A New Coding Algorithm for Trees

Vladimir B. Balakirsky

Euler Institute for Discrete Mathematics
and its Applications (EIDMA)
c/o Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
v.b.balakirsky@ele.tue.nl

Abstract

We construct a one–to–one mapping between binary vectors of length n and preorder codewords of regular, ordered, oriented, rooted, binary trees having $N \approx n + 2\log n$ nodes. The mappings in both directions can be organized in such a way that complexities of all transformations are measured by linear functions of n. The approach is then completely extended to non–regular binary trees and partially extended to D–ary trees with $D > 2$.

1 Statement of the problem and formulation of the result

Suppose that any given binary vector of length n should be bijectively mapped to a regular, ordered, oriented, rooted, binary tree (we will say "regular binary tree"). Since the number of trees having N nodes can be expressed as the Catalan number [1], the problem can be solved if and only if N is odd and

$$2^n \leq \frac{1}{N}\binom{N}{(N-1)/2}. \tag{1}$$

Any regular binary tree T having N nodes can be specified by a binary vector of length N, called the preorder codeword [2], which is recurrently introduced as

$$\mathbf{c}(T) \triangleq \begin{cases} 0, & \text{if } T \text{ is an isolated vertex} \\ (1, \mathbf{c}(T_0), \mathbf{c}(T_1)), & \text{otherwise} \end{cases}$$

where T_0 and T_1 are the left and right subtrees of T, respectively. We will refer to the mapping of a given binary vector of length n to the preorder codeword of a tree as *encoding* and refer to the mapping of a given preorder codeword of a tree to the binary vector of length n as *decoding* (these mappings are also called "ranking" and "unranking"). We assume that *coding* stands for the pair of mappings (encoding, decoding).

The problem under consideration, when N is the smallest odd integer satisfying (1), was addressed by many authors ([3], [4], [5], and other papers). However, implementation of the known coding algorithms seems to be difficult if n is big enough. We develop the approach, proposed for solving the data transmission problem in asynchronous mode [6], and include a small redundancy in the representation, which allows us to avoid the difficulties. The inequality

$$N \geq n + (3/2)\log n$$

which follows from (1) and Stirling's approximation formula for the factorial, can be helpful to evaluate the number of additional redundant bits.

Theorem 1. *Let n be a given odd integer and let δ be the smallest odd integer satisfying the inequality*

$$n + 1 \leq \frac{1}{\delta}\binom{\delta}{(\delta-1)/2}.$$

For any binary vector $\mathbf{x}$ of length n, there is an explicit construction of a preorder codeword $f(\mathbf{x})$ for a regular binary tree having

$$N = n + 2\delta - 2 \tag{2}$$

nodes, which can be implemented in such a way that complexities of encoding and decoding are measured by linear functions of n.

Some numerical results are given in Table 1. If n is not very small, then $\delta \approx \log n + (3/2)\log\log n$, as it follows from Stirling's approximation for the factorial.

The idea of our approach to the coding problem can be summarized as follows. We translate the encoding and decoding procedures to the logarithmic scale, i.e., instead of dealing with 2^n different regular binary trees having approximately n nodes, we deal with $n+1$ trees having approximately $\log n$ nodes. It turns out that such a translation can be derived if N is expressed by (2).

The paper is organized as follows. The encoding and decoding algorithms, which lead to the proof of Theorem 1, are presented in Section 2. The justification of all transformations, included in their description, is given in Section 3. Possible extensions of our approach to D–ary trees are presented in Section 4. In particular, we show that all previous considerations are valid for non–regular binary trees after minor modification.

2 Encoding and decoding algorithms

We assume that the list of preorder codewords for regular binary trees having δ nodes can be formed in advance and stored in the memory in some order (see Table 2 where $\delta = 9$ and the codewords are listed in the lexicographic order).

Encoding algorithm :

- Find $j \in \{0, \ldots, n\}$ such that the inversion of the first j components of the vector $\mathbf{x} = (x_0, \ldots, x_{n-1})$ transforms it to *the balanced vector* $\mathbf{x}' = (x'_0, \ldots, x'_{n-1})$ having $(n+1)/2$ zeroes and $(n-1)/2$ ones, i.e.,

$$x'_0 + \ldots + x'_{n-1} = (n-1)/2$$

  where

$$x'_i \triangleq \begin{cases} x_i \oplus 1, & \text{if } \ i \in \{0, \ldots, j-1\} \\ x_i, & \text{if } \ i \in \{j, \ldots, n-1\}. \end{cases}$$

- Find $k \in \{0, \ldots, n-1\}$ such that the cyclic shift of the vector $\mathbf{x}'$ by k positions to the right transforms it to the vector $\mathbf{z} = (z_0, \ldots, z_{n-1})$ having the dominating property, i.e.,

$$z_0 + \ldots + z_{t-1} \geq t/2, \ \text{ for all } \ t = 1, \ldots, n-1$$

where

$$z_i \triangleq \begin{cases} x'_{n+i-k}, & \text{if } i \in \{0,\ldots,k-1\} \\ x'_{i-k}, & \text{if } i \in \{k,\ldots,n-1\}. \end{cases}$$

– Let $\mathbf{a} = (a_0,\ldots,a_{\delta-1})$ and $\mathbf{b} = (b_0,\ldots,b_{\delta-1})$ are the preorder codewords numbered k and j in the list of codewords for regular binary trees having δ nodes. Construct the vector $f(\mathbf{x})$ in the following way : delete the last bit 0 in the vectors $\mathbf{z},\mathbf{a},\mathbf{b}$, concatenate the received vectors, and attach bit 0. Thus,

$$f(\mathbf{x}) = (z_0,\ldots,z_{n-2},a_0,\ldots,a_{\delta-2},b_0,\ldots,b_{\delta-2},0). \tag{3}$$

Example of encoding for $n = 13$ is given in Table 3, where we use data of Table 2 to encode k and j.

Decoding algorithm :

The decoder constructs preorder codewords $\mathbf{a},\mathbf{b}$ based on the last $2\delta - 1$ bits of the given vector $f(\mathbf{x})$. These codewords and the list of codewords for regular binary trees having δ nodes specify integers k and j. Furthermore, the decoder constructs the preorder codeword $\mathbf{z}$ based on the first $n-1$ bits of the vector $f(\mathbf{x})$. The cyclic shift of the vector $\mathbf{z}$ by k positions to the left and inversion of the first j components reconstructs the vector $\mathbf{x}$.

3 Justification of the algorithms

3.1 Transformation of a given binary vector to a balanced vector : $\mathbf{x} \to \mathbf{x}'$

The transformation $\mathbf{x} \to \mathbf{x}'$ is based on the following fact discovered by D. E. Knuth [7]. Let $\mathbf{x}_0 \triangleq \mathbf{x}$ and

$$\mathbf{x}_i \triangleq (x_0 \oplus 1,\ldots,x_{i-1} \oplus 1, x_i,\ldots,x_{n-1})$$

for all $i = 1,\ldots,n$. Let w_i denote the number of ones in the vector $\mathbf{x}_i$ (the Hamming weight of $\mathbf{x}_i$) for all $i = 0,\ldots,n$. Then the path on the plane (see Figure 1), specified by the points $(0,w_0),\ldots,(n,w_n)$, has increments ± 1 and there always exists a $j \in \{0,\ldots,n\}$ such that $w_j = (n-1)/2$.

Let $s_0 \triangleq 0$ and let

$$s_t \triangleq 2(x_0 + \ldots + x_{t-1}) - t, \quad \text{for all} \;\; t = 1, \ldots, n$$

denote the difference between the number of ones and the number of zeroes in the first t positions of the vector $\mathbf{x}_0$. Then

$$x_0 + \ldots + x_{j-1} = (s_j + j)/2$$

and

$$x_j + \ldots + x_{n-1} = (s_n - s_j + n - j)/2.$$

Therefore, since

$$(x_0 \oplus 1) + \ldots + (x_{j-1} \oplus 1) = j - (x_0 + \ldots + x_{j-1})$$

the requirement that $\mathbf{x}_j$ is a balanced vector is equivalent to the equation

$$s_n = 2s_j - 1. \tag{4}$$

We define j as the smallest index, for which (4) holds.

3.2 Transformation of a balanced binary vector to the vector having the dominating property : $\mathbf{x}' \to \mathbf{z}$

The transformation $\mathbf{x}' \to \mathbf{z}$ is based on the following statement, known as Raney's lemma [8], pp. 344–346 (this result is also mentioned in [9] with the reference to the paper by W. A. Whitworth published in 1878) : *for any odd n, the set of all binary vectors of length n containing $(n+1)/2$ zeroes and $(n-1)/2$ ones can be uniquely partitioned in*

$$\frac{1}{n}\binom{n}{(n-1)/2}$$

cyclic groups, and each group contains exactly one vector having the dominating property. Furthermore, if this vector has to be received by the cyclic shift of a given balanced vector $\mathbf{x}'$ by k positions to the right, then the value of the parameter k is defined by

$$\begin{cases} s'_i > s'_k, & \text{if } i \in \{0, \ldots, k-1\} \\ s'_i \geq s'_k, & \text{if } i \in \{k, \ldots, n-1\} \end{cases} \tag{5}$$

where $s'_0 \triangleq 0$ *and*

$$s'_t \triangleq 2(x'_0 + \ldots + x'_{t-1}) - t, \quad \textit{for all} \ \ t = 1, \ldots, n.$$

It is easy to see that

$$s'_i = \begin{cases} -s_i, & \text{if } \ i \in \{0, \ldots, j-1\} \\ s_i - 2s_j, & \text{if } \ i \in \{j, \ldots, n-1\}. \end{cases}$$

Thus, the knowledge of the sequence $s_1, \ldots, s_n$ is sufficient for organizing the transformations $\mathbf{x} \to \mathbf{x}' \to \mathbf{z}$.

3.3 Construction of the codeword $f(\mathbf{x})$ as a concatenation of three vectors having the dominating property

It is easy to see that (3) and the inequalities $z_0 + \ldots + z_{t-1} \geq t/2$ for all $t = 1, \ldots, n-1$ and $a_0 + \ldots + a_{i-1}$, $b_0 + \ldots + b_{i-1} \geq i/2$ for all $i = 1, \ldots, \delta - 1$ imply $y_0 + \ldots + y_{t-1} \geq t/2$ for all $t = 1, \ldots, N-1$, where $y_0, \ldots, y_{N-1}$ denote components of the vector $f(\mathbf{x})$. Furthermore

$$y_0 + \ldots + y_{N-2} = (n-1)/2 + 2(\delta - 1)/2 = (N-1)/2$$

and $y_{N-1} = 0$ leads to the conclusion that $f(\mathbf{x})$ is the preorder codeword for a tree having N nodes.

4 Extension of the approach to D–ary trees

4.1 Coding algorithm for D–ary regular trees

Any node of a regular, ordered, oriented, rooted D–ary tree T (we will say "regular D–ary tree") is either a terminal node, or it has D sons numbered $1, \ldots, D$. The tree is uniquely specified by the binary vector $\mathbf{c}_D(T)$, which can be defined by the following recurrent formula [8]

$$\mathbf{c}_D(T) \triangleq \begin{cases} 0, & \text{if } T \text{ is an isolated vertex} \\ (1, \mathbf{c}_D(T_1), \ldots, \mathbf{c}_D(T_D)), & \text{otherwise} \end{cases}$$

where $T_1, \ldots, T_D$ are the ordered subtrees of T.

Let $K_D(n)$ be the total number of nodes of a regular D–ary tree. Then $K_D(n) = 0$ if $n-1$ is not divisible by D and

$$K_D(n) = \frac{1}{n}\binom{n}{(n-1)/D} \tag{6}$$

otherwise. Note that if n is not very small, then

$$\frac{\log K_D(n)}{n} \approx \log D - \frac{D-1}{D}\log(D-1) \tag{7}$$

as it follows from Stirling's approximation for the factorial.

Raney's lemma (see Section 3.2) is true for any D–ary regular tree as well in a sense that, for any binary vector of length n having $(n-1)/D$ ones, there is a $k \in \{0, \ldots, n-1\}$ such that the cyclic shift of this vector by k positions to the right transforms it to the binary vector $\mathbf{z}$ satisfying the inequalities

$$z_0 + \ldots + z_{t-1} \geq t/D, \quad \text{for all } \; t = 1, \ldots, n-1.$$

The value of the parameter k can be transfered to to the preorder codeword for a regular D–ary tree and attached to the vector $\mathbf{z}$ in a similar way as for regular binary trees. As a result, we come to the following statement.

Theorem 2. *Let n be a given integer such that $n-1$ is divisible by D and let δ be the smallest integer satisfying the inequality*

$$n + 1 \leq K_D(\delta)$$

where $K_D(\delta)$ is expressed in (6) *with the δ substituted for n. For any binary vector $\hat{\mathbf{x}}'$ containing $n - (n-1)/D$ zeroes and $(n-1)/D$ ones, there is an explicit construction of a preorder codeword $f(\hat{\mathbf{x}}')$ for a regular D–ary tree having*

$$N = n + \delta - 1$$

nodes, which can be implemented in such a way that complexities of encoding and decoding are measured by linear functions of n.

The result of Theorem 2 is weaker than the result of Theorem 1, where we were capable to form a balanced vector by attaching relatively few bits to any binary vector $\mathbf{x}$ of length n (in the case of regular D–ary trees, the vector is "balanced" if it contains $(n-1)/D$ ones). However, the exponent of $K_D(n)$ differs from $\log D$ for all $D > 2$, by (7). Therefore, a similar transformation is in principle impossible in this case since we are supposed to change the length of the vector by a quantity, which linearly depends on n (if the vector to be transformed is one of D^n D–ary vectors of length n, then we have to introduce redundancy in the representation). Nevertheless, the result of Theorem 2 is helpful because we reduce the problem of generating preorder codewords for regular D–ary trees and the coding problem to the known procedures that construct binary vectors having the fixed Hamming weight by searching for a path in the Pascal triangle [11], [12]. The development consists in the note that the dominating property of the vectors does not have to be controlled.

4.2 Coding algorithm for D–ary non–regular trees

Any node of a non–regular, ordered, oriented, rooted D–ary tree T (we will say "non–regular D–ary tree") is either a terminal node, or it has $d \in \{1, \ldots, D\}$ sons numbered $1, \ldots, d$. The tree T is uniquely specified by the vector $\tilde{\mathbf{c}}_D(T)$, which can be defined by the following recurrent formula [10]

$$\tilde{\mathbf{c}}_D(T) \triangleq \begin{cases} 0, & \text{if } T \text{ is an isolated vertex} \\ (d, \tilde{\mathbf{c}}_D(T_1), \ldots, \tilde{\mathbf{c}}_D(T_d)), & \text{if the root of } T \text{ has } d \geq 1 \text{ sons} \end{cases}$$

where T_d is the d–th subtree of T.

Given integers n and $D \geq 2$, let

$$\tilde{\mathcal{S}}_D(n) \triangleq \Big\{ \mathbf{n} = (n_0, \ldots, n_D) : \ n_0, \ldots, n_D \in \{0, \ldots, n\}, \quad \sum_{d=0}^{D} n_d = n, \ \sum_{d=0}^{D} (d-1) n_d = -1 \Big\} \tag{8}$$

For all $\mathbf{n} \in \tilde{\mathcal{S}}_D(n)$, let $\tilde{K}_D(\mathbf{n})$ denote the number of ordered, oriented, rooted trees having the spectrum $\mathbf{n}$, i.e., each tree under consideration has n_0 terminal nodes, n_1 nodes with 1 son, …, n_D nodes with D sons. Then [8]

$$\tilde{K}_D(\mathbf{n}) = \frac{1}{n} \begin{pmatrix} & n & \\ n_1 & \ldots & n_D \end{pmatrix} = \frac{(n-1)!}{n_0! \cdot n_1! \cdot \ldots \cdot n_D!}$$

and the total number of D–ary trees having n nodes can be expressed as

$$\tilde{K}_D(n) = \sum_{\mathbf{n} \in \tilde{\mathcal{S}}_D(n)} \frac{(n-1)!}{n_0! \cdot n_1! \cdot \ldots \cdot n_D!} \tag{9}$$

If n is big enough, then the order of magnitude of $\tilde{K}_D(n)$ is determined by the largest term and

$$\frac{\log \tilde{K}_D(n)}{n} \approx \max_{\mathbf{p} \in \tilde{\mathcal{P}}_D} \sum_{d=0}^{D} p_d \log \frac{1}{p_d} \tag{10}$$

where

$$\tilde{\mathcal{P}}_D \triangleq \Big\{ \mathbf{p} = (p_0, \ldots, p_D) : \; p_0, \ldots, p_D \in (0,1), \\ \sum_{d=0}^{D} p_d = 1, \; \sum_{d=0}^{D} (d-1) p_d = -1 \Big\}$$

The probability distribution specified by the vector $\mathbf{p}$, which maximizes the sum on the right–hand side of (10), can be found using the Lagrange multipliers method. Namely,

$$\tilde{p}_d = 2^{\lambda d} \left(\sum_{d'=0}^{D} 2^{\lambda d'} \right)^{-1}, \quad \text{for all } \; d = 0, \ldots, D$$

where the parameter λ has to be chosen in such a way that

$$\sum_{d=0}^{D} (d-1) 2^{\lambda d} = - \sum_{d=0}^{D} 2^{\lambda d}. \tag{11}$$

It is easy to see that the statement similar to Theorem 2 can be presented for non–regular trees if we replace *binary vector* $\hat{\mathbf{x}}'$ *containing* $n - (n-1)/D$ *zeroes and* $(n-1)/D$ *ones* in the formulation of Theorem 2 with the *vector* $\hat{\mathbf{x}}'$ *containing* n_0 *symbols* $0, \ldots, n_D$ *symbols* D, *where* $(n_0, \ldots, n_D) \in \tilde{\mathcal{S}}_D(n)$. In the remark, given in the previous subsection, we in fact noticed that if $(\tilde{p}_0, \ldots, \tilde{p}_D)$ is a non–uniform probability distribution, then we have to change the length of the vector by a quantity, which linearly depends on n. The set $\tilde{\mathcal{P}}_D$ does not contain the uniform distribution if the parameter λ, satisfying

(11), is non–zero. However, if $D = 2$, then the vector $(1/3, 1/3, 1/3)$ belongs to the set $\tilde{\mathcal{P}}_D$, and we arrive at a stronger statement.

Theorem 3. *Let n be a given integer and let δ be the smallest integer satisfying the inequality*

$$n + 2 \leq \tilde{K}_2(\delta)$$

where $\tilde{K}_2(\delta)$ is expressed in (9) with the δ substituted for n. For any ternary vector $\hat{\mathbf{x}}$ of length n, there is an explicit construction of a preorder codeword $f(\hat{\mathbf{x}}) \in \{0, 1, 2\}$ for a non–regular binary tree having

$$N = n + 2\delta - 1$$

nodes, which can be implemented in such a way that complexities of encoding and decoding are measured by linear functions of n.

Encoding algorithm :

– Set

$$\hat{x}_n \triangleq \begin{cases} 1, & \text{if } \ n - n_1 \ \text{ is odd} \\ 0, & \text{if } \ n - n_1 \ \text{ is even} \end{cases}$$

where n_1 is the number of 1's in the vector $(\hat{x}_0, \ldots, \hat{x}_{n-1})$.

– Find $j \in \{0, \ldots, n+1\}$ such that the substitutions $0 \to 2$ and $2 \to 0$ at the first $j - 1$ components of the vector $\hat{\mathbf{x}} = (\hat{x}_0, \ldots, \hat{x}_n)$ transforms it to the vector $\hat{\mathbf{x}}' = (\hat{x}'_0, \ldots, \hat{x}'_n)$ such that

$$(\hat{x}'_0 - 1) + \ldots + (\hat{x}'_n - 1) = -1$$

where

$$\hat{x}'_i \triangleq \begin{cases} 2, & \text{if } \hat{x}_i = 0 \text{ and } i \in \{0, \ldots, j-1\} \\ 0, & \text{if } \hat{x}_i = 2 \text{ and } i \in \{0, \ldots, j-1\} \\ \hat{x}_i, & \text{if } \hat{x}_i = 1 \text{ or } i \in \{j, \ldots, n\}. \end{cases}$$

– Find $k \in \{0, \ldots, n\}$ such that the cyclic shift of the vector $\hat{\mathbf{x}}'$ by k positions to the right transforms it to the vector $\hat{\mathbf{z}} = (\hat{z}_0, \ldots, \hat{z}_n)$ having the dominating property, i.e.,

$$(\hat{z}_0 - 1) + \ldots + (\hat{z}_{t-1} - 1) \geq 0, \ \text{ for all } \ t = 1, \ldots, n$$

where

$$\hat{z}_i \triangleq \begin{cases} \hat{x}'_{n+i-k}, & \text{if } \ i \in \{0, \ldots, k-1\} \\ \hat{x}'_{i-k}, & \text{if } \ i \in \{k, \ldots, n\}. \end{cases}$$

– Let $\mathbf{a} = (a_0, \ldots, a_{\delta-1})$, $\mathbf{b} = (b_0, \ldots, b_{\delta-1}) \in \{0,1,2\}^\delta$ are the preorder codewords numbered k and j in the list of codewords for non–regular binary trees with δ nodes. Delete the last bit 0 in the vectors $\hat{\mathbf{z}}, \mathbf{a}, \mathbf{b}$, concatenate the received vectors, and attach bit 0. Thus,

$$f(\hat{\mathbf{x}}) = (\hat{z}_0, \ldots, \hat{z}_{n-1}, a_0, \ldots, a_{\delta-2}, b_0, \ldots, b_{\delta-2}, 0).$$

Example of encoding for $n = 7$ is given in Table 4, where we use the preorder codewords $\mathbf{c}_0, \ldots, \mathbf{c}_8$ for non–regular binary trees in Figure 2 to encode k and j.

In the ternary case, the balanced vector is understood as the vector with the difference between the number of 0's and the number of 2's equal to 1. We cannot construct such a vector by substitutions $2 \to 0$ and $0 \to 2$ if $n - n_1$ is even. Therefore we increase the length of the input vector by 1 and place "a parity symbol" at the $(n+1)$–st position. This symbol should be 1 if $n - n_1$ is odd and either 0 or 2 if $n - n_1$ is even. After the first step all further transformations become similar to the previous ones for regular binary trees : in fact, we process the vector of length

$$n' \triangleq \begin{cases} n - n_1, & \text{if } n - n_1 \text{ is odd} \\ n + 1 - n_1, & \text{if } n - n_1 \text{ is even} \end{cases}$$

which is obtained if we puncture all 1's in the vector $(\hat{x}_0, \ldots, \hat{x}_n)$. Namely (see (4) and (5)), j can be defined as the smallest index, for which

$$\hat{s}_n = 2\hat{s}_j - 1$$

and k is defined by

$$\begin{cases} \hat{s}'_i > \hat{s}'_k, & \text{if } i \in \{0, \ldots, k-1\} \\ \hat{s}'_i \geq \hat{s}'_k, & \text{if } i \in \{k, \ldots, n-1\} \end{cases}$$

where $\hat{s}_0 = \hat{s}'_0 \triangleq 0$ and

$$\begin{aligned} \hat{s}_t &\triangleq (x_0 - 1) + \ldots + (x_{t-1} - 1) \\ \hat{s}'_t &\triangleq (x'_0 - 1) + \ldots + (x'_{t-1} - 1) \end{aligned}$$

for all $t = 1, \ldots, n$.

Table 1: Some parameters of the code for regular binary trees.

n	δ	N	n/N
7	9	23	0.3043
15	9	31	0.4839
31	11	51	0.6078
63	11	83	0.7590
127	13	151	0.8411
255	15	283	0.9011
511	15	539	0.9480
1023	17	1055	0.9697
2095	17	2127	0.9850
4095	19	4131	0.9913
8191	19	8227	0.9956
16383	21	16423	0.9976
32767	23	32811	0.9987

Table 2: The list of preorder codewords for regular binary trees having 9 nodes.

i	$\mathbf{c}_i$
0	101010100
1	101011000
2	101100100
3	101101000
4	101110000
5	110010100
6	110011000
7	110100100
8	110101000
9	110110000
10	111000100
11	111001000
12	111010000
13	111100000

Table 3: Example of encoding for regular binary trees, $n = 13$.

		$\mathbf{x} =$	1011011000000
$j = 11$		$\mathbf{x}' =$	0100100111100
$k = 6$		$\mathbf{z} =$	1111000100100
$k = 6$	$\to$	$\mathbf{a} =$	110011000
$j = 11$	$\to$	$\mathbf{b} =$	111001000
$N = 29$		$f(\mathbf{x}) =$	3611322720 (octal)

Table 4: Example of encoding for non–regular binary trees, $n = 7$.

		$\hat{\mathbf{x}} =$	0112221
$w = 3$		$\hat{x}_7 =$	0
$j = 5$		$\hat{\mathbf{x}}' =$	21100210
$k = 3$		$\hat{\mathbf{z}} =$	21021100
$k = 3$	$\rightarrow$	$\mathbf{a} =$	12100
$j = 5$	$\rightarrow$	$\mathbf{b} =$	11200
$N = 17$		$f(\hat{\mathbf{x}}) =$	2102110121011200

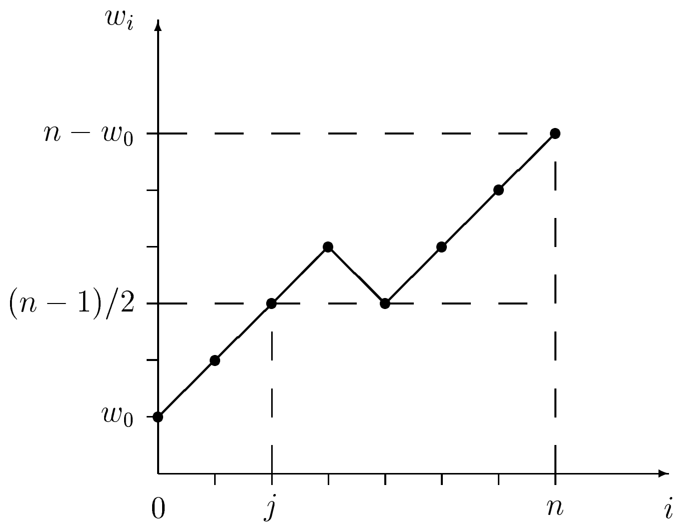

Figure 1: The Hamming weights of the vectors constructed by the inversion of the first i components of the vector $\mathbf{x} = 0001000$, where $i = 0, \dots, n$ and $n = 7$. The balanced vector $\mathbf{x}' = 1101000$ is received by the inversion of the first $j = 2$ components.

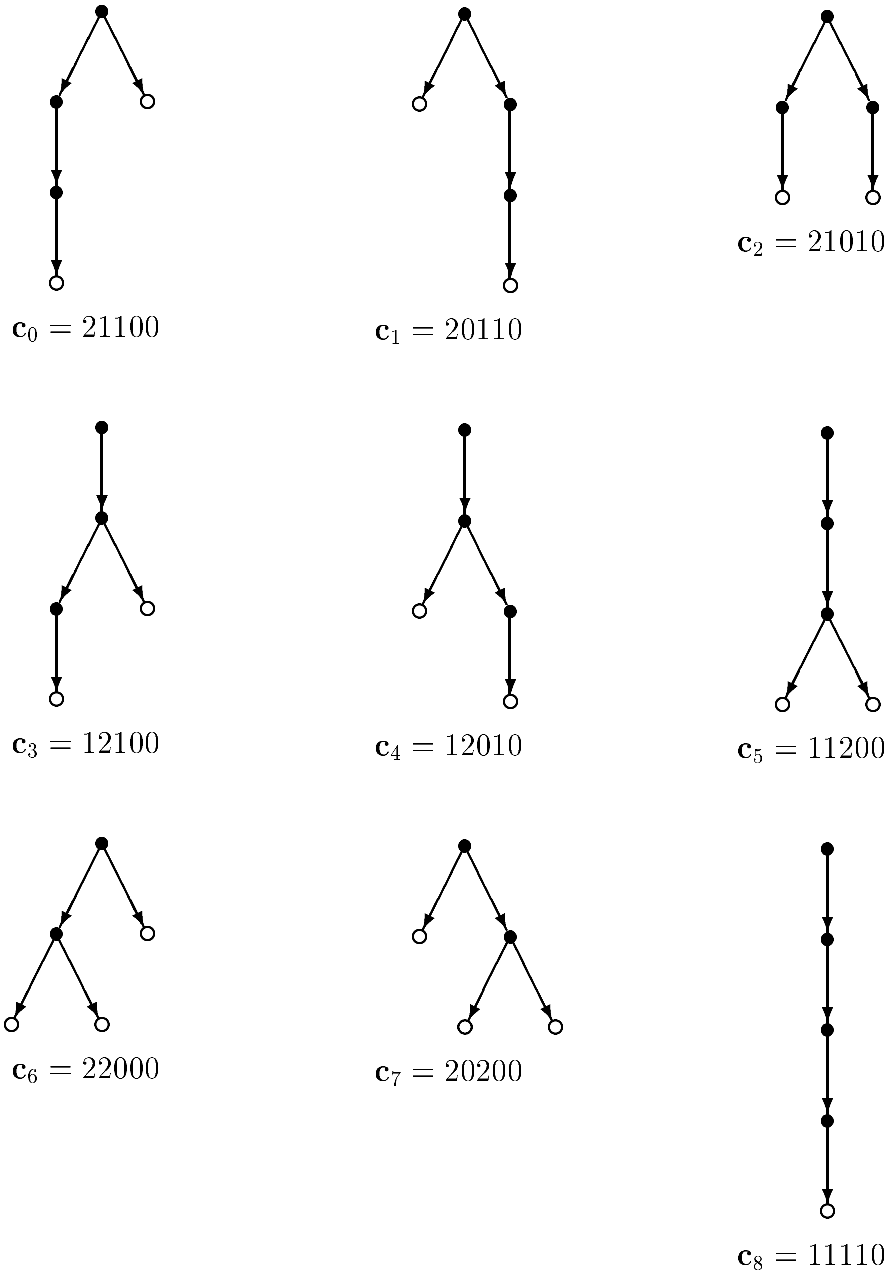

Figure 2: The non–regular binary trees having 5 nodes and corresponding preorder codewords.

References

[1] Even, S. (1973) *Algorithmic Combinatorics.* Macmillan, London.

[2] Mäkinen, E. (1996) A survey on binary tree codings. *Comp. J.,* **34**, 438–443.

[3] Ruskey, F., and Hu, T. C. (1977) Generating binary trees lexicographically. *SIAM J. Comput.,* **6**, 745–758.

[4] Pallo, J. M. (1986) Enumerating, ranking and unranking binary trees. *Comp. J.,* **29**, 171–175.

[5] Sekharan, C. N., Del Greco, J. G. and Rathi, S. (1996) Space–time trade–offs in the relative unranking of binary trees. *Comp. J.,* **39**, 36–44.

[6] Balakirsky, V. B. (1994) A class of block codes and their decoding for transmission without synchronization. *Proc. 1994 IEEE International Symposium on Information Theory,* Trondheim, Norway, 64.

[7] Knuth, D. E. (1986) Efficient balanced codes. *IEEE Trans. Inform. Theory,* **32**, 51–53.

[8] Graham, R. L., Knuth, D. E. and Patashnik, O. (1989) *Concrete Mathematics.* Addison–Wesley, New York.

[9] de Bruijn, N. G. and Morselt, B. J. M. (1967) A note on plane trees. *J. Combinatorial Theory,* **2**, 27–34.

[10] Er, M. C. (1988) A simple algorithm for generating non–regular trees in lexicographical order. *Comp. J.,* **31**, 61–64.

[11] Babkin, V. F. (1971) A universal encoding method with nonexponential work expenditure for a source of independent messages. *Probl. Inform. Transmission,* **7**, 288–294.

[12] Cover, T. M. (1973) Enumerative source coding. *IEEE Trans. Inform. Theory,* **19**, 73–77.