}_1(ij)$ otherwise. Note that for a fixed $q \in L_{k-1}$, Proposition 4.2 says that there is a different position ℓ for each $p \in L_k(q)$, using the maps λ^{n-k+1}_0, or λ^{n-k+1}_1, if $g^1_s = g^1_t$, or $g^1_s \neq g^1_t$, respectively. Next, we insert the sequence $BRGC((p,g^\ell),\delta') = \Big((p,h^1),\ldots,(p,h^{2^{n-k}})\Big)$ between (q,g^ℓ) and $(q,g^{\ell+1})$, where $h^1 = g^\ell$, $h^{2^{n-k}} = g^{\ell+1}$ and δ' is the smallest letter such that $g^\ell_{\delta'} \neq g^{\ell+1}_{\delta'}$.


```
1  procedure expand(π, k) begin
2      if k > n then write π ;
3      else
4          π_k ← s_k × k;
5          expand(π, k + 1);
6          for j ← 1 to k − 1 do
7              if |π_j| = j then
8                  aux ← π_j;
9                  if sgn(π_k) == sgn(π_j) then π_j ← π_k;
10                 else π_j ← −π_k;
11                 π_k ← aux;
12                 expand(π, k + 1);
13                 π_k ← s_k × k;
14                 π_j ← aux;
15             end
16         end
17         π_k ← −π_k;
18         expand(π, k + 1);
19         s_k ← −s_k;
20     end
21 end
```

Algorithm 2: CAT implementation of Algorithm 1.

4. A Gray code with minimal Hamming distance

In this section we develop a second Gray code for the signed involutions. Although more elaborated than the first code, the present code has the smallest possible Hamming distance, since the difference between any two consecutive elements will be a transposition or at most a paired sign. Our approach is based on the Binary Reflected Gray Code [5], $BRGC_n$, which we briefly describe next.

The $BRGC_1$ is the list $(0,1)$, and the $BRGC_{n+1}$ is obtainable by first listing $BRGC_n$, with each word prefixed by 0, and then listing the $BRGC_n$ in reverse order with each word prefixed by 1. For instance, $BRGC_2 = (00,01,11,10)$ and $BRGC_3 = (000,001,011,010,\ 110,111,101,100)$.

Each one of the 2^n elements of the sequence $BRGC_n$ differs from the previous one by only one bit, and the same is true for the last and the first elements of the sequence. Moreover, it is possible to identify the element

in the ℓ-th position of the $BRGC_n$ code by the map $\alpha^n : [2^n] \to [0, 2^n - 1]$ defined below.

Letting $(x_k)_2$ and $(y_k)_2$ be the binary expansions of the integers $x = \sum_{k=1}^{n} x_k 2^{n-k}$ and $y = \sum_{k=1}^{n} y_k 2^{n-k}$, define

$$x \oplus y := (x_k \oplus y_k)_2 \equiv \sum_{k=1}^{n} (x_k \oplus y_k) 2^{n-k},$$

where $x_k \oplus y_k$ is the binary exclusive or –XOR– operation. It is easy to check (see [7]) that the element in the ℓ-th position of the $BRGC_n$ code is given by

$$\alpha^n(\ell) = (\ell - 1) \oplus \left\lfloor \frac{\ell - 1}{2} \right\rfloor.$$

In the following, we introduce some auxiliar functions and notations needed for the construction of our second Gray code.

Definition 4.1. Given a positive integer $n \geq 3$, let T_n be the set of all transpositions of $\mathfrak{S}_n$. For each $n \geq 3$, we define two functions $\lambda_0^n : T_n \to [2^n]$ and $\lambda_1^n : T_{n-1} \to [2^n]$ with

$$\lambda_0^n(ij) = \begin{cases} 2^{n-i} + 2^{n-j} + 1, & \text{if } i+1 < j \\ 2^{n-i} + 1, & \text{if } i+1 = j < n \\ 2^n, & \text{if } i+1 = j = n \end{cases};$$

and $\lambda_1^n(k\ell) = 2^{n-k+1} - 2^{n-\ell} + 1$.

Proposition 4.2. (1) *The functions λ_0^n and λ_1^n are injective.*
(2) *For all $(ij) \in T_n$ and $(k\ell) \in T_{n-1}$, $\lambda_0^n(ij) \neq \lambda_1^n(k\ell)$.*

Proof: (1) We start by showing that λ_0^n is injective, for any $n \geq 3$. By the uniqueness of the binary expansion of a natural number, it follows that any two numbers of the form $2^{n-i} + 2^{n-j} + 1$, with $i \neq j$, are distinct. The same is true for any two numbers of the form $2^{n-i} + 1$. We must show that these two classes of numbers and the number 2^n do not overlap. The elements in the range of λ_0^n are written in one of the following forms: 2^n, or $2^a + 2^b + 1$ with $1 \leq b + 1 < a < n$, or $2^c + 1$ with $2 \leq c \leq n - 1$. Since $n, c > 0$, the number $2^c + 1$ is an odd number while 2^n is even, so that these two numbers cannot be equal. Again, by the uniqueness of the binary expansion we have that $2^a + 2^b + 1 = 2^c + 1$ if and only if $a = b = c - 1$, but this contradicts the inequality $b + 1 < a$. Using the same argument, we see that $2^a + 2^b + 1 = 2^n$ if and only if $a = 1$, $b = 0$ and $n = 2$, which is not possible since we are assuming $n \geq 3$.

the difference between two consecutive elements of this sequence is a single sign or a paired sign. ∎

Our second Gray code for $\mathfrak{I}_n$ is constructed by layers: in the first stage we construct the code for the involutions in L_0^B as the sequence $(id, g^1), \ldots, (id, g^{2^n})$, where $(g^1, \ldots, g^{2^n}) = BRGC_n$. Having the code for $L_{\leq k-1}^B$, we construct the code for $L_{\leq k}^B$ by inserting sequences of involutions in L_k^B between two consecutive elements of L_{k-1}^B such that each element of the sequence for $L_{\leq k}^B$ differs from the previous by a transposition or at most a paired sign. For each $p \in L_k$, this insertion process is described below.

```
1 procedure insert(p)
2 begin
3 |   Let q be the unique involution in L_{k-1} such that p ∈ L_k(q)
  |   with p = q · (st).
4 |   Let (q, r) and (q, r̃) be two consecutive elements of L^B_{k-1} such
  |   that r_s = r̃_s and r_t = r̃_t.
5 |   Let δ be the smallest letter such that r_δ ≠ r̃_δ.
6 |   Introduce the sequence BRGC((p, r), δ) between the elements
  |   (q, r) and (q, r̃).
7 end
```

The elements (q, r) and $(q, \widetilde{r})$ will be found using the maps introduced in Definition 4.1. We are now in conditions to construct an algorithm which outputs a cycle Gray code with Hamming distance 2 for the signed involutions in $\mathfrak{S}_n^B$, for $n \geq 3$.

```
1 L_0^B ← BRGC(id, 1).
2 for k ← 1 to ⌊n/2⌋ do
3 |   L^B_{≤k} ← L^B_{≤k-1}
4 |   for each p ∈ L_k do
5 |   |   insert(p)
6 |   end
7 end
```

Algorithm 3: Algorithm for the second Gray code.

Theorem 4.8. *The sequence obtained by the successive application of Algorithm 3 gives a cyclic Gray code for the signed involutions in $\mathfrak{I}_n$, $n \geq 3$, where each element differs from its successor either by a transposition, or a single sign, or a paired sign.*

element of this new code is a given $g \in B_n^p$, and the last element differs from the first only in a given position δ. This variation is achieved by the following procedure, which have as input an involution $\sigma \equiv (p, g)$, where $p \in L_k$ with $k \geq 1$, and an integer $\delta \in [n]$, distinct from $m(p)$ and from any closure of p, and outputs the binary code $BRGC(\sigma, \delta)$.

```
1 procedure BRGC(σ, δ):
2 begin
3 |  Let ĝ := Ω_{p,δ}(g).
4 |  Construct the sequence ĝ^1, ..., ĝ^{2^{n-k}} of binary words, where
  |  each ĝ^i := ĝ ⊕ x^i = ĝ^i_1 ··· ĝ^i_{n-k}, with
  |  (x^1, ..., x^{2^{n-k}}) = BRGC_{n-k}.
5 |  Let BRGC(σ, δ) := ((p, g^1), ..., (p, g^{2^{n-k}})), where each
  |  g^i = Ω^{-1}_{p,δ}(ĝ^i).
6 end
```

Notice that $\left(g^1, \ldots, g^{2^{n-k}}\right) = \Omega_{p,\delta}^{-1}\left(\Omega_{p,\delta}(g) \oplus BRGC_{n-k}\right)$. Moreover, $g^1 = g$ and $g^{2^{n-k}}$ differs from g only in position δ, if δ is a position fixed letter of p, and also in position p_δ if δ is an opener of p.

When $\sigma = id$, we set $BRGC(id, \delta) := \left((id, x^1), \ldots, (id, x^{2^n})\right)$, where $\left(x^1, \ldots, x^{2^n}\right) = BRGC_n$.

Example 4.1. For instance, if $\sigma \equiv (2134, 1100)$ and $\delta = 4$, the code $BRGC(\sigma, 4)$ is achieved as follows. First, note that $p = 2134$ with $m(p) = 1$. Therefore, $\widehat{p} = 431$, $\widehat{g} = \Omega_{p,\delta}(1100) = 001$ and the procedure outputs the code

$$BRGC(\sigma, 4) = ((p, 1100), (p, 0000), (p, 0010), (p, 1110), (p, 1111), (p, 0011), (p, 0001), (p, 1101)).$$

Lemma 4.7. *Given $\sigma \equiv (p, g) \in L_k$ and $\delta \in [n]$, the difference between each consecutive words of the binary code $BRGC(\sigma, \delta)$ is a single sign or a paired sign of σ.*

Proof: The sequence $(x^1, \ldots, x^{2^{n-k}})$ is the $BRGC_{n-k}$, and each word $\widehat{g}^i$, in step 4 of procedure $BRGC(\sigma, \delta)$, is obtained from x^i by changing all the signs x^i_j where $\widehat{g}_j = 1$. Therefore, the sequence $\widehat{g}^1, \ldots, \widehat{g}^{2^{n-k}}$ is a rearrange of the $BRGC_{n-k}$ preserving the difference between consecutive words. In step 5, each word of this sequence is expanded repeating the signs, associated to the openers in p, on the positions of the corresponding closures. Thus,

We will now prove that λ_1^n is an injection. The range of λ_1^n is the set of the numbers $2^p - 2^q + 1$ with $1 < q + 1 < p \leq n$. It is not difficult to see that for $1 < q + 1 < p$,

$$2^{p-1} < 2^p - 2^q + 1 < 2^p.$$

It follows that it not possible for an integer to be written in the form $2^p - 2^q + 1$ in two distinct ways.

(2) First we notice that $2^p - 2^q + 1 = 2^c + 1$ if and only if $c = q = p - 1$. Since we are assuming that $q < p - 1$, this equality cannot occur. For $b + 1 < a$, we have

$$2^a < 2^a + 2^b + 1 < 2^{a+1},$$

which implies that if $2^a + 2^b + 1 = 2^p - 2^q + 1$, then $a + 1 = p$ and thus $2^a + 2^b + 1 = 2^{a+1} - 2^q + 1 \Leftrightarrow 2^a = 2^b + 2^q \Rightarrow q = b = a - 1$. Since $b < a - 1$ we deduce that $2^a + 2^b + 1 \neq 2^p - 2^q + 1$ for any $b + 1 < a$ and any $q + 1 < p$.

Finally $2^p - 2^q + 1 = 2^n$ if and only if $p = n$ and $q = 0$, which is not possible since $q \geq 1$. ∎

Lemma 4.3. *For a fixed $n \geq 3$ and $i < j$, we have the following conditions:*

(1) *If $\lambda_0^n(ij) \neq 2^n$, $\alpha^n\left(\lambda_0^n(ij)\right) \equiv (x_k)_2$ and $\alpha^n\left(\lambda_0^n(ij) + 1\right) \equiv (y_k)_2$, then $x_i = y_i = x_j = y_j$;*
(2) *If $\lambda_0^n(ij) = 2^n$, $\alpha^n\left(2^n\right) \equiv (x_k)_2$ and $\alpha^n(1) \equiv (y_k)_2$, then $x_i = y_i = x_j = y_j$;*
(3) *If $\alpha^n\left(\lambda_1^n(ij)\right) \equiv (x_k)_2$ and $\alpha^n\left(\lambda_1^n(ij) + 1\right) \equiv (y_k)_2$, then $x_i = y_i \neq x_j = y_j$.*

Proof: (1) Let i, j be two integers in $[n]$ such that $i + 1 < j < n$. As the number $\lambda_0^n(ij) = 2^{n-i} + 2^{n-j} + 1$ is odd, its successor in the $BRGC_n$ code only differs from it in the last position, which implies that $x_i = y_i$ and $x_j = y_j$. It remains to show that $x_i = x_j$. By the definitions of the functions α^n and λ_0^n,

$$\begin{aligned}\alpha^n\left(\lambda_0^n(ij)\right) &= \left\lfloor \frac{2^{n-i} + 2^{n-j} + 1 - 1}{2} \right\rfloor \oplus \left(2^{n-i} + 2^{n-j} + 1 - 1\right) \\ &= \left(2^{n-i-1} + 2^{n-j-1}\right) \oplus \left(2^{n-i} + 2^{n-j}\right) \\ &= 2^{n-i} + 2^{n-(i+1)} + 2^{n-j} + 2^{n-(j+1)},\end{aligned}$$

since $i + 1 < j < n$, and thus $x_i = x_j = 1$.

Assume now that i, j are positive integers such that $i + 1 < j = n$. The number $\lambda_0^n(in) = 2^{n-i} + 2^{n-n} + 1 = 2^{n-i} + 2$ is equivalent to 2 (mod 4), since $n - i \geq 2$. Thus, in the $BRGC_n$ code, the successor of $\lambda_0^n(in)$ is obtained by changing the digit in the penultimate position. Since both i and n are

different from $n-1$, we have that $x_i = y_i$ and $x_n = y_n$. As in the previous case, we compute

$$\begin{aligned}\alpha^n\left(\lambda_0^n(ij)\right) &= \left\lfloor \frac{2^{n-i}+2-1}{2} \right\rfloor \oplus \left(2^{n-i}+2-1\right)\\ &= 2^{n-i-1} \oplus \left(2^{n-i}+1\right)\\ &= 2^{n-i} + 2^{n-(i+1)} + 2^{n-n},\end{aligned}$$

since $i+1 < n$, and thus $x_i = x_n = 1$.

Let now $j = i+1 < n$. The number $\lambda_0^n(ij) = 2^{n-i}+1$ is odd which implies that its successor in the $BRGC_n$ code only differs from it in the last position. It follows that $x_i = y_i$ and $x_j = y_j$. Again, we compute

$$\begin{aligned}\alpha^n\left(\lambda_0^n(ij)\right) &= \left\lfloor \frac{2^{n-i}+1-1}{2} \right\rfloor \oplus \left(2^{n-i}+1-1\right)\\ &= 2^{n-i-1} \oplus \left(2^{n-i}\right)\\ &= 2^{n-i} + 2^{n-(i+1)},\end{aligned}$$

and thus $x_i = x_{i+1} = 1$.

(2) $\lambda_0^n(ij) = 2^n$ if $i = n-1$ and $j = n$. In the last element of the $BRGC_n$ code we have that $x_1 = 1$ and $x_k = 0$, for $k \neq 1$, and in the first one, $y_k = 0$ for all values of k. Since $n \geq 3$, it follows that $x_{n-1} = x_n = 0$.

(3) The domain of λ_1^n is T_{n-1} and then $i < j < n$. As before, we notice that $\lambda_1^n(ij) = 2^{n-i+1} - 2^{n-j} + 1$ is odd. Therefore the successor of $\lambda_1^n(ij)$ in the $BRGC_n$ code only differs from it in the last position, which implies that $x_i = y_i$ and $x_j = y_j$.

Once again we compute

$$\begin{aligned}\alpha^n\left(\lambda_0^n(ij)\right) &= \left\lfloor \frac{2^{n-i+1}-2^{n-j}+1-1}{2} \right\rfloor \oplus \left(2^{n-i+1}-2^{n-j}+1-1\right)\\ &= \left(2^{n-i}-2^{n-j-1}\right) \oplus \left(2^{n-i+1}-2^{n-j}\right)\\ &= \sum_{k=j+1}^{i+1} x_k 2^{n-k} \oplus \sum_{k=j}^{i} x_k 2^{n-k}\\ &= 2^{n-i} + 2^{n-(j+1))},\end{aligned}$$

and thus $x_i = 1 \neq 0 = x_j$. ∎

Definition 4.4. Given $n \in \mathbb{N}$, and $0 \leq k \leq \lfloor n/2 \rfloor$, let L_k denote the set of all involutions in $\mathfrak{S}_n$ having exactly k transpositions, and let

$$L_{\leq k} = \bigcup_{i=0}^{k} L_k$$

be the set of all involutions having at most k transpositions. For a sign involution $\sigma \equiv (p, g)$, we say that σ is in L_k^B whenever $p \in L_k$.

Given an involution $q \in L_{k-1}$, $k \geq 1$, we define

$$L_k(q) := \{q \cdot (ij) \,:\, m(q) < i < j,\, q_i = i,\, q_j = j\} \subseteq L_k.$$

We recall that $m(q)$ is the maximum of the openers in q, and $m(q) = 0$ when q is the identity.

Lemma 4.5. (a) $L_k = \bigcup_{q \in L_{k-1}} L_k(q)$;

(b) *if* $q \neq q'$, *then* $L_k(q) \cap L_k(q') = \emptyset$.

Proof: Follows from the uniqueness representation of an involution $p \in L_k$ as a product of disjoint transpositions $p = (a_1 b_1) \cdot (a_2 b_2) \cdots (a_k b_k)$ with $a_1 < a_2 < \cdots < a_k$. ∎

Definition 4.6. Given an involution $p \in L_k$, $k \geq 1$, let $s = m(p)$ and $\delta \in [n]$ an integer distinct from s and from any closure of p.

(a) Define the word

$$\chi_{p,\delta} := \widehat{p}$$

obtained from p by removing the closures of all transpositions of p, as well as the letters δ and s, and then inserting δ in the leftmost position, and s in the rightmost position, *i.e.* setting $\widehat{p}_1 = \delta$, $\widehat{p}_{n-k} = s$.

(b) Let $B_n^p = \{g \in B_n : (p, g) \in \mathfrak{I}_n\}$, and define the map

$$\Omega_{p,\delta} : B_n^p \longrightarrow B_{n-k},$$

where $\Omega_{p,\delta}(g) := \widehat{g}$ is the binary word corresponding to $\widehat{p}$, *i.e.* $\widehat{g}_i = g_{\widehat{p}_i}$.

For each pair (p, δ) in the conditions above, the map $\Omega_{p,\delta}$ is a one-to-one correspondence between the sets B_n^p and B_{n-k}, since it replaces each paired signal by a single sign, and rearranges the remaining word.

We will use the map $\Omega_{p,\delta}$ to construct a variation of the Binary Reflexive Gray Code, where each paired signal is treated as a single sign, the first