A tight upper bound on the size of the antidictionary of a binary string

Hiroyoshi Morita[1†] and Takahiro Ota[2]

[1]*Graduate School of Information Systems, University of Electro-Communications, Chofugaoka 1–5–1, Chofu, Tokyo, 182–8585 Japan.* [2]*Dept. of Electronic Engineering, Nagano Prefectural Institute of Technology, Shimonogo 813–8, Ueda, Nagano, 386–1211 Japan.*

A tight upper bound of the size of the antidictionary of a binary string is presented. And it is shown that the size of the antidictionary of a binary sting is always smaller than or equal to that of its dictionary. Moreover, an algorithm to reconstruct its dictionary from its antidictionary is given.

Keywords: antidictionary, minimum forbidden words, suffix trees, data compression, ECG

1 Introduction

An antidictionary is a set of words that never appear in a binary string. In 2000, Crochemore et al. (2000) presented a compression algorithm of binary text using antidictionary called DCA. Their coding algorithm has been tested on the Calgary Corpus, and their experimental results show that we get compression ratios equivalent to those of most common compressors such as pkzip. Recently, an online source coding scheme based on DCA is presented to apply for compressing losslessly ECG (ElectroCardioGram) in Ota and Morita (2004). Experimental results show that their algorithm achieved 10% smaller compression ratio than LZ ones.

In this article, we present au upper bound of the size of the antidictionary of a binary string. The upper bound we obtained is stronger than that in Crochemore et al. (1998). And it is tight in the sense there exists a string to attain the bound. We also proved that the antidictionary of a binary string is always smaller than or equal to that of the dictionary of the same string. Moreover, we give an algorithm to reconstruct the dictionary from the antidictionary.

This article is organized as follows. Section 2 gives definitions on antidictionary with some examples. In Sections 3 and 4, we investigate the size of the antidictionary of a given string and derive a tight upper bound on its size. Section 5 presents an algorithm to reconstruct the dictionary from the antidictionary of a given string and Section 6 summarizes our results.

2 Definitions on Antidictionary

Let $\mathcal{A}$ be the binary alphabet $\{0, 1\}$ and $\mathcal{A}^*$ be the set of all finite-length strings over $\mathcal{A}$ including the null string of length zero, denoted by λ. The dictionary $\mathcal{D}(\boldsymbol{x})$ of a binary string $\boldsymbol{x} = x_1x_2 \dots x_n \in \mathcal{A}^*$ is defined as the set of all the substrings of $\boldsymbol{x}$:

$$\mathcal{D}(\boldsymbol{x}) = \{x_\ell x_{\ell+1} \cdots x_m | 1 \leq \ell \leq m \leq n\} \cup \{\lambda\}.$$

For example, if $\boldsymbol{x} = 01011$, then $\mathcal{D}(\boldsymbol{x})$ is given by

$$\mathcal{D}(01011) = \{\lambda, 0, 1, 01, 10, 11, 010, 011, 101, 0101, 1011, 01011\}.$$

Letting $\mathcal{D}^c(\boldsymbol{x}) = \mathcal{A}^* \backslash \mathcal{D}(\boldsymbol{x})$, a string $\boldsymbol{v} = v_1v_2 \cdots v_m \in \mathcal{D}^c(\boldsymbol{x})$ such that

$$v_1v_2 \cdots v_{m-1} \in \mathcal{D}(\boldsymbol{x}) \text{ and } v_2v_3 \dots v_m \in \mathcal{D}(\boldsymbol{x})$$

Tab. 1: LIST OF ANTIDICTIONARY OF SEVERAL $\boldsymbol{x}$'S.

$\boldsymbol{x}$	$\mathcal{AD}(\boldsymbol{x})$	$\boldsymbol{x}$	$\mathcal{AD}(\boldsymbol{x})$
0	$\{00\}$	000	$\{0000\}$
1	$\{11\}$	001	$\{000, 10, 11\}$
00	$\{000\}$	010	$\{00, 101, 11\}$
01	$\{00, 10, 11\}$	011	$\{00, 111, 10\}$

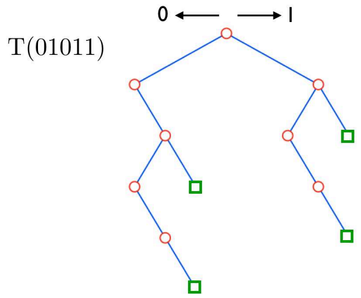

Fig. 1: Suffix trie of $\boldsymbol{x} = 01011$.

is called a minimal forbidden word (MFW) of $\boldsymbol{x}$. The antidictionary of $\boldsymbol{x}$, denoted by $\mathcal{AD}(\boldsymbol{x})$, is defined as the set of all the MFW's of $\boldsymbol{x}$. In case of $\boldsymbol{x} = 01011$, $\mathcal{AD}(\boldsymbol{x}) = \{00, 110, 111, 1010\}$. Table 1 shows the antidictionaries of several binary strings.

Let $\mathcal{S}(\boldsymbol{x})$ be the set of suffices of $\boldsymbol{x}$:

$$\mathcal{S}(\boldsymbol{x}) = \{\lambda\} \cup \{x_i x_{i+1} \ldots x_n | 1 \le i \le n\}.$$

The suffix trie $T(\boldsymbol{x})$ is a tree structure (Gusfield, 1997) such that every suffix of $\boldsymbol{x}$ is stored as a path from the root to a node in $T(\boldsymbol{x})$ where every edge is labeled with a symbol in $\mathcal{A}$. Figure 1 shows the suffix trie of $\boldsymbol{x} = 01011$. Note that some suffices are implicitly represented as paths from the root to internal nodes in $T(\boldsymbol{x})$. In fact, every string in $\mathcal{D}(\boldsymbol{x})$ can be represented as a path from the root to a node. Reversely, for every node p in $T(\boldsymbol{x})$, a string represented by a path from the root to p is in $\mathcal{D}(\boldsymbol{x})$. Hence, we obtain the following statement.

Proposition 1 (Suffix Trie representing Dictionary). *A node in $T(\boldsymbol{x})$ corresponds uniquely to an element in $\mathcal{D}(\boldsymbol{x})$ and vice versa.*

3 A Necessary Condition on MFW's

A necessary condition that a string in $\mathcal{A}^*$ is an MFW of $\boldsymbol{x}$ can be derived by adding new nodes to $T(\boldsymbol{x})$ as follows: If p is a leaf in $T(\boldsymbol{x})$, then create two nodes connected to p. Otherwise, and if p has only a child node, create a new node connected to p. The obtained tree, denoted by $T_{\mathsf{ex}}(\boldsymbol{x})$, is a binary tree such that every internal node has two child nodes (See Fig. 2). Moreover, let $\boldsymbol{w}(r)$ be the string associated with the path from the root to a node r in $T_{\mathsf{ex}}(\boldsymbol{x})$.

It is shown that every MFW of $\boldsymbol{x}$ is represented by a path from the root to a leaf p in $T_{\mathsf{ex}}(\boldsymbol{x})$ since for its parent node q, $\boldsymbol{w}(q) \in \mathcal{D}(\boldsymbol{x})$ but $\boldsymbol{w}(p) \notin \mathcal{D}(\boldsymbol{x})$ and there exists $a \in \mathcal{A}$ such that $\boldsymbol{w}(p) = \boldsymbol{w}(q)a$. Hence, we obtain the following proposition.

Proposition 2. *If $\boldsymbol{v} \in \mathcal{A}^*$ is an MFW of $\boldsymbol{x}$, then there exists a leaf p in $T_{\mathsf{ex}}(\boldsymbol{x})$ such that $\boldsymbol{v} = \boldsymbol{w}(p)$.*

From Propositions 1 and 2, we can derive a rough estimation on the size of $\mathcal{AD}(\boldsymbol{x})$. Throughout this article, the size, or the cardinality of a set $\mathcal{S}$ is denoted by $\#\mathcal{S}$.

Theorem 1. *For $\boldsymbol{x} \in \mathcal{A}^*$, we have*

$$\#\mathcal{AD}(\boldsymbol{x}) \le \#\mathcal{D}(\boldsymbol{x}) + 1.$$

[†]This work was supported by Grant-in-Aid for Scientific Research no.173601782332 of Japan Society for the Promotion of Science.

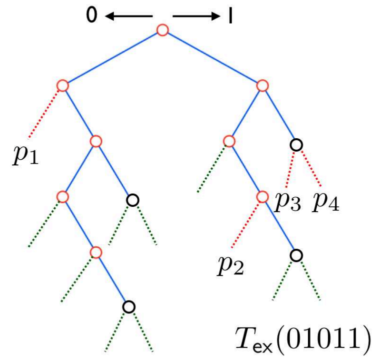

Fig. 2: $T_{\mathsf{ex}}(\boldsymbol{x})$ corresponding to the suffix trie in Figure 1.

Proof. Let m_k be the number of nodes having k child nodes in $T(\boldsymbol{x})$ where $0 \le k \le 2$. From Proposition 1, we have

$$\#\mathcal{D}(\boldsymbol{x}) = m_0 + m_1 + m_2.$$

Since $m_0 = m_2 + 1$, we can rewrite it as

$$\#\mathcal{D}(\boldsymbol{x}) = 2\, m_0 + m_1 - 1.$$

On the other hand, the number of leaves in $T_{\mathsf{ex}}(\boldsymbol{x})$ is upper bounded by $2\, m_0 + m_1$. Hence Proposition 2 gives

$$\#\mathcal{AD}(\boldsymbol{x}) \le 2\, m_0 + m_1.$$

Therefore, we have $\#\mathcal{AD}(\boldsymbol{x}) \le \#\mathcal{D}(\boldsymbol{x}) + 1$. □

Recently, Janson et al. (2004) investigated the average size of a dicitionary of a random binary string generated by a mixing model. It is asymptotically equal to $n^2/2$. In the same paper, they did more precise analysis on the average behavior of the number of distinct substrings in a string of length n over an alphabet of size A. The size of an antidictionary, however, is much smaller than that of the dictionary on average as we will discuss below.

4 A Tight Upper Bound on Size of Antidictionary

Hereafter, for any node r in $T_{\mathsf{ex}}(\boldsymbol{x})$, let $\sigma(r)$ be a node such that $\boldsymbol{w}(\sigma(r))$ is equal to the string obtained by removing the first symbol of $\boldsymbol{w}(r)$. The following theorem gives a necessary and sufficient condition that $\boldsymbol{w}(p)$ for a leaf p in $T_{\mathsf{ex}}(\boldsymbol{x})$ is an MFW of $\boldsymbol{x}$.

Theorem 2 (A necessary and sufficient condition on MFW). *For a leaf node p in $T_{\mathsf{ex}}(\boldsymbol{x})$, $\boldsymbol{w}(p)$ is an MFW of $\boldsymbol{x}$ if and only if $\sigma(p)$ is an internal node in $T_{\mathsf{ex}}(\boldsymbol{x})$.*

Proof. Suppose that p is a leaf in $T_{\mathsf{ex}}(\boldsymbol{x})$ and q is its parent node. Then there exists $a \in \mathcal{A}$ such that $\boldsymbol{w}(p) = \boldsymbol{w}(q)a$. From the definition of $T_{\mathsf{ex}}(\boldsymbol{x})$, $\boldsymbol{w}(p) \notin \mathcal{D}(\boldsymbol{x})$ while $\boldsymbol{w}(q) \in \mathcal{D}(\boldsymbol{x})$. If $\sigma(p)$ is an internal node in $T_{\mathsf{ex}}(\boldsymbol{x})$, then $\boldsymbol{w}(\sigma(p)) \in \mathcal{D}(\boldsymbol{x})$ from Proposition 1. Moreover, there exists $b \in \mathcal{A}$ such that $\boldsymbol{w}(p) = b\boldsymbol{w}(\sigma(p))$. Hence $\boldsymbol{w}(p)$ is an MFW of $\boldsymbol{x}$.

Conversely, assume that $\boldsymbol{w}(p)$ is an MFW of $\boldsymbol{x}$ for a leaf p in $T_{\mathsf{ex}}(\boldsymbol{x})$. Rewriting $\boldsymbol{w}(p)$ as $c\boldsymbol{u}$ with a certain symbol $c \in \mathcal{A}$, string $\boldsymbol{u}$ corresponds to node $\sigma(p)$, that is, $\boldsymbol{u} = \boldsymbol{w}(\sigma(p))$. Since $\boldsymbol{w}(p)$ is an MFW, $\boldsymbol{w}(\sigma(p)) \in \mathcal{D}(\boldsymbol{x})$. Therefore, $\sigma(p)$ is an internal node in $T_{\mathsf{ex}}(\boldsymbol{x})$. □

Corollary 1. *Suppose that p is a leaf in $T_{\mathsf{ex}}(\boldsymbol{x})$ and its parent node q is an internal node in $T(\boldsymbol{x})$. Then, $\boldsymbol{w}(p)$ is an MFW of $\boldsymbol{x}$ if and only if node $\sigma(q)$ has two child nodes in $T(\boldsymbol{x})$.*

Proof. If node $\sigma(q)$ has two child nodes, one of them is $\sigma(p)$. Hence $\sigma(p)$ is a node in $T(\boldsymbol{x})$. Thus, it is also an internal node in $T_{\mathsf{ex}}(\boldsymbol{x})$. Conversely, assume that $\sigma(p)$ is an internal node in $T_{\mathsf{ex}}(\boldsymbol{x})$. Since p is a leaf in $T_{\mathsf{ex}}(\boldsymbol{x})$, its parent node q has two child nodes including p. Hence, $\sigma(q)$ does so too. □

Corollary 1 shows that the size of $\mathcal{AD}(\boldsymbol{x})$ is at least m_2 where m_2 is the number of nodes having two child nodes in $T(\boldsymbol{x})$ as defined in Theorem 1. In Figure 2, strings $00, 1010, 110$ and 111 are corresponding to leaves p_1 to p_4, respectively. Since their parents satisfy the conditions of Corollary 1, these strings are MFW's of $\boldsymbol{x} = 01011$. And there are no other leaves whose parents do so.

Theorem 3 (MFW sprouting from leaves in $T(\boldsymbol{x})$). *For a leaf q in $T(\boldsymbol{x})$, string $\boldsymbol{w}(p)$ associated with p that is one of q's child nodes in $T_{\mathsf{ex}}(\boldsymbol{x})$ is an MFW if and only if the path from the root to q is the shortest one among all the leaves in $T(\boldsymbol{x})$.*

Proof. First, we assume that q is the leaf q^* with the shortest path in $T(\boldsymbol{x})$. Then $\sigma(q^*)$ is an internal node in $T(\boldsymbol{x})$. Thus, $\sigma(q^*)$ has at least one child node r in $T(\boldsymbol{x})$. Since there exists a child node p^* of q^* such that $r = \sigma(p^*)$, $\boldsymbol{w}(p^*)$ is an MFW of $\boldsymbol{x}$.

Conversely, if $q \neq q^*$, then $\boldsymbol{w}(q^*)$ is a suffix of $\boldsymbol{w}(q)$ since $\boldsymbol{w}(q)$ is strictly longer than $\boldsymbol{w}(q^*)$ and both of them are suffices of $\boldsymbol{x}$. Let g be a child node of q in $T_{\mathsf{ex}}(\boldsymbol{x})$ such that $\boldsymbol{w}(p^*)$ is a suffix of $\boldsymbol{w}(g)$ where $\boldsymbol{w}(p^*)$ is an MFW defined above. Thus neither $\boldsymbol{w}(g)$ is in $\mathcal{D}(\boldsymbol{x})$. Therefore any suffices of $\boldsymbol{w}(g)$ that are longer than or equal to $\boldsymbol{w}(p^*)$ are not in $\mathcal{D}(\boldsymbol{x})$. Hence, $\boldsymbol{w}(g)$ is not an MFW. Taking the contraposition completes the proof of Theorem 3. □

For example, two leaves associated with strings 110 and 111 in $T_{\mathsf{ex}}(01011)$ of Figure 2 are MFW's that satisfy the condition of Theorem 3. Finally, we have that $\mathcal{AD}(01011) = \{00, 110, 1010, 111\}$.

Theorem 4 (An improved bound of Theorem 1). *Given a binary string $\boldsymbol{x}$ of length n, we have*

$$\#\mathcal{AD}(\boldsymbol{x}) \leq n + 1.$$

And if $\boldsymbol{x}$ is the all-one string $1 \cdots 1$ or the all-zero string $0 \cdots 0$, then

$$\#\mathcal{AD}(\boldsymbol{x}) = 1.$$

Proof. Combining the results of Corollary 1 and Theorem 3, we have

$$\#\mathcal{AD}(\boldsymbol{x}) \leq m_2 + 2.$$

Since $m_0 \leq n$ and $m_0 = m_2 + 1$, the above inequality is evaluated further from above as follows:

$$\#\mathcal{AD}(\boldsymbol{x}) \leq m_0 + 1 \leq n + 1.$$

If $\boldsymbol{x}$ is the all-one string $1 \cdots 1$ of length n, the all-one string of length $n+1$ is an MFW of $\boldsymbol{x}$ and any other strings are not. Therefore, $\#\mathcal{AD}(\boldsymbol{x}) = 1$. In case $\boldsymbol{x}$ is the all-zero string of length n, the same argument derives the equality. □

Since the equality holds for $\boldsymbol{x} = 01$ (see Table 1), the upper bound obtained in Theorem 4 is tight. In case of the binary alphabet, Corollary 9 in Crochemore et al. (1998) is translated into

$$\#\mathcal{AD}(\boldsymbol{x}) \leq \begin{cases} 3 & \text{if } |\boldsymbol{x}| \leq 2, \\ 2 & \text{else if } \boldsymbol{x} \text{ is the all-one string } 1 \cdots 1 \text{ or the all-zero string } 0 \cdots 0, \\ 2n - 2 & \text{else} \end{cases}$$

where $|\boldsymbol{x}|$ is the length of $\boldsymbol{x}$. For $n \geq 4$, the results in Theorem 4 is stronger than the above one.

Moreover, since all the suffices of $\boldsymbol{x}$ including the null string are contained in $\mathcal{D}(\boldsymbol{x})$, we have $\#\mathcal{D}(\boldsymbol{x}) \geq n + 1$. Therefore, we obtain the following corollary.

Corollary 2. *For $\boldsymbol{x} \in \mathcal{A}^n$,*

$$\#\mathcal{AD}(\boldsymbol{x}) \leq \#\mathcal{D}(\boldsymbol{x}).$$

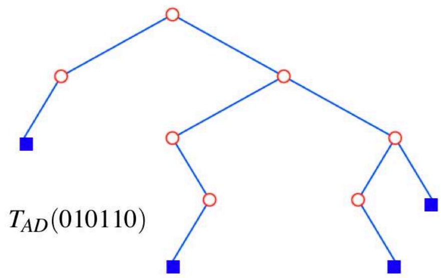

Fig. 3: An example of digital trie $T_{AD}(010110)$.

5 From Antidictionary to Dictionary

The antidictionary $\mathcal{AD}(\boldsymbol{x})$ of a given string $\boldsymbol{x} \in \mathcal{A}^*$ can be represented as a digital trie denoted by $T_{AD}(\boldsymbol{x})$. As an example, Figure 3 shows a digital trie representing the antidictionary of $\boldsymbol{x} = 010110$ where $\mathcal{AD}(010110) = \{00, 1010, 1101, 111\}$. An MFW in $\mathcal{AD}(\boldsymbol{x})$ is corresponding to a path from the root to a leaf in $T_{AD}(\boldsymbol{x})$.

In Algorithm **AD2D** described below, starting with $T_{AD}(\boldsymbol{x})$, nodes in $T(\boldsymbol{x})$ will be reproduced step by step. However, it will be determined on each process of the algorithm whether those nodes are internal or external nodes in $T(\boldsymbol{x})$. Hence, nodes created by the algorithm are tentatively called 'neutral' nodes until we know their connectivity, that is, the number of their child nodes and their directions. Besides, at the initial state of the algorithm, internal nodes in $T_{AD}(\boldsymbol{x})$ should be treated as neutral nodes since their connectivity are not known at the initial state.

(Algorithm **AD2D**)

0. Let T be $T_{AD}(\boldsymbol{x})$.
1. For each neutral node p in T in the breadth-first order:
   1-1. If $\sigma(p)$ has two internal nodes as child nodes, create one or two new nodes connecting to p so that p has two child nodes.

   1-2. If $\sigma(p)$ has only one internal node q as a child node, create a new node connecting to p so that p has a child node with the same direction as q.

   1-3. If $\sigma(p)$ has no child nodes, let p be an external node.
2. Remove all the external nodes in T_{AD} from T.
3. Output T as $T(\boldsymbol{x})$. Then stop.

Figure 4 depicts the process of reconstructing the dictionary of $\boldsymbol{x} = 010110$ from its antidictionary.

Theorem 5. *The suffix trie $T(\boldsymbol{x})$ is reproduced from $T_{AD}(\boldsymbol{x})$ by means of Algorithm* **AD2D**.

Proof. Let p be an internal node in $T_{AD}(\boldsymbol{x})$. From the definition of MFW, $\boldsymbol{w}(p)$ is in $\mathcal{D}(\boldsymbol{x})$. Suppose that p has only a child node. Then, Algorithm **AD2D** sprouts a new child node q from p if $\sigma(q)$ is not an external node in T. The new node q becomes neutral and $\boldsymbol{w}(q) \in \mathcal{D}(\boldsymbol{x})$.

The connectivity of a neutral node r is determined by that of $\sigma(r)$. That is, if $\sigma(r)$ has k child nodes in $T(\boldsymbol{x})$ where $k = 0, 1, 2$, then r does so. Since the algorithm processes neutral nodes in the breadth-first order, the connectively of $\sigma(r)$ is known when r is processed. Thus, all the suffices of $\boldsymbol{x}$ will be reproduced in Step 1 of the algorithm. Removing all the external nodes in T_{AD} from T, we obtain $T(\boldsymbol{x})$. □

Since $\boldsymbol{x}$ is equal to $\boldsymbol{w}(p)$ for a certain leaf p that has the longest path among leaves in $T(\boldsymbol{x})$, Algorithm **AD2D** can reproduce $\boldsymbol{x}$ from $\mathcal{AD}(\boldsymbol{x})$. Hence we have the following corollary.

Corollary 3. *The original string $\boldsymbol{x}$ can be reproduced from $\mathcal{AD}(\boldsymbol{x})$.*

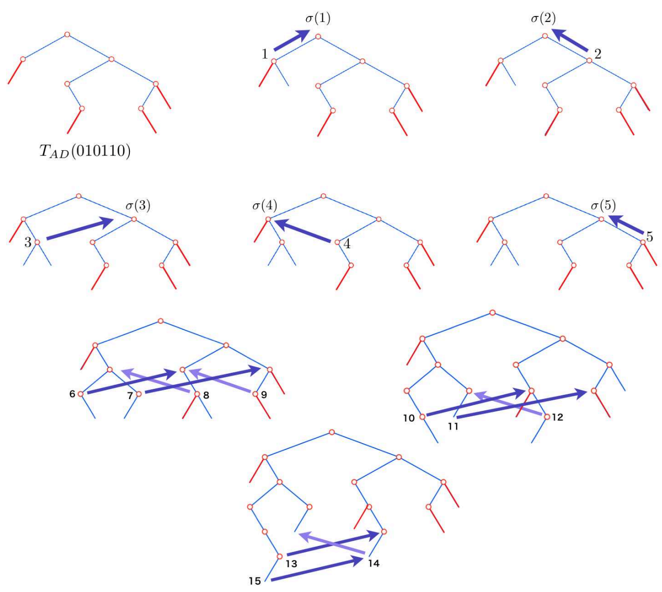

Fig. 4: $T(010110)$ is reproduced from $T_{AD}(010110)$ by Algorithm **AD2D** where non-external nodes are indexed by numbers.

6 Conclusions

In this article, we derived an upper bound on the size of the antidictionary of a given binary string $\boldsymbol{x}$. And we proved that the antidictionary of $\boldsymbol{x}$ is always smaller than or equal to the dictionary of $\boldsymbol{x}$. Moreover, we gave an algorithm to reconstruct the dictionary of $\boldsymbol{x}$ from the antidictionary of $\boldsymbol{x}$.

Acknowledgements

The authors express their thanks to Philippe Flajolet of INRIA, Rocquencourt, France to call their attention to Janson et al. (2004).

References

M. Crochemore, F. Mignosi, and A. Restivo. Automata and forbidden words. *Information Processing Letters*, 67(3):111–117, 1998.

M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi. ata compression using antidictionaries. *Proc. IEEE*, 88(11):1756–1768, 2000.

D. Gusfield. *Algorithms on strings, trees, and sequences: Computer Science and Computational Biology*. Cambridge Univ. Press, 1997.

S. Janson, S. Lonardi, and W. Szpankowski. On average sequence complexity. *Theoretical Computer Science*, 326:213–227, 2004.

T. Ota and H. Morita. One-path ecg lossless compression using antidictionaries. *IEICE Trans. Fundamentals (Japanese Edition)*, J87-A(9):1187–1195, 2004.